Lodz University of Technology, Poland
Faculty of Technical Physics, Information Technology and Applied Mathematics
Institute of Information Technology

Computer Game Innovations 2023

Edited by:
Dominik Szajerman and Rafał Szrajber

Lodz University of Technology Monograph
Łódź 2024

Editors:
Dominik Szajerman
Rafał Szrajber

Reviewed by:
prof. dr hab. inż. Ewa Napieralska-Juszczak
prof. dr hab. inż. Liliana Byczkowska-Lipińska

Technical editor:
Dominik Szajerman

Reproduced from materials supplied by the Editors.

ISBN: ???-??-?????-??-?

Łódź University of Technology Press
93-005 Łódź, 223 Wólczańska St.
E-mail: zamowienia@info.p.lodz.pl
www.wydawnictwo.p.lodz.pl

Contents

Grass Plant Simulation in Real-Time Applications Using GPU

Radosław Bednarski[1][0000–0003–0468–6401], **Bartłomiej Drągowski**, **Dominik Szajerman**[2][0000–0002–4316–5310]

[1]*Stefan Batory Academy of Applied Sciences*
Institute of Information and Technology Science
Batorego 64c, 96-100 Skierniewice, Poland
rbednarski@ansb.pl
[2]*Lodz University of Technology*
Institute of Information Technology
Politechniki 8, 93-590 Łódź, Poland
dominik.szajerman@p.lodz.pl

Abstract. *The article presents a method of simulating a large amount of interactive grass in computer games with the use of a graphics card, optimizing the number of calculations. In addition to wind simulation, the method is also to take into account the bending of the grass under moving objects and under the influence of the gust generated by low-flying helicopters.*
Keywords: *computer games, plant simulation, real time rendering*

1. Introduction

Real-time applications, such as computer games, play an increasingly important role in the modern world. Despite the development of hardware and more and more powerful computers, optimization considerations are an important factor that is a challenge for developers, affecting the operation of applications. This is especially visible in the case of three-dimensional computer games, where we usually have a few milliseconds to process a single image frame. This is a very small time window, in which, in addition to drawing and displaying the image on the screen, you often have to take into account, among others, physics simulation, animation calculation. To meet these requirements, computer game developers often have to resort to various types of simplifications or tricks aimed at reducing the number of calculations performed. More and more graphics cards are used, the processors of which are much better than typical general-purpose processors. One of the elements of the three-dimensional world that requires this type of solution is vegetation, especially grass. These are objects that usually occur en masse in open space. Displaying so many objects on the screen is a challenge, and the whole thing becomes even more complicated if we also want to include the physics simulation.

2. Related work

YouTube user Jonas Mølgaard, in a series of videos posted on his channel, presented a method of simulating grass in computer games, using the physics system available in the Unreal Engine. Due to its specificity, this method focuses on slightly larger plants with clearly separated branches[1]. In this method, each plant is treated as a separate three-dimensional model. This model has a bone system configured in such a way that there is one bone for each branch of the plant, which allows for independent movement of each branch.

To alleviate the performance problem, Mølgaard also proposed optimizing the entire system. It involves a sphere attached to the player's character that detects nearby plants. If the plant is outside this sphere, the physical objects assigned to it are excluded from the simulation process. Thanks to this, only plants close to the player are processed, which is often the same as foreground objects visible on the screen Fig. 1 Asbjørn Thirslund, a video game developer known as Brackeys

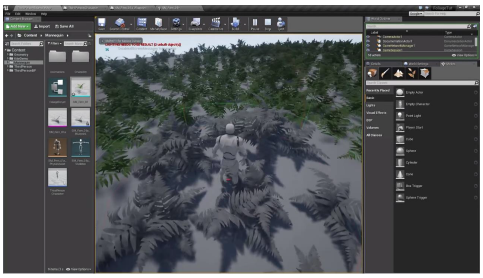


Figure 1. Visualization of only nearby plants being activated [2].

in the Unity community, has created a YouTube tutorial on how to create swinging grass. The technique he showed is kind of the opposite of the previously mentioned Jonas Mølgaard method and is intended to simulate thick grass. Although it only takes into account wind movement, it uses the GPU for this purpose and shows well the practical use of the Vertex Shader, which is the basis for possible developments [3]. The grass on the stage consists of many objects with very simple geometry arranged next to each other Fig. 6. Each such object consists of several intersecting planes and does not require a bone system. This solution facilitates the use of instanced rendering on a large scale, which will allow you to draw thousands of objects using one command. A grass texture is applied to each such plane. To animate it, the method requires appropriate distribution of these

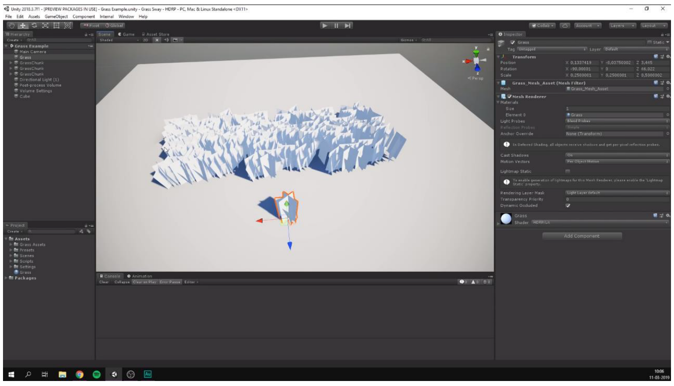


Figure 2. Grass object geometry [4].

planes on the UV map. Each plane must be laid out so that its lower part (the grass root) is at the very bottom of the UV map and its upper part (the top of the grass) is at the very top. Thanks to this, the vertical axis of the UV map determines the gradient, which serves as the deformation factor of the grass geometry Fig. 3. The deformation consists in shifting the geometry vertices by the appropriate vector in the plane perpendicular to the grass. If we take Y as the vertical axis, then the shift is performed in the XZ plane. A solution that takes into account both interac-

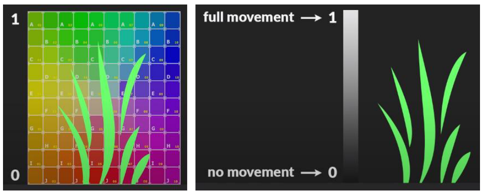


Figure 3. a) Grass UVW map b) Gradient resulting from UVW map [5].

tions with other objects and a gust of wind was presented in the work Animating Prairies in Real-Time[6], and an extension of this method can be found in the work A procedural approach to animate interactive natural sceneries [7]

To display large amounts of grass on the screen efficiently, the method uses the Level Of Detail (LOD) mechanism. This mechanism assumes that the model exists in at least two versions with different levels of detail. The version displayed depends on how far the object is in the field of view. To optimize this process, the scenery was divided into small areas using a grid. Thanks to this, instead of setting the LOD for each grass object separately, it is set for the area within which the grass group is displayed. The method uses three levels of detail Fig. 4:

- 3D - used in areas closest to the camera. Each blade of grass has its own geometry.

- 2.5D - used in areas in the background. The grass is displayed using planes, similar to Asbjørn Thirslund's method.
- 2D - used in background areas. It involves projecting a flat grass texture onto a distant area.

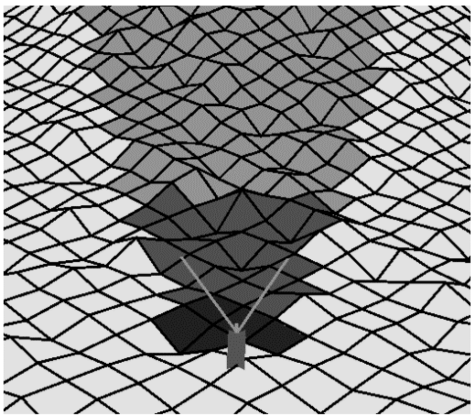


Figure 4. The principle of operation of the LOD mechanism based on the terrain mesh [6].

Only 3D and 2.5D levels are animated. For each grass model, a set of versions with different degrees of deflection has been prepared. These versions were generated using computer simulation. During rendering, the direction of deflection and its degree are determined for each object, defined as an index in the range $[-1, 1]$ (Fig. 5). Based on the index value, a version of the model representing a given degree of deflection is selected. The direction and degree of tilt (index) are stored in the graphics card memory as a texture. A slightly different solution, taking into ac-

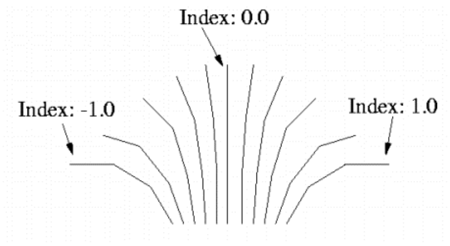


Figure 5. Grass geometry tilt based on predefined, indexed versions [6].

count both wind and collision with objects, was presented in the work "Simulation and Rendering for Millions of Grass Blades" [8]. As in the previous example, the scenery is divided into areas using a grid and also takes into account three levels of LOD. The main difference is that each blade of grass is processed individually, both for rendering and simulation, regardless of the LOD level.

A blade of grass is stored in memory as a polyline curve with 16 vertices. During rendering, the blade geometry is generated based on the mentioned curve. The easiest way to achieve this is using Geometry Shader, but the creators point out that greater efficiency is achieved by generating the vertices in advance and saving them in the graphics card's memory buffer Fig. 6 Each area consists of

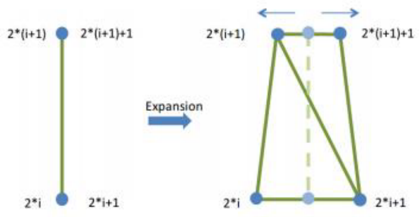


Figure 6. Creating grass blade geometry based on a polyline curve [8].

64 blades. However, the LOD level determines how many of them are displayed. Starting from the most detailed level, 64, 32 and 16 shoots per area are displayed successively. Because information about the blades in individual areas is stored in the graphics card's memory, such an LOD system effectively uses instanced rendering and transformation in the vertex shader Fig. 7. The simulation and an-

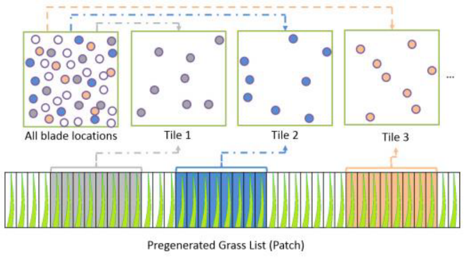


Figure 7. Memory sharing for grass blade [8].

imation process is divided into two, independent parts. The first one concerns a gust of wind and is implemented in a classic way, i.e. using a Vertex Shader (as in the example presented by Thirslund). The second part concerns the reaction of the grass to collisions with other objects. The process of collision detection and reaction to them (grass bending) is carried out for each blade separately. It is done using a computational shader that bends curves that are broken under the pressure of other objects, using the Verlet algorithm. Afterwards, it restores the curves to their original form.

From the above analyses, it can be concluded that the main role in the interactive grass simulation process is played by the optimal use of the graphics card. This also applies to situations when we use a ready-made physical system. Although it most often performs calculations on the CPU, it should be borne in mind that modern systems can also use computational shaders for this purpose. Appropriate optimization at each of the three stages (rendering, animation, simulation) is equally important, both in terms of calculations performed and memory usage. The key to optimal rendering is the appropriate LOD mechanism. In the case of animation, vertex transformation in the vertex shader is often used, which is a relatively simple and effective solution. The greatest diversity occurs in simulation methods. A solution using a texture representing grass parameters on the map seems to be the optimal one. Although it does not provide fully realistic effects, their quality is good enough to be successfully used in video games. This solution is not complicated, uses little memory, and can be easily integrated with the animation.

3. Proposal of an original solution

The Unity engine (version 2019.4.16f1 LTS) was used to implement the project in a configuration based on the scriptable URP (Universal Render Pipeline) rendering pipeline.On an empty scena, a fragment of flat terrain with dimensions of 150 × 150 was created, which was painted with a grass texture. Two controllable vehicles were placed on the stage - an off-road vehicle and a helicopter. The 3D models, textures and scripts used for this purpose were downloaded from the Unity Asset Store [9]. To add aesthetic value, a basic configuration of lighting, shadows and post-processing was carried out Fig. 8. The next stage was to prepare a three-

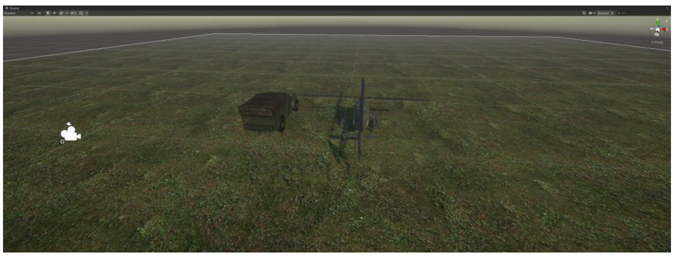


Figure 8. View of the project scene.

dimensional grass object that could be displayed on the screen in large quantities. For this purpose, the method described in Section 2 was used, with intersecting planes on which a texture with transparency was applied. The grass blade texture was downloaded from the Unity Asset Store [9]. The planes were created in Blender v2.82a in three levels of geometry complexity (LOD) Fig. 9. The model

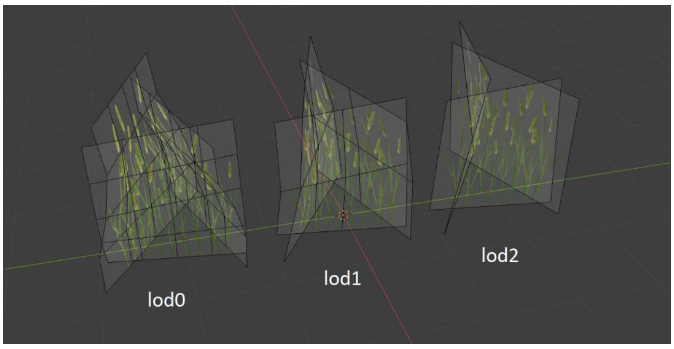


Figure 9. Three dimensional grass model.

prepared in this way must be properly displayed in the Unity environment Fig. 10. Since the animation will be implemented through vertex transformation, it is necessary to create your own shader. For this purpose, the Shader Graph tool was used to create a shader in the PBR Master configuration, which, after providing lighting parameters, carries out the entire shading process in the PBR (Physically-Based Rendering) model.

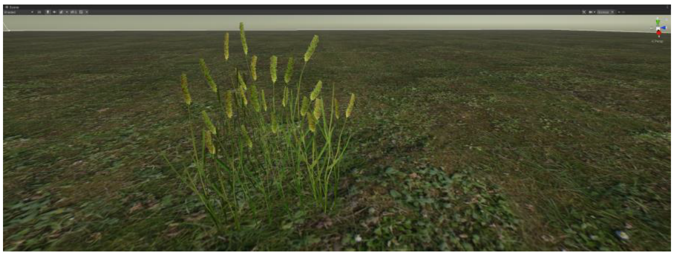


Figure 10. Grass in Unity Engine.

3.1. Procedural grass rendering

Many thousands of objects are needed to fill the entire scene with grass. Arranging them manually is not only tedious, but also not very efficient. Despite the use of instanced rendering, each such object will participate in the process of determining its visibility on the screen, which, with a very large number of objects, is a significant burden on the CPU. For this reason, procedural rendering was implemented based on the method described in the article "Simulation and Rendering for Millions of Grass Blades" [8]. The scene area was divided into smaller, virtual sectors with dimensions of 5×5, which, with an area of 150×150, gave 900 sectors within which grass objects are displayed. The Unity CullingGroup API was used to determine the visibility of a given sector on the screen. This tool also allows you to define distance groups from the camera, thanks to which in each image frame Unity provides information about the sectors visible on the screen, divided into groups, which we can use as LOD levels. The grass LOD levels configured in the project are presented in Table 1.

Table 1. Grass LOD levels

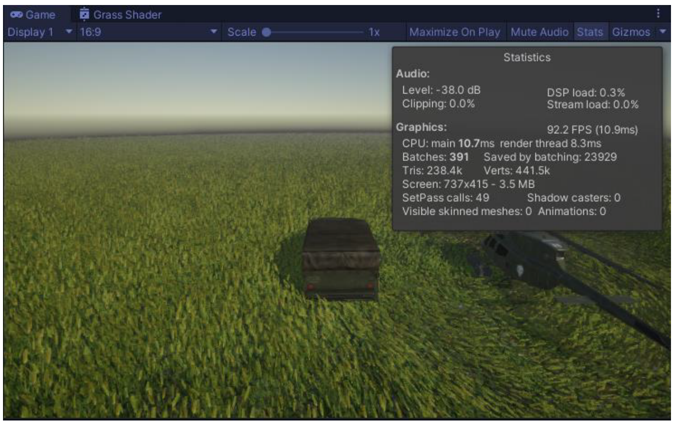
LOD Level	Maximum distance	Grass Geometry	Shadowing
0	5	lod0	Yes
1	25	lod1	Yes
2	60	lod2	Yes
3	100	lod2	No

Two buffers have been used to solve potentially high memory consumption. The first contains a set of four-dimensional vectors filled with pseudorandom floating-point numbers with the range and use given in Table 2. The second buffer is the indexing buffer, which contains as many pseudorandom integers as there are sectors. Each of them represents the index offset in the first buffer. The density of the displayed grass is 36 objects per sector, arranged in a 6×6 grid. Each object within the sector is indexed from 0 to 35. Since we can also index sectors (in the case of the tested scene - from 0 to 899), we are able to assign one of the buffer vectors to each grass object in the sector. The x, y, and z components of this vector serve as an additional, pseudo-random shift of the object from its point on the grid. In the coordinate system used in Unity, the translation takes place in the XZ plane and the rotation takes place around the Y axis. The project generates 3,600 vectors in the buffer, which provides a relatively large variety in the arrangement of the grass, using about 60 kB of memory including the indexing buffer. The last com-

Table 2. The composition of the random vector

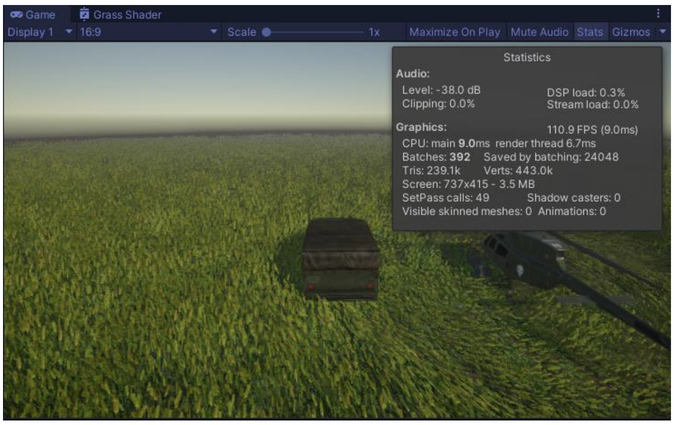
Vector component	Range	Application
X	$[-1, 1]$	Offset object from a grid point (on the plane)
Y	$[-1, 1]$	
Z	$[0, 2]$	Rotation around the height axis
W	$[0, 1]$	Additional visibility factor

ponent of the vector is the additional visibility factor, which is used to optimize the displayed image. This optimization consists in saving the processing power of the graphics card at the expense of a slight decrease in image quality, consisting in a smooth reduction of the grass density with increasing distance from the camera. In addition to the division into distance groups provided by CullingGroup, its exact distance from the camera is calculated individually for each object. This distance is normalized to the interval [0,1] and compared with the w component of the random vector. If this value is greater than the w component, the object is not drawn. This is done by moving it outside the visible frame. Although such an object is still processed by the graphics card as part of the vertex shader, it is completely cut off from the shader fragment, which is much more computationally demanding. In Fig. 11 and Fig. 12 we see an almost 20 % increase in performance after enabling the mentioned optimization.



```
Statistics
Audio:
  Level: -38.0 dB              DSP load: 0.3%
  Clipping: 0.0%               Stream load: 0.0%
Graphics:                      92.2 FPS (10.9ms)
  CPU: main 10.7ms  render thread 8.3ms
  Batches: 391    Saved by batching: 23929
  Tris: 238.4k    Verts: 441.5k
  Screen: 737x415 - 3.5 MB
  SetPass calls: 49         Shadow casters: 0
  Visible skinned meshes: 0  Animations: 0
```

Figure 11. Optymization off - average 93 FPS



```
Statistics
Audio:
  Level: -38.0 dB              DSP load: 0.3%
  Clipping: 0.0%               Stream load: 0.0%
Graphics:                      110.9 FPS (9.0ms)
  CPU: main 9.0ms  render thread 6.7ms
  Batches: 392    Saved by batching: 24048
  Tris: 239.1k    Verts: 443.0k
  Screen: 737x415 - 3.5 MB
  SetPass calls: 49         Shadow casters: 0
  Visible skinned meshes: 0  Animations: 0
```

Figure 12. Optymization on - average 110 FPS

3.2. Grass animation

In the presented method, grass animation focuses on the deformation of the grass geometry according to. given parameters inside the vertex shader. The deformation method and its assumptions are based on the method presented by Asbjørn

Thirslund. In addition to the requirement to properly distribute the geometry on the UV map, its center (position 0,0,0) must also be located in the lower part of the model. In this way, the y-value (height axis) of each vertex's position simultaneously defines its height from the grass root. Two parameters are needed to perform the deformation:

- a two-dimensional unit vector defining the direction of the grass inclination on the XZ plane.
- inclination coefficient in the range [0, 1], where 1 means full inclination and 0 - no inclination (initial state of the geometry).

Both parameters come from the parameter map processed in the simulation stage. The animation consists of three stages. The first one is to determine the inclination angle, which is a value in the range $[0, \frac{\pi}{2}]$. For this purpose, the following formula (1) is used, using the gradient determined by the UV map.

$$\alpha = s * \frac{\pi}{2} * \sqrt[4]{y_{uv}} \tag{1}$$

Where:

- α- tilt angle
- s - tilt factor
- y_{uv} - gradient resulting from the UV map

The second step is to move the vertex to its target position. It consists of two parts

- a shift on the XZ plane according to formula (2)
- a shift along the Y axis according to formula (3)

These patterns rotate the vertex p relative to the point $p_x, 0, o_z$, which is its ground geometry representing the grass root. For this purpose, they use the fact that the y component of the vector's position is also its distance from the root. This ensures that the moved vertex is at the same distance from its root as in the original position. In this way, the effect of unnatural stretching of the grass geometry is minimized.

$$p'_{xz} = p_{xz} + (d_{xz} * \sin\alpha * p_y) \tag{2}$$

$$p'_y = p_y * \cos\alpha \tag{3}$$

Where:

- p'_{xyz} - target vertex position

- p_{xyz} - primary vertex position
- α - tilt angle
- d_{xz} - direction angle

The last, third stage is the rotation of the normal vector assigned to the vertex. This requires specifying a vector indicating the axis around which the normal vector should be rotated. To rotate it in the direction specified by the dxy vector, the axis vector must be perpendicular to it and in the same plane. Since the direction vector always lies in the XZ plane, a cross product was made on it with the vector pointing to the Y axis.

3.3. Map of animation parameters

Based on the solution presented in the article [7], another map was introduced to the proposed solution. This map contains grass animation parameters in its channels. The red and green channels contain the X and Y components of the vector determining the direction of inclination. The tilt factor is in the alpha channel. The blue channel is not used, which potentially creates the possibility of introducing an additional parameter. The resolution of the created map is 1024×1024 pixels. For an area of 150×150, such a high resolution is not needed, but it allowed for better and more accurate analysis and visualization of the simulation process. In this case, 512×512 is completely sufficient, and after enabling MSAA anti-aliasing [10] for the rendering texture that constitutes the map, even 256×256. In order for the grass shader to be able to sample the animation parameter map, it must be provided with information about the position and size of the terrain. With this data, it is possible to normalize the vertex position in the game world to a value in the range [0,1], used when the graphics card samples the texture.

In order for the grass shader to be able to sample the animation parameter map, it must be provided with information about the position and size of the terrain. Thanks to this data, it is possible to normalize the vertex position in the game world to a value in the range [0,1], used when the graphics card samples the texture.

3.4. Simulation primitives

Following the solution presented in [7], animation primitives were used to update the parameter map, which for the purposes of the presented solution were called simulation primitives. They are a kind of two-dimensional masks that manipulate a specific area on the map. As mentioned in subsection 3.3, the parameter map consists of a render texture. This is a special type of texture that allows the graphics card to "draw" on it. In other words, the graphics card can use it as image output - as if it were an additional monitor. In each image frame, the texture content is cleared and then all active primitives in the scene are drawn on it again. For this purpose, the Unity GL API was used.

Two types of primitives were implemented in the project. The first one is rectangular directional. This is a basic primitive that sets a uniform direction and tilt factor on a rectangular area inside the map. The script controlling this primitive takes into account the height of the assigned object in the Y axis. Thanks to this, it is possible to calculate the tilt factor so that the grass geometry transformed in the animation visually reaches the level of the primitive. The direction of the tilt is determined based on the route it takes in the three-dimensional world while the application is running. It was used in a car whose shape resembles a rectangle from above. Fig. 13shows the configuration of the car simulation primitives. It is worth emphasizing the creation of separate primitives for the wheels and the slightly higher chassis. Thanks to this, by more precisely matching the animated grass to the body of the vehicle, it appears more realistic. When primitives overlap, the one with the highest skew becomes dominant. This is due to a properly configured blending stage that the graphics card performs based on the alpha channel storing this coefficient.

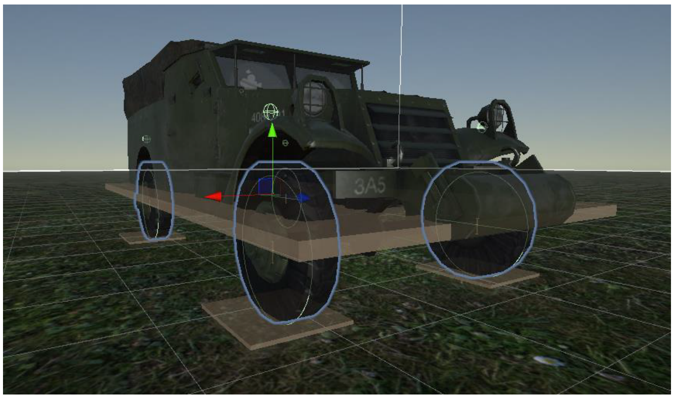


Figure 13. Car simulation primitives.

The second primitive is the “whirlwind” primitive. It is much more customizable and finds its application in the gust of wind generated by the helicopter blades. It has the shape of a circle whose size is determined by two radii - external and internal Fig. 14. They are used to interpolate the camber coefficient between the maximum value at the center and the minimum (zero) value outside the wheel. Two factors influence the maximum camber ratio. The first one is the height above the ground of the object to which the primitive is attached. Unlike the rectangle primitive, here the height is not used directly. It is first normalized to the interval [0,1] based on the previously determined maximum height. The second factor is optional noise. In the case of helicopter propellers, a sine function moving in time was used. This relatively simple and basic trigonometric function, when properly set its scale and speed, reproduces the gust generated by rotating helicopter propellers surprisingly well.

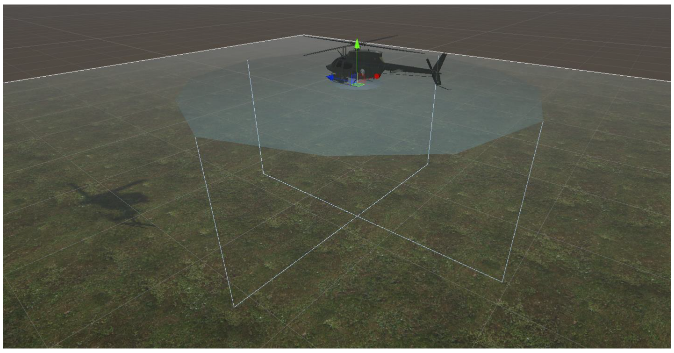


Figure 14. Helicopter simulation primitives.

3.5. Conception of simulation and target texture

Directly using the rendering texture described in section 3.3 has a serious drawback - it does not remember its previous state. Changes in the location of primitives result in rapid changes in the parameter map and, consequently, in the animation of the grass. If the sloping grass is no longer affected by the primitive, it will immediately return (within one image frame) to its original form, instead of linearly returning to it over time as it does in reality. A similar jumping effect also occurs in the other direction, when the grass is tilted.

For this reason, the parameter map was divided into two textures with the same dimensions. The first one is the previously described rendering texture, which serves as the target texture. The second one is the simulation texture, which remembers its previous state. This means that with each image frame its values are not overwritten with new ones, but are updated accordingly. The simulation texture update is done using a compute shader. For each map pixel, it takes the values of both textures and then interpolates the simulation texture value towards the value of the target texture. The interpolation taking place here is not linear, but depends on the difference between both values.

The first interpolated parameter is the tilt coefficient. The method of its interpolation depends primarily on whether the target value means a greater or lesser degree of inclination. If the grass is tilted (pressed against the ground), we expect the visible effect to occur quickly due to the (most common) collision between the grass and the object. In the opposite situation, the return of the grass to its original form should be much slower due to the air resistance that actually occurs. The new tilt coefficient and its interpolation speed are calculated successively using the formulas (4,5)

$$\begin{aligned}
&s_s' = s_s + \Delta s \\
&\Delta s = ch(s_s, s_t, s_{speed}) \\
&ch(a,b,p) = \begin{cases} b-a & |b-a| > p*\Delta t \\ p*\Delta t * sgn(b-a) & |b-a| \leqslant p*\Delta t \end{cases}
\end{aligned} \tag{4}$$

$$s_{speed} = \begin{cases} 5*(s_t - s_s) + 1 & s_t > s_s \\ \frac{\sqrt[8]{s_s}}{4} * min(10*(s_s - s_t, 1)) & s_t \leqslant s_s \end{cases} \tag{5}$$

where:

- s_s - value from simulation texture,
- s_s' - the new value,
- s_t - value from target texture,
- s_{speed} - speed of factor interpolation,
- Δt- time elapsed since the last update.

The process of updating the tilt direction is a little more complex. Its change is performed in a circle, so first the direction values are converted from a two-dimensional vector to radians using the formula (6). The change of direction itself looks similar to the change of the coefficient and is performed using formulas (8) and (9). Not only the difference between the current and target direction is taken into account, but also the previously calculated coefficient difference and the tilt coefficient itself. This is due to the introduction of the inertia coefficient, which determines the ability of the grass object to change direction. In a vertical position, the grass has no problem bending in any direction, but in a tilted position, changing direction is much more difficult. Additionally, changing direction is only possible during a positive change in the tilt coefficient. Thanks to this, if the direction on the target texture suddenly changes significantly while maintaining the same coefficient, the grass blades will not rotate unnaturally - since they have already been "squeezed" by the object, they should theoretically not be able to move freely until they are released. In the calculations, the inertia parameter changes the value of the target direction, which is visible in the formula (7)

$$\beta_s = \arctan(d_s, xy) \qquad \beta_t = \arctan(d_{t,xy}) \tag{6}$$

$$\begin{aligned}
&i = \sqrt{s_s} \\
&\beta_t' = \begin{cases} \mathrm{lerp}(\beta_s,\beta_t, i) & \Delta s > 0 \\ \beta_s & \Delta s \leqslant 0 \end{cases} \\
&lerp(a,b,t) = a + (b-a)*t \qquad t \in [0,1]
\end{aligned} \tag{7}$$

$$
\begin{aligned}
d'_{s,xy} &= [\sin(\beta_s + \Delta\beta), \cos(\beta_s + \Delta\beta)] * {s'_s}^2 \\
\Delta\beta &= ch(\beta_s, \beta'_t, \beta_{speed})
\end{aligned}
\tag{8}
$$

$$\beta_{speed} = 10 * lerp(1, |\Delta s|, s_s) \tag{9}$$

where:

- s_s, β_s - direction from simulation texture,
- d'_s - new direction,
- $d_t, \beta t$ - direction from target texture,
- β'_s - target direction resulting from inertia,
- i - inertia factor,
- d_{speed}- direction interpolation speed.

The multiplication performed in formula (8) is worth attention. It affects the length of the vector but not its direction, which is crucial from the point of view of the entire algorithm. This multiplication means that the smaller the tilt factor, the shorter the direction vector. Although it does not change the operation of the algorithm itself, this procedure stabilizes the interpolation process between pixels when the graphics card samples the texture. Without this operation, you can notice clear jumps in the grass direction on the edges of the primitives, resulting from interpolation between pixel values. It is this simulation texture that is sampled by the grass shader as a texture containing the final parameters ready to be visualized through a grass animation.

3.6. Wind simulation

The wind simulation created in this work is an extension of the method presented by Asbjørn Thirslund [3]. It was performed in a grass shader. The result of its operation are the parameters of the direction and the slope coefficient of the grass, which are then mixed with the values read from the map. Unlike Asbjørn Thirslund's method, a previously generated texture with Perlin noise was used as noise. The noise texture was generated using Perlin Noise Maker[11]. In the grass shader, this texture is sampled separately for the X component of the tilt vector and the Z component. Sampling coordinates are determined based on the position of the grass object in world space and two configurable parameters: NoiseSize and GWindSpeed. The first one determines the scale of the noise texture, the second one - the speed at which it moves in time. The length of the resulting vector determines the value of the tilt coefficient based on the GWindForce parameter.

Blending the received parameters with the map parameters is done using a variable that determines the intensity of the blended wind direction in relation to that coming from the map. The wind should only be an addition to the parameters derived from the simulation and should only have an effect in the free (vertical) position of the grass. For this purpose, Hermite's cubic interpolation [12] (smoothstep) was used based on the tilt coefficient derived from the map. The resulting direction and coefficient are calculated using formula (10) and (11).

$$s = max(s_s, s_w) \tag{10}$$

$$d = lerp(d_s + s_w, d_s, smoothspte(s_s) \tag{11}$$

where:

- s_s - factor from the map,
- s_w - wind factor,
- s- the resulting tilt factor,
- d_s - direction from the map,
- d_w - wind direction,
- d - the resulting ilt direction.

The direction vector thus obtained is finally normalized. Before that,it is further increased by epsilon $\epsilon = 0.0001$ in the X axis to avoid the impossibility of normalizing the zero vector.

Theoretically, it is possible to implement the wind directly in the computational shader when processing the texture of the simulation, but it was found that implementation directly in the shader provides greater possibilities and the quality itself does not depend on the map resolution.

4. The final result

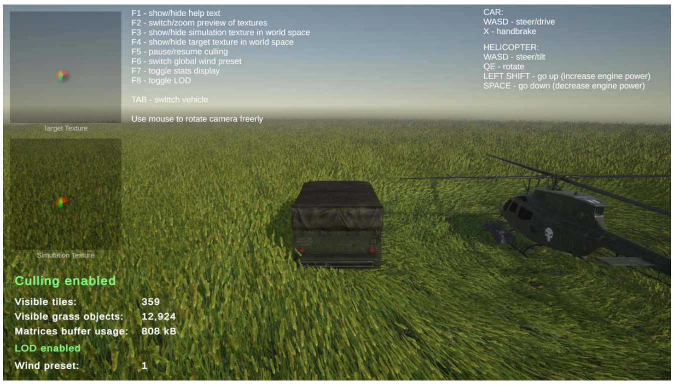
As mentioned in the previous sections, the final project created in this work consists of a fragment of flat terrain fully covered with grass. In this area, the user has two controllable vehicles available - an off-road vehicle and a helicopter, between which they can switch using the TAB key. All information regarding the control method and available functions is visible on the screen in the form of text, which can be turned off using the F1 key. On the left side of the screen, at the top, you can see a preview of the target texture and the simulation texture. Their location next to each other allows for a more detailed comparison and analysis of the operation of the implemented solution. Using the F2 key you can toggle their visibility, including full screen preview. These textures can also be visualized in three-dimensional space, as an additional layer displayed on the terrain.

The visualization can be activated independently for each of the two textures using the F3 and F4 keys. The bottom left corner of the screen displays statistics related to the procedural grass rendering described Subsection 3.1. They inform about the number of sectors and grass objects visible on the screen, as well as the degree of memory consumption of the buffer storing the transformation matrices. Using the F5 key, you can pause the process of cutting the visibility to the field of view (culling process), which also results in stopping the updating of the displayed statistics. This functionality has several applications. It not only allows you to more accurately analyze the number of visible objects and LOD levels, but also allows you to "create" an empty area without grass, allowing for better visualization of textures in three-dimensional space. You can use the F8 key to disable the LOD system, which forces all grass objects to be rendered in the most detailed version, and also disables the optimization of the density of grass objects. The last implemented functionality is the ability to change wind settings. Using the F6 key, you can switch wind parameters between three predefined settings. The parameters used for each setting are included in Table 3.

Table 3. Wind setup available in application

Parameter	Setup #1	Setup#2	Setup#3
NoiseSize	64	32	256
GWindSpeed	0.5	0.4	2.0
GWindForce	0.2	0.3	0.5
GWindInfluence	0.005	0.005	0.005

The effects of the application are presented in Figures 15, 16, 17.



F1 - show/hide help text
F2 - switch/zoom preview of textures
F3 - show/hide simulation texture in world space
F4 - show/hide target texture in world space
F5 - pause/resume culling
F6 - switch global wind preset
F7 - toggle stats display
F8 - toggle LOD

TAB - switch vehicle

Use mouse to rotate camera freerly

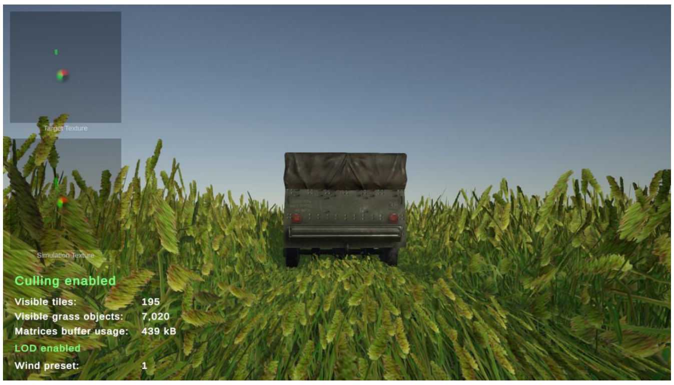
CAR:
WASD - steer/drive
X - handbrake

HELICOPTER:
WASD - steer/tilt
QE - rotate
LEFT SHIFT - go up (increase engine power)
SPACE - go down (decrease engine power)

Culling enabled

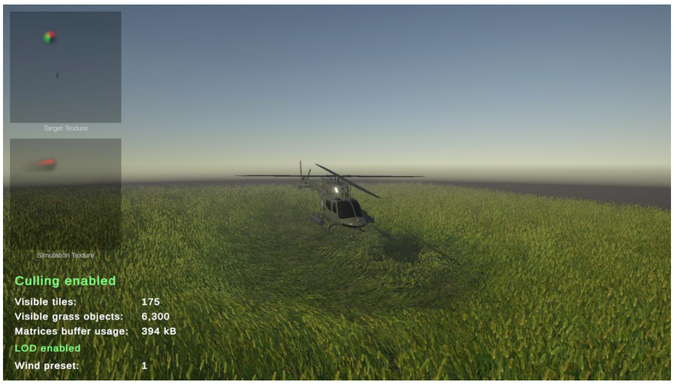
```
Visible tiles:           359
Visible grass objects:   12,924
Matrices buffer usage:   808 kB
LOD enabled
Wind preset:             1
```

Figure 15. Application start screen.



```
Culling enabled
Visible tiles:          195
Visible grass objects:  7,020
Matrices buffer usage:  439 kB
LOD enabled
Wind preset:            1
```

Figure 16. The impact of the car on the grass.



```
Culling enabled
Visible tiles:          175
Visible grass objects:  6,300
Matrices buffer usage:  394 kB
LOD enabled
Wind preset:            1
```

Figure 17. The impact of the helicopter on the grass.

5. Performance analysis

In the case of this type of applications, one of the most important aspects is their performance measured in the number of displayed image frames per second. It depends primarily on the power of the hardware on which the application is run. A computer with an Intel Core i7-7700HQ @ 2.8 GHz processor and an NVIDIA GeForce GTX 1050 Ti 4 GB graphics card was used for testing. The MSI Afterburner program was used to measure the number of frames per second.

Due to the limited amount of available materials and the lack of executable files available for download, the performance was compared only with the solution described in [8]. Of the solutions analyzed in the section 2, the authors of the mentioned work included the most extensive performance analysis. Additionally, the test platform used in it (processor: AMD A10-6800K @ 4.1 GHz, graphics card: AMD Radeon HD 7970) is characterized by performance very similar to the platform used by the authors, which is confirmed by numerous comparative tests conducted by existing online technology services [13], [14]. The only unknown is the screen resolution used by the authors, which significantly affects the rendering performance. The demonstration video attached to their work was recorded in 1280 × 720 resolution, so this was adopted for most of the tests [15].

The first test performed by the authors of the reference work was a test of rendering performance with various scene settings and LOD system activity. The results they achieved (in frames per second) are presented in 4. Unfortunately, the authors did not specify how to interpret the tested parameters shown in the table. It is not known whether the number of grass blades per sector concerned a sector of the same size or whether it was enlarged appropriately to obtain the same grass density. The same applies to the indicated numbers of sectors - it is not specified whether they refer to the total number of sectors on the stage (stage size) or the number of sectors visible on the screen. In the first case, it was decided to gradually increase the density of objects per sector while maintaining its size. In the second case, three scenarios were tested. In the first one, the number of sectors was increased, thus increasing the size of the area. In the second one, the grass drawing distance was increased, thus forcing a larger number of sectors visible on the screen. The third scenario is not related to sectors and concerns the screen resolution. It was additionally carried out to compare the difference in performance between the FullHD resolution of 1920 × 1080 and the assumed 1280 × 720. The results are presented in Tables 5, 6, and 7. They express the number of image frames displayed per second on the application start screen, which guarantees the same camera position in each case.

Table 4. Rendering performance [FPS] in [15]

Tile No.	2K		4K		8K		16K	
Blades/Tile	LOD		LOD		LOD		LOD	
	no	yes	no	yes	no	yes	no	yes
64	71	73	66	67	54	56	28	44
128	72	76	56	62	32	60	15	41
256	57	72	30	49	16	36	7	23
512	31	50	16	35	8	20	4	12

From Table 5, we can see that the size of the map in a sector has a marginal impact on performance. This is due to the fact that it does not affect the number of sectors actually processed and displayed. The only thing that changes is the number of objects whose visibility must be determined by the CullingGroup in each frame of the image. The efficiency of the tool provided by Unity is very high, even with several dozen thousand objects.

The results in Tables 6 and 7 are consistent with our predictions. The increased

Table 5. Performance depending on the map size [FPS]

	Map size in sectors							
	30 × 30		60 × 60		120 × 120		240 × 240	
Sector density	LOD		LOD		LOD		LOD	
	no	yes	no	yes	no	yes	no	yes
6 × 6	53	75	52	74	52	73	52	70
9 × 9	29	43	29	43	29	42	28	42
12 × 12	20	30	19	29	19	29	19	28
17 × 17	12	18	12	18	12	18	12	18

Table 6. Performance depending on the grass drawing distance [FPS]

	Grass drawing distance							
	+0%		+33%		+66%		+100%	
Sector density	LOD		LOD		LOD		LOD	
	no	yes	no	yes	no	yes	no	yes
6 × 6	53	75	41	61	33	50	28	43
9 × 9	29	43	22	34	18	28	15	24
12 × 12	20	30	15	23	12	19	10	16
17 × 17	12	18	9	14	7	11	5	9

grass drawing distance also increases the number of sectors visible on the screen, which directly translates into the number of objects drawn. However, you can see that increasing this distance by 33% still ensures high and stable frame rate above 60 frames per second. In the case of a higher resolution with more than twice as many pixels, the performance drop is not that great. Although it can be seen that the increased number of pixels puts more strain on the shader, the performance is still at a satisfactory level, maintaining smoothness around 60 frames per second.

The second test carried out by the authors was the analysis of the execution time of individual stages within a single image frame. The test results are shown in Table 8. For a similar analysis proposed by the authors of the medot, the performance profiler built into Unity was used. Data from the key stages of the project are presented in Table 9

Table 8 shows that in the reference work, rendering and simulation take almost the same amount of image frame time. The data presented in Table 9 show the very high efficiency of the CullingGroup system. On the other hand, it is clear that the bottleneck in the presented method is rendering that takes up almost the entire frame time. The high efficiency of the simulation stage based on the parameter map and primitives may be surprising.

Table 7. Performance depending on the screen resolution[FPS]

	Screen resolution			
	1280×720		1920×1080	
Sector density	LOD		LOD	
	no	yes	no	yes
6×6	53	75	40	57
9×9	29	43	25	41
12×12	20	30	17	24
17×17	12	18	11	15

Table 8. Image frame analysis in [15]

	GPU (ms)	Vertex Shader	Pixel Shader	Compute Shader	Texture
Grass simulation	5.6	0	0	99.99%	99,7%
Grass rendering	5.6	99.9%	75.0%	0	32.1%
Skybox	0.3	0.11%	31.3%	0	92.2%
Terrain	0.3	99.5%	72.7%	0	10.9%

6. Conclusion

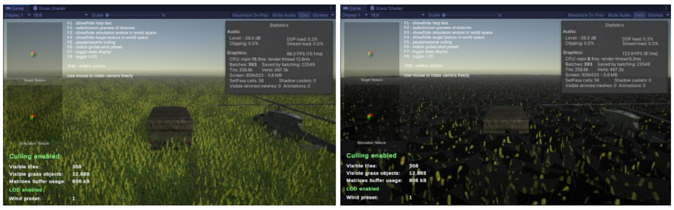
To sum up, the method proposed in the article is similar in terms of performance to the performance of the reference method described in the article [8]. In favor of the presented one is the much higher efficiency of the simulation stage, which is based on a parameter map instead of an independent simulation of each blade of grass separately. Grass rendering is slightly more efficient in the method proposed in the article [8]. Analyzing more closely the principle of operation of both methods, there are two main reasons that may affect the lower rendering efficiency of the method proposed by the authors.

The first is the high complexity of the grass shader, or more precisely, its most demanding part - the fragment shader. This is due to two reasons - alpha clipping and the PBR lighting model. Alpha clipping allows you to "clip the geometry" of a rendered object where its texture is fully transparent, thus adjusting the depth buffer only to visible fragments of the geometry. This allows overlapping grass textures to be displayed correctly on the screen. Unfortunately, alpha clipping requires the use of the discard operation. Its side effect is to disable the hardware optimization of the depth test on the graphics card, which has a significant negative impact on shader performance [42]. In the reference method, each blade of grass has its own geometry adapted to it, which does not require alpha clipping. On the other hand, creating geometry tailored to the grass blades may result in very high complexity, thus burdening the vertex shader. As for the lighting model, the PBR model used performs the process of shading the object based on the physical

Table 9. Analysis of image frame occupancy

Step	Image frame occupancy
CullingGroup	0.03 ms
Grass rendering	9.8 ms
Parameter map simulation	0.24 ms

properties of its surface, such as roughness or metallicity. This model requires the use of much more complex mathematical formulas than, for example, the classic Phong lighting model [43]. Unfortunately, the Shader Graph tool does not allow you to change the lighting model and for this purpose it would be necessary to write your own shader. A simpler lighting model would certainly improve performance, but it is difficult to estimate how much and the reduction in visual quality of the rendered grass must also be taken into account. The impact of alpha clipping and lighting on performance is shown in Fig. 18 and 19.



```
F1 - show/hide help text
F2 - switch/zoom preview of textures
F3 - show/hide simulation texture in world space
F4 - show/hide target texture in world space
F5 - pause/resume culling
F6 - switch global wind preset
F7 - toggle stats display
F8 - toggle LOD

TAB - switch vehicle

Use mouse to rotate camera freely
```

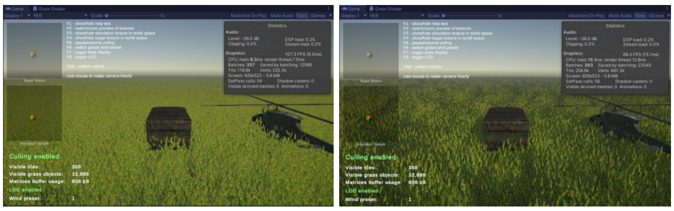
```
Statistics
Audio:
  Level: -38.0 dB              DSP load: 0.2%
  Clipping: 0.0%               Stream load: 0.0%
Graphics:                      66.3 FPS (15.1ms)
  CPU: main 15.1ms  render thread 12.6ms
  Batches: 393   Saved by batching: 23549
  Tris: 258.6k      Verts: 467.3k
  Screen: 929x523 - 5.6 MB
  SetPass calls: 56        Shadow casters: 0
  Visible skinned meshes: 0  Animations: 0
```

```
Culling enabled
Visible tiles:             358
Visible grass objects:     12,888
Matrices buffer usage:     806 kB
LOD enabled
Wind preset:               1
```

```
Statistics
Audio:
  Level: -38.0 dB              DSP load: 0.3%
  Clipping: 0.0%               Stream load: 0.0%
Graphics:                      123.9 FPS (8.1ms)
  CPU: main 8.1ms  render thread 6.3ms
  Batches: 393   Saved by batching: 23549
  Tris: 258.6k      Verts: 467.3k
  Screen: 929x523 - 5.6 MB
  SetPass calls: 56        Shadow casters: 0
  Visible skinned meshes: 0  Animations: 0
```

```
Culling enabled
Visible tiles:             358
Visible grass objects:     12,888
Matrices buffer usage:     806 kB
LOD enabled
Wind preset:               1
```

Figure 18. a) Alpha clipping on - avg. 66 FPS. b) Alpha clipping off - avg. 123 FPS.



```
F1 - show/hide help text
F2 - switch/zoom preview of textures
F3 - show/hide simulation texture in world space
F4 - show/hide target texture in world space
F5 - pause/resume culling
F6 - switch global wind preset
F7 - toggle stats display
F8 - toggle LOD

TAB - switch vehicle

Use mouse to rotate camera freely
```

```
Statistics
Audio:
  Level: -38.0 dB              DSP load: 0.2%
  Clipping: 0.0%               Stream load: 0.0%
Graphics:                      107.3 FPS (9.3ms)
  CPU: main 9.3ms  render thread 7.0ms
  Batches: 357   Saved by batching: 12569
  Tris: 118.6k      Verts: 220.3k
  Screen: 929x523 - 5.6 MB
  SetPass calls: 54        Shadow casters: 0
  Visible skinned meshes: 0  Animations: 0
```

```
Culling enabled
Visible tiles:             358
Visible grass objects:     12,888
Matrices buffer usage:     806 kB
LOD enabled
Wind preset:               1
```

```
Statistics
Audio:
  Level: -38.0 dB              DSP load: 0.2%
  Clipping: 0.0%               Stream load: 0.0%
Graphics:                      66.3 FPS (15.1ms)
  CPU: main 15.1ms  render thread 12.6ms
  Batches: 393   Saved by batching: 23549
  Tris: 258.6k      Verts: 467.3k
  Screen: 929x523 - 5.6 MB
  SetPass calls: 56        Shadow casters: 0
  Visible skinned meshes: 0  Animations: 0
```

```
Culling enabled
Visible tiles:             358
Visible grass objects:     12,888
Matrices buffer usage:     806 kB
LOD enabled
Wind preset:               1
```

Figure 19. a) PBR light off - avg. 107 FPS. b) PBR light on - avg. 66 FPS.

The second reason concerns the overhead of the Unity engine on the overall application performance. The authors of the reference method implemented it by creating an application in C++ using the DirectX 11 library. This way, they had full control over the operation of individual stages and the calculations performed by the hardware. Unfortunately, the greatest advantage of a ready-made game engine can also be its greatest disadvantage. The game engine, in order to be as universal as possible, performs calculations in the background that are often unnecessary. The reason for this is that it is difficult to predict when a programmer will want to use a given functionality or how. For this reason, the ready-made engine performs some of the calculations in advance so that the programmer does not have to deal with them when necessary. In the case of the DrawMeshInstanced function used in the proposed method, the Unity engine performs a number of additional operations on the matrices provided in the buffer, such as determining the inverse transformation matrix, which are not necessary for rendering. The scale of the problem in question is shown by an experiment performed by Nick Caston, a YouTube user [16]. He aimed to create an identical game in Unity and in his own engine using C++ and the OpenGL library. In extreme cases, the efficiency of your own engine can be even an order of magnitude higher. On the other hand, creating your own engine for this purpose requires not only advanced knowledge and skills, but also a lot of time.

Despite the problems described above, the achieved system efficiency is at a high level. Although the procedural rendering takes a significant part of the image frame time, the partial drop in performance at FullHD 1920×1080 resolution does not result in any noticeable drop in fluidity. It is also worth mentioning that all tests performed concern extreme situations when almost only grass is visible on the screen. In reality, however, scenes in computer games are a bit more varied. Sculpting the terrain or placing additional objects on it may, contrary to appearances, significantly improve the performance of grass rendering. This is due to the fact that CullingGroup according to Unity documentation also takes into account the presence of other objects in the scene when determining visibility. This means that in such a case the number of sectors visible on the screen decreases significantly because sectors hidden by, for example, buildings are not processed. This reduces the amount of grass objects processed and drawn, thus improving the performance of the entire system.

References

[1] Mølgaard, J. Physics interactable foliage. URL `https://www.youtube.com/playlist?list=PLT4Seaj0a6mbF_zINb7qE5yc7-g1FaqaP`.

[2] Nearby plant visualization. URL `https://www.youtube.com/watch?v=ZFXwhkchN28`.

[3] A.Thirslund. Grass sway in unity - shader graph. URL `https://www.youtube.com/watch?v=L_Bzcw9tqTc`.

[4] Grass object geometry. URL `https://www.youtube.com/watch?v=L_Bzcw9tqTc`.

[5] Grass uvw map. URL https://www.youtube.com/watch?v=L_Bzcw9tqTc.

[6] Perbet, F. and Cani, M.-P. Animating prairies in real-time. *Proceedings of the 2001 symposium on Interactive 3D graphics*, 2001.

[7] Sylvain, G., Perbet, F., Raulo, D., Faure, F., and Cani, M.-P. A procedural approach to animate interactive natural sceneries. *Proceedings of the 16th International Conference on Computer Animation and Social Agents*, 2003.

[8] Fan, Z., Li, H., Hillesland, K., and Sheng, B. Simulation and rendering for millions of grass blades. *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*, 2015.

[9] Unity asset store, 2023. URL https://assetstore.unity.com.

[10] Anti-aliasing. URL https://learnopengl.com/Advanced-OpenGL/Anti-Aliasing.

[11] Perlin noise marker. URL http://www.kitfox.com/projects/perlinNoiseMaker/index.html.

[12] Hermit interpolation. URL https://en.wikipedia.org/wiki/Smoothstep.

[13] Amd radeon hd 7970 vs nvidia geforce gtx 1050 ti. URL https://www.gpucheck.com/pl-pln/compare/amd-radeon-hd-7970-vs-nvidia-geforce-gtx-1050-ti/intel-core-i7-2600k-3-40ghz-vs-intel-core-i7-6700k-4-00ghz/.

[14] Nvidia geforce gtx 1050 ti vs amd radeon hd 7970. URL https://technical.city/en/video/GeForce-GTX-1050-Ti-vs-Radeon-HD-7970.

[15] Simulation and rendering for millions of grass blades [video],. URL https://dl.acm.org/doi/10.1145/2699276.2699283.

[16] Unity dots vs handbuilt: Sample project. URL https://www.youtube.com/watch?v=tInaI3pU19Y.

Creating an Advanced Combat System in an CRPG Game Based on Similarity of Symbols Drawn by the Player Using Neural Networks

Igor Budzyński[0009–0003–5500–4535], **Damian Pęszor**[0000–0002–3754–2212]

Silesian University of Technology
Department of Graphics, Computer Vision and Digital Systems
Akademicka 16, 44-100 Gliwice, Poland
igobud01@gmail.com
damian.peszor@polsl.pl

Abstract. *The aim of the article is to present a designed combat system for an action game based on the recognition of hand-drawn symbols using machine learning. The neural network model, which handles the classification of drawn symbols, uses deep-metric learning algorithms. The project focused on the ability to combine multiple actions into a time sensitive of dexterity-based tests of skill in both reflex, accuracy and planning.*
Keywords: *computer games, artificial intelligence*

1. Introduction

Machine learning is one of the branches of artificial intelligence that involves the use of algorithms and data to increase the accuracy of the results obtained by imitating human learning. The algorithms used in machine learning are designed to find correlations and patterns in data sets and make the best decisions based on them. A programmed model requires a large set of data from which it will learn. In supervised machine learning, which is used in this study, each data sample on which the model is trained is provided with information about the correct result, thanks to which the network knows what result it should receive. Once trained, one can use another dataset to test the reliability of the neural network's predictions. This technology is increasingly used in all aspects of science and technology, such as the selection of advertisements displayed on websites, weather forecasts or modern home robots.

Computer Role Playing Games (cRPG), are a genre [1, 2] of computer games in which the player takes on the role of a hero, often created by himself, and experiences an adventure in the fictional world presented. Of the many aspects of gameplay, combat is usually a very important element, which greatly affects how the genre is perceived in the general taxonomy of computer games [3, 4, 5]. Over the years in which the genre has developed and taken various forms, several frequently recurring patterns can be distinguished. The first combat mechanics in cRPG games were very similar to tabletop Role Playing Games (RPG) they were inspired by. The character had its own characteristics, and whether a certain action was achieved was determined by checking whether the value of a given characteristic was high enough, usually with some degree of randomness, so that probability distribution of success was based on character's attributes. An example of such a system is the *Baldur's Gate* series. Later, this system evolved and took various forms. Especially in the action cRPG (acRPG) subgenre, random elements began to be more of an addition to the combat systems themselves. Whether a characted managed to hit the opponent was determined by player's skill, reducing the application of randomness to the damage inflicted by the attack. In modern acRPG games, combat increasingly relies on good timing and player reflexes, and less often on randomizing the outcomes of events. Productions whose gameplay focuses on fighting using melee weapons have different directions of attacks and blocks, which makes the player have to demonstrate even more dexterity. Those games can be represented by the *Witcher* series.

The aim of this study is to create an arcade combat system for a cRPG game using machine learning, specifically neural networks. The key contribution is defined by utilising a machine learning algorithm to read the symbol drawn by the player and, based on it, recognize which type of attack it is most similar to. In the sample demonstration game, the recognition result of the neural network determines what spell the dueling wizard character will cast and how effective it will be. This system is characterized by a combat that involves the use of the player's reflexes combined with skill in mapping symbols. A very important feature of the proposed system is that the player is encouraged to choose spells carefully, as the elements of the spells will react with each other, causing stronger or changed effects. The purpose of this approach is to force the player to think about what combination is best to use.

The remainder of this paper is structured as follows. Section **??** states the specific problem that this article addresses, while Section 3 describes the methods used for the proposed solution. Section 4 presents the results obtained during the experiments performed. Finally, Section 5 concludes the article with an additional discussion of its influence.

2. Introduction

Machine learning is one of the branches of artificial intelligence that involves the use of algorithms and data to increase the accuracy of the results obtained by imitating human learning. The algorithms used in machine learning are designed to find correlations and patterns in data sets and make the best decisions based on them. A programmed model requires a large set of data from which it will learn. In supervised machine learning, which is used in this study, each data sample on which the model is trained is provided with information about the correct result, thanks to which the network knows what result it should receive. Once trained, one can use another dataset to test the reliability of the neural network's predictions. This technology is increasingly used in all aspects of science and technology, such as the selection of advertisements displayed on websites, weather forecasts or modern home robots.

Computer Role Playing Games (cRPG), are a genre [1, 2] of computer games in which the player takes on the role of a hero, often created by himself, and experiences an adventure in the fictional world presented. Of the many aspects of gameplay, combat is usually a very important element, which greatly affects how the genre is perceived in the general taxonomy of computer games [3, 4, 5]. Over the years in which the genre has developed and taken various forms, several frequently recurring patterns can be distinguished. The first combat mechanics in cRPG games were very similar to tabletop Role Playing Games (RPG) they were inspired by. The character had its own characteristics, and whether a certain action was achieved was determined by checking whether the value of a given characteristic was high enough, usually with some degree of randomness, so that probability distribution of success was based on character's attributes. An example of such a system is the *Baldur's Gate* series. Later, this system evolved and took various forms. Especially in the action cRPG (acRPG) subgenre, random elements began to be more of an addition to the combat systems themselves. Whether a characted managed to hit the opponent was determined by player's skill, reducing the application of randomness to the damage inflicted by the attack. In modern acRPG games, combat increasingly relies on good timing and player reflexes, and less often on randomizing the outcomes of events. Productions whose gameplay focuses on fighting using melee weapons have different directions of attacks and blocks, which makes the player have to demonstrate even more dexterity. Those games can be represented by the *Witcher* series.

The aim of this study is to create an arcade combat system for a cRPG game using machine learning, specifically neural networks. The key contribution is defined by utilising a machine learning algorithm to read the symbol drawn by the player and, based on it, recognize which type of attack it is most similar to. In the sample demonstration game, the recognition result of the neural network determines what spell the dueling wizard character will cast and how effective it will be. This system is characterized by a combat that involves the use of the player's reflexes combined with skill in mapping symbols. A very important feature of the proposed system is that the player is encouraged to choose spells carefully, as the elements of the spells will react with each other, causing stronger or changed effects. The purpose of this approach is to force the player to think about what combination is best to use.

The remainder of this paper is structured as follows. Section **??** states the specific problem that this article addresses, while Section 3 describes the methods used for the proposed solution. Section 4 presents the results obtained during the experiments performed. Finally, Section 5 concludes the article with an additional discussion of its influence.

3. Materials and methods

While various algorithms were tested in the initial part of the project, the final framework is based on a deep metric learning approach, due to very good results in classification problems, which is a framework that fits symbol recognition very well [6]. The design of the neural network model was done using the PyTorch Metric Learning library [7].

3.1. Layers description

- **Convolution layer**: the first convolution layer, accepts images with a single channel and uses a 3x3 filter and generates an output with 32 channels. These filters are moved over the image with an offset of 1, detecting local features. The result is a set of 32 feature maps that represent the input data processed.

- **Activation function** Rectified Linear Unit (ReLU): After the convolution layer, the activation function Rectified Linear Unit, or ReLU for short, is used, which introduces nonlinearity by eliminating negative values and enhancing positive activations. Many image and spatial data analysis problems are not linearly separable. This means that they cannot be solved effectively using only linear operations. Activation functions such as ReLU introduce non-linearity, which allows the creation of more complex decision boundaries.

- **Second Convolution Layer**: The second convolution layer processes the output of the previous layer using a 3x3 filter size shifted by 1, generating a set of 64 feature maps. Each convolution layer extracts specific features from the input image. The second convolutional layer can analyze the features extracted by the first layer, allowing it to learn more complex and abstract features that are combinations of lower-level features. Each successive convolutional layer operates on increasingly abstract levels of features. This allows the model to automatically learn features of increasing complexity, which is crucial for image tasks.

- **Merging layer**: after the second convolution layer, a max pooling operation with a 2x2 kernel is used, which reduces the size of the processed feature maps and, at the same time, extracts relevant features from local areas. There are several reasons to choose to use a pooling layer. First, the pooling layer reduces the size of the feature map, which helps reduce the number of parameters and calculations in the network, which in turn can improve performance and reduce the risk of overfitting. Second, the merging layer makes the model more robust to small shifts in images. As a result, pattern recognition is not too sensitive to the exact position of a feature in an image.
- **Dropout layer**:Next placed is the dropout layer, which randomly shuts down some neurons zeroing channels to prevent model overfitting. This helps with generalization and increases the model's ability to deal with new data.
- **Linear layer**: Then, the data is flattened into a vector and passed to a linear layer with 9216 inputs and 128 outputs. The weights of this layer are adjusted in the learning process.
- **Batch normalization**: After the linear layer, batch normalization is applied to standardize the activation distribution, which helps stabilize the learning process and accelerates convergence.
- **Output Layer - L2 Normalization**: Finally, before returning the results, the output of the linear layer is normalized based on the L2 norm to create unit-length embeddings, thus obtaining solutions in a unit hypersphere, which will make all the results encapsulated in a cyclic space.

3.2. Selection of the loss function

One of the most relevant hyperparameters of the model is the loss function, which is used in the learning process. This subsection addresses the analysis and selection of loss functions used in deep learning metrics, which allow manipulation of the distance between objects based on their similarity. The goal is to obtain the most efficient representation of features for different objects. The differences and uses of these functions will be presented. The following loss functions will be described in the following subsection:

- Triplet Margin Loss
- Angular Loss
- Circle Loss
- Contrastive Loss

- Large-Margin Softmax Loss
- Lifted Structured Loss
- Margin Loss
- Neighbourhood Components Analysis Loss
- Normalized Softmax Loss
- N Pairs Loss
- Proxy Anchor Loss

3.3. Triplet Margin Loss

Triplet Margin Loss [8] is a loss function used in deep learning, especially in facial identification and verification systems. It is a special case of image pair learning where three images are used instead of two. The three images are: a reference anchor image, a positive image, which is the same object as the reference image but in a different view or context, and a negative image, which is a different object compared to the reference image.

The goal is to minimize the distance between the reference image and the positive image while increasing the distance between the reference image and the negative image. In other words, the model is trained so that images of the same object are close together in feature space, while images of different objects are far apart. The function can be described using Eq. 1.

$$L_{triplet} = [d_{ap} - d_{an} + m]_+ \tag{1}$$

where:

- L_{triplet}: is the triplet loss function.
- d_{ap}: is the squared Euclidean distance between the anchor and a positive sample in the feature space.
- d_{an}: is the squared Euclidean distance between the anchor and a negative sample in the feature space.
- m: is the margin, a positive value that we add to the difference between d_{ap} and d_{an}. The margin is used to force the model to learn features that are at least m apart.
- $[x]_+$: is the Rectified Linear Unit (ReLU) function, which returns x if $x > 0$, otherwise returns 0. means that $[d_{ap} - d_{an} + m]_+$ returns $d_{ap} - d_{an} + m$ if it's greater than zero, otherwise returns 0.

3.4. Angular Loss

Angular Loss is a special type of loss function that is used in deep neural networks. It is one of the techniques used to improve the quality of embedding, which is the process of mapping input data into a space where similar objects are close to each other and different objects are far apart. Angular Loss focuses on minimizing the angle between embedding vectors for pairs of objects that are considered similar and maximizing the angle between objects that are considered different. It works by using the structure of the embedding space to affect the relationships between objects. Angular Loss is used, for example, in neural networks for image classification, facial identification, or facial expression analysis [9]. Angular loss is defined as in Eq. 2.

$$l_{ang}(B) = \frac{1}{N} \sum_{x_a \epsilon B} \{\log[1 + \sum_{\substack{x_a \epsilon B \\ y_n \neq y_a, y_p}} \exp(f_{a,p,n})]\} \tag{2}$$

$$f_{a,p,n} = 4 \tan^2 \alpha (x_a + x_p)^T x_n - 2(1 + \tan^2 \alpha) x_a^T x_p \tag{3}$$

where:

- $l_{\text{ang}}(B)$: is the angular loss function for the set B.
- N: is the number of elements in set B.
- x_a: is the reference point for which we calculate the loss.
- B: is the set of all data points.
- y_n: is the negative label, different from y_a and y_p.
- y_a: is the label of the reference point x_a.
- y_p: is the label of the positive point.
- $f_{a,p,n}$: is the function that calculates the difference between the distances of the reference, positive and negative points.
- α: is the angle between points in the embedding space.

3.5. Circle Loss

Circle Loss is a loss function developed [10] to improve the performance of machine learning models in the context of supervised and unsupervised learning. This is particularly important for classification tasks where traditional loss functions, such as cross-entropy, may not be optimal. Circle Loss is designed to minimize the distance between samples of the same class, while increasing the distance between samples of different classes. It achieves this by introducing a margin between positive and negative samples, which leads to better class separation.

Circle Loss focuses on improving feature discrimination in feature space, which is crucial for tasks such as face recognition and image classification. Unlike other loss functions, Circle Loss not only minimizes the distance between positive samples and their corresponding class centers, but also maximizes the distance between negative samples and class centers. Circle Loss is particularly useful for large datasets where samples may be unevenly distributed in feature space.

Circle Loss is defined based on the relationship between distance and angle in feature space. During the learning process, the circle loss seeks to minimize the angle between the feature vectors of positive samples and their corresponding class centers, while it seeks to increase the angle between the feature vectors of negative samples and the class centers. This approach leads to efficient separation of classes and their classification [10]. The loss is defined by Eq. 4.

$$\mathcal{L}_{circle} = \log[1 + \sum_{j=1}^{L} \exp(\gamma\alpha_n^j(s_n^j - \Delta_n)) \sum_{i=1}^{K} \exp(\gamma\alpha_p^j(s_p^j - \Delta_p))] \tag{4}$$

where:

- $\mathcal{L}_{\text{circle}}$: is the circle loss.
- γ: is the scale factor that controls how fast the loss decreases.
- α_n^j and α_p^j: are the margin parameters for negative and positive pairs.
- s_n^j and s_p^j: are the similarities between the anchor sample and negative and positive samples.
- Δ_n and Δ_p: are pre-defined margins for negative and positive pairs.
- L: is the total number of negative samples.
- K: is the total number of positive samples.

3.6. Contrastive Loss

3.6.1. Function description

Contrastive Loss is a loss function used in machine learning, particularly in problems where pairs of objects are compared, such as face verification. The goal is to minimize the distance between similar pairs and maximize the distance between different pairs in the feature space. For this purpose, the contrast loss is calculated for a pair of objects, which usually includes one similar object and one dissimilar object.

In supervised learning, pairs of objects are labeled, where the label 0 indicates that the objects are similar and the label 1 indicates that the objects are different. Contrastive Loss is calculated on the basis of the difference between the objects in a pair and their labels [11]. Contrastive Loss function is defined as in Eq. 5.

$$L_{contrastive} = [d_p - m_{pos}]_+ + [m_{neg} - d_n]_+ \tag{5}$$

where:

- d_p: is the distance between positive pairs of examples, i.e., the distance between two examples that are of the same type or class.
- m_{pos}: is the positive margin, i.e., the minimum distance we want to maintain between positive pairs of examples.
- d_n: is the distance between negative pairs of examples, i.e., the distance between two examples that are of different types or classes.
- m_{neg}: is the negative margin, i.e., the maximum distance we want to maintain between negative pairs of examples.
- $[x]_+$: Denotes the ramp function, which is equal to x for $x \geq 0$ and 0 for $x < 0$. is used to ensure that the values inside the brackets are non-negative.

3.7. Large-Margin Softmax Loss

3.7.1. Function description

Large-Margin Softmax Loss is a loss function designed to improve the classification ability of neural networks. It is an important component of machine learning that helps optimize the model by minimizing the difference between predicted outputs and actual labels. The standard softmax function is often used in classification problems but can lead to problems when the classes are very close together or when the data are unevenly distributed. Large-Margin Softmax Loss introduces a margin, an additional parameter that forces the model to increase the distance between different classes, leading to better separation of classes and, in turn, improving the generalization ability of the model. It should be noted that Large-Margin Softmax Loss is particularly useful in the context of problems such as face recognition, where classes are very close to each other and require more accurate separation [12]. The Large-Margin Softmax Loss function is expressed as in Eq. 6.

$$L_i = -\log(\frac{e^{\|W_{y_i}\|\|x_i\|\psi(0_{y_i})}}{e^{\|W_{y_i}\|\|x_i\|\psi(0_{y_i})} + \sum_{j \neq y_i} e^{\|W_j\|\|x_i\| \cos(0_j)}}) \tag{6}$$

$$\psi(0) = (-1)^k \cos(m0) - 2k, \quad 0\epsilon[\frac{k\pi}{m}, \frac{(k+1)}{\pi}m] \tag{7}$$

where:

- L_i: is loss for the i-th sample in the dataset.
- W_{y_i}: is row of weight matrix corresponding to the true class of sample x_i.
- x_i: is feature vector of the i-th sample.
- $\psi(0_{y_i})$: is function modifying the angle between W_{y_i} and x_i.
- 0_{y_i} is angle between W_{y_i} and x_i.
- 0_j: is angle between W_j and x_i for the j-th class, which is not the true class of sample x_i.
- m: is hyperparameter controlling the margin.
- k: is the largest integer such that 0_{y_i} belongs to the interval $\left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}\right]$.

3.8. Lifted Structured Loss

Lifted Structured Loss is a loss function used in deep learning to improve feature embedding by defining a new structure prediction target. Traditionally, machine learning algorithms aim to minimize the distance between samples of the same class and maximize the distance between samples of different classes. However, such a strategy can lead to suboptimal solutions in cases where the differences between classes are subtle [13].

Structure prediction is related to the matrix of pairwise distances within batches when training a neural network. The matrix of pairwise distances within batches is a matrix that contains the distances between all pairs of samples in a given batch of data processed by a neural network. This new objective is based on the raised dense matrix of pairwise distances, which allows both positive and negative sample pairs to be considered during training, helping to find the optimal embedding of features that are suitable for specific tasks, such as classification, regression or search. The raised dense pairwise distance matrix is a special type of this matrix that takes into account the extra dimension of the data, adding additional information about the structure of the data. In the context of the loss function algorithm, lifting refers to the application of a loss function that promotes similar or pairs of samples that belong to the same class and degrades pairs of samples that are different or belong to different classes.

In practice, this algorithm is based on learning metrics to extract features that are more discriminating and suitable for specific tasks. In this way, the raised structural loss allows to improve the quality of feature embedding and facilitates image processing for different tasks [13]. The raised structural loss function is defined as in Eq. 8.

$$\tilde{J}_{i,j} = \log\Big(\sum_{(i,k)\epsilon N} \exp\{\alpha - D_{i,k}\} + \sum_{(j,l)\epsilon N} \exp\{\alpha - D_{j,l}\}\Big) + D_{i,j} \tag{8}$$

$$\tilde{J} = \frac{1}{2|P|} \max(0, \tilde{J}_{i,j})^2 \tag{9}$$

where:

- $\tilde{J}_{i,j}$: is the value of the loss function for the pair of indices (i, j), representing elements in the data set being compared. It represents a measure of the distance between elements i and j, taking into account comparisons with other elements (k and l) from the neighborhood.
- N: is the neighborhood set, consisting of indices k and l. Relative to element i, these are the indices of other elements with which i is compared.
- α: is a parameter that can influence the weight of terms in the exponentials.
- $D_{i,k}$ and $D_{j,l}$: These are distance measures between elements i and k, and j and l respectively.
- $\tilde{J}$: is the normalized value of the loss function, which is squared and divided by twice the number of elements in set P.

3.9. Margin Loss

Margin Loss is a loss function used in machine learning, particularly in models based on neural networks. This function is used to minimize the distance between positive examples, i.e., samples that have the same labels, and their negative counterparts, i.e., samples that have different labels in the embedding space. Margin Loss is particularly useful in applications where it is important for embeddings of objects of the same class to be close to each other, while objects of different classes are separated by a certain margin. For example, in face recognition, we want the embeddings of different images of the same person to be close to each other, while the embeddings of different people are separated. Margin Loss forces the model to learn embeddings that meet these conditions by imposing a penalty on pairs of embeddings that are too close together or too far apart [14]. This loss function is defined by Eq. 10

$$\text{minimize} \sum_{i,j} l^{margin}(i, j) + \nu(\beta^{(0)} + \beta^{(class)}_{c(i)} + \beta^{(img)}_{i}) \tag{10}$$

$$l^{margin}(i, j) := (\alpha + y_{ij}(D_{ij} - \beta))_+ \tag{11}$$

where:

- $\sum_{i,j}$: Denotes summation over all pairs (i, j).
- $l^{\text{margin}}(i, j)$: is the margin loss function for the pair (i, j).
- α: is a constant controlling the margin, i.e., the difference between positive and negative pairs.
- y_{ij}: is the label taking the value +1 for positive pairs and -1 for negative pairs.
- D_{ij}: is the distance between the i-th and j-th element.
- β: is a parameter learned during the optimization process.
- $(\cdot)_+$: Represents the ReLU function, which is $max(0, \cdot)$.
- ν: is the regularization parameter controlling the influence of the regularization term on the objective function.
- $\beta^{(0)}$: is a constant regularization term.
- $\beta^{(\text{class})}_{c(i)}$: is the regularization term associated with the class of element i.
- $\beta^{(\text{img})}_{i}$: is the regularization term associated with image i.

3.10. Neighborhood Components Analysis Loss

Neighborhood Components Analysis is a learning method for the non-linear mapping of an input data set to a space in which the distances between elements reflect their similarity to each other. The key element of this approach is a loss function that is optimized during the learning process. The goal of this process is to find a mapping that increases the probability that the nearest neighbor of a given point in the mapped space will be of the same class as the reference point. The loss function is defined as the difference between the true class distribution and the predicted distribution, which is calculated based on the distance between elements in the mapped space. It is important to note that this function does not assume any particular class structure or decision boundary structure, relying solely on the strong regularization introduced by the restriction to linear transformation of inputs [15]. The definition of the Neighborhood Components Analysis Loss function is expressed as in Eq. 12.

$$g(A) = \sum_i \log(\sum_{j \epsilon C_i} p_{ij}) = \sum_i \log(p_i) \tag{12}$$

$$p_i = \sum_{j \epsilon C_i} p_{ij} \tag{13}$$

$$p_{ij} = \frac{\exp(-||Ax_i - Ax_j|^2)}{\sum_{k \neq i} \exp(-||Ax_i - Ax_k|^2)} \tag{14}$$

where:

- $g(A)$: is the loss function, which is the sum of logarithms of probabilities p_i for all data points i.
- p_i: is the probability that data point i is chosen as a reference point for other points in the same class C_i. It is the sum of probabilities p_{ij} for all points j in the same class as i.
- p_{ij}: is the probability that data point j is chosen as the nearest neighbor to point i. It is determined based on the distance between Ax_i and Ax_j in the space transformed by matrix A. The smaller the distance, the higher the probability.
- A: is the transformation matrix, which is learned to minimize the cost function $g(A)$.
- Ax_i: is data point i transformed by matrix A.
- Ax_j: is data point j transformed by matrix A.
- Ax_k: is data point k transformed by matrix A.
- C_i: is the set of data points belonging to the same class as point i.

3.11. Normalized Softmax Loss

Normalized Softmax Loss is a loss function that plays a key role in the learning process of deep neural network models. This function is designed to address problems associated with the traditional Softmax Loss function, such as optimization problems in high-dimensional space and the problem of normalizing feature vectors. In the traditional Softmax Loss function, the output of the network is transformed to probabilities for different classes, and then the probability of belonging to the true class is calculated. The loss function is calculated on the basis of the difference between the true and predicted probabilities. Normalized Softmax Loss differs in that it normalizes the feature vectors before calculating the probabilities. This normalization of feature vectors helps improve the generalization ability of the model, as it reduces the impact of different scales and facilitates optimization of the[16]. The definition of the Normalized Softmax Loss function is as in Eq. 15.

$$L_{norm} = -\log(\frac{\exp(x^T p_y/\sigma)}{\sum_{z \epsilon Z} \exp(x^T p_z/\sigma)} \tag{15}$$

where:

- L_{norm}: is the normalized soft max loss.
- x: is the feature vector of the object currently under consideration.
- p_y: is the weight vector for the true class of the object.
- p_z: is the weight vector for each possible class from the set of classes Z.
- Z: is the set of all possible classes.
- σ: is a scaling parameter used for the normalization of results.
- exp: denotes the exponential function.

3.12. N-Pairs Loss

N-Pairs Loss is a loss function used in deep learning metrics that seeks to improve the quality and convergence of the learning process used in triplet loss. In its basic form, triplet loss aims to minimize the distance between examples belonging to the same classes, called positive samples, and maximize the distance between examples belonging to different classes, called negative samples. However, this can lead to problems related to the difficulty in selecting appropriate negative examples, which in turn affects the learning process and the quality of representation [17].

N-Pairs Loss was introduced to deal with these problems. The main idea is to force the simultaneous separation of positive and negative examples by introducing more negative examples for each positive example. In practice, N negative examples are selected for each positive example, leading to a more balanced learning process.

The task of N-Pairs Loss is to minimize the value of the loss function by maximizing the distance between positive and negative examples. By introducing more negative examples, the model is forced to create more separable and focused feature representations, which in turn translates into better ability to distinguish between different classes in the classification or identification process [17].

As a result, N-Pairs Loss contributes to improved quality of feature representation and better model performance in tasks such as image classification, face recognition, and pattern search. Its application enables a more efficient use of training data, which has a significant impact on the final performance and effectiveness of the model in real-world applications. The N-Pairs Loss function is defined in 16.

$$\mathcal{L}_{N-pair}(\{(x_i, x_i^+)\}_{i=1}^{N}; f) = \frac{1}{N}\sum_{i=1}^{N}\log(1 + \sum_{j\neq 1}\exp(f_i^\top f_j^+ - f_i^\top f_i^+)) \tag{16}$$

where:

- $\mathcal{L}_{N\text{-pair}}$: is the N-pair loss function.
- x_i: is the i-th image in the dataset.
- x_i^+: is the positive example for image x_i.
- N: is the number of image pairs in the dataset.
- f: is the function that maps input images to feature vectors.
- f_i: is the feature vector of image x_i, obtained by applying function f to x_i.
- f_i^+: is the feature vector of image x_i^+, obtained by applying function f to x_i^+.
- $f_i^\top$: is the transpose of the feature vector f_i.

In the above equation, for each pair of images x_i and x_i^+, the difference between the scalar product of $f_i^t op f_j^+$ and $f_i^t op f_i^+$ for all $j \neq i$ is calculated, and then the exponent of this difference is calculated. One sums these exponents for all $j \neq i$, adds 1, and then calculates the logarithm of this result. Finally, the average of these logarithms for all i is calculated.

3.13. Proxy Anchor Loss

Proxy Anchor Loss is a loss function developed to improve the performance of deep representation learning models on image datasets. This function is designed to manage the difficulties associated with learning deep neural networks, such as large-scale optimization and processing a large number of classes of [18].

The model uses image embeddings, representing input images in the embedding space, and generates proxies for each class, which are reference embeddings for all images in the class. Proxy Anchor Loss uses these proxies to minimize the distance between images in the same class and maximize the distance between images in different classes.

During learning, the model minimizes the difference between an image embedding and the closest proxy of the same class and maximizes the difference between an image embedding and the closest proxy of a different class. As a result, the model will learn representations that effectively separate images of different classes and closely group images of the same class [18]. The Proxy Anchor Loss function is mathematically defined as in Eq. 17.

$$l(x) = \frac{1}{|P^+|} \sum_{p \epsilon P^+} \log(1 + \sum_{x \epsilon X_p^+} e^{-\alpha(s(x,p)-\delta)}) + \frac{1}{|P|} \sum_{p \epsilon P} \log(1 + \sum_{x \epsilon X_p^-} e^{\alpha(s(x,p)+\delta)}) \quad (17)$$

where:

- $l(x)$: is the anchor loss function.
- P^+: is the set of all positive pairs, i.e., pairs of images that are considered similar.
- P: is the set of all image pairs, both positive and negative.
- X_p^+: is the set of all positive images for a given reference image p.
- X_p^-: is the set of all negative images for a given reference image p.
- α: is a hyperparameter controlling the weight of the exponent in the equation.
- $s(x, p)$: is the similarity function between image x and the reference image p.
- δ: is the margin controlling how large differences between a positive pair and a negative pair are considered acceptable.

The values of $s(x, p)$, α, and δ are set during the learning process. The model attempts to adjust these parameters to minimize the anchor loss function $l(x)$, leading to improved model performance.

3.14. Classification

In order to collect research data, several steps were carried out in addition to training the model. First, all the outputs of the neural network were collected and stored in a list. These steps were needed to prepare the KNN classifier based on the training dataset, which in this case were the hypersphere positions returned by the learning neural network model [19, 20].

The above steps had to be performed to get the KNN to find nearest neighbors in the newly created metric during evaluation. During model testing, the outputs of the network and their corresponding data labels are recorded. Finally, the accuracy of the model is calculated by comparing the KNN's predicted labels from the outputs saved during evaluation with their actual labels.

In addition, while training the model in each iteration, a loss value is recorded, based on which a graph is created showing the change in the average loss value over time.

3.15. Chosen symbols set

The Modified National Institute of Standards and Technology (MNIST) database was used as a training and test collection [21]. This is a popular data set used in the field of machine learning to test and evaluate image recognition algorithms.

The original MNIST database contains a set of 60,000 training images and 10,000 test images that were collected from handwritten samples by staff at the US National Institute of Standards and Technology. These images were then processed and scaled to be 28x28 pixels in size and grayscale.

Each image in the MNIST database represents a single digit from 0 to 9, and a label assigned to each image indicates which digit is represented. The database has been used as a test set and benchmark in many scientific studies and as a teaching tool for learning and demonstrating classification algorithms, neural networks, and other techniques related to image analysis.

The MNIST database was one of the first and most recognized datasets in the field of machine learning. It helped lay the groundwork for new algorithms and models, as well as popularize the field of image recognition and deep learning. Although MNIST is a relatively simple dataset, it is still used as a starting point for learning and experimenting with new machine learning techniques.

This database was chosen because of its popularity, which allows one to study how accurate a neural network based on deep learning metrics is compared to other types of machine learning and algorithms for image classification. A major advantage of selecting a popular database is the easy reproducibility of the results.

4. Results

In order to obtain the best results for the accuracy of the prediction of the model, the model was repeatedly trained using different loss functions. The results obtained are shown in Figure 1. As can be seen, the best accuracy of 99.43% was obtained using the margin loss function.

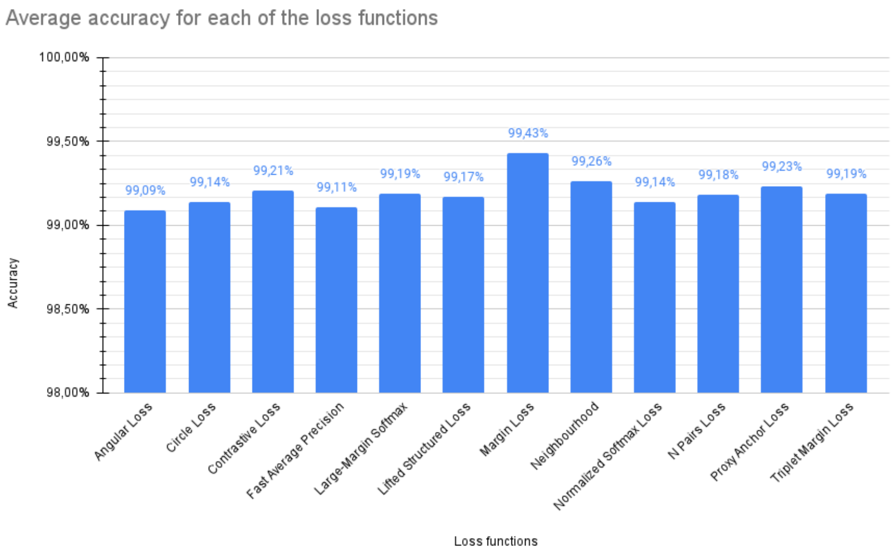


Figure 1. The diagram shows the average accuracy for each of the loss functions tested.

5. Conclusions

This article proposes a real-time, machine-learning combat system based on symbol-drawing. In such context, different loss functions were compared. Further work on the development of the system can be taken in several directions. One is to further develop it with further spells to increase the player's capabilities when skirmishing with the enemy. Another possibility is to create additional combinations between effects to allow the creation of the strongest possible combinations in combat. All available mechanics could be accompanied by a much better graphical representation using models and advanced shading systems. A final future development could be to translate an external neural network prediction application into an internal script, which would certainly optimize the game.

An important part of the project was to design a neural network to classify hand-drawn symbols. The deep learning metrics algorithms used for this obtained very good results. It turned out to be crucial to select an appropriate loss function to maximize the accuracy of the model.

References

[1] Wolf, M. J. 6 genre and the video game. In *The medium of the video game*, pages 113–134. University of Texas Press, 2002.

[2] Apperley, T. H. Genre and game studies: Toward a critical approach to video game genres. *Simulation & gaming*, 37(1):6–23, 2006.

[3] Aarseth, E., Smedstad, S. M., and Sunnanå, L. A multidimensional typology of games. In *DiGRA Conference*. 2003.

[4] Elverdam, C. and Aarseth, E. Game classification and game design: Construction through critical analysis. *Games and culture*, 2(1):3–22, 2007.

[5] Lee, J. H., Karlova, N., Clarke, R. I., Thornton, K., and Perti, A. Facet analysis of video game genres. *IConference 2014 Proceedings*, 2014.

[6] Kaya, M. and Bilge, H. Ş. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

[7] Musgrave, K., Belongie, S. J., and Lim, S.-N. Pytorch metric learning. *ArXiv*, abs/2008.09164, 2020.

[8] Ha, M. L. and Blanz, V. Deep ranking with adaptive margin triplet loss. *arXiv preprint arXiv:2107.06187*, 2021.

[9] Wang, J., Zhou, F., Wen, S., Liu, X., and Lin, Y. Deep metric learning with angular loss. In *Proceedings of the IEEE international conference on computer vision*, pages 2593–2601. 2017.

[10] Sun, Y., Cheng, C., Zhang, Y., Zhang, C., Zheng, L., Wang, Z., and Wei, Y. Circle loss: A unified perspective of pair similarity optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6398–6407. 2020.

[11] Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[12] Liu, W., Wen, Y., Yu, Z., and Yang, M. Large-margin softmax loss for convolutional neural networks. *arXiv preprint arXiv:1612.02295*, 2016.

[13] Oh Song, H., Xiang, Y., Jegelka, S., and Savarese, S. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4004–4012. 2016.

[14] Wu, C.-Y., Manmatha, R., Smola, A. J., and Krahenbuhl, P. Sampling matters in deep embedding learning. In *Proceedings of the IEEE international conference on computer vision*, pages 2840–2848. 2017.

[15] Goldberger, J., Hinton, G. E., Roweis, S., and Salakhutdinov, R. R. Neighbourhood components analysis. *Advances in neural information processing systems*, 17, 2004.

[16] Zhai, A. and Wu, H.-Y. Classification is a strong baseline for deep metric learning. *arXiv preprint arXiv:1811.12649*, 2018.

[17] Sohn, K. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016.

[18] Kim, S., Kim, D., Cho, M., and Kwak, S. Proxy anchor loss for deep metric learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3238–3247. 2020.

[19] Fix, E. and Hodges, J. L. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, 57(3):238–247, 1989.

[20] Cover, T. and Hart, P. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[21] Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Moving Object Detection on VGG Networks in Games with Moving Camera

Michał Domarecki, Dominik Szajerman[0000−0002−4316−5310]

Lodz University of Technology
Institute of Information Technology
Politechniki 8, 93-590 Łódź, Poland
dominik.szajerman@p.lodz.pl

Abstract. *The main goal of this work was to extend the existing reference method to make it more useful in the problem of creating game testing agents. Such an agent should obtain exactly the same data as the player while playing. This is of course a video stream from the game. A quite difficult problem to solve, which occurs in most games, is distinguishing the background from moving game objects. This is even more difficult with a moving camera.*

The DASFE method was chosen as it is used in analogous problems for video streams from the real world. A modification was proposed to improve its results in artificial image analysis conditions and compared with the reference method. In addition, a test dataset was created based on a selected computer game.

Keywords: *computer games, deep networks, game testing*

1. Introduction

Detecting moving objects is an issue that has been widely studied. It is an important and one of the fundamental issues in the area of image analysis. There are many challenges associated with detecting moving objects. They are caused by many different factors. They can appear simultaneously in one image. The most common problems are [1, 2]: lighting change, dynamic background, moving camera, camouflage effect, occlusion, appearance change, motion blur, and image noise.

The purpose of this paper was to implement one of the methods for detecting moving objects in a computer game and modify it to adapt to new conditions. An additional goal was to create a dataset using a computer game.

2. Related Works

Many methods have been developed to detect moving objects. They can be divided into 4 groups according to the approach to the problem being solved [3]: methods based on frame difference, based on background modeling and subtraction, using optical flow, methods based on deep learning.

Frame difference is one of the basic, but efficient techniques. It involves subtracting two consecutive frames to obtain differences in pixel values that define the movement of the object. The factors that most affect its accuracy are: noise in the video sequence, change in lighting, dynamic background, and shadows casting by moving objects [4]. Many modifications have been proposed to eliminate issues caused by these factors[3].

The principle behind background modeling and subtraction methods is to obtain the pixels and subtract their values from the frame. The resulting differences are included in the foreground. For a static background it is enough to get one image representing the background. This is an extremely simple and efficient method. However, in the case of a dynamic background, it is required to obtain and constantly update the background image - i.e. its modeling. These simple models cannot cope with camera shaking, lighting changes, and dynamic backgrounds. Gaussian functions or kernel density estimator[2] are used for better detection in the case of dynamic backgrounds.

Optical flow is a vector field that represents the pattern of movement of objects, surfaces, and edges. It approximates the velocity for each pixel and determines where that pixel will be in the next frame. The key assumption is that the brightness of corresponding pixels does not change. Thus groups of pixels moving at the same speed in the same direction can be detected. They are considered moving objects. Methods for determining optical flow are demanding and computationally expensive [3].

2.1. Deep learning

Methods based on deep learning often use the principles of modeling and background subtraction methods, but they use convolutional neural networks. However, convolutional networks were designed with image classification in mind. For foreground detection, a classification of each pixel in the form is required as a mask, not single answer. One solution is to divide the network into two parts: encoder and decoder. The encoder creates a map of the features of the image and creates a pixel classification as an output, often in the form of the image reduced several times. The decoder is a network responsible for convolutionally reproducing the encoder result to enlarge it and obtain the classification of each pixel [5]. Methods based on deep learning achieve good results for most challenges [3]. However, they require the creation of a training set first. The most famous architectures are [6]: AlexNet, VGG, GoogleNet, and ResNet.

AlexNet was the first to achieve innovative results in image recognition. It used filters of size 11×11, 5×5, and 3×3. The number of filters used in the layers were 96, 256, and 384. It used 60 million parameters. It contributed to renewed interest in convolutional networks and significantly accelerated their further development [7].

With the development of convolutional networks, a new architecture was proposed - Visual Geometry Group (VGG), distinguished by its simple structure. Only smaller 3×3 filters were used in the convolutional layers, showing that a smaller filter with fewer parameters can achieve the same accuracy as larger filters [7]. In this group, there are several variants marked VGG-N, where N denotes the number of neural layers used. The most The VGG-16 network is popular in this group.

The goal of the GoogleNet architecture was to achieve a high level of accuracy with reduced computational cost. It proposed an innovative concept of the inception block in the context of convolutional networks. Thanks to innovative blocks and sparse connections in the network, the number of parameters was reduced to 5 million [7].

The ResNet architecture uses additional cross-layer connections that are parameterless and data-independent. In this group, there are many types of networks denoted ResNetN, where N is the number of layers. Despite using more layers, the ResNet network has much lower computational complexity compared to the VGG [7] network.

2.2. Application in computer games

With the development of image analysis, a new approach to teaching AI to play a game began to be used. These were computer vision methods to teach AI how to play a game based solely on the displayed image, i.e. the information the player receives. For this approach Kempka [8] et al. decided. Using the Doom game, they created the ViZDoom platform for testing agents. They created a convolutional neural network to control the agent. Venkatesh used a similar approach to create an agent playing Super Smash Brothers Melee [9]. He created the SmashNet network, whose task was to recognize 4 characters in the game and determine their position. Its architecture was based on the VGG-16 network architecture.

3. Method

The reference method chosen in this work is DASFE (Deep Atrous Spatial Feature Extractor) neural network which is based on convolutional neural network VGG-16 [10]. On real world datasets it achieved very high results in moving object detection. It was able to detect positions and approximate shape of the objects. The input takes the image in width $\times$ height $\times$ 3 resolution, where 3 means the number of image channels (RGB in this case). The size of the filters used is 3×3, and their number is increased twice (64, 128, 256, 512) in subsequent layers according to the VGG-16 network architecture. To remove negative values, a rectified linear activation function (ReLU) is added to each layer. The first two blocks of the DASFE network are of the same form as the VGG-16 network. They use standard convolutional layers. Their goal is to detect basic features of objects such as edges, color, corners. Each of these blocks reduces the size of the input image by half via max-pooling layers. To train the network faster, the weights of the first two blocks are got from the pre-trained VGG-16 network.

The blocks belonging to Atrous Convolution Layers use convolutional layers with dilation to extract mid- and high-level features. This increases the field of view of the network without increasing the number of parameters to train [11]. Convolution filters can be obtained by enlarging it and inserting 0 in the appropriate cells. The dilatation sizes used in the layers are $3, 4, 5$ to increase the filter field of view from 3×3 to 7×7, 9×9, 11×11. At the end of the first ACL block, the pooling layer reduces the image size by half, resulting in an 8-fold reduction of the input image at this stage. The low resolution of high-level features often reduces the accuracy of pixel-level predictions, so in next blocks the pooling layers are removed. After each convolutional layer in them, a spatial dropout method was added to generalize the network and reduce the number of parameters by resetting the entire feature map to zero.

The feature map at the output of the DASFE network has been reduced 8 times compared to the input image. Convolutional reconstruction of the feature map is required to obtain pixel-level predictions. An upsampling network (UPN) is used for this purpose. It consists of 3 blocks. Each block contains a convolutional layer with a 1×1 filter and a convolutional layer with a 3×3 filter as the first two layers. The feature map is then enlarged twice using bilinear interpolation. Then the enlarged feature map is refined by another convolutional layer with 3×3 filter. Each convolutional layer uses ReLU as its activation function. The last layer of the UPN network uses a sigmoid function to calculate the probability that a given pixel belongs to the foreground or background. The output is an image in resolution width×height×1, which is a mask showing the detected moving objects.

3.1. VGG modification

The DASFE network architecture is based on the VGG-16 network. It is popular in selecting network architecture for feature extraction in image analysis. In the VGG group, there is also the VGG-19 architecture. It can achieve better results at the expense of a much larger number of operations and a larger number of parameters (approximately 5 million more parameters). The difference in the architecture of these two networks is the addition of one convolutional layer with the number of filters corresponding to a given block to the three middle blocks, for

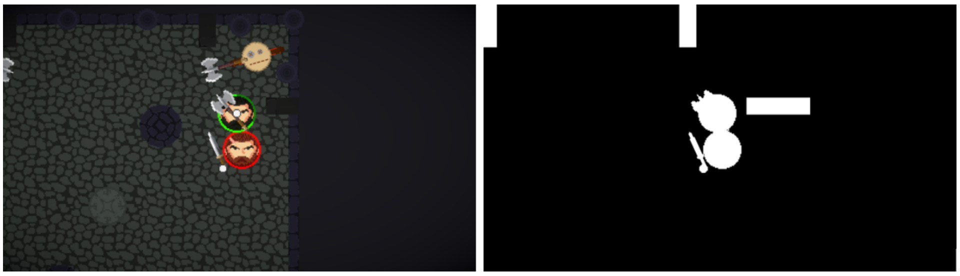


Figure 1. Sample frames from the dataset with correct results marked

a total of three convolutional layers with a filter size of 3×3.

We decided to add three convolutional layers in ACL blocks to reproduce the architecture of the VGG-19 network. Additionally, we changed the size of the dilations in these blocks. Four sets of dilations were used: initial 3, 4, 5, 6, then 2, 3, 4, 5, then exponentially increasing 1, 2, 4, 8 and finally 1, 2, 3, 4, thus affecting the field of view.

4. Experiment

The experiment compares the results achieved in the task by the reference and modified DASFE variants using our own built dataset.

4.1. Dataset

The dataset was created based on the game Blade Spinners [12]. It consists of frames containing gameplay and marked correct results as a mask (Fig. 1). All objects that have changed their position or rotation since the last check were marked as a moving object. If the object has stopped moving, it is not selected in the result frame. All UI elements and the particle system have been disabled when rendering the dataset to avoid possible errors. The basic moving objects in this dataset are players, swords, barrels, puppets and walls (Fig. 2). The player's basic movement is translation. The movement of the sword depends on the player's movement and the player can also rotate or put it out. Barrels are items that can be moved by the player or the wall. The walls move independently and cannot be influenced.

4.2. Parameters

Depending on the subset, 50 to 500 manually selected frames from the dataset were used to train the network. 10% of the selected frames were used for network validation. The network was trained for 50 epochs with a training step of 0.0001, which is reduced if there is no progress in learning. The dropout rate used is 0.5.

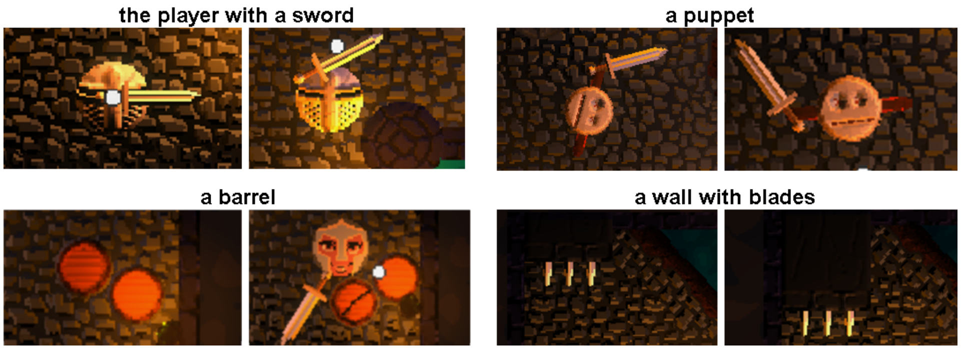


Figure 2. Sample frames with moving elements and their possible movement

5. Results

The dataset included frames from several game maps. Only frames from the first map were used for training. Results from the experiment are shown in Table 1. All maps were created in the same style, but have significant changes to game elements (moving walls, puppets, moving barrels). Additionally, the player's appearance was changed.

Table 1. Comparison of results for the dataset

	DASFE				
	VGG-16	VGG-19, dilation values			
measure		3, 4, 5, 6	2, 3, 4, 5	1, 2, 4, 8	1, 2, 3, 4
Precision	0.4253	0.369	0.393	0.3627	0.7115
Recall	0.1947	0.5656	0.6829	0.616	0.8053
Specificity	0.9913	0.9682	0.9653	0.9644	0.9893
FP rate	0.0087	0.0318	0.0347	0.0356	0.0107
FN rate	0.81	0.4344	0.3171	0.384	0.1947
PWC	3.4042	4.4652	4.3711	4.6726	1.6606
f-measure	0.2671	0.4467	0.4989	0.4565	0.7555

The addition of three convolutional layers increased the number of parameters from less than 15 million to more than 20 million. This slightly lengthened the learning time of one epoch. The network was trained on a set created from one map and tested on several maps. The modified network with smaller dilatation values (1, 2, 3, 4) achieved the best results for most of the calculated measures. They are better than for the reference method.

The visual results for the reference method are shown in Figure 3. The network was unable to detect smooth walls as moving objects. However, it was able to

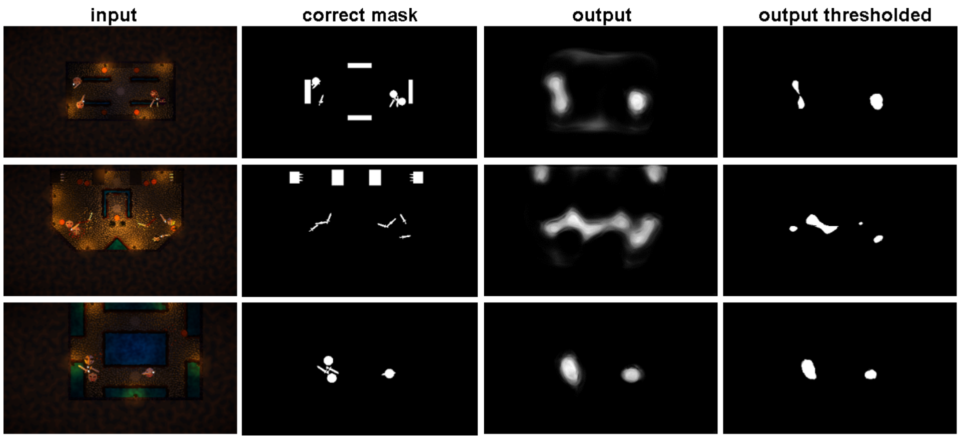


Figure 3. Sample training results for VGG-16

slightly point out the moving swords. The detected shape of objects was very inaccurate. A large proportion of objects were combined into one object in the resulting image when these objects were close to each other.

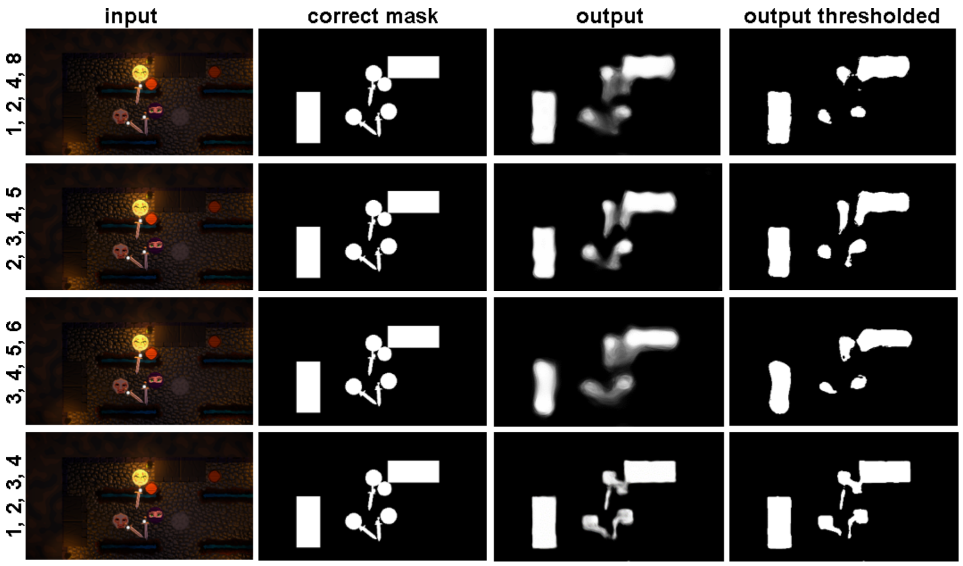


Figure 4. Sample results for the modified - VGG-19 network. The numbers on the left indicate the dilation values used

Sample results of the VGG-19 network are shown in Fig. 4. Modified networks with smaller dilation values created a less blurry and more angular output image, which made them better at detecting swords and moving walls. This also resulted in less objects merging together.

Example frames from training the modified network are shown in Figure 5. The modified network often marked players, but when players were not moving (especially at the beginning of the game) these markings had little pixel values. The puppets were pointed at most of the time, but only those that moved at some point. If the puppet did not move throughout the game, it was not indicated. The swords were pointed out with great uncertainty. All moving walls were indicated correctly with high confidence, but sometimes they were indicated with a hole in the middle of the wall area. Blades on the wall were not indicated or were combined with the wall into one object in the resulting image. Most of the barrels were not indicated or were combined with another element into one object.

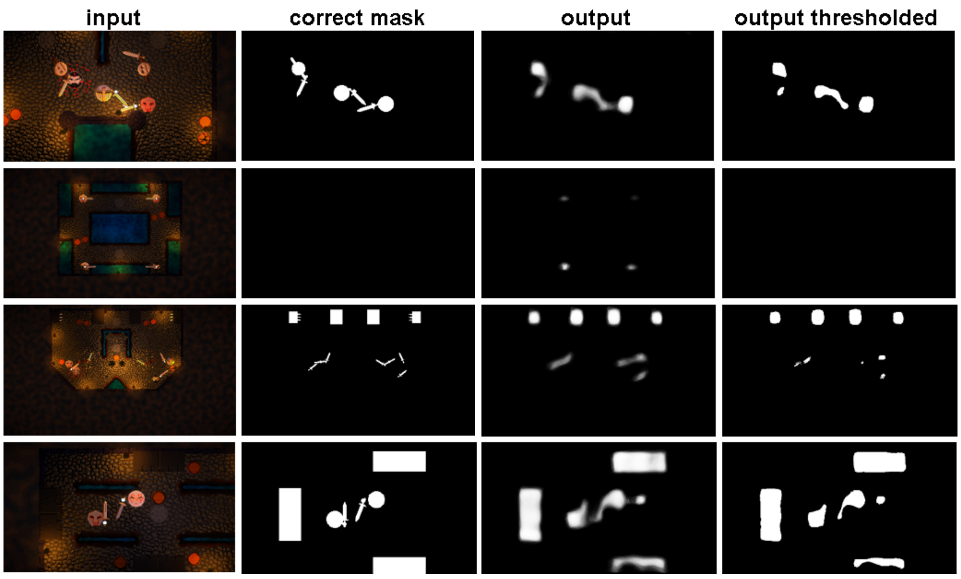


Figure 5. Example results for the modified - VGG-19 network with the best dilation values selected (1, 2, 3, 4) for various scenes

6. Conclusions

From the obtained results, it can be concluded that the DASFE network learns the appearance and colors of moving objects. Therefore, it is effective for objects that move all the time. For objects with intermittent motion, the network has difficulty detecting when the object stopped moving and when its motion began. However, in the case of objects whose movement is only possible during contact with another object, the network is more often able to determine the transient states of it. Our modification allowed for better detection of object movement at the expense of the number of parameters and therefore the object detection time. With more parameters, it is able to learn more maps.

It can be concluded that a larger number of parameters allowed learning not only the shape and color of the moving object, but also partially its position by recognizing the moving object as the background in a given place. This makes it better at detecting when an object has started moving.

Future works could consider better detection of intermittent object motion by changing the network input to accept two consecutive frames from a video sequence. Taking as input the result from the previous frame can also improve results. However, such a modification may result in a significant increase in the number of parameters and calculation time, making such a solution unsuitable for real-time applications.

References

[1] Yazdi, M. and Bouwmans, T. New trends on moving object detection in video images captured by a moving camera: A survey. *Computer Science Review*, 28:157–177, 2018. ISSN 1574-0137. doi:10.1016/j.cosrev.2018.03.001.

[2] Chapel, M.-N. and Bouwmans, T. Moving objects detection with a moving camera: A comprehensive review. *Computer Science Review*, 38:100310, 2020. ISSN 1574-0137. doi:10.1016/j.cosrev.2020.100310.

[3] Roy, S. D. and Bhowmik, M. K. A comprehensive survey on computer vision based approaches for moving object detection. In *2020 IEEE Region 10 Symposium (TENSYMP)*. IEEE, 2020. doi:10.1109/tensymp50017.2020.9230869.

[4] Husein, A. M., Calvin, Halim, D., Leo, R., and William. Motion detect application with frame difference method on a surveillance camera. *Journal of Physics: Conference Series*, 1230(1):012017, 2019. ISSN 1742-6596. doi:10.1088/1742-6596/1230/1/012017.

[5] Wang, Y., Yu, Z., and Zhu, L. Foreground detection with deeply learned multi-scale spatial-temporal features. *Sensors*, 18(12):4269, 2018. ISSN 1424-8220. doi:10.3390/s18124269.

[6] Canziani, A., Paszke, A., and Culurciello, E. An analysis of deep neural network models for practical applications, 2016. doi:10.48550/ARXIV.1605.07678.

[7] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., and Farhan,

L. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1), 2021. ISSN 2196-1115. doi:10.1186/s40537-021-00444-8.

[8] Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaskowski, W. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016. doi:10.1109/cig.2016.7860433.

[9] Venkatesh, A. *Object Tracking in Games using Convolutional Neural Networks*. Ph.D. thesis, Robert E. Kennedy Library, Cal Poly. doi:10.15368/theses.2018.56.

[10] Shahbaz, A. and Jo, K.-H. Moving object detection based on deep atrous spatial features for moving camera. In *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)*. IEEE, 2020. doi:10.1109/isie45063.2020.9152237.

[11] Seo, J. D. Understanding 2d dilated convolution operation with examples in numpy and tensorflow with interactive code, 2018. URL `https://towardsdatascience.com/understanding-2d-dilated-convolution-operation-with-examples-in-numpy-and-tensorflow-with-d376b3972b25`.

[12] Blade spinners, 2022. URL `https://dawid-piech.itch.io/blade-spinners`.

Extending OptaPlanner Solver with Implementation of Ruin&Recreate Principle

Michał Dudkiewicz, Michał Karbowańczyk[0000−0003−3875−1101]

Lodz University of Technology
Institute of Information Technology
al. Politechniki 8, 93-590 Łódź, Poland
michal.dudkiewicz96@gmail.com

Abstract. *Optimization problems today are commonly addressed using specialized tools like the open-source software OptaPlanner. It rapidly delivers satisfactory solutions for constrained optimization challenges, benefiting various global industries and institutions by cutting costs, enhancing service quality, and reducing carbon emissions. These problems are often NP-hard or NP-complete, making greedy or brute-force algorithms unsuitable. Therefore, OptaPlanner utilizes metaheuristics within local search. They may, however, occasionally get stuck in local maxima. To counter this, the Ruin and Recreate (R&R) rule is implemented, progressively ruining and recreating parts of the solution in each solving iteration. This approach yields favorable results across diverse problem types and search space sizes. With proper ruin configuration, it often finds superior solutions within the same timeframe, avoiding local maxima pitfalls. While not fully integrated with the OptaPlanner engine due to implementation details, the RedHat team will continue researching and adjusting the R&R rule's application and partial reimplementation in the engine's development.*

Keywords: *optimization, local optima, ruin&recreate*

1. Introduction

Organizations grapple with planning challenges, seeking efficient product or service delivery within constraints like employees, assets, time, and money. RedHat's open-source software, OptaPlanner[1], optimizes planning, enhancing operations while conserving resources via constraint programming. OptaPlanner is a versatile planning engine, addressing various optimization problems like employee shift scheduling, production planning, and more. Object-oriented modeling handles resource allocation and constraints, influencing the objective function.

Optimization problems are often NP-complete or NP-hard [2][3], with vast search spaces preventing exhaustive exploration. OptaPlanner utilizes local search algorithms [4], like “tabu search” [5][6], “simulated annealing” [7], and “late acceptance ”[8], to escape local maxima. An unutilized yet promising principle is the “Ruin and Recreate” (R&R) approach, initially introduced by IBM[9].

This work extends OptaPlanner with the R&R rule, assessing its effectiveness across diverse problem types, modeling approaches, and search space characteristics. It aims to compare R&R-aided solutions with non-R&R solutions, evaluating its impact on escaping local maxima and overall optimization success. The analysis also explores the evolution of local search over time, identifying factors influencing ruining effectiveness and proposing universal parameters, such as ruin size and its variant, drawing inspiration from IBM’s extensive research.

2. Ruin and Recreate principle in OptaPlanner engine

2.1. OptaPlanner search phases

2.1.1. Construction Heuristics phase

OptaPlanner engine utilizes local search algorithms for solving optimization problems. The solving process is divided into phases, with the first one being the phase of construction heuristics. Its primary goal is the initialization of the initial solution.

A typical approach in this phase is “first-fit”[1], which involves iteratively assigning the best possible values to decision variables. In practice, these decision variables are often modeled as planning objects following the object-oriented programming paradigm. Upon completing this phase, the solution is fully initialized and ready to be used as a starting point for the local search phase.

2.1.2. Local Search phase

The Local Search[4] phase is a crucial element of the optimization process using OptaPlanner. It begins with the initial solution from the previous phase. In this phase, all possible moves are evaluated, and the best one is chosen based on the objective function's values.

These moves involve changing the positions of planning objects according to specified rules. The goal is to minimize violations of hard and soft constraints. The algorithm iteratively improves the solution, replacing the current solution until a certain termination criterion is met, such as a time limit for calculations.

2.2. Avoiding local optima

An important challenge during local search is avoiding getting stuck in local maxima. These maxima are points where a better solution cannot be found by performing single moves. OptaPlanner implements several metaheuristics to assist in this task.

One of these metaheuristics is "tabu search"[5][6], which involves creating a list of "tabu" moves that are not executed, even if they are possible. This list is updated successively to avoid repeating the same moves and prevent getting trapped in local maxima.

Another metaheuristic is "simulated annealing"[7], which differs from "tabu search" in that it accepts moves that worsen the objective function's value with a certain probability. This probability decreases over time, allowing for exploration of a broader solution space.

Yet another technique is "late acceptance"[8], which evaluates only a subset of moves and accepts them if they do not worsen the objective function's value or are better than the objective function's value from a certain number of previous steps.

2.3. Ruin and Recreate extension

In addition to the methods implemented in the OptaPlanner engine to avoid local maxima, other approaches exist in the literature, such as the "random restart" of the current solution to find a completely new solution after destruction, thereby enabling the escape from local optima[9].

The general procedure for local search with the application of the R&R rule is as follows:

1. Create (or partially recreate) an initial solution using construction heuristics

2. Conduct local search to improve the solution

3. Stop the local search after a long period without improvement (potential local maximum)

4. Ruin a selected part of the solution

5. Return to step 1 and repeat until the completion of solving

2.4. Ruin phase

The ruin phase is a proprietary extension of the OptaPlanner engine and aims to partially destroy the solution after the completion of the local search phase. The degree of ruin is configurable to leverage previous best solutions. Ruin involves resetting decision variables, which can be either random or controlled in some dimension, referred to as “neighborhood” or “radial” ruin[9].

Various dimensions of the solution can undergo ruin, and the user controls which objects/decision variables are “close” to each other in a certain dimension. The distance metric is defined by the user and is implemented in the NearbyDistanceMeter interface[1].

2.5. Recreate phase

The previously ruined solution must be recreated in a manner that differs from the original solution. Construction heuristics are used to reconstruct the part of the solution that was previously destroyed. After reconstruction, another iteration of the local search phase can begin, starting from a completely new, previously unattainable solution.

If a significant portion of the solution has been ruined, there is freedom to create an entirely new solution, making it easier to tackle complex problems with intricate search spaces.

3. Test scenarios

The analysis aims to compare the R&R rule’s application in OptaPlanner across various variants against solutions without it, focusing on three optimization problems: the traveling salesman problem (TSP)[10], the vehicle routing problem (VRP)[11], and the cloud resource balancing problem[1]. Each problem uses different modeling approaches and metaheuristics to avoid local optima, and we’ve limited the analysis to these specific problems due to extended search times.

For each problem, we test three approaches: no ruin, random ruin, and defined neighborhood ruin. We also examine five ruin magnitudes: 1%, 2%, 5%, 10%, 20%, and 50% of the solution. Dataset sizes represent small, medium, and large solution spaces, with a constraint that all problems on all available machine threads should be solved in less than about two days.

To eliminate statistical errors, each test is repeated five times, with the median run selected for comparison. Results are presented in box plots. The experiment is conducted in two variants: one with a limit on ruin phases (5) and the other with a time limit (15 minutes) for a direct comparison of ruin variants to the basic variant.

Additionally, the analysis considers the autocorrelation coefficient to assess solution space "roughness", indicating the smoothness of the space based on the correlation of matching solution vectors. A higher coefficient signifies a smoother space[12].

4. Results

The autocorrelation coefficient for the knapsack and VRP increases with the arbitrary search space size (see Figure 1). This makes larger search spaces less complex and rough, facilitating search within them and thereby reducing the potential gain from applying the R&R principle. The opposite relationship occurs in the case of the TSP.

In studies on the time-constrained knapsack problem, the application of R&R has shown positive results compared to the no-R&R variant. In most cases, the final objective function value for R&R variants is better than the median for the no-R&R variant (see Figure 2). Larger ruins lead to better results. Random ruins seems more effective than neighborhood ruins, although the results are not entirely conclusive.

For the limited search space knapsack problem, it was observed that some algorithm runs fall into the local optimum trap, leading to worse results than without ruining (see Figure 3). In the average search space for the knapsack problem, neighborhood ruin tends to fall into the local optimum trap, especially for ruin sizes of 10-20%. In the case of a large search space for the knapsack problem, ruins of 10-20% tend to get stuck in a local optimum. Here too, random ruins perform better, especially those with a size of 50%.

For the TSP, R&R brings benefits, especially for medium and large-sized problems (see Figures 4 and 5). It is observed that the arrangement of cities affects the

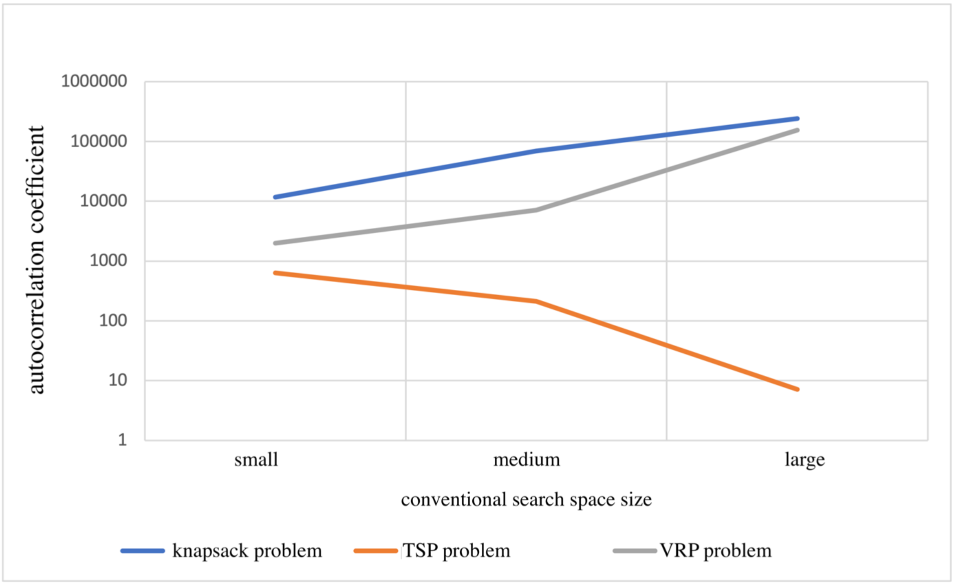


Figure 1. Correlation coefficient's dependence on the arbitrary search space size for each type of solved problem.

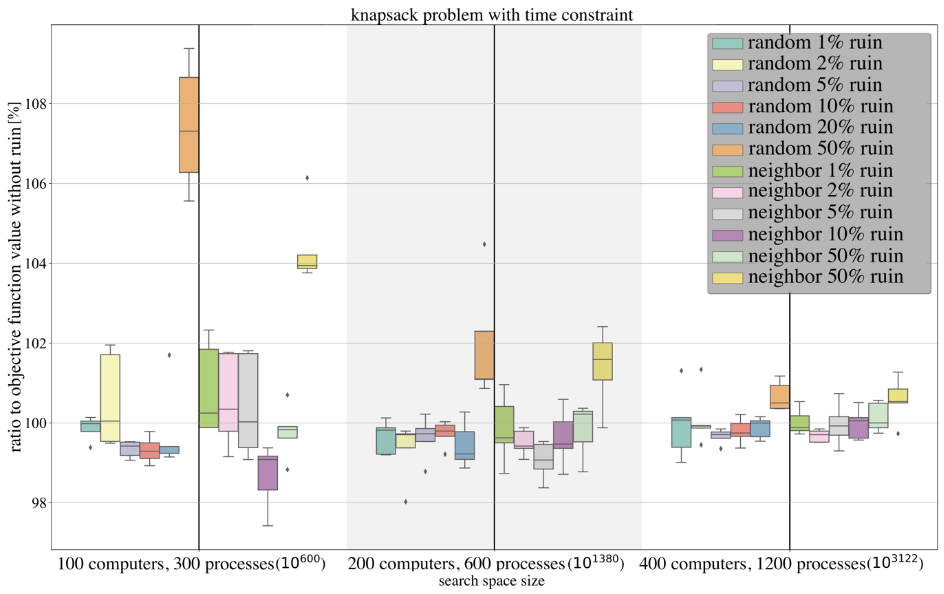


Figure 2. Percentage ratio of the final objective function value for a run with ruins to a run without ruins for various search space sizes of the knapsack problem with a 15-minute time limit for finding a solution. A higher value (above 100%) indicates a greater improvement in effectiveness due to the use of ruining.

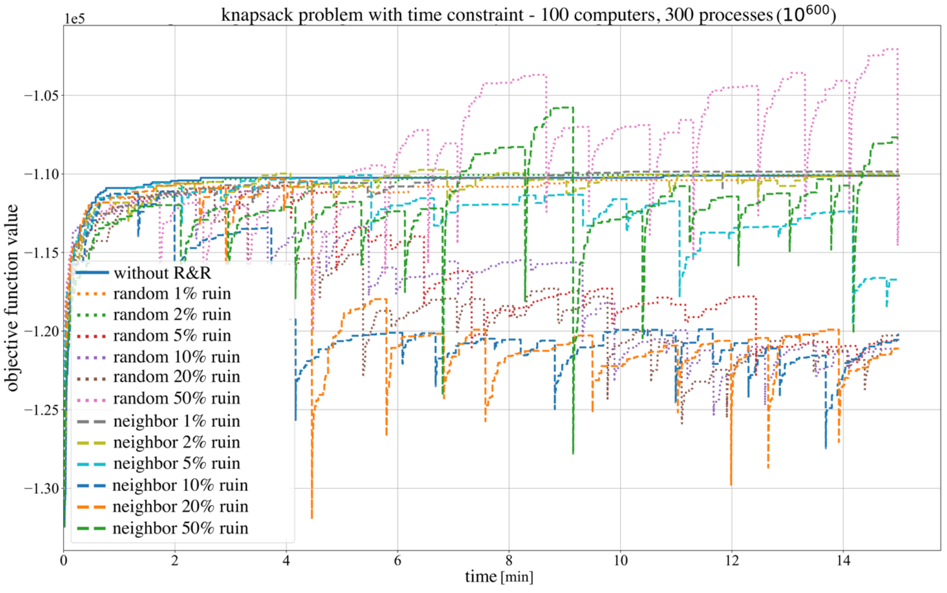


Figure 3. Percentage ratio of the final objective function value for a run with ruins to a run without ruins for various search space sizes of the knapsack problem with a 15-minute time limit for finding a solution. A higher value (above 100%) indicates a greater improvement in effectiveness due to the use of ruining.

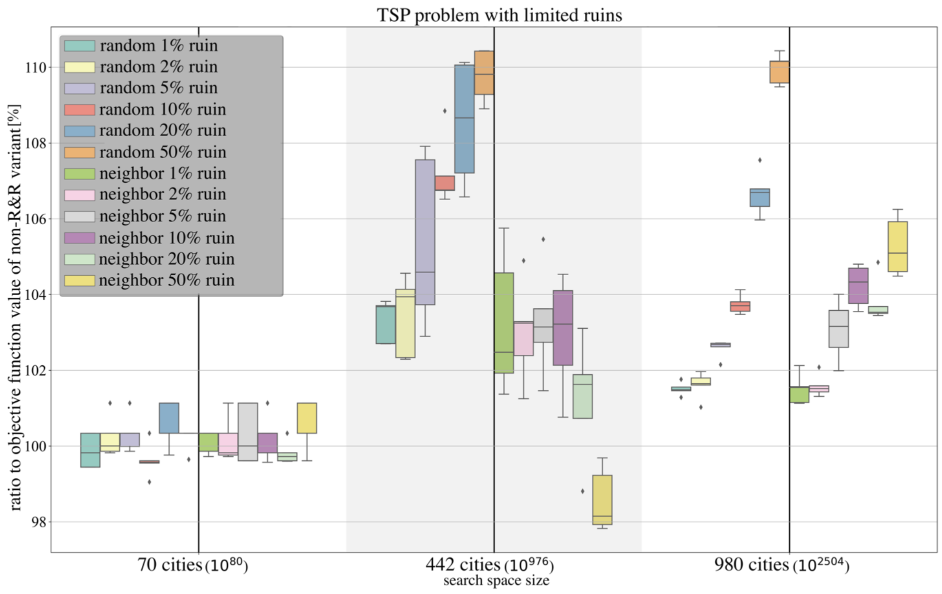


Figure 4. A box plot of the percentage ratio of the final objective function value for a run with ruins to a run without ruins for various search space sizes of the traveling salesman problem with a limitation on the occurrence of five ruins. A higher value (above 100%) indicates a greater improvement in effectiveness due to the use of ruining.

effects of ruining. The more orderly and less random the data layout, the better the effects of ruining. In a small search space for the traveling salesman problem, small ruins are not very effective, and the local optimum trap effect is noticeable. In the average search space for the traveling salesman problem, ruining does not bring significant benefits, but the local optimum trap effect is absent. In large search spaces for the traveling salesman problem, larger ruins are more effective but quickly get stuck in a local maximum. Typically, random ruining is more effective than neighborhood ruining.

For the VRP, R&R is beneficial, especially for large search space problems. The larger the ruin size, the better the results, but only up to a certain point, beyond which the effects of R&R decrease (see Figure 6). In a small search space, ruins of large sizes have a chance to find a better solution. In the average search space for the routing problem, ruins do not provide significant benefits, but the local

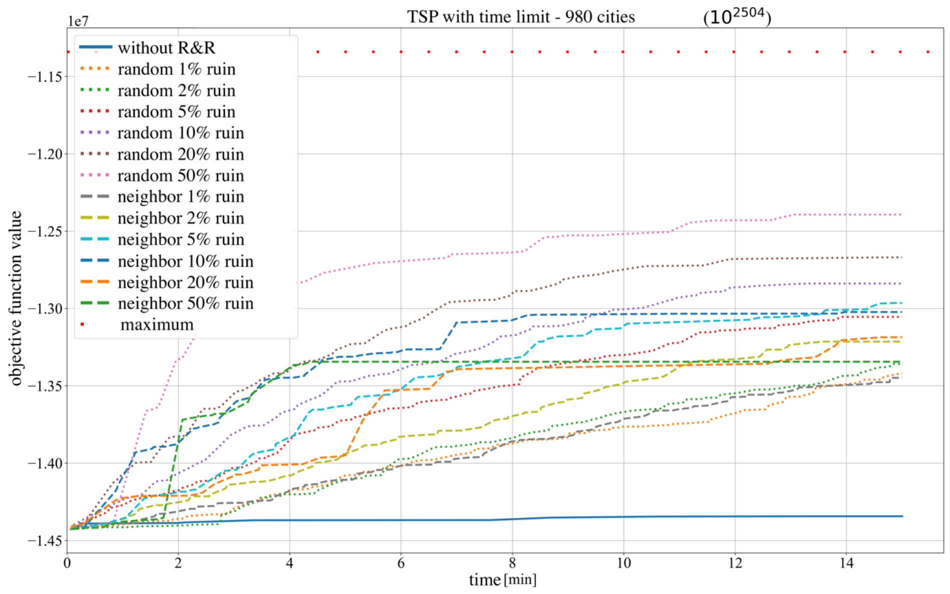


Figure 5. The course of the best solutions for various versions of the traveling salesman problem for a large search space and with a limited computation time of 15 minutes. Additionally, the global maximum known from the literature[13] has also been marked.

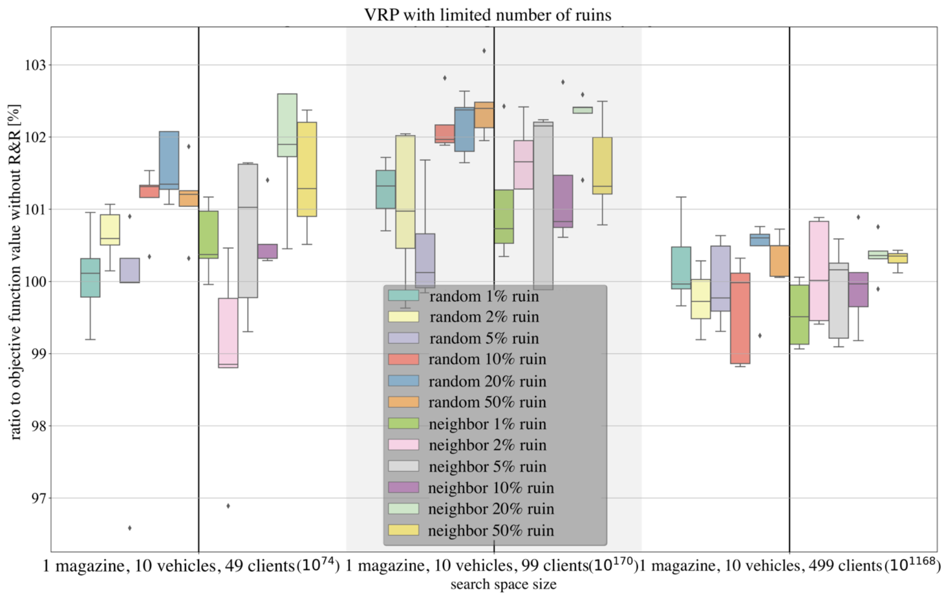


Figure 6. A box plot of the percentage ratio of the final objective function value for a run with ruins to a run without ruins for various search space sizes of the routing problem with a limitation on the occurrence of five ruins. A higher value (above 100%) indicates a greater improvement in effectiveness due to the use of ruining.

optimum trap effect is absent, because of big autocorrelation values and ease of recreating solution (see Figure 7). For large search spaces, ruins of large sizes are more effective, but the increasing autocorrelation coefficient simplifies the search space, leading to significantly longer computation times for the fixed ruins count variant (subsequent ruins occur after longer intervals).

5. Concluding remarks

In optimization problems, applying the R&R rule in the OptaPlanner engine often leads to a reduction in the time needed to find a better solution in a larger search space. The larger the search space, the more favorable the effects of R&R. However, the effectiveness of R&R depends on the complexity of the search space. In most cases, R&R is more effective in challenging search spaces. In the vast ma-

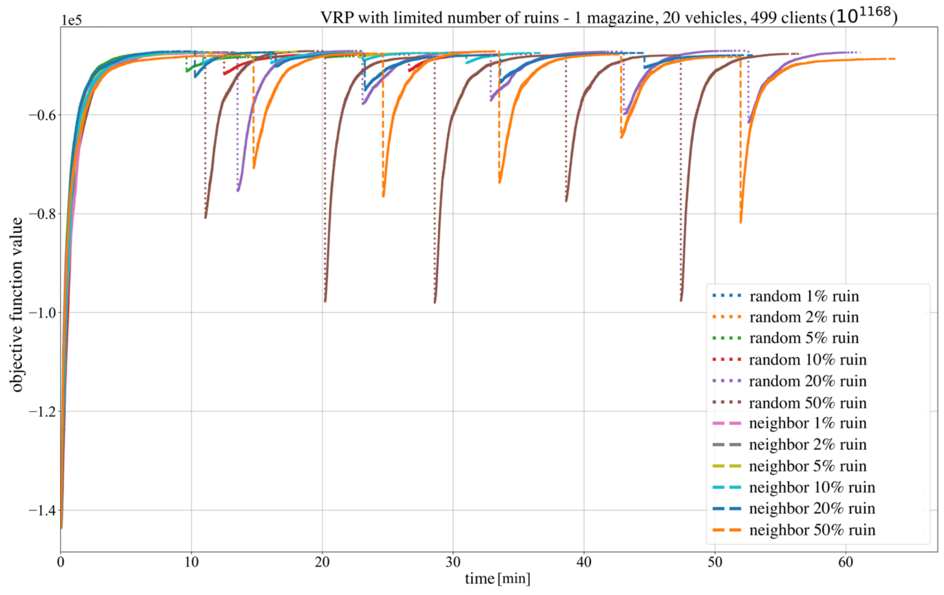


Figure 7. The course of solution steps for various versions of the routing problem for a large search space and with a limited number of ruins set to five.

jority of cases, completely random ruin is better than user-defined neighborhood ruin.

The ruin size matters - the closer to the optimal solution, the larger ruin is more effective. In most cases, large ruins are best for VRP and knapsack problems.

This leads to the conclusion that the more advanced the stage of solving, the larger the ruin should be applied. However, one should avoid using excessively large ruins at the beginning to prevent falling into the local optimum trap.

It is advisable to use metaheuristics for accepting solutions after reconstruction. Research indicates that accepting every mutation of a solution over successive R&R iterations can also lead to getting stuck in a local optimum.

Implementing the R&R rule into the OptaPlanner engine is a complex process. It requires integrating ruin as a separate move in scope of local search - rather than a separate phase after it - to fully use available metaheuristics that are provided. The research results, however, show that this effort should be profitable, as the efficiency of solving optimization problems is generally improved if ruining step is involved. The authors hope that the RedHat team will be able to reimplement the R&R rule in the future.

Acknowledgment

The authors would like to express gratitude to everyone who played a role in research works presented in this article, particularly to Lukáš Petrovický (former OptaPlanner project leader at RedHat, now Timefold[14]) and other RedHat staff involved. We are also grateful to Barbara Krupa for language and editorial support.

References

[1] Opta planner - the ai constraint solver. `https://www.optaplanner.org/` [Accessed: September 2023].

[2] Johnson, D. S., Papadimitriou, C. H., and Yannakakis, M. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988. ISSN 0022-0000. doi:10.1016/0022-0000(88)90046-3.

[3] Hämäläinen, W. Class NP, NP-complete, and NP-hard problems, 2006. `https://www.cs.joensuu.fi/pages/whamalai/daa/npsession.pdf` [Accessed: September 2023].

[4] Michiels, W., Aarts, E., and Korst, J. *Theoretical Aspects of Local Search (Monographs in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 3540358536.

[5] Glover, F. Tabu search—part i. *ORSA Journal on Computing*, 1(3):190–206, 1989. doi:10.1287/ijoc.1.3.190.

[6] Glover, F. Tabu search—part ii. *ORSA Journal on Computing*, 2(1):4–32, 1990. doi:10.1287/ijoc.2.1.4.

[7] Henderson, D., Jacobson, S. H., and Johnson, A. W. *The Theory and Practice of Simulated Annealing*, page 287–319. Springer US, 2003. ISBN 978-0-306-48056-0. doi:10.1007/0-306-48056-5_10.

[8] Burke, E. K. and Bykov, Y. The late acceptance hill-climbing heuristic. *European Journal of Operational Research*, 258(1):70–78, 2017. ISSN 0377-2217. doi:10.1016/j.ejor.2016.07.012.

[9] Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., and Dueck, G. Record breaking optimization results using the Ruin and Recreate principle. *Journal of Computational Physics*, 159(2):139–171, 2000. ISSN 0021-9991. doi: 10.1006/jcph.1999.6413.

[10] Hoffman, K. L. and Padberg, M. *Traveling Salesman Problem (TSP)*, page 849–853. Springer US, 2001. ISBN 978-1-4020-0611-1. doi:10.1007/ 1-4020-0611-X_1068.

[11] Agárdi, A. *The generalization of the Vehicle Routing Problem: Mathematical model, ontology, algorithms, fitness landscape analysis and applications*. Ph.D. thesis, University of Miskolc, 2023.

[12] Humeau, J., Liefooghe, A., Talbi, E., and Verel, S. ParadisEO-MO: from fitness landscape analysis to efficient local search algorithms. *Journal of Heuristics*, 19:881–915, 2013. ISSN 1381-1231. doi:10.1006/jcph.1999. 6413.

[13] National Travelling Salesman Problems : TSP tour of 980 populated locations in Luxembourg. `https://www.math.uwaterloo.ca/tsp/world/lutour.html` [Accessed: September 2023].

[14] Timefold - the open source AI solver. `https://timefold.ai/` [Accessed: September 2023].

Automated Escape Room Game Progress Monitoring and Moderating System

Tycjan Fortuna,
Filip Andrzejewski,
Małgorzata Komorowska,
Wojciech Michałowski,
Michał Karbowańczyk[0000–0003–3875–1101]

Lodz University of Technology
Institute of Information Technology
al. Politechniki 8, 93-590 Łódź, Poland
242213@edu.p.lodz.pl

Abstract. *This paper presents the development and implementation of an application called the Automated Game Progress Monitoring and Moderating System in an escape room game. The objective of the project was to enhance the escape room experience by providing automated game progress monitoring and moderating capabilities. The report outlines the main points of the project, including the introduction, idea finding process, solution implementation, ways of verification, and conclusion, which provide further details about the prototype of the web application.*

Keywords: *escape room, automation, immersion, monitoring, moderating, self-improvement, visualization*

1. Introduction

Escape rooms as a game format have become a global phenomenon. Since the creation of it in its current form, which, according to the Nicholson, S. (2015), happened in 2007 in Japan[1], more and more venues offering puzzle solving entertainment are emerging around the world. Research presented at the Escape Room Industry Conference 2019[2] shows that as of 2019, there were between 50,000 and 60,000 active escape rooms with the number still rapidly growing. Emerging of this branch of entertainment and constant search for improvement and optimisation poses challenges unique to its field.

The emergence of this branch of entertainment has sparked a continuous quest for improvement and optimization, presenting challenges that are distinct to its field. Escape room designers and operators find themselves at the forefront of innovation, pushing the boundaries of what is possible in interactive entertainment. They are tasked with crafting intricate narratives that transport players to new worlds, devising puzzles that simultaneously challenge and delight, and integrating cutting-edge technology to enhance the overall experience. Moreover, safety and accessibility remain paramount concerns. Ensuring that escape rooms are not only thrilling but also safe for participants, including those with physical disabilities, demands meticulous attention to detail and rigorous standards.

Customer feedback plays an integral role in this industry. Operators actively seek insights from participants, using their valuable input to refine and elevate the escape room experience continually. The feedback loop between designers and players fuels a cycle of innovation and improvement.

Table 1. Overview of escape room popularity statistics in chosen countries. Table presents name of the country, its population in millions and Number of escape rooms. Both gross and per one million inhabitants. Data collected by exitgames.co.uk[3]

Country	**Population (m)**	**Rooms (/m pop)**
NL	17	612 (36)
Poland	38	986 (26)
Luxembourg	0.57	11 (19)
USA	323	6000 (18.6)
UK	65	855 (13.2)
Singapore	5.6	68 (12.1)
France	67	811 (12.1)
Belgium	11	120 (11)
Australia	24	195 (8.1)
Austria	8.7	65 (7.5)

1.1. Background Information

Escape rooms are popular entertainment experiences where teams of two to ten players work together to solve puzzles, find clues, and complete tasks within a set time limit, usually between 45 to 60 minutes. Puzzlebreak.us blog[4] lists some of the most popular kinds of challenges. These include pattern recognition, word games, riddles or logic puzzles.

Participants immerse themselves in fictional settings like prisons or space stations, encountering challenges that match the theme of the room. Before starting, players receive a brief explanation of the rules[5][6] and goals through video, audio, or live interaction with a gamemaster. Once inside the room, the countdown begins, and players explore, search for clues, and solve puzzles to progress through the game.

Players in escape rooms can ask for hints when they get stuck, usually through a gamemaster who knows the script and can guide them toward solutions. However, there is a scalability issue. According to data collected by Nicholson (2015) [1], around 24% of escape room facilities offer one room, while 27% offer two rooms and 18% offer three rooms. The rest are facilities with even more game settings.

When rooms' gameplay continuity rely solely on human intervention, the number of simultaneous groups that needs to be supervised impacts profitability. This raises the question for business owners: How can delivering hints to players be automated, so that the owner of the escape rooms can serve several playing groups at once? The main focus of this work is to deliver the solution for this problem.

1.2. Problem Finding

The problem was identified using tools from a Problem Based Learning [7] method toolkit and a combination of qualitative and quantitative research. Following the "Double Diamond" [8] PBL method, the problem discovery and definition process were divided into two stages.

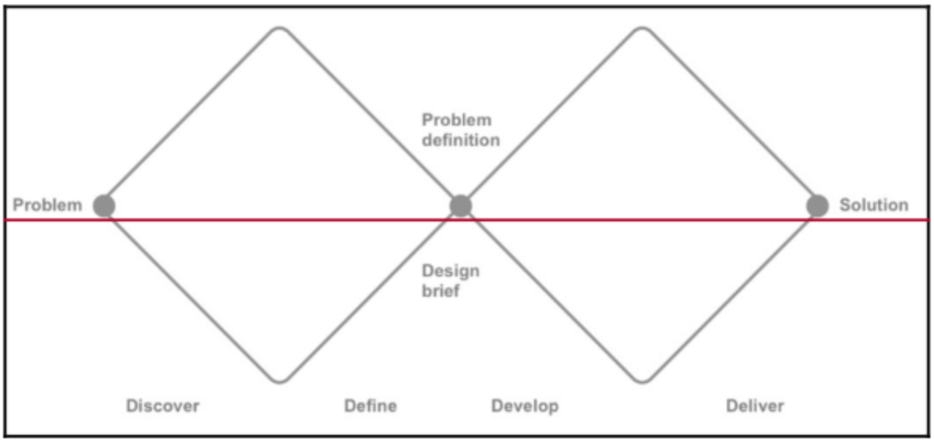


Figure 1. Double Diamond design process model. The whole approach was used in the project, although problem finding aspect, being considered in this section ends in the middle. Source: ResearchGate, The Design Council, 2007 [8]

Escape rooms and video games have been compared due to their shared characteristics [1, 9]. Both provide players with a sense of agency and control over their environment, requiring them to solve puzzles and overcome obstacles to advance. This similarity suggests that key measures used for assessing game design quality, such as functionality, usability, and gameplay, can be applied to evaluate escape rooms. Among these measures, playability, which encompasses the combined aspects of usability and functionality, holds particular significance in determining the overall success of the experience. According to the definition:

> “Playability represents the degree to which specified users can achieve specified goals with effectiveness, efficiency and, especially, satisfaction and fun in a playable context of use.”
> S̃anchez et al. [10]

Its seven main attributes are Satisfaction, Learnability, Effectiveness, Immersion, Motivation, Emotion, and Socialization.

To gain market insights and understand the perspectives of target groups, a survey and interview were conducted. The survey had over 100 participants, primarily aged 18-25, with the following key findings:

- Most participants have been to an escape room (73) or can imagine themselves in one (39).
- The ideal duration of an escape room game varied, but the most common suggestion was 1 hour.
- The majority prefer to receive hints when time is running out (107), while a smaller number prefer not to (18).
- Participants were divided on “dead-end” puzzles, with similar numbers finding them interesting (63) or annoying (62).
- A significant majority prefer to try a more difficult escape room (120) rather than an easier one (5).
- Opinions on microphone hints varied, with roughly equal numbers finding them to ruin the atmosphere (49), not ruin it (56), or having other opinions (20).
- Most participants feel safe in escape rooms (113), but a minority feel unsafe (12).
- Getting lost in an escape room does not significantly change the sense of safety for most participants (81), but some feel less safe (44).

An escape room operator was interviewed to verify the data gathered from online sources and to gain insights into the games and hint delivery systems.

Another PBL tool that has proven effective to the problem definition was POV defined for an escape room owner, which after the research looks as follows:

Escape room owner wants:

- SATISFACTION of his clients (which turns into profit)
- IMMERSION of his rooms
- CONTROL over technologies used in his games

Taking into account the previous discussion on the playability factors, to achieve client's satisfaction the game needs to have a level of feeling of control over it, for the player to feel accomplishment and involvement. Satisfaction is also rooted in game's challenge for completion and encouragement to do so, what is a game-related case of more general Self-Determination Theory [11]. Yannakakis and Hallam [12] showed that one of the key aspects that improve game satisfaction is its self-adaptation of its difficulty level to the measured skill level of the player. On the other hand Yildirim [13] points at time pressure as one of the key factors that have to be considered if the goal is to optimize player's satisfaction. Hence, the need for delivering hints, most preferably adapted to the level of the player, so that the challenge part would be fulfilled in desired time. That leaves a question of immersion to be answered by a system that would not reveal itself directly to the players while at the same time providing information necessary for them to progress in a satisfactory pace.

Factors of playability can be improved in order to increase satisfaction of the clients, and therefore profitability of the business, and at the same time have potential to make use of the hint delivery system, the following problem definition emerged: **There is a need for a system that will incorporate any method of gathering input from the room and automate control over the game**.

2. Idea Finding

2.1. State of the Art

While conducting research on potential solutions the main focus was to get information on the current state of modern escape rooms. How a game is usually conducted, what technological solutions are used to make a game easier to control and run by a room manager and lastly what possible improvements are currently being researched to enhance both user experience and automation of giving clues to the player during the game.

A number of innovative ideas were found, many of which included usage of virtual reality and augmented reality technology [14, 15]. Additionally, materials regarding automatization and digitalization were found, focusing on the implementation of embedded systems to escape room games and were mainly focused on the educational aspect of such games [16, 17]. However, the overall number of escape rooms incorporating such methods in purpose of automating hint delivery is close to none with the most popular solution for escape rooms automatization being Queen – a system based on Arduino devices with predefined scenarios for purchase developed by company Escape Room Doctor [18]. According to sources on the website, the services of this company were implemented in 125 escape rooms around the world since 2014 which compared to the records in Table 1 leaves much to be desired.

2.2. Innovative Ideas

The main focus of the development team was to find ideas for gathering input from an escape room while also keeping in mind previously mentioned properties of a game valued amongst escape room owners and designers: functionality, usability, playability and control for escape room owner, immersion and satisfaction for escape room customers. Current systems of gathering information about players' position are mostly video surveillance cameras which are not ideal for the purpose of automatization of giving clues and hints during a session.

Ideas were measured mainly based on their influence on the game design, which according to the data gathered during the interview should be as low as possible, and their potential to help fully automate the process of giving clues to the players. The best ideas were put on the 2 by 2 comparison diagram. A simplified version of this diagram as well as the list of the best ideas can be found in Figure 2.

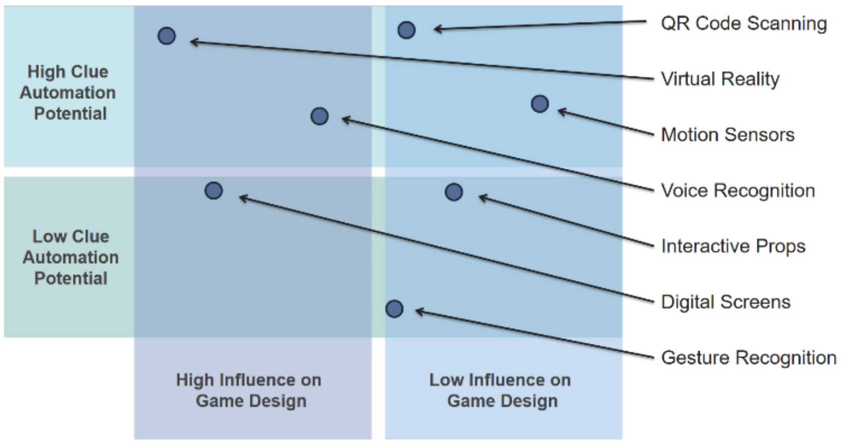

High Clue Automation Potential
Low Clue Automation Potential
High Influence on Game Design
Low Influence on Game Design
QR Code Scanning
Virtual Reality
Motion Sensors
Voice Recognition
Interactive Props
Digital Screens
Gesture Recognition

Figure 2. Simplified 2 by 2 comparison diagram of our best ideas for gathering player input.

Three highest rated ideas have been developed further by our team and thoroughly analyzed.

2.2.1. QR Code Scanning

This innovative approach significantly reduces the need for direct human intervention during gameplay, granting players access to guidance precisely when required. It is probably the most cost effective solution and presents itself with relatively low influence on the game design. The mechanics of this solution are outlined as follows:

1. QR Code Integration: Escape room designers strategically position QR codes throughout the physical space, typically in proximity to puzzles that may pose challenges to players. Each QR code corresponds to a specific clue relevant to the accompanying puzzle.

2. Player Initiation: When players find themselves struggling with puzzles or need assistance, they can employ their smartphones or provided devices to scan nearby QR codes, thus activating the automated clue delivery system.

3. Clue Dispensation: Upon scanning a QR code, players receive digital clues on their devices, offering nuanced hints or instructions to aid in game progression. The system can be configured to provide a range of hints, enabling players to control the level of assistance they receive.

4. Minimal Gameplay Disruption: This solution minimizes disruptions to the gameplay experience, as players have the autonomy to determine when and how frequently they access clues. It preserves the integrity of the game's design by seamlessly integrating automation into the room's setup.

5. Feedback and Monitoring: Operators can analyze QR code usage patterns to gain insights into which puzzles commonly challenge players, facilitating iterative improvements to game design.

Potential problems:

1. Device Dependency and Technological Proficiency: The proposition of leveraging QR codes presupposes the possession of smartphones or designated devices by participants, thereby potentially marginalizing individuals who lack access to such technology or possess limited proficiency in its operation.

2. Dilution of Immersion: The act of diverting participants' attention towards their personal devices for QR code scanning carries the inherent risk of momentarily punctuating the immersive ambiance of the escape room, engendering an experiential discontinuity.

3. Technical Vulnerabilities: The susceptibility to technical perturbations, including but not limited to connectivity issues, device compatibility anomalies, or QR code scanning malfunctions, emerges as a concern, capable of disrupting the fluidity of gameplay and engendering participant dissatisfaction.

2.2.2. Virtual Reality

An alternative and cutting-edge solution involves crafting escape rooms within the realm of virtual reality (VR). This approach offers a high degree of automation and control over clue delivery but additionally induces changes in the escape room design:

1. Escape room scenarios are meticulously recreated in a virtual reality game. The game allows players to physically interact with objects and puzzles using VR controllers or haptic gloves. The virtual environment faithfully replicates the layout and complexity of real-world escape rooms and can easily expand beyond it.

2. Automated Guidance: Within the VR experience, a set of predefined instructions or an AI-driven guide is integrated to provide clues upon player request. Players can communicate their will to receive a clue by multiple different means including button sequences on controllers, interactable objects programmed inside the game or voice commands. The clues can be provided directly onto user's VR goggles or can be represented by a change inside the world of the game

3. Immersion: Despite the automation involved, players should remain fully immersed in the game, as the VR environment can simulate various scenarios and atmospheres, enhancing the overall experience.

4. Versatility and Replayability: VR-based escape rooms offer the advantage of easy scenario modifications and the introduction of dynamic elements, ensuring each playthrough is unique and engaging for repeat sessions.

Potential problems:

1. Cost and Accessibility: The establishment of VR-based escape rooms carries substantial economical implications, spanning both the acquisition of VR hardware and the development of VR-specific content. This economic barrier may restrict access to a particular demographic.

2. Virtual Reality-Induced Nausea: VR experiences may induce heavy or in some cases even health threatening motion sickness in some individuals, detracting from the overall playability of the game.

2.2.3. Motion Sensors

Motion sensors represent an additional avenue for automating the escape room experience, particularly in terms of monitoring player progress and identifying when assistance is required. This solution has the lowest influence on the design of the game:

1. Strategic Sensor Placement: Motion sensors are strategically positioned throughout the escape room to monitor player movements and interactions with objects and puzzles.

2. Progress Monitoring: These sensors can detect when players spend an extended period in a specific area, signaling potential challenges. This data triggers the automated clue delivery system.

3. Automated Clue Delivery: When a player is detected as being stuck on a puzzle, the system can automatically provide a clue through numerous potential means like audio prompts, visual cues, or digital displays.

4. Enhanced Gameplay Flow: This solution ensures that players receive assistance precisely when needed and without need of any user input, preserving a smooth and immersive gameplay experience.

Potential problems:

1. Determining Clue Needs: The main problem with motion sensor system is the difficulty in accurately determining whether a player requires a puzzle clue solely by monitoring their physical location within the escape room. Depending solely on positional data can lead to scenarios where players are mistakenly considered in need of assistance

2. Maintenance Complexities: The operation of motion sensor systems necessitates routine calibration and maintenance to ensure optimal performance. Lapses in maintenance could introduce disruptive elements during game scheduling.

3. Sensory Precision: The reliability of motion sensors in terms of data interpretation presents a fundamental challenge.

During development and analysis of potential solutions a potential concern was raised that not every important variable was considered. Factors such as playability, influence on the gameplay and integration with the theme are very subjective and can differ between many escape rooms as they are very diverse in both visuals and gameplay. Two main methods of grading such properties of specific implementations are creating large scale surveys that gather opinions from active escape room players or employing qualified escape room game designers. Finally all three of the potential solution candidates were rejected in favor of a new idea. A new concept was born to think of the problem from further perspective.

The development team came to the conclusion that the final solution shall not impose any major changes to already existing escape rooms and will deliver a system capable of maintaining control over the escape room game by encapsulating various player input methods.

2.3. Main Idea Selection and Justification

Ultimately the solution chosen to be developed further was a system capable of encapsulating almost any information gathering method. A system that can be applied to any existing escape room without imposing changes to both visual design and gameplay design, but with ability to control the time players spend on the game by using either timed hints or distractions in order to come closer to a desired optimal playing time. This solution was chosen mainly because it provides a lot of flexibility both for the developers and the client as well as providing an opportunity to truly automate the process of running escape room games without supervision. To ensure proper functioning, implementation of an input gathering technology is necessary inside the room. The choice of method for that functionality depends on the escape room owner.

Another reason for choosing this solution is that it incorporates methods and technologies already known to the team. It is a relatively simpler approach than implementing a solution based on technologies like virtual or augmented reality. That said, the solution still does not reject such technologies. As mentioned before, it was designed to take any method of gathering player input and position assuming that it produces appropriate signals that can be sent to our system.

3. Solution Implementation

The solution is a website offering an API that automates and visualizes the distribution of hints in concurrently played games. The game's state is monitored and controlled, with the option to track and consider players' positions if necessary. Users can define and customize the structure and progression graphs of rooms within the application. These defined steps guide players from the beginning to the end of the game and are accompanied by descriptions and approximate time estimates relative to previously completed steps. The time estimates are continually updated and evaluated based on prior games played in the same room, as well as the distribution of hints and slowdowns. The communication between the application and the client's API is highly customizable, allowing each step to have a client-defined functionality module that determines the communication process.

This flexibility enables tailored interactions based on the client's specific requirements, such as integrating with different sensors or existing systems. The created room structures can be easily exported and imported, fostering a collaborative escape room community for sharing ideas and experiences. Furthermore, games scheduled for specific rooms and timeframes can be organized with player details. The dashboard panel allows users to view scheduled games, games currently being played, and games with issues.

Once a game starts, the system captures and interprets user interactions, allowing for changes to the virtual state of the game. Based on mathematical estimations and previous game data, it can be anticipated when hints or slowdowns might be needed. The deployment of hints and slowdowns is automated but can also be done manually if necessary. Storage for players' opinions about rooms and statistics on game duration compared to a reference point is provided, enabling room improvements based on feedback. The application supports internationalization in English and Polish, with the ability to add and support additional languages. The settings panel allows users to manipulate technical and visual elements, enhancing website interactivity.

3.1. Technical Details

The solution was implemented using several key technologies: Vue.js (JavaScript framework), Laravel (PHP framework), Bootstrap, Docker, and PostgreSQL. These technologies enabled efficient development, easy feature integration, and reliable testing. Two separate GitHub repositories were created - one for the main application and another for the simulator, which was instrumental in testing and emulating real-world scenarios with multiple concurrent games. This setup ensured the smooth functioning of visual and technical elements, minimizing flaws, bugs, and misunderstandings. The project benefited from utilizing Jira and Confluence platforms for issue and feature management from the outset. The clear distinction between frontend and backend layers facilitated rapid development and project growth, offering ample opportunities for skill enhancement and technology exploration.

The website's graphical design and appearance were achieved through a component-based hierarchy, each offering distinct appearance, high responsiveness, and planned functionality. Components encompass HTML, CSS, and JavaScript logic to handle user events and actions. A RESTful API was also implemented, allowing seamless communication with backend services using the HTTP protocol. All interactions are designed to be asynchronous, ensuring a smooth user experience. To provide real-time data websockets were used. Websockets were utilized to ensure real-time data transmission and communication within the application. The page is embellished with custom-made SVG assets, along with Bootstrap's functional components and icon makers to enhance the user experience. A secure login and registration system were implemented to safeguard the application.

The backend follows the Model-View-Controller design pattern, communicating with a PostgreSQL database to store escape room structures, game data, and statistics. API endpoints are prepared with documentation available. The API is designed for handling and processing requests from various devices, enabling concurrent gameplays. Data security is prioritized, requiring authentication and proper access for all users connecting or using the application as depicted in Figure 3.

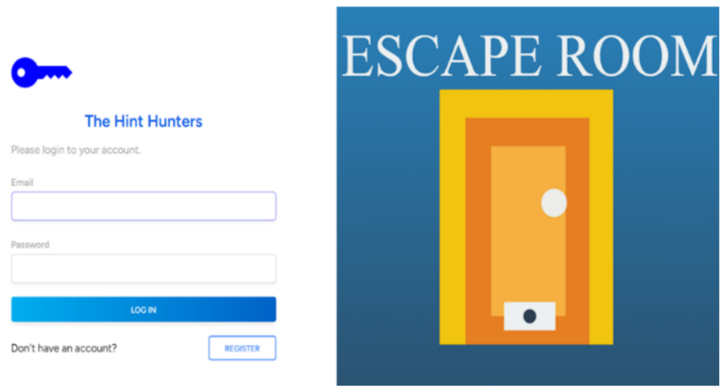
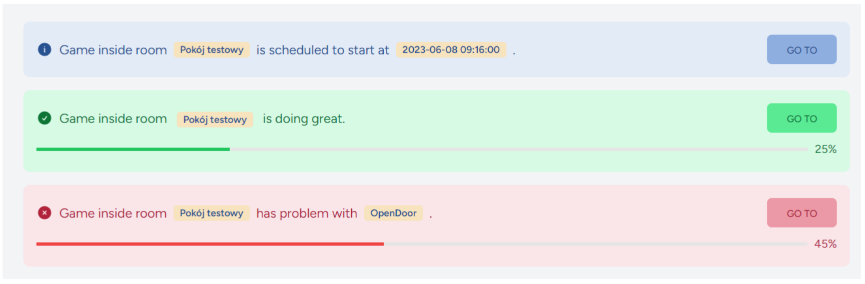


Figure 3. Authentication page of the application.

Once granted access to the application, users can securely log in and proceed with setting up room structures, scheduling games, and performing various other activities. Each game can exist in three states: scheduled, successfully playing, or suspected of a problem. The status of games can be monitored through the dashboard panel, providing an overview of ongoing activities, as shown in Figure 4.



Game inside room Pokój testowy is scheduled to start at 2023-06-08 09:16:00 .

Game inside room Pokój testowy is doing great.

25%

Game inside room Pokój testowy has problem with OpenDoor .

45%

Figure 4. Dashboard panel game monitoring preview presenting each allowed state of the game.

The system requires each room to have two specified graphs: one representing the progression and the other depicting the internal structure of the room. Examples of such graphs are shown below in Figures 5 and 6.

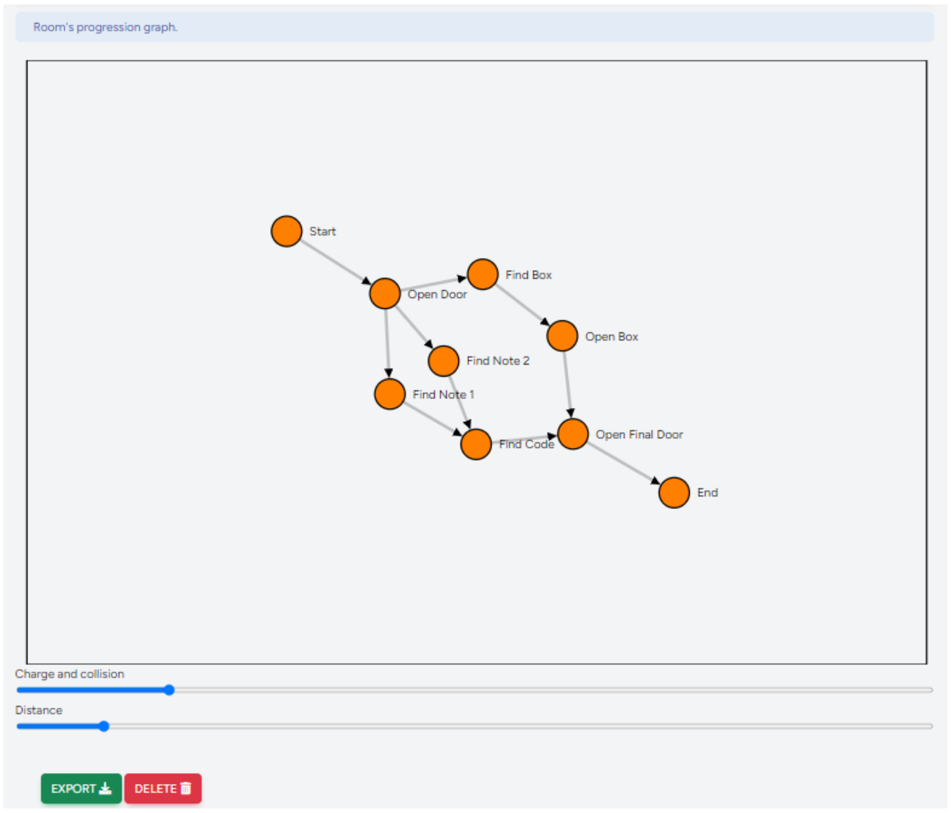


Figure 5. Example of an unidirectional graph representing the progression structure.

3.1.1. Graph simulation principles

A fully integrated graph designer was developed using JavaScript interactions with SVG library, which makes it possible to create and manage of graphs with as many as millions of nodes. The graph may manifest as either two-sided or one-sided, with each node possessing the potential for acquiring n parents and n children. It accommodates graphs of varying dimensions, facilitating scalability, mobility, as well as import and export functionalities.

Some following equations were used to develop working graph simulation:

The attraction or repulsion force between connected nodes based on Hooke's law:

$$F_{\text{link}}(d) = -k \cdot (d - d_0)$$

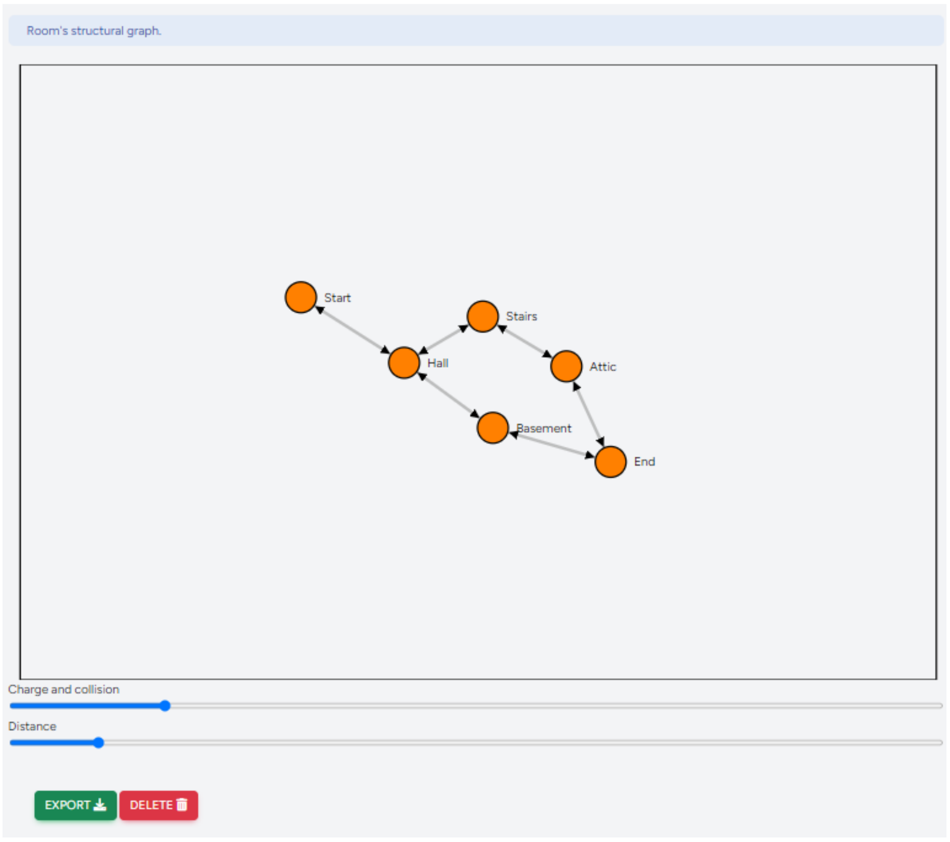


Figure 6. Example of a bidirectional graph representing the structural interconnections between different sections.

- $F_{\text{link}}(d)$ is the force.
- k is the spring constant.
- d is the current distance between the nodes.
- d_0 is the desired/resting distance.

The charge force models the repulsion between nodes and involves Coulomb's law expressed as follows:

$$F_{\text{charge}}(d) = \sum_{i \neq j} \frac{k \cdot q_i \cdot q_j}{d_{ij}^2}$$

- $F_{\text{charge}}(d)$ is the force.
- k is Coulomb's constant.

- q_i and q_j are the charges of nodes i and j.
- d_{ij} is the distance between nodes i and j.

The center force attracts nodes towards the center of the simulation area. It can be represented as a force proportional to the distance from the center:

$$F_{\text{center}}(d) = k_{\text{center}} \cdot (d - d_{\text{center}})$$

- $F_{\text{center}}(d)$ is the force.
- k_{center} is a constant.
- d is the distance of a node from the center.
- d_{center} is the desired center.

The collision force prevents nodes from overlapping and can involve pairwise interactions. A simple representation might be:

$$F_{\text{collide}}(d) = \begin{cases} k \cdot (r_i + r_j - d) & \text{if } d < r_i + r_j \\ 0 & \text{otherwise} \end{cases}$$

- $F_{\text{collide}}(d)$ is the force.
- k is a spring constant.
- r_i and r_j are the radii of nodes i and j.
- d is the distance between the centers of the nodes.

For each structure, two nodes are mandatory: the starting and ending nodes, allowing control over the game's state. Nodes can be categorized into three different states: "done", "waiting", and "problem". By analyzing these statuses during gameplay, a mathematical model is employed to detect potential issues. Additionally, preventive measures are offered for situations where teams are progressing too quickly, such as triggering events to make it harder or temporarily turning off the lights in the escape room. It's important to note that a system of embedded devices is not provided with the application, but rather offer an open API that allows clients to integrate their existing devices or choose new ones according to their preference. Standard implementations and request structures are provided inside technical documentation at GitHub repository for sending impulses to indicate the need for subsequent hints or slowdowns. However, these can be modified as nodes can incorporate client-specified functionality by creating easily connectable modules.

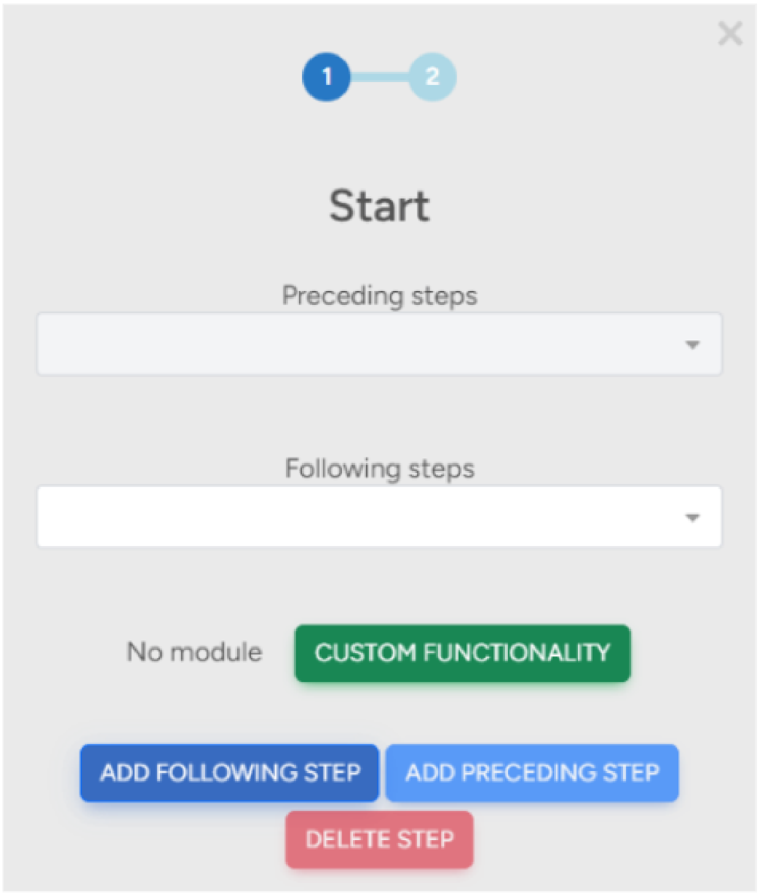


Figure 7. Popup presenting how the graph can be created along with applying custom functionality.

A popup of the graph creator when clicking on subsequent nodes is shown in Figure 7. For each game currently being played, a real-time visualization of the game's state is provided. This visualization is not only fully automated but also allows for manual actions to be performed if necessary. Additionally, the player's position can be registered and visualized if required. An example of the changing game state and represented player positions can be seen in Figure 8.

In addition to its core functionalities, the application incorporates a statistical panel that presents the duration of games in comparison to an ideal game duration, which has been determined through the research and analysis. This statistical panel offers valuable insights into the actual gameplay experience and allows for a quantitative assessment of how well games align with the expected timeframe. By comparing the actual duration to the chosen reference value, users can gain a

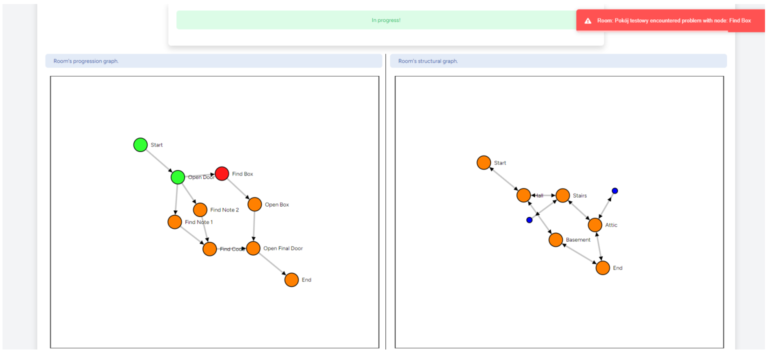


Figure 8. One of the game's panel representing both change of node's states and player movement.

deeper understanding of the performance and efficiency of their escape rooms.

3.2. Ways of verification

The Automated Game Progress Monitoring and Moderating System was tested using various verification methods. The performance, user satisfaction, and overall impact of the application were evaluated.

A system was implemented to allow forms to be filled out by players after the game in order to gather feedback. Quantitative and qualitative data for analyzing the strengths and weaknesses of the application were provided by the feedback.

The functionality, reliability, and impact of the application on the escape room industry were evaluated by consulting with the escape room manager. A deeper understanding of the strengths and areas for improvement in our application was obtained through her professional opinion.

In the future, long-term observation is planned to further assess the effectiveness of the application. This will involve closely monitoring its performance, collecting player feedback, and evaluating its impact on the overall escape room experience.

4. Conclusions

The system for was developed and implemented by the team to enhance the overall experience, following the PBL approach. Players' progress was tracked by the system, real-time updates and feedback were provided, and moderating features were offered to adjust difficulty and provide hints.

Moving forward, user feedback will be actively sought, and regular updates will be conducted to address any identified weaknesses or limitations. The application's functionality, usability, and performance will be enhanced over time through the adoption of an iterative approach.

Valuable insights into the effectiveness and functionality of the application were provided by the verification methods, including user testing, feedback forms, and expert evaluation. Long-term observations will be conducted to ensure that expectations are met by the application. Refinement and optimization of the system based on their specific requirements and preferences will be done through collaboration with escape room game designers and owners. Continuous user feedback and iterative improvements are crucial for ensuring the ongoing success and relevance of the application.

Acknowledgment

We would like to thank respondents of our survey, thanks to whom we were able to improve our application taking into account their responses.

References

[1] Nicholson, S. Peeking behind the locked door: A survey of escape room facilities., 2015. `http://scottnicholson.com/pubs/erfacwhite.pdf`.

[2] Ferguson, K. International escape room markets analysis, 2019. `https://thelogicescapesme.com/news/international-escape-room-markets-analysis/` [Accessed: June 2023].

[3] Ferguson, K. By the numbers: Uk vs the rest of the world, 2017. `http://exitgames.co.uk/blog/2017/09/14/by-the-numbers-uk-vs-the-rest-of-the-world/` [Accessed: June 2023].

[4] PuzzleBreak. 7 of the most common types of puzzles in escape rooms. `https://www.puzzlebreak.us/blog/7-of-the-most-common-types-of-puzzles-in-escape-rooms/` [Accessed: June 2023].

[5] The Vault. `https://thevaultne.com/escape-room-rules/` [Accessed: June 2023].

[6] 60out Blog. Everything you need to know about escape room rules, 2019. `https://www.60out.com/blog/advantages-to-escape-room-rules/` [Accessed: June 2023].

[7] PBLWorks. What is PBL. `https://www.pblworks.org/what-is-pbl/` [Accessed: June 2023].

[8] The Design Council. Design processes for OBM firms in the NPD process, 2007. https://www.researchgate.net/figure/Double-Diamond-diagram-The-Design-Council-2007_fig1_327200965/ [Accessed: June 2023].

[9] Brown, A. Laura E. Hall on how escape rooms are more like video games than you'd expect, 2021. https://www.nme.com/features/gaming-features/laura-e-hall-on-how-escape-rooms-are-more-like-video-games-than-youd-expect-3058147/ [Accessed: June 2023].

[10] González-Sánchez, J., Padilla-Zea, N., and Vela, F. Playability: How to identify the player experience in a video game. *Video Game*, pages 356–359, 2009. doi:10.1007/978-3-642-03655-239. URL https://www.researchgate.net/profile/Francisco-Luis-Vela/publication/260247169_2012_INDEXADA_BIT/links/00b7d5304e07db0e1a000000/2012-INDEXADA-BIT.pdf.

[11] Deci, E. and Ryan, R. Self-determination theory. *Handbook of Theories of Social Psychology*, (1):416–437, 2012. URL https://isbndb.com/book/9780857029607.

[12] Yannakakis, G. and Hallam, J. Real-time game adaptation for optimizing player satisfaction. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(2):121–133, 2012. doi:TCIAIG.2009.2024533.

[13] Yildirim, I. Time pressure as video game design element and basic need satisfaction. *MSc dissertation at Middle East Technical University*, 2015. doi:10.13140/RG.2.1.4107.0564.

[14] Zeng, H., He, X., and Pan, H. Implementation of escape room system based on augmented reality involving deep convolutional neural network. *Virtual Reality*, (25):585–596, 2021. doi:10.1007/s10055-020-00476-0.

[15] Mystakidis, S., Papantzikos, G., and Stylios, C. Virtual reality escape rooms for STEM education in industry 4.0: Greek teachers perspectives. *Proceedings of the 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, 2021. URL https://www.researchgate.net/publication/354545146_Virtual_Reality_Escape_Rooms_for_STEM_Education_in_Industry_40_Greek_Teachers_Perspectives.

[16] Lubyagin, I., Ivanova, A., and Loginov, S. Microcontrollers in automation systems of escape rooms. *Human. Environment. Technologies. Proceedings*

of the Students International Scientific and Practical Conference, 0(21):164–169, 2017. URL `http://journals.ru.lv/index.php/HET/article/view/3559/3567`.

[17] Pfeifer, M., Völker, B., Böttcher, S., Köhler, S., and Scholl, P. M. Microcontrollers in automation systems of escape rooms. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 1103–1109, 2021. URL `https://www.researchgate.net/publication/349840093_Teaching_Embedded_Systems_by_Constructing_an_Escape_Room`.

[18] EscapeRoomDoctor. QUEEN - your powerful solution to create and control escape rooms. `https://escaperoomdoctor.com/queen/` [Accessed: June 2023].

Teaching Game Development in the Białystok Desert of the Game Industry: The Game Development Track

Krzysztof Gawryluk[0000−0002−5677−6719]

Uniwersytet w Białymstoku
Wydział Fizyki, Pracownia Fizyki Układów Mezoskopowych
ul. Ciołkowskiego 1L, 15-500 Białystok, Poland
k.gawryluk@uwb.edu.pl

Abstract. *This article presents information regarding education in the field of physics with a specialization in 'Computer Game Physics and Robotics' offered at the Faculty of Physics in Białystok. The focus is on introducing the concept of the 'game development track', which involves the use of ready-made tools related to game development, differing from the 'programming track' described in [1]. The 'programming track' covers subjects dedicated to 3D graphics programming, physical engine creation, and graphic engine development. The innovative fusion of physics in the context of games and robotics takes place in the city of Białystok, where the video game industry is practically nonexistent.*
Keywords: *education, gamification, computer games, blender, godot*

1. Introduction

The specialization 'Physics of Computer Games and Robotics' (PCGR) was established in 2016 as a response from the Faculty of Physics (FoP) at the University of Bialystok (UwB) to the declining number of candidates enrolling in the physics program. Despite FoP's high scientific ranking (A category in ranking of Ministry of Science and Higher Education consistently since 2010), demographic shifts [2], a reluctance to study the exact sciences (especially at universities[1]), population migration from Białystok [3, 4], and the desire to study outside one's hometown in larger scientific centers are the primary reasons for the decreasing number of FoP students[2]. Enhancing the educational offering with a specialization

[1]observations of the employees at the Faculty, after numerous conversations with candidates during recruitment meetings, show that young people expect well-paid jobs in emerging industries. However, working at the university is more commonly associated with a future career in academia.

[2]it's difficult to pinpoint a dominant cause; instead, one should consider the superposition of these factors

that aligns with current employment trends, especially the growing role of information technology (IT) fields, seemed to be a constructive strategy that leveraged the expertise of FoP's teaching staff[3]. After seven years of offering the PCGR specialization, it is time for further reflection[4]. It involves sharing experiences with other institutions and starting a dialogue on the possibility of combining physics education with the creative field of video game development (game-dev). An additional curiosity (or rather a challenge in achieving this goal) is the absence of a game-dev industry in the Podlaskie Voivodeship, especially in the city of Białystok, where until 2017, there were only three registered companies [5]. By 2022, the number had increased to five [6]. These numbers are peculiar and unique on a national scale because the IT sector itself thrives in Podlaskie, but the video game industry seems to have been marginalized (replaced?). This latter observation is a subject of analysis more suitable for sociologists and will not be pursued further, with the focus remaining on the attempt to combine the desire to study physics with game-dev.

2. Game Development Track

Studies in the PCGR specialization encompass typical subjects in mathematics and physics, with an additional significant focus (approximately 30% of the classes) on IT-related subjects. Among them, we distinguish:

- Operating systems (covering Linux and Windows, with an emphasis on Linux system work).
- Programming (commencing with the learning of the C language, followed by C++ in the next semester, and then scripting languages - with a focus on Python).
- Graphic tools (2D graphics: Gimp[7] and Inkscape[8], 3D: Blender[9]).
- Utilization of ready-made game engines (Godot[10]) and acquainting students with modern game development methods.

The purpose of the 'game development track' is to teach students how to use ready-made tools in the context of game development. This approach differs from the 'programming track' [1], where students utilize their acquired knowledge in physics, mathematics, and numerical methods to build their own physics

[3]FoP employees regularly employ advanced IT technologies, including programming and parallel programming on high-performance computing clusters (HPC)

[4]in the fourth year of running the PCGR specialization, changes were introduced that involved reshuffling the sequence of didactic subjects, bringing in easier academic subjects specialized classes (graphics software like Gimp and Inkscape) more quickly at the expense of more challenging academic subjects classes in mathematical analysis or various physics branches

and graphics engines. The selection of the specific tools in the 'game development track' was influenced by the preferences of the instructors and the desire to promote the concept of open-source software among students. The latter aspect is significant in the context of potentially forming a small game development team (in Poland, nearly one-third of game development studies employ up to 5 people [11]). Software costs are, therefore, crucial, and this issue is addressed through the use of open-source software. The chosen programs are also well-known and appreciated in large game development studios (e.g., Blender), or are gaining recognition (e.g., Godot). The decision regarding software selection may have (and likely does have) a significant impact on how the PCGR specialization is perceived in the future. A discussion regarding this choice is provided at the end of this article.

3. 2D Graphics

The aim of the "2D Graphics" subject in the PCGR specialization is to acquaint students with the fundamentals of both raster and vector graphics. We concentrate on Inkscape and Gimp programs. This block comprises 30 hours of hands-on laboratory practicals. Our goal is to teach typical concepts related to graphics, as well as user interface commands (repeated in every other graphic software, not just the ones we have chosen), and basic skills in image manipulation (e.g., image scaling, merging multiple images, and more). We instruct on working with layers, understanding color spaces, photo editing (including cloning image fragments, removing backgrounds, and drawing with various tools/brushes). We don't profess to be artists from the academy of fine arts - our objective is to present the programs, their capabilities, and typical applications. The skills acquired in

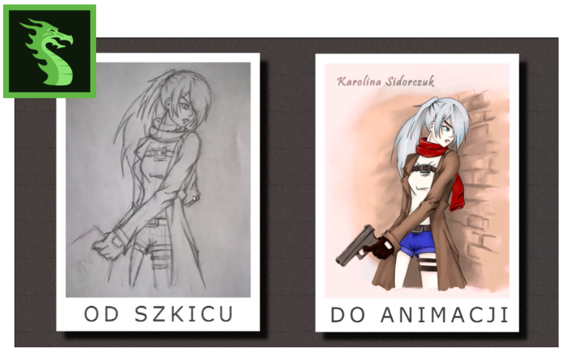


Figure 1. Sample Student Work - 2D Animation. Software: Dragon Bones. Source: http://gry.fizyka.uwb.edu.pl.

this course are also beneficial in the everyday and professional life of an FOP graduate, whether they become a scientist, teacher, or work in another industry.

With the ability to create simple 2D graphics, our graduate is capable of preparing high-quality educational materials, such as charts (we also offer a separate 'Data Visualization' course, spanning 60 hours, where, in addition to creating charts using gnuplot, we focus on programming in Python, utilizing the popular Matplotlib library), flowcharts, and electronic circuit diagrams, all without the need for specialized tools. A scientist, on the other hand, can employ these skills in preparing publications. Industries other than those of a scientist or teacher can also benefit from these abilities.

Depending on the proficiency of the students participating in the course, we sometimes manage to expand the course content to include one or two sessions on 2D animation. For this purpose, we use Dragon Bones, which features all the typical elements of the interface found in other similar programs. Occasionally, there are artistically talented students who, thanks to the tools they have learned, create their own projects (a small step toward building their own portfolio). An example of such work is presented in Fig. 1. Students delve deeper into animation during the '3D Graphics' course.

4. 3D Graphics

Our students learn the fundamentals of 3D graphics in two thematic blocks. One of them consists of classes from the subject '3D Design and Printing' (part of the robotics course where students focus on CAD tools and use the department's 3D printers for their own projects[5]. The other block focuses on '3D Modeling' classes, which entirely center on Blender and the creation of objects (assets) for games. Similar to the '2D Graphics' classes, we do not emphasize artistic qualities (which we lack) but rather focus on acquiring practical skills - mastering the program (especially typical menu items), techniques, and concepts. We highlight aspects of model performance in the context of their use in 3D games (number of vertices used, object detail, and applied modifiers) as well as selecting the appropriate texture resolution for a model.

In the scope of "3D Modeling" (with a dimension of 45 laboratory hours), students learn hard surface modeling, low and high-poly modeling, typical modifiers, material creation (including procedural materials), animations (involving the use of curves and camera movement along them). Discussions also cover texture mapping (UV mapping), material baking (normal/diffuse/etc maps) in the context of effective techniques for creating 3D game assets. We emphasize the creation of "optimized" materials to ensure better video game performance. We often make use of ready-made text and object repositories available on the internet. Students also become acquainted with Blender extensions (plugins), which are useful for

[5]discussion of these classes will be omitted as they are not related to the game development theme

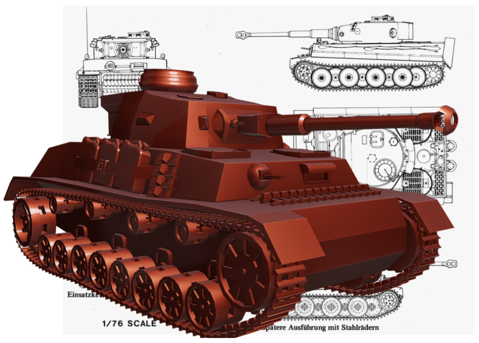


Figure 2. Sample Student Homework - 3D Modeling (Without Texturing, Based on a Technical Drawing). Software: Blender. Source: http://gry.fizyka.uwb.edu.pl.

later learning, such as for creating buildings [12]. We're aware that Blender is a powerful tool highly regarded in the game development community [13], and it's impossible to cover all of its functionalities. As a result, we omit 3D sculpting, video editing, or node-based programming. Instead, we dedicate our time to modeling from technical drawings (known as blueprints, widely available on the internet), which doesn't require artistic skills.

We place a strong emphasis on students' independent work, so they complete around 10 assignments per semester, which is a substantial number considering the typical 15 sessions per semester. These assignments are often time-consuming (instructors adjust their complexity to allow completion within 2 – 3 hours). With a small number of students, we can individualize these assignments - each student receives a unique task to perform[6], making it challenging for them to download pre-made solutions from the internet. Furthermore, this approach allows students to slowly build their portfolios. See Fig. 2 for a comparison.

Within the "3D Modeling" course, we also spend time exploring skeletal animation, not only in the obvious context of humanoid characters[7]. Instead, the focus is on acquainting students with the technique to enable them to create their own animations, such as an opening chest (using bones) or creatures with 3/7 limbs. Regarding animations within the Blender environment, we also discuss the use of Shape Keys (to later apply this technique when discussing game engines).

[6]for example, modeling tanks from World War II, each student gets a different model to replicate in Blender

[7]such animations are available in high quality on free websites like mixamo.com, eliminating the need to replicate them

5. Game Engine

During this course segment, students learn to utilize a modern “all-in-one” solution (game engine) for game development. We’ve chosen the Godot environment for this purpose. While we previously experimented with Unity for 3D projects and libGDX for 2D games, after a few years of experience, we shifted our approach and standardized on Godot. This choice was also influenced by the clarity of its scripting language, namely GDScript[8], which is similar to Python, taught in the PCGR specialization. Another advantage of the Godot engine is its lightweight nature, allowing effective work on nearly any contemporary personal computer or laptop. This is particularly important for our students who work at home and have various hardware setups.

The increasing popularity of the Godot engine, along with a growing community and the availability of numerous pre-made solutions, also played a role in our decision. Additionally, Godot offers virtually all typical features found in dominant commercial solutions like Unity or Unreal Engine. Further discussion of this choice will be presented towards the end of this article.

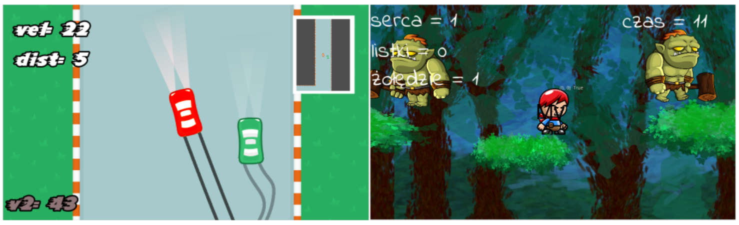


Figure 3. Two examples of coursework from the “2D Game Programming” course. From left to right: a “racing game” with AI elements (computer opponent) and a platformer with custom 2D graphics. Software used: Godot, Gimp, Krita. Source: http://gry.fizyka.uwb.edu.pl.

The first encounter with the Godot engine takes place during the “2D Game Programming” course, where students are introduced to basic concepts of creating scripts associated with objects. The course covers the sequence of engine operations (program flow: rendering, user interaction, physics engine, signals). Students complete these sessions with a mandatory assignment (unique for each student), where they can use internet assets, and the assessment is based on their application of engine techniques and programming skills.

“3D Game Programming” is another opportunity for students to work with the Godot engine, this time in combination with “3D Modeling.” Students expand

[8]it is also possible to program in C#. However, due to its user-friendliness and ease of programming, we primarily focus on GDScript

their experience with Godot, focusing on the transition of their projects into the 3D realm. This stage requires understanding concepts such as 3D transformations, rotation matrices, and quaternion principles. However, the engine itself largely handles these aspects, relieving students from coding them independently.

Students can utilize their skills with Blender (though they can still access ready-made assets online, as we don't emphasize artistic creativity). Additionally, they learn to write their shaders (based on pre-existing ones from sources like godotshaders.com), and they animate humanoid characters (using resources from mixamo.com). We don't hesitate to encourage the use of these pre-made solutions because it illustrates the collaboration of all elements in a larger project.

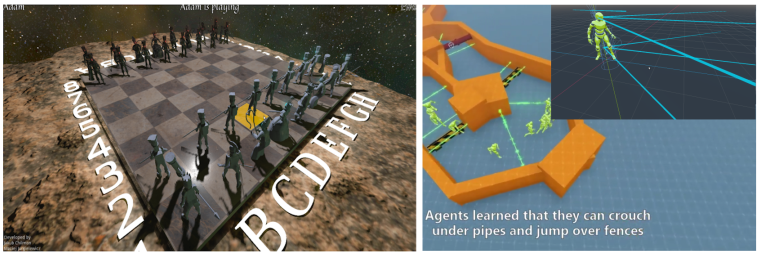

Adam is playing
Agents learned that they can crouch under pipes and jump over fences

Figure 4. Assignments from the "3D Game Programming" course - a chess project with custom graphical assets and a machine learning project. Software used: Godot, Blender. Source: http://gry.fizyka.uwb.edu.pl.

Students receive their grades based on their final project presentations, either individually or in pairs. These projects should not be expected to feature elaborate graphics or complex narratives. Experience shows that students typically spend about a month on these assignments, given their other coursework responsibilities. However, we expect the creation of a game prototype, which should include a start screen, gameplay elements, and an end screen.

Students often surprise us with their diligent project work. For example, a pair of students jointly created a chess game project (seen in the left panel of Fig. 4). Noteworthy in this project is their work on custom assets - the chess pieces were individually modeled and textured. Furthermore, the creators set a self-imposed limit of 16,000 vertices for each of the chess pieces.

Godot is not solely intended for game development. It can be used for various purposes, including creating physical simulations, virtual architectural tours, or other types of visualizations, such as machine learning. In the right panel of Fig. 4, we present another project where machine learning is employed to train non-playable characters (NPCs) to navigate obstacle courses. Each NPC is equipped with built-in sensors that detect obstacles, which the algorithm seeks to avoid while optimizing their movement in a winding corridor. NPCs earn points for reaching designated locations (marked by a visible light beam in the image) or lose points upon collision with walls. Machine learning refines the NPC's movements over successive generations of training.

6. Gamification

In our classes, we strive to employ gamification methods, which involve scoring both in-class and homework assignments. We create a ranking system (on the university's website) visible to the entire class, complete with specific "checkpoints" representing earned course grades. Additionally, students can earn extra points ("bonus points") for completing additional content in their homework, finishing it more quickly (e.g., the first two students receive bonuses), or even exceeding a certain point limit first ("flight bonus"). These methods have been well-received by our students, as they increase engagement in the subject. However, they require additional effort from the instructor, and not all instructors find this approach suitable. It has been observed that these methods increase student engagement but can also foster competition among students, potentially at the expense of collaboration. Therefore, gamification methods work well for individual subjects such as computer graphics or modeling but may not be as suitable for game design courses or typical computational exercises where student cooperation is expected.

7. Student Success - Cyberiada

It appears that our students are somewhat reluctant to participate in competitions, likely due to the demanding nature of their physics studies, which involve a lot of material to cover and numerous assignments (such as preparing reports from physical experiments in the "laboratory of physics"). Nevertheless, students specializing in PCGR took on the challenge of the "Cyberiada" (https://cyberiada.info), the "Computer Game Creation Championship," in 2022. The competition involved the development of the main game, where teams earned points by presenting specific progress in their own projects, referred to as "milestones." They were also required to complete additional tasks prepared by the organizers, such as creating a game description document or recording video blogs. Additionally, participants had to take part in two game jams. The competition spanned seven months and included teams from Lublin, Wrocław, Bydgoszcz, and Białystok.

The team from FoP reached the finals and secured second place. This success was acknowledged by the Dean of FoP, who sponsored a trip to the PGA trade fair in Poznań and a presentation of the students' projects. The staff of the PCGR specialization hoped that this achievement would break down barriers and encourage more students to participate in national competitions. This has partially come to fruition, as in 2023, we participated with two projects in the "Team-based

Computer Game Creation" (better known as ZTGK in Polish) competition, successfully qualifying for the finals. The positive results in both competitions have had a positive impact on the morale of our students.

8. Alumni

It turns out that the majority of our graduates find employment in the IT industry, capitalizing on their knowledge of Linux operating systems, programming experience (skills in C, C++, Python), database programming, and web development. However, the skills related to the "game development path" discussed here don't necessarily correlate with employment success (with a few exceptions). This is likely due to the lack of a game industry presence in Podlaskie, which means that other sectors are the ones hiring our graduates and offering them jobs locally. Another factor is the choice of specific tools, with the most controversial one being the Godot engine with GDScript, rather than the popular Unity with C#. Yet, similar arguments can be made against the latter choice, such as the absence of Unreal Engine and the emphasis on C++.

We are aware of the ongoing debate in the game development community regarding Unity and Unreal Engine, particularly at national game development conferences like the Game Industry Conference (GIC) in Poznań and Digital Dragons (DD) in Kraków, where there are varying expectations for Unity and Unreal. We recognize that our PCGR specialization does not follow the typical gaming curriculum, as we don't offer specific tracks in game design or game art but rather focus on the application of physics in game development.

9. Summary

The Faculty of Physics introduced an innovative specialization in 2016, combining physics with elements of game development. The goal was to use physics knowledge to create computer simulations that could enhance popular game engines or even build them from scratch. The second approach was to utilize existing solutions, useful not only in game development but also for attractive visualizations and simulations.

The emphasis on teaching hard skills (specialized software handling) without forcing an artistic aspect seemed like a good idea since game projects are teamwork-based, where everyone has an assigned role. The choice of open-source software was a deliberate decision by the Faculty of Physics, student-friendly (no need to buy licenses or rely on incomplete free licenses) and part of a broader plan to use open-source tools. Teaching programming languages fulfills employer expectations (C, C++, Python), but specific tools (Blender, Gnuplot, GIMP) often lose importance when the final outcome matters (although it's a good approach for freelancers, employment in larger companies usually prioritizes paid solutions/programs, often based on Windows OS).

The participation of our students in national competitions is undoubtedly a good move, though somewhat late in implementation. An open question remains: can we attract high school students to study physics on the border of video games, the target audience for our educational offer?

We sincerely thank you for any feedback on this matter (the author's contact information can be found at the beginning of the article).

Acknowledgments

The author would like to express gratitude for productive discussions with Rafał Szrajber, Jarosław Andrzejczak and Piotr Napieralski.

References

[1] Karpiuk, T. Ucząc programowania gier na białostockiej pustyni branży gier: ścieżka programistyczna. *TEWI (Technology, Education, Knowledge, Innovation) 2021*, 2023.

[2] Cierniak-Piotrowska, M., Dąbrowska, A., Potocka, M., Góral-Radziszewska, K., Stelmach, K., and Woźnic, M. Sytuacja demograf iczna polski do 2022 r. 2023. URL `https://stat.gov.pl/obszary-tematyczne/ludnosc/ludnosc/`.

[3] Godlewska, A. and Szeszko, A. Ludność, ruch naturalny i migracje w województwie podlaskim w 2020 r. *Urząd Statystyczny w Białymstoku*, 2021. URL `https://bialystok.stat.gov.pl/publikacje-i-foldery/ludnosc/`.

[4] Godlewska, A. and Szeszko, A. Sytuacja demograczna województwa podlaskiego w 2021 r. *Urząd Statystyczny w Białymstoku*, 2021. URL `https://bialystok.stat.gov.pl/publikacje-i-foldery/ludnosc/`.

[5] Miodońska, P. and Raczyk, A. Producenci gier wideo w polsce. *Czasopismo Geograficzne –90(2) 2019*, pages 131–155, 2019.

[6] Polskigamedev.pl. pages 1–2. Fundacja Indie Games Polska, 2022. ISBN 978-83-958502-3-3.

[7] URL `https://gimp.org`.

[8] URL `https://inkscape.org`.

[9] URL `https://blender.org`.

[10] URL `https://godotengine.org`.

[11] Jakub, M., Sławomir, B., and Eryk, R. The game industry of poland — report 2023. page 14. Polish Agency for Enterprise Development, 2023. ISBN 978-83-7633-487-5.

[12] URL `https://blender-addons.org/building-tools-addon/`.

[13] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. Binary robust independent elementary features. pages 778–792, 2010. doi:10.1007/978-3-642-15561-1_56.

Visual Consistency Assessment for a Series of Graphic Design Works: A Machine Learning Approach

Krzysztof Guzek$^{1[0000-0002-8060-7077]}$**, Maja Garkowska**

^{1}*Lodz University of Technology*
Institute of Information Technology
Wólczańska 215, 90-924 Łódź, Poland
krzysztof.guzek@p.lodz.pl

Abstract. *This paper presents an automated visual consistency assessment method for a series of generative graphic design works. A comprehensive evaluation of visual consistency is a complicated, multi-faceted task, as it needs to consider the full range of aspects associated with image perception. In this work, we focus on three main differentiating factors: the colour relationships, the type of graphic design elements used, and the ratio of positive to negative space. The problem in question is addressed using a machine learning-based approach. For the purpose of this study, we developed a convolutional neural network model, which was trained on a data set prepared in consultation with a visual arts expert. The predictions of the system were validated through subjective perceptual tests. As indicated by the results, the approach presented has the potential to enhance the process of graphic design, while also unveils the areas for further research.*
Keywords: *visual consistency, generative design, machine learning, convolutional neural networks, perceptual tests, design automation*

1. Introduction

Today's graphic artists face various challenges, from meeting deadlines and clients' expectations to staying creative and competitive, which results in an increasing integration of automation into graphic design workflow. The automated design tools using rule-based procedural mechanisms, driven largely by AI technology, provide the ability to instantly produce hundreds of potential outcomes based on a set of input data [1]. The tools based on machine learning (ML) offer a great potential for automated assessment of graphic design features based on a set of aesthetic criteria, enhancing significantly the generative design process, while also making the assessment process more objective. For example, Wang et al. [2] proposed a deep element selection network (DESN), which allows users to automate the selection of graphic design elements. Given a layout file with annotated elements, new graphical elements are selected to form a graphic design based on aesthetics and consistency criteria, such as colour distribution and the ratio of positive to negative space. The network is trained using a complex reinforcement learning based system, with a novel reward function that accounts for both visual aesthetics and consistency, providing the ability to generate the most visually readable and aesthetic drafts. Another example of an intelligent graphic design system is Vinci [3], a solution designed to support the automatic generation of advertising posters. Given the product image and taglines specified by the user, Vinci applies a deep generative model to match the product image with a set of design elements and layouts for generating an aesthetic poster. The system also incorporates an online editing-feedback mechanism that assists the users in editing the poster and supports them in updating the generated outcomes according to their design preference. In [4], the author demonstrated the application of AI technology to an aesthetic analysis of graphic design. The system employs a convolution-automatic encoder (CAE) to measure visual features such as symmetry, composition and readability.

Machine learning models can also be used to analyse graphic designs in terms of shared features to determine the degree of similarity between design concepts, and thus detect potential instances of plagiarism. A major difficulty associated with the similarity learning process in a traditional deep neural network is the insufficient number of training samples. This problem can be solved using the Generative Adversarial Network (GAN) [5]. An example of a GAN-based approach was presented in [6], where the generator of GAN was used to create similar graphics following the common plagiarism features of logo design. The traditional discriminator, which judges the authenticity of the generated image and the original image, was modified to calculate the perceptual similarity of the graphics pair.

In a design concept comprising a collection of graphic works, such as book or magazine covers, visual identity or user interface projects [7], it is important that all the works in the collection should demonstrate a certain degree of unity to create a strong, recognisable character of the series as a whole and foster immediate recall among the audience even after a short exposure to any of the individual components, while also maintaining the system's capability to generate multiple variations of the work without losing the unique character of the outcomes. This may be referred to as visual consistency - a concept that should not be confused with similarity. There are many examples of graphic design series which exhibit significant differences in terms of colour palette and types of graphic elements used, but are still perceived as consistent, as the differences observed fall within the framework of adopted visual rules. The consistency classification thresholds in a series of graphic designs should be established in such a way as to maintain harmony and coherence between the constituent works. However, such a description of consistency conditions does not define a precise and universally applicable set of rules to be followed by a designer. The number and character of discriminating factors adopted for a graphic design depends largely on the very nature of a specific design case and might not be transferable to other cases. Moreover, the audience's interpretation may depend on personal perception and aesthetic sensitivity. Thus, the subjectivity factor in the assessment process makes it difficult to determine what a "consistent style" or "consistent colour palette" exactly means. The confrontation of human, subjective vision with "raw" image analysis based on machine learning may bring some unexpectable results. Therefore, the final judgement regarding visual consistency should be left to a graphic design specialist.

2. Visual consistency

Due to the fact that "consistency", in the common understanding of the word, is sometimes considered synonymous with "similarity", there seems to be a need to draw a clear distinction between these two terms. Similarity describes a relation between objects definable only in reference to shared properties, which allows one to classify these objects as belonging to the same group. However, grouping by similarity makes sense only if it proceeds from a common base and if placed in the proper context [8]. Consistency, on the other hand, denotes "close correspondence" or "uniformity" among the parts of a complex thing - the term "consistent" may be regarded as synonymous to "coherent", "harmoniously connected", "consequent". In fine arts, the term "consistent" is used to refer to internally uniform art, design or architecture pieces, whose constituent elements are closely connected to one another, in terms of both subject matter and style, forming a harmonious, indivisible whole. A formal definition that probably best represents the idea of consistency applied in visual arts is that of a "connected space" used in topology. A topological space is said to be disconnected if it is the union of two or more disjoint non-empty sets [9]. Formally, a disconnected space is a union of two disjoint open or closed sets:

Definition 1. *Space (X, T) is connected if there are no non-empty sets $U, V \in T$ such that $X = U \cup V$, $U \cap V = \emptyset$. Subset $A \subset X$ is said to be connected if subspace $(A, T \mid A)$ is connected.*

In this paper, the term "visual consistency" is interpreted as a concept related to visual perception and understood as "visual uniformity". It is said that human perception starts with the grasping of shape, and then colour [8]. A shape may consist of a point - single, isolated; a line - a point in motion, moving vertically, horizontally or diagonally; or a plane figure - a closed geometrical shape or an open arrow- or cross-shaped figure [10]. These shapes may be integrated to form in an ordered, symmetrical whole, or arranged in a chaotic, random manner. Human perception is governed by the so called Gestalt Principles that describe how humans group objects using differentiating factors such as proximity, similarity or figure-ground relationship [11]. Colours, as the second most important perceptual data in graphic design may serve to expand the meaning of a piece of graphic design work by interacting with other colours and evoking certain emotions in the minds of the audience. Such means of expression, combined and used repeatedly throughout a piece of work, have a direct influence on visual consistency. The various means of expression - a consistent colour palette, unified typography, repetition of design elements - add up to a specific set of features that also accommodates the need for variation. When designing individual works of a graphic design series, it is important to maintain a balance between consistency and diversity to avoid excessive repeatability. However, there are no precisely defined values or ranges of variation beyond which the series ceases to appear as consistent.

3. Experiment

The experiment presented aims to verify the performance of an automated visual consistency assessment method as applied to a series of graphic design works. It employs a convolutional neural network (CNN) trained specifically for the purpose of this study. The results are compared with those obtained through subjective perceptual tests.

3.1. Visual data

The network was trained on a series of graphic works with a controlled variation degree obtained using the TuneVis generative graphic design system developed by Maja Garkowska. The system was implemented in the Processing environment using the Python language. According to the idea of generative art [12, 13], the system generates the graphics for a collection of music tracks according to a predefined rule-based procedure using a selection of data: the title of the album, title of the song, as well as key, length, tempo, acousticness and energy of the track. The tracks come from MusicOSet [14], a freely-available data set with enriched access to metadata, developed by Universidade Federal de Minas Gerais. Each music track is represented as a square graphic form comprising systematically distributed shapes - squares, circles, straight line segments, arcs - placed against a colourful gradient background. The parameters of particular graphic design elements depend on a set of data values and stochastic variables ascribed to them.

Three differentiating factors were applied: the colour palette, the selection of the graphic design elements and the ratio of positive to negative space. Each factor was assigned a three-step variation scale to allow orderly generation of graphics with a higher or lower degree of visual diversity, which enabled generating series of outputs that could be classified by the author as either consistent or inconsistent.

Each file generated by the system presents a set of graphic works arranged in a row, intended as visual representations of music album tracks. Figs.1-3 illustrate three series of graphics generated for the album titled “It’s About Time” comprising songs “Right Here”, “Put Your Hand In Mine”, and “Weak”, with the three-step variation scale applied to each of the differentiating factors.

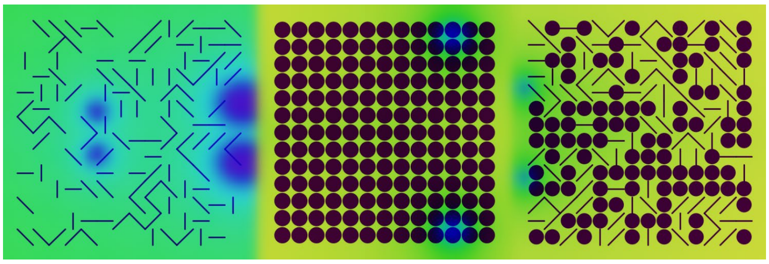


Figure 1. A series of graphic works for the album “It’s About Time” with the lowest variation degree

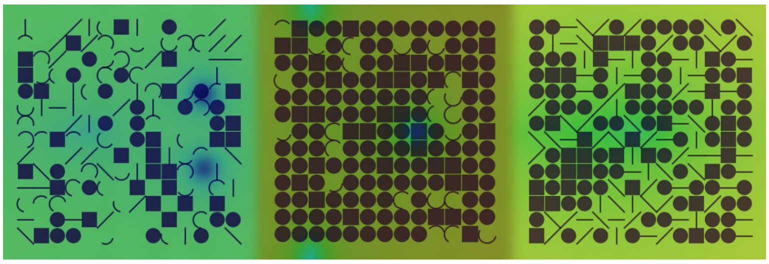


Figure 2. A series of graphic works for the album “It’s About Time” with an average variation degree

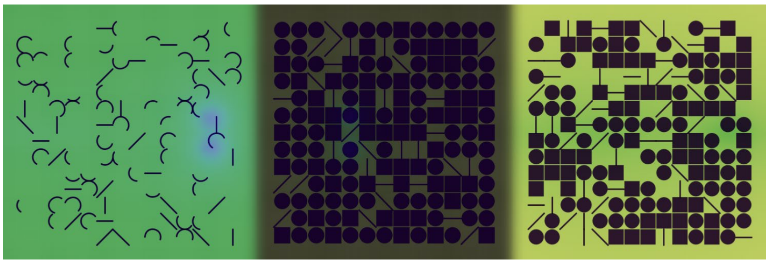


Figure 3. A series of graphic works for the album “It’s About Time” with the highest variation degree

The series of graphic designs, given as 32-bit PNG files, generated for the music albums, each containing a different number of tracks (2, 3, 4, 5, 7) were then used as training and validation data for the neural network.

3.2. Consistency assessment using a deep neural network

The classification of images in terms of visual consistency was performed using a convolutional neural network (CNN), a type of artificial neural network that has been successfully employed in the analysis of image features, such as shapes, edge patterns and texture information [15, 16]. The CNN classifier was implemented using the Python language and TensorFlow [17], an open source library dedicated to machine learning experimentation: building, training and implementing ML models, with a particular focus on deep neural networks.

The input data set consisted of 675 series of graphic works: those generated in a random manner and those that allow variation of only one or none of the features. Each series was classified in terms of visual consistency by a graphic design expert. The set was divided into three subsets - for training (437 series), validation (125 series) and testing (63 series).

The hyperparameters of the network were optimized, resulting in a 12-layer CNN architecture (Fig. 4). The Conv2D layer expects images (InputLayer) in 256x256 pixel format, with three colour channels. It has 16 filters of size 3x3 and stride of 1. ReLU is used as the activation function. In the MaxPooling2D layer, the feature map is reduced by selecting the maximum pixel value in each position. This is followed by more pairs of convolutional and pooling layers. The network also incorporates Dropout - a regularization technique that works by randomly deactivating a portion of neurons during each training update. This helps to prevent overfitting, which could impair the visual consistency assessment of subsequent graphics. In the Flatten layer, the output from the last pooling layer is converted to a single-dimensional vector and input to the Dense layers for classification.

For final configuration, which precedes the training of the model, a stochastic gradient descent (SGD) was used. Binar Cross Entropy was applied as a loss function to measure the difference between predicted binary outcomes and actual binary labels. It is the most common loss function applied to binary classification problems. The model's classification performance was monitored using the accuracy metric.

The training process took 20 epochs on the training and validation subsets, resulting in a relatively small training and validation loss. Thus, there is no indication of either overfitting or underfitting of the model [18]. Over the last epochs, the loss function seemed to stabilize, which implies that the model was optimized (Fig. 5).

The network's classification accuracy was tested on data previously unseen by the model. The following metric values were achieved: accuracy - 0.97 (a general measure of classification performance for both classes, including both correctly classified positive and negative instances), precision - 0.81 (the portion of positive instances correctly classified among the total number of instances classified as positive), sensitivity - 0.89 (a measure of how many positive instances the model was

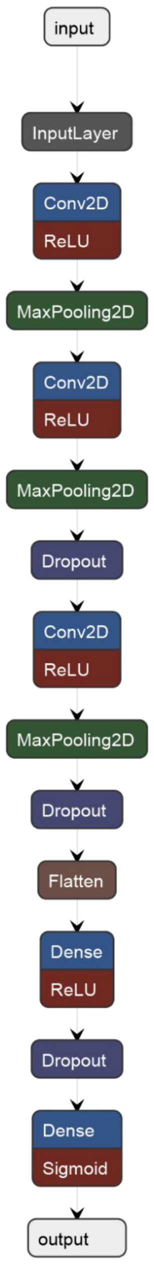


Figure 4. The proposed CNN model architecture

able to correctly identify). These results demonstrate a good ability of the model to correctly identify consistent series of images, with a slightly lower precision score

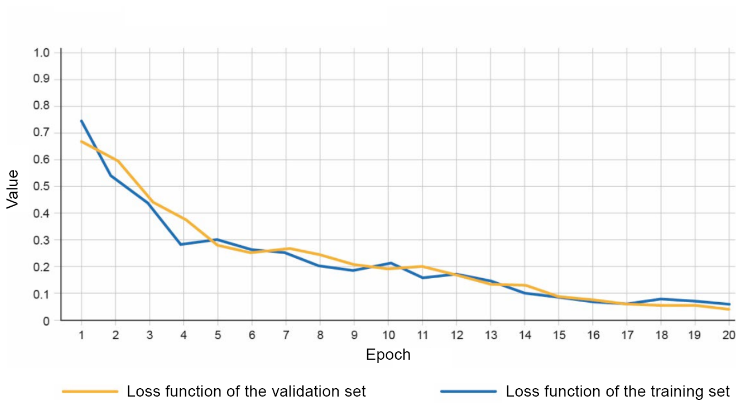


Figure 5. Training and validation loss over epochs

indicating that the model can sometimes misclassify a negative (inconsistent) sample as positive.

3.3. Perceptual tests

The experiment included perceptual tests which aimed to compare the classification results obtained by the neural network with the opinions of graphic design practitioners - the domain-specific knowledge and experience of the observers is a critical factor in obtaining reliable and valid results. The visual consistency assessment undertaken by both the CNN model and domain specialists was performed on the same set consisting of 32 series of graphics, 8 for each of the 4 albums, with a different variation degree. A questionnaire was prepared, consisting of the introductory part, including the description of the task, the average time required to complete the questionnaire and explanation of the term “visual consistency” to make sure it has the same meaning for all respondents, and the main body including a set of questions regarding the visual consistency of the series of graphics under examination. A 1-5 rating scale was applied, providing the respondents with the following answer options: “Very consistent”, “Consistent”, “Hard to assess”, “Inconsistent”, “Very inconsistent”. The display order was the same for each respondent.

The questionnaire was completed by 24 respondents (12 women and 12 men) with academic (19 of those questioned) and/or professional background (15 of those questioned) in the field of visual arts. The average age of the respondents was 27 (ranging from 21 to 49). The average length of professional experience among those interviewed was 5 years.

The average amount of time that respondents took to complete the questionnaire was 4 minutes 37 seconds, which means an average of 8.65 seconds per one series of graphic works. In the subjective evaluation of visual consistency, for only 7 out of the 32 series the agreement among the respondents reached a level of at least 75 %. A detailed distribution of the perceptual evaluation results for the albums under examination can be seen in Fig. 6. To enable numerical analysis of the results, the answer options “Very consistent”, “Consistent”, “Hard to assess”, “Inconsistent”, “Very inconsistent” were assigned values from 5 to 1, respectively. The calculation of an average assessment result, variance and median for each graphic work was followed by the two-factor analysis of variance with replication. The sum of squares within the groups confirms that the respondents differ in their assessment of the same graphics.

4. Analysis of the results and discussion

4.1. Analysis of the results

The results of CNN prediction and respondents’ evaluation of the 32 graphic design series generated for 4 albums using different parameters are summarised in the histograms below (Fig. 7). It can be seen that more than half of the responses are comparable to each other, with a maximum difference of 0.2. However, if the data are compared in binary terms: consistent / inconsistent, the distribution will improve significantly. For this purpose, values equal to or higher than 0.5, representing samples identified as “consistent”, were assigned a rank of 2, whereas values lower than 0.5, representing samples identified as “inconsistent”, were assigned a rank of 1. This comparison is presented in the form of a cumulative bar chart, shown in Fig. 8. The bars of equal distribution represent the graphics for which the consistency assessment results obtained from the respondents are identical with the CNN predictions. The bars of unequal distribution - showing a dominance of either respondents’ evaluation or CNN prediction - represent the graphics for which the results were not compatible. As can be seen from the chart, the results of CNN prediction differed from those of respondents’ evaluation in only 6 cases, which means a compatibility rate of over 80%. The mean squared error obtained is 0.03 - this indicates a low standard error of the model, signifying better performance in terms of consistency prediction. Spearman’s rank correlation coefficient equal to 0.72 indicates a close alignment between the model’s results and the respondents’ ratings. The correlation coefficient equal to 0.69 indicates a close association between the classes. The values of the statistical measures indicate a strong agreement between the neural network’s predictions and respondents’ evaluation outcomes, which indicates that the model can be effectively applied to visual consistency assessment in graphic design.

The distribution of variation degrees across the three differentiating parameters in the graphics identified as consistent by the expert, the CNN model and the

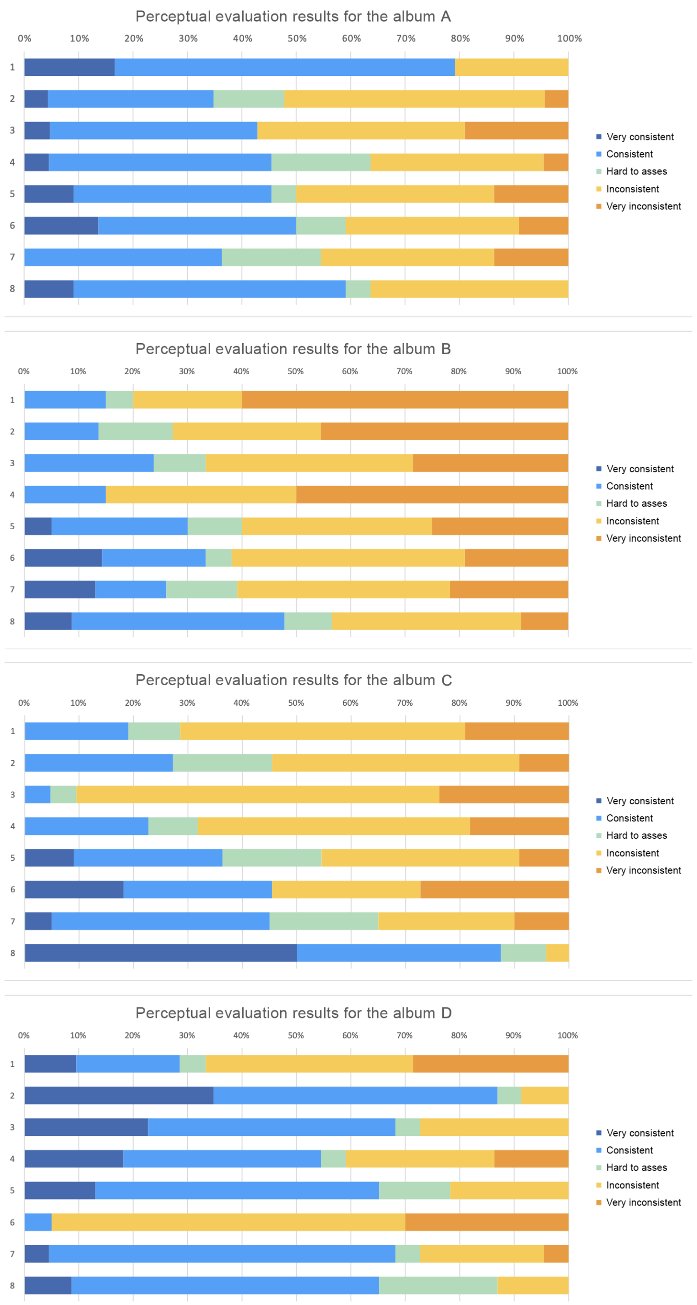


Figure 6. Perceptual tests results for albums A, B, C, D. The bar chart represents the percentage split of consistency assessment results for each series of graphic works

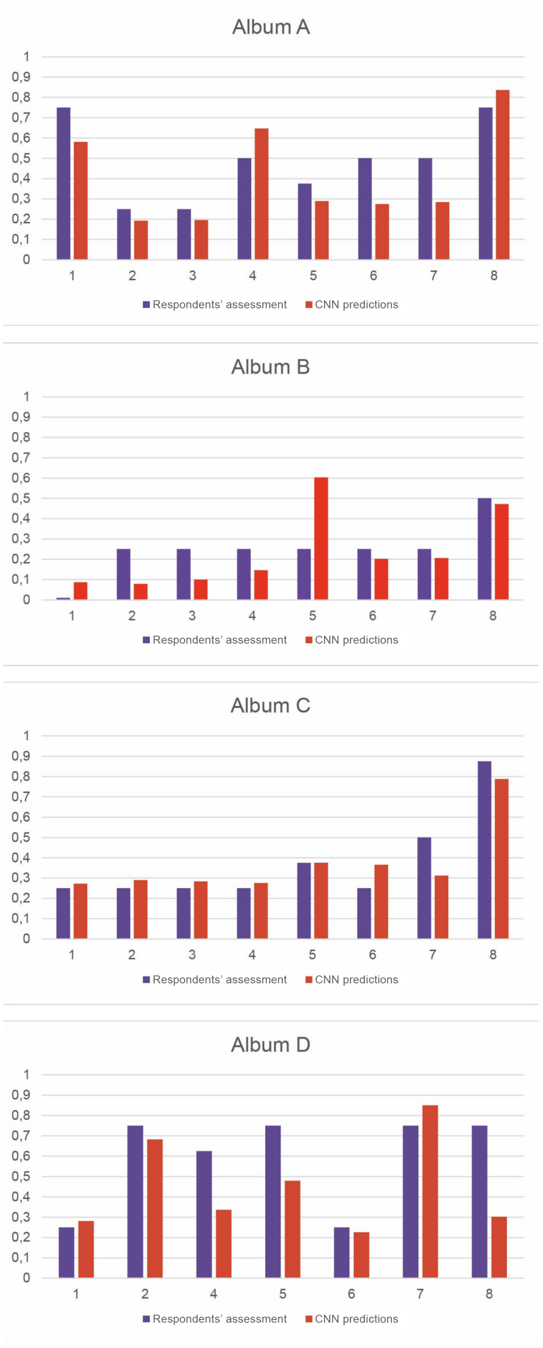


Figure 7. Comparison of visual consistency assessment results for albums A, B, C, D (median of respondents' assessments, CNN predictions, in 0-1 scale)

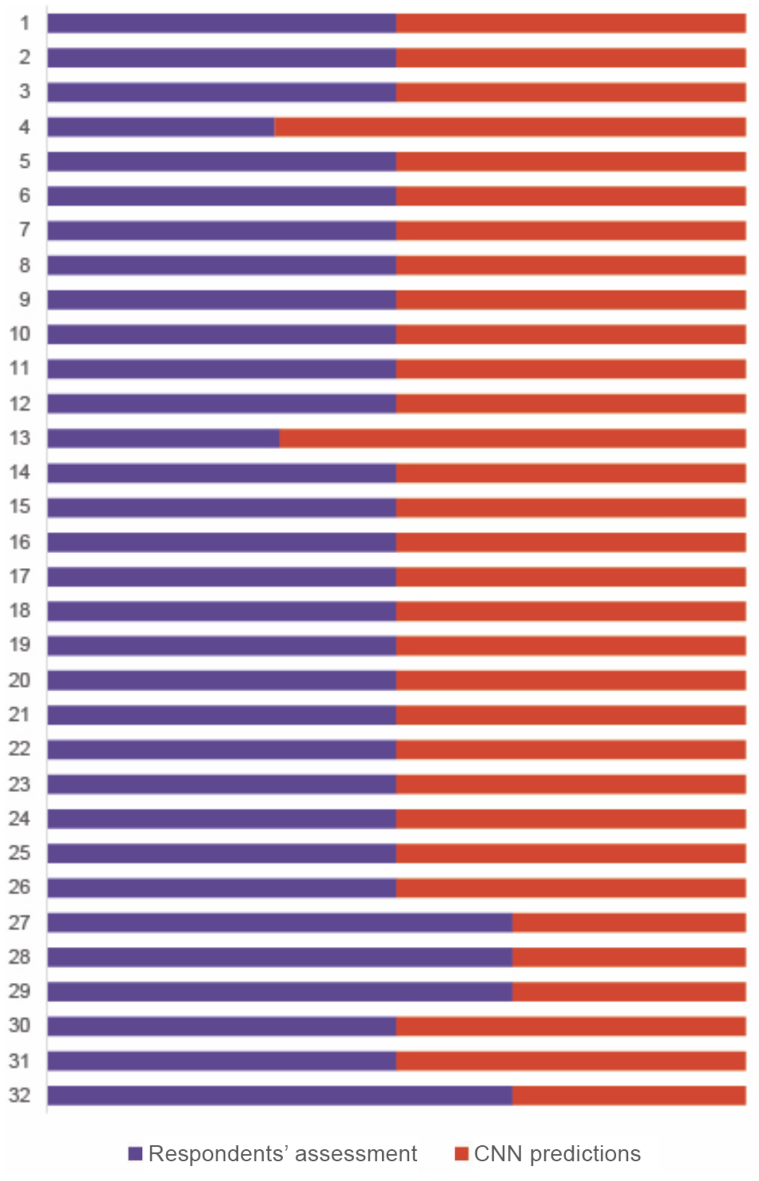


Figure 8. Cumulative results of consistency assessment (respondents' evaluation, CNN prediction)

respondents is shown in Fig. 9. The vertical axis represents the distribution of percentage of each parameter by variation degree.

As predicted, the majority of the samples identified as consistent are those

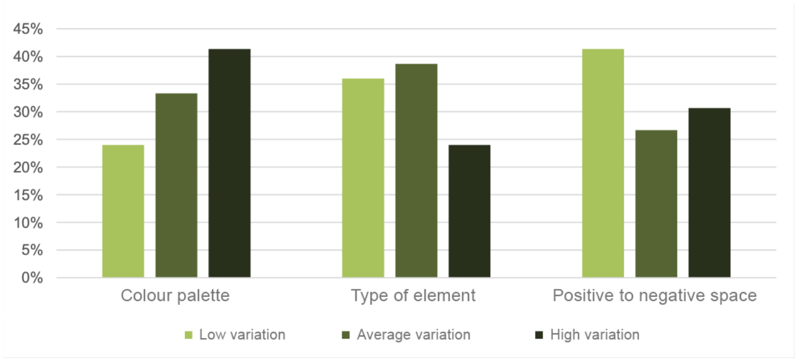


Figure 9. The distribution of variation degrees (low, average and high) across three differentiating parameters (colour palette, type of graphic design elements and ratio of positive to negative space) in samples identified as consistent

with the lowest variation degree related to the ratio of positive to negative space. Surprisingly, the images identified as consistent have also a very high degree of variation related to the colour palette. Neither of the variation degrees adopted appears as dominant among all three differentiating parameters. The cumulative variation degree distribution also does not provide any clear indication - the lowest variation degree can be observed in 34%, average - in 33%, and highest - in 32% of the samples.

4.2. Discussion

The presented results of automated visual consistency assessment were obtained using a CNN model trained on visually similar data. Undoubtedly, the model's performance could improve if it was trained on a more numerous and diverse data set.

Due to the different distribution of parameter values observed among the samples identified as consistent, both by the CNN model and by the human evaluators, it was impossible to determine a clear consistency boundary. The set of elements to choose from - two plane shapes and two lines - can influence the background proportions, due to the obvious difference in the space occupied by a filled square and a thin line.

The CNN predictions differed from the respondents' ratings in 19% of all assessment results. This discrepancy could be attributed to an error in the model's performance or misjudgement of one of the respondents, which influenced the median. Moreover' the process of perceptual testing could have been influenced by the fact that the graphics were always presented to the observers in the same order. The presentation of samples in groups while changing the display order might have improved the reliability of the results.

5. Conclusions and further research

The findings of this study suggest that ML-powered tools may enhance significantly the process of visual consistency verification, which constitutes a vital aspect of generative graphic design. Even with a relatively small set of training data, the results provided by the CNN model demonstrated a 81% compliance with those obtained through subjective perceptual evaluation.

Due to the interconnectedness of two differentiating factors, namely the type of design elements and the ratio of positive to negative space, resulting in variation of these factors' values among the series classified as consistent, it was impossible to draw a clear distinction between "consistent" and "inconsistent". As often observed in graphic design practice, different visual elements interact with one another and form relationships that are perceived as related. Perhaps these connections should be identified at an early stage of analysis and considered as a single, complex differentiating factor.

This study makes an original contribution to research on visual consistency assessment, which might lead to further developments in this field. In future studies, it would be interesting to investigate a wider range of discriminating factors, such as changing composition patterns and typographical design elements.

Acknowledgment

The authors would like to express their deepest gratitude to Professor Lesław Miśkiewicz of the Strzemiński Academy of Fine Arts in Łódź for providing an expert opinion on the consistency of the training data set, and for his invaluable, constructive feedback and insightful comments throughout this research endeavour.

References

[1] Shi, Y., Gao, T., Jiao, X., and Cao, N. Understanding design collaboration between designers and artificial intelligence: A systematic literature review. *Proc. ACM Hum.-Comput. Interact.*, 7(CSCW2), 2023. doi: 10.1145/3610217. URL `https://doi.org/10.1145/3610217`.

[2] Wang, G., Qin, Z., Yan, J., and Jiang, L. Learning to select elements for graphic design. In *Proceedings of the 2020 International Conference on Multimedia Retrieval*, ICMR '20, page 91–99. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450370875. doi: 10.1145/3372278.3390678. URL `https://doi.org/10.1145/3372278.3390678`.

[3] Guo, S., Jin, Z., Sun, F., Li, J., Li, Z., Shi, Y., and Cao, N. Vinci: An intelligent graphic design system for generating advertising posters. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21. Association for Computing Machinery, New York, NY, USA, 2021. ISBN 9781450380966. doi:10.1145/3411764.3445117. URL `https://doi.org/10.1145/3411764.3445117`.

[4] Liu, Y. Design of graphic design assistant system based on artificial intelligence. *Int. J. Inf. Technol. Syst. Appoach*, 16(3):1–13, 2023. ISSN 1935-570X. doi:10.4018/IJITSA.324761. URL `https://doi.org/10.4018/IJITSA.324761`.

[5] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, 2020. ISSN 0001-0782. doi:10.1145/3422622. URL `https://doi.org/10.1145/3422622`.

[6] Yang, B. Perceptual similarity measurement based on generative adversarial neural networks in graphics design. *Appl. Soft Comput.*, 110(C), 2021. ISSN 1568-4946. doi:10.1016/j.asoc.2021.107548. URL `https://doi.org/10.1016/j.asoc.2021.107548`.

[7] Mahajan, R. and Shneiderman, B. Visual and textual consistency checking tools for graphical user interfaces. *IEEE Trans. Softw. Eng.*, 23(11):722–735, 1997. ISSN 0098-5589. doi:10.1109/32.637386. URL `https://doi.org/10.1109/32.637386`.

[8] Arnheim, R. *Art and Visual Perception, Second Edition: A Psychology of the Creative Eye*. Art Psychology. University of California Press, 2004. ISBN 9780520243835. URL `https://books.google.pl/books?id=9RktoatXGQ0C`.

[9] Błaszczyk, A. *Topologia*. Wydawnictwo Naukowe PWN, Warszawa, 2023. ISBN 978-83-01-23173-6. doi:https://doi.org/10.53271/2023.107.

[10] Frutiger, A. *Człowiek i jego znaki*. d2d.pl, Kraków, 2022. ISBN 978-83-959016-6-9.

[11] Ellis, W. *A Source Book of Gestalt Psychology*. The Gestalt Legacy Press seminal series. Gestalt Legacy Press, 1938. ISBN 9781892966001. URL `https://books.google.pl/books?id=kvjXAAAAMAAJ`.

[12] Składanek, M. *Sztuka generatywna. Metoda i praktyki [Generative art. Method and practices]*. SZTUKA/MEDIA/KULTURA. Wydawnictwo Uniwersytetu Łódzkiego, 2017. ISBN 978-83-8088-402-1.

[13] Galanter, P. What is generative art? complexity theory as a context for art theory. In *In GA2003–6th Generative Art Conference*. 2003.

[14] Silva, M. O., Rocha, L. M., and Moro, M. M. Musicoset: An enhanced open dataset for music data mining. In *XXXIV Simpósio Brasileiro de Banco de Dados: Dataset Showcase Workshop, SBBD 2019 Companion*, pages 408–417. 2019. ISSN 2016-5170.

[15] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 1106–1114. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[16] Ramprasath, M., Anand, M. V., and Hariharan, S. Image classification using convolutional neural networks. *International Journal of Pure and Applied Mathematics*, 119(17):1307–1319, 2018.

[17] Vishal, B. L., Pragathi, M., Amulya, P. K., and Kumar, N. S. Image classification using neural networks and tensor-flow. *Test Eng. Manage*, 83:20087–20091, 2020.

[18] Ce, P. and Tie, B. An analysis method for interpretability of cnn text classification model. *Future Internet*, 12(12):228, 2020.

Optimizing the GMRES Implementation for Solving Multiple Large-Scale Matrix Systems of Equations

Wojciech Jakieła[0009–0008–2311–3776],
Bartosz Czernic-Goławski[0009–0004–7327–2119],
Korneliusz Dyszczyński[0009–0004–7219–7532]

Warsaw University of Technology
Department of Electrical Engineering
plac Politechniki 1, 00-661 Warszawa, Poland

Abstract. *The project described and compared implementations of procedures for solving large systems of linear equations arising from the solution of partial differential equations. The implementations were modelled on the GMRES method. The research focused on improving the efficiency of the calculations performed, ensuring computational effectiveness and accuracy.*
Keywords: *CSC format, GMRES, Julia, sparse matrix, matrix systems of equations, differential equations, finite element method, finite element analysis*

1. Introduction

Differential equations serve as indispensable instruments in the realm of mathematical modeling across various domains of science and technology. Their utility extends across the disciplines of physics, chemistry, biology, and medicine. This exposition is dedicated to the meticulous exploration of solving extensive linear matrix equation systems, stemming from the resolution of differential equations through the finite element method (FEM). The finite element analysis (FEA) technique has gained pervasive adoption in multifarious sectors, encompassing aerospace, automotive, nuclear engineering, and fluid mechanics, among others, as underscored by Dhatt [1].

Within the scope of the research elucidated herein, it is of paramount significance to acknowledge that the matrices representing the linear equations derived from the solutions of partial differential equations via the FEM method exhibit a sparsity characteristic. This inherent sparsity emanates from the intrinsic nature of the relationships between the unknowns within an individual equation of a given system, which predominantly engender connections with a sparse subset of other unknowns. The crux of securing an efficacious solution to an expanded equation system transcends the mere attainment of a veracious outcome, one that aligns with the presumed precision. It also necessitates meeting performance criteria. In contemporary digital age, the voluminous surge in data volumes and the burgeoning complexity of computational tasks underscore the imperative for the tools at hand to not only meet stringent quality benchmarks and predefined temporal constraints but also to curtail extravagant resource consumption.

In light of these considerations, the experiments conducted within this research endeavor were rigorously designed to discern an apt method for resolving matrix equation systems. Subsequently, the emphasis was placed on crafting a meticulous and resource-efficient implementation of the chosen method.

2. Overview of available methods

In the extensive review of the pertinent literature concerning the subject at hand, an array of methodologies employed in addressing matrix systems of equations has been unveiled. The strategies elucidated for the resolution of linear equation systems inherently furnish solutions of an approximate nature. The systematic construction of a series of said approximate solutions is recognized as an iterative procedure. It is worth noting that all three methodologies expounded within this passage adhere to an iterative paradigm. Diverging from their direct method counterparts, which aim to achieve an exact solution within a finite number of operations and are thereby susceptible to approximation errors, these iterative approaches commence with an initial estimate and systematically engender progressively refined approximations, constituting an infinite sequence converging to the precise solution.

2.1. Jacobi method

In the domain of numerical linear algebra, Jacobi's method stands as a prominent algorithm designed to determine solutions for systems of linear equations, specifically those exhibiting diagonal dominance. This method entails a systematic procedure where each diagonal element is individually solved, and an initial approximation is incorporated. The iterative cycle is then repeated until convergence is achieved. Notably, Jacobi's method adheres to a protocol where all values of the unknowns are updated in each iteration prior to any incorporation of new information in the calculation process, as documented in [2].

However, it is imperative to acknowledge that the Jacobi method exhibits suboptimal convergence, particularly when applied to large matrices. This sluggish convergence necessitates a considerable number of iterations for completion, as elucidated in [3]. Consequently, earnest endeavors have been undertaken to enhance the efficacy of this method. Some of these enhancements include the introduction of pre-conditioning techniques, commonly referred to as 'squeezing,' or the development of innovative approaches aimed at deriving parameters conducive to the Jacobi method. A noteworthy example of such refinement is the 'Scheduled Relaxation Jacobi' (SRJ) method, expounded upon in [4].

2.2. Gauss-Seidel method

The Gauss-Seidel method represents a modification of Jacobi's method, introducing the practice of referencing the most recently computed, and thus, the more accurate values of the solution for the system of equations during successive iterations. While this method tends to converge more swiftly than Jacobi's approach in many cases, its convergence is often characterized as too sluggish to serve as a standalone method for solving large systems of matrix equations. Nevertheless, it is essential to recognize that both of these methods offer distinct advantages. Notably, the avoidance of computationally intensive scalar products stands as one such advantage.

Hence, there exist specific domains where both of the methods introduced above find applicability. An illustrative instance can be found in computer graphics. Within this context, a randomized parallel method, known as 'Vivace,' has been introduced to address sparse linear systems with a rate of convergence akin to that of the Gauss-Seidel method, as detailed in the referenced paper [5].

The convergence of the Gauss-Seidel method can further be expedited through the introduction of a relaxation parameter, giving rise to the method referred to as 'Successive Overrelaxation' (SOR). The utilization of the generalized SOR (GSOR) approach is exemplified in the work of [6].

2.3. GMRES method

The primary focus of this document is the *Generalized Minimal Residual* (GMRES) method, which belongs to the family of Krylov iterative methods [7] and possesses the unique capability to adapt autonomously to the properties of the matrix. Detailed construction and convergence rate estimation can be found in [8]. The method's name is derived from its objective to minimize the residual, measured in the Euclidean norm, at the k-th iteration. As previously explained, our goal is to solve the matrix equation system $Ax = b$. Denoting r as the residual, x_0 as the initial approximation, and K_k as the Krylov space the aim to achieve is:

$$||r_k||_2 = ||b - Ax_k||_2 \leq ||b - Ax||_2 \quad \forall x \in x_0 + K_k \tag{1}$$

The minimization task has a unique solution. Let V_k be the matrix with columns forming a basis for K_k, and let $x_k = x_0 + V_k a_k$, where a_k is the solution to the least squares problem with respect to $a \in \mathbb{R}^{dimK_k}$. Thus:

$$||r_0 - AV_k a||_2 = min! \tag{2}$$

In practice, to efficiently solve the minimization problem, techniques such as orthogonalizing the basis of the Krylov space using the Arnoldi method are commonly employed [3]. It is noteworthy that until the solution with the desired accuracy (which can be determined through the analysis of the residual, r_k) is attained, there is no need to compute either a_k or x_k. Consequently, the cost of one GMRES iteration is of order $O(kN)$, rather than $O(k^2)$. GMRES does not require matrix symmetry or positive definiteness, making it a versatile choice. To minimize resource consumption at the expense of convergence speed, a restarted version is sometimes utilized - employing the approximate initial solution obtained after a fixed number of iterations.

3. Algorithm implementation and testing

In developing the solution, primary focus was to maximize computational efficiency. Given the inherent characteristics of the problem at hand, the Compressed Sparse Column (CSC) Matrix Storage format was adopted. [9]. An alternative version employing the standard matrix format was also implemented. Notably, we eliminated a conventional stopping condition that terminated the algorithm upon minor changes in the iteration results. Initial tests indicated a substantial increase in both memory consumption and processing time due to result calculation in every iteration. Subsequent modifications led to the final result being computed after a specified number of iterations, contingent upon the coefficient matrix's size. We are actively researching an equation that characterizes the optimal number of iterations for this process.

3.1. CSC sparse matrix format

The implementation operating on matrices in the CSC format for sparse coefficient matrices is expected to yield a noticeable computational advantage over calculations performed using the standard matrix format. Experiments involved varying sizes of sparse coefficient matrices, and the impact of CSC on both computation time and memory usage was examined. These tests were conducted under a fixed number of iterations, and the results were averaged for robust analysis. In these experiments scrutinized the equivalence of outcomes between both matrix formats.

As illustrated in Figure 1, a significant reduction in computation time is observed when employing the CSC format, with only a negligible memory overhead. Within the examined range of coefficient matrix sizes, it is possible to transform the computational complexity from exponential to near-linear. The average execution times for various matrix sizes are summarized in Table 1. In the case of small matrices, such as 40×40, a minor time overhead was noted. However, for all other cases, execution times were reduced by 35% to as much as 87%. The advantages of

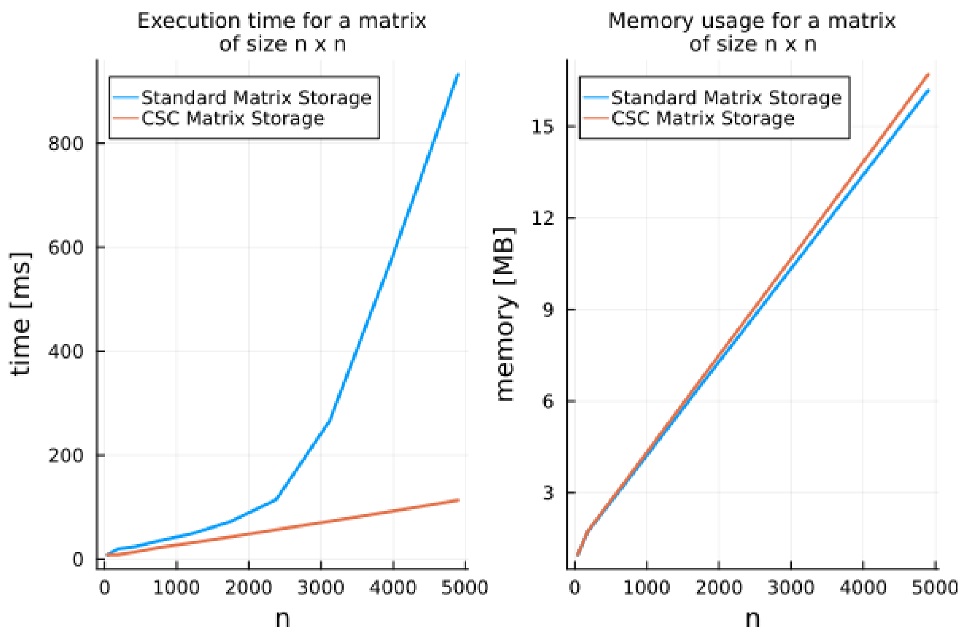


Figure 1. Comparison of computational and memory complexity of methods with and without CSC format

the CSC format became more pronounced as the coefficient matrix size increased. Consequently, the CSC-based method was selected for further experimentation.

Table 1. Equation solution times depending on the matrix format used and the size of the coefficient matrix (n × n).

n	**time for std format [ms]**	**time for CSC format [ms]**	**difference [%]**
40	8.35	8.43	+0.97
180	19.64	8.45	-56.97
420	24.18	13.66	-43.51
760	35.44	22.61	-36.20
1200	48.64	31.29	-35.67
1740	72.11	42.6	-40.92
2380	114.61	56.79	-50.45
3120	266.0	72.92	-72.59
3960	570.28	91.6	-83.94
4900	932.74	113.34	-87.85

3.2. Optimal number of iterations

Experimental investigations were undertaken to derive a mathematical equation elucidating the ideal number of iterations relative to the size of the coefficient matrix. In the course of these experiments the number of iterations were ascertained at which the mean squared error (MSE), as stipulated in reference [10], surpassed the threshold of 1e-9. The reference vector of solutions was obtained from the 'IterativeSolvers.gmres' function [11]. Utilizing the amassed dataset, the polynomial equation that best approximated the data in a least-squares sense was computed, as outlined in reference [12]. The results are graphically presented in Figure 2.

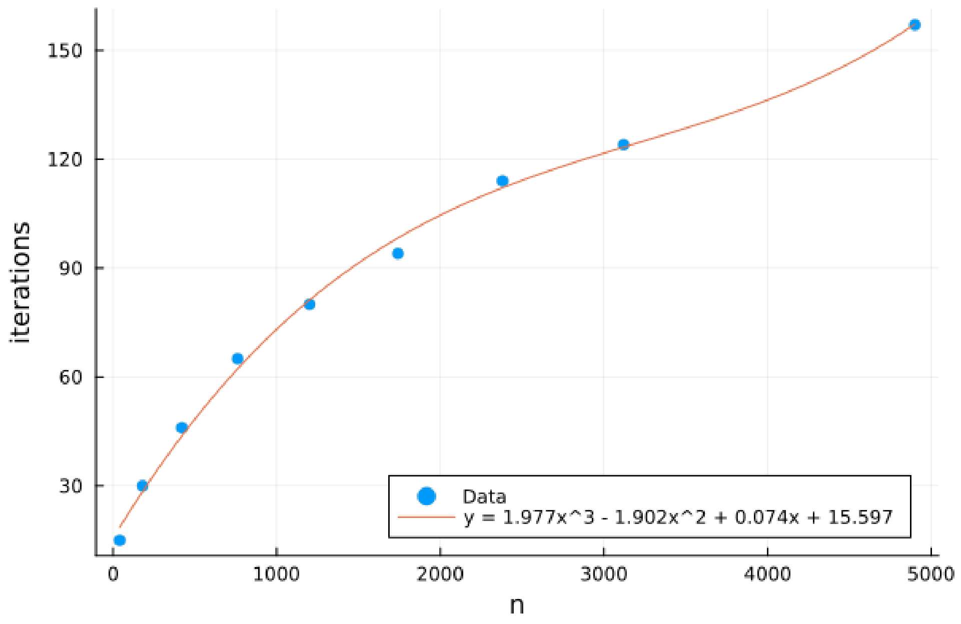


Figure 2. The number of iterations needed to achieve MSE with IterativeSolvers.gmres at 1e-9 and the fitted third-degree polynomial. Measurements for matrices of size $n \times n$.

After determining the polynomial equation, our method was modified to terminate calculations after completing a number of iterations equal to $y(n)$, where y represents the polynomial equation derived, and n is the size of the coefficient matrix, denoted as $n \times n$.

One drawback of this approach is the rapid and potentially unjustified escalation in the number of iterations for matrices larger than those used in the modeling process. This issue is discernible in Figure 3. A potential solution may involve fine-tuning the model for larger matrices by identifying a new polynomial equation. Alternatively, for matrices beyond the scope of the model's data, a linear regression model based on the most recent samples can be applied.

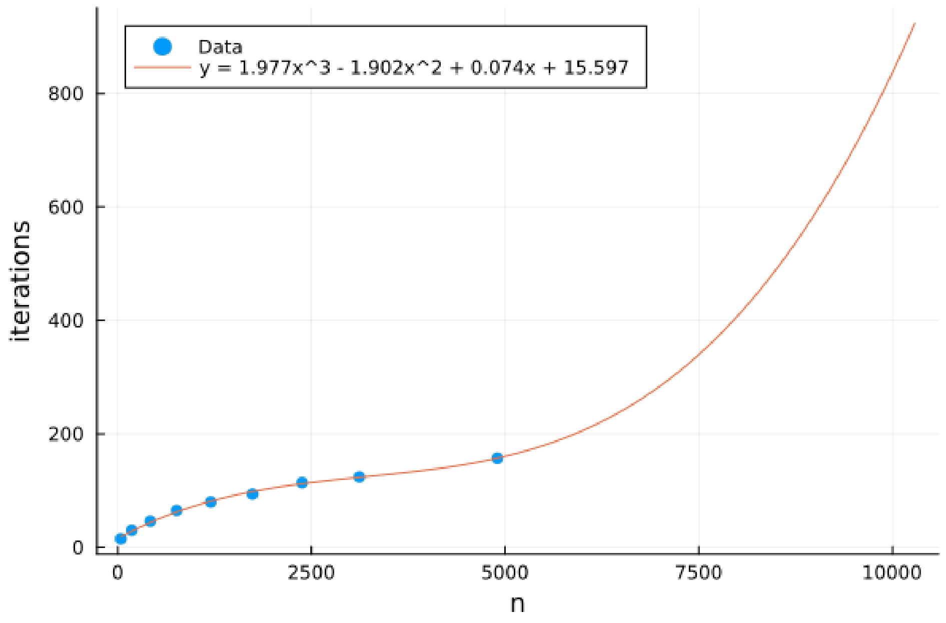


Figure 3. Rapid growth of the calculated polynomial outside the known dataset

3.3. Comparison of method accuracy

The accuracy of the implemented method variants was rigorously examined by means of a comparison of the mean squared error (MSE). The reference solution vector was derived from the 'IterativeSolvers.gmres' method, which was considered the gold standard. Three distinct variants were tested and designated as GMRes, EpsGMRes, and AutoGMRes.

In succession evaluated an implementation with a fixed, manually specified number of iterations (GMRes), an implementation with a maximum number of iterations coupled with an epsilon ϵ-stop condition (EpsGMRes), and an implementation that automatically determined the number of iterations based on the polynomial model presented in the 'Optimal number of iterations' section. Notably, the number of iterations was set to 100 for both the GMRes and EpsGMRes methods, and ϵ for the EpsGMRes method was set at 1e-5.

The results comprehensively summarized in Table 2.

Across all matrix sizes, the AutoGMRes method consistently exhibited the smallest average MSE, demonstrating the robustness of this approach over the entire matrix size range. Moreover, the solution vectors produced by the AutoGMRes method exhibited a remarkable degree of consistency with the 'IterativeSolver.gmres' method, surpassing a precision of ϵ=1e-3. Such a high level of consistency was not observed for the other two methods, GMRes and EpsGMRes, which displayed reduced efficiency for matrices of extreme size. In particular, the GMRes method, with its manually preset fixed number of iterations, exhibited the least computational precision among the three evaluated variants.

Table 2. MSE against IterativeSolvers.gmres for different sizes of n x n matrices

n	**MSE**		
	GMRes	**EpsGMRes**	**AutoGMRes**
40	1.014e-2	2.650e-1	1.903e-8
180	7.250e-1	1.509e-11	1.292e-8
420	5.267e-1	1.318e-10	1.724e-7
760	1.489e-11	1.087e-10	4.660e-8
1200	5.573e-11	9.980e-10	9.980e-10
1740	1.824e-10	6.960e-10	2.811e-10
2380	2.000e-7	2.000e-7	7.733e-10
3120	1.262e-5	1.262e-5	4.994e-10
3960	2.410e-5	2.410e-5	1.262e-8
4900	1.608e-4	1.608e-4	9.328e-10
Średnia	1.262e-1	2.652e-2	2.671e-8

3.4. Performance and memory comparison

The GMRes, EpsGMRes, and AutoGMRes methods, as elucidated in the preceding section, underwent a comprehensive evaluation encompassing performance and memory utilization. To establish a benchmark, the 'IterativeSolvers.gmres' method was selected. Comprehensive results are depicted in Figures 4 and Tables 3 and 4.

Upon a thorough examination of the results presented in Figure 4, as well as Tables 3 and 4, a conspicuous trend emerges, underscoring the temporal benefits achieved over the reference 'gmres' method from the 'IterativeSolvers' package. This temporal advantage is most pronounced in the case of the 'AutoGMRes' method, evident across nearly all sizes of the coefficient matrix. Notably, the average execution time was halved in comparison to the 'IterativeSolvers.gmres' method. It is important to note, however, that 'AutoGMRes' lags behind 'gmres' from 'IterativeSolvers' in terms of memory complexity.

Of the methods under evaluation, 'EpsGMRes' emerged as the least performant, exhibiting disparities in both time and memory complexity in comparison to

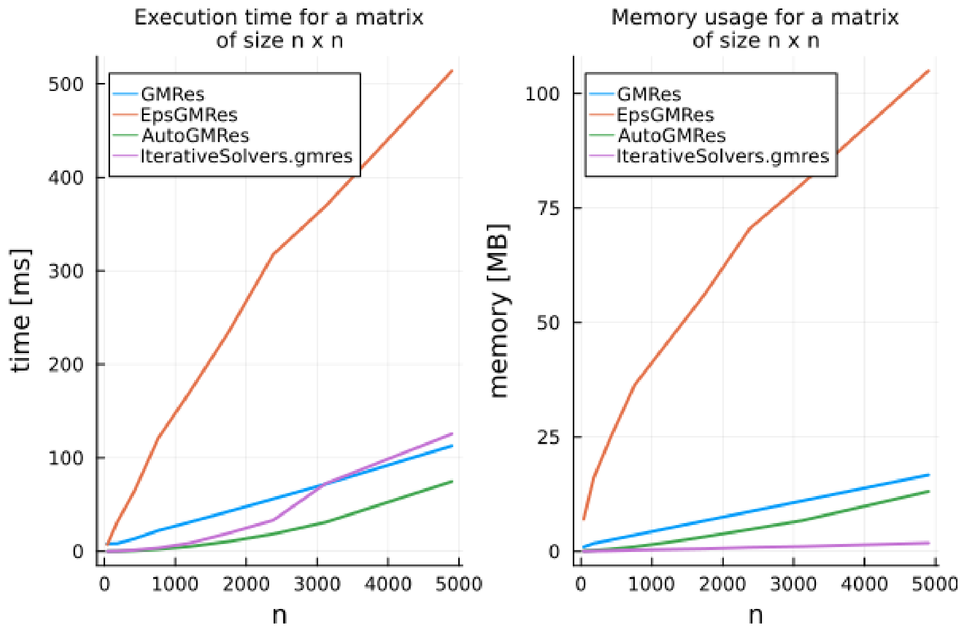


Figure 4. Comparison of calculation time and memory consumption for the different methods as a function of the size of the $n \times n$ coefficient matrix

Table 3. Execution times of individual methods depending on the size of the input matrix

n	**Execution time [ms]**			
	GMRes	**EpsGMRes**	**AutoGMRes**	**Iterative Solvers .gmres**
40	8.3024	8.0837	0.0272	0.0153
180	8.3164	32.057	0.1102	0.2702
420	13.384	64.576	0.6580	1.3631
760	22.455	119.16	2.2196	3.6611
1200	31.330	171.33	5.1355	8.8781
1740	42.670	230.98	10.505	19.744
2380	56.385	317.53	18.757	30.396
3120	72.440	366.15	31.500	65.476
3960	90.867	460.17	48.719	102.60
4900	113.21	512.53	75.768	116.86
Average	45.937	228.26	19.340	34.927

Table 4. Memory consumption of individual methods depending on the size of the input matrix

n	**Memory usage [MB]**			
	GMRes	**EpsGMRes**	**AutoGMRes**	**Iterative Solvers .gmres**
40	0.9707	7.0544	0.1071	0.0174
180	1.7493	16.063	0.2077	0.0835
420	2.5018	24.941	0.4874	0.1523
760	3.5711	36.413	1.0190	0.2985
1200	4.9687	45.238	1.9159	0.4201
1740	6.6869	56.056	3.1854	0.5690
2380	8.7049	70.510	4.8015	0.8704
3120	11.046	80.277	6.7428	1.0751
3960	13.713	96.078	9.2947	1.3071
4900	16.706	104.99	13.095	1.8314
Average	7.0619	53.762	4.0857	0.6625

its counterparts due to need to storing and comparing previous solution and calculating error. Regrettably, it failed to provide any discernible advantages over the other methods.

4. Conclusions

The experiments were meticulously designed with the principal objective of optimizing the algorithm for solving systems of partial differential equations in a manner that significantly enhances execution time. To achieve this, the utilization of the Compressed Sparse Column (CSC) sparse matrix storage format was employed, yielding substantial gains in execution time, all the while maintaining memory efficiency. A further refinement entailed the replacement of the conventional stop condition with the automated determination of the optimal number of iterations, a process informed by the determined polynomial. Consequently, the cumulative optimizations not only surpassed the execution time achieved by the reference method ‘IterativeSolvers.gmres’ without losing computational precision on the matrices subjected to testing, but in some cases even reduced it by half.

Acknowledgment

The authors would like to thank Doctor Bartosz Chaber and Doctor Damian Pęszor for introducing them to the problem, inspiring their work in this field, and preparing a solid foundation for research.

References

[1] G. Dhatt, G. T., E. Lefrançois. Finite element method. *Wiley-ISTE 1st ed.*, 2012.

[2] Karunanithi, D. S., Gajalakshmi, N., Malarvizhi, M., and Saileshwari, M. A study on comparison of jacobi, , gaussseidel and sor methods for the solution in system of linear equations. *International Journal of Mathematics Trends and Technology*, 56, 2018.

[3] P. Krzyżanowski, L. P. Matematyka obliczeniowa ii, 2013.

[4] Adsuara, J. E., Cordero-Carrión, I., Cerdá-Durán, P., and Aloy, M. A. Scheduled relaxation jacobi method: Improvements and applications. *Journal of Computational Physics 321*, 2016.

[5] M. Fratarcangeli, F. P., V. Tibaldo. Vivace: a practical gauss-seidel method for stable soft body dynamics. *ACM Transactions on Graphics*, 35, 2016.

[6] D. K. Salkuyeh, V. E., D. Hezari. Generalized sor iterative method for a class of complex symmetric linear system of equations. *International Journal of Computer Mathematics*, 92, 2015.

[7] Brown, P. N. A theoretical comparison of the arnoldi and gmres algorithms. *SIAM Journal on Scientific and Statistical Computing*, 12, 1991.

[8] Saad, Y. Iterative methods for sparse linear systems, 2003.

[9] Julia sparse arrays. URL `https://docs.julialang.org/en/v1/stdlib/SparseArrays/`.

[10] Hodson, T. O., Over, T. M., and Foks., S. S. Mean squared error, deconstructed. *Journal of Advances in Modeling Earth Systems 13.12*, 2021.

[11] Julia sparse arrays. URL `https://docs.juliahub.com/IterativeSolvers/ef2NV/0.8.5/linear_systems/gmres/`.

[12] Gavin, H. P. The levenberg-marquardt algorithm for nonlinear least squares curve-fitting problems. 2019.

IF-THEN Rules in Searching Outliers

Marcin Kacprowicz[1[0000–0003–4381–1359]],
Monika Bartczak[1[0000–0002–1361–3805]],
Adam Niewiadomski[1[0000–0001–7346–5472]]

Lodz University of Technology
Institute of Information Technology
Wólczańska 215, 90-924 Łódź, Poland
jan.kowalski@example.com

Abstract. *Processing of large data sets can interfere when inaccurate or incomplete data is included in the set. This is why it is so important that this type of data - called OUTLIERS or ANOMALIES - be detected early. The literature contains many different definitions of outliers. However, these definitions are based on numerical information (not on data expressed in vague terms). Due to this, we performed outlier finding based on the fuzzy logic method: IF-THEN. In this paper, we present a little information about this method. The proposed method will find its application in medicine, banking, cyber security, and commerce, among others.*
Keywords: *outliers in databases, fuzzy rules, detection of outliers, outlying objects*

1. Introduction

With the monumental growth in information technology, data accumulation has reached unprecedented scales, posing substantial challenges to human analysts. The collected data often contains imprecise, incomplete, or irregular entries, complicating the process of analysis. Moreover, the absence of a definitive model to characterize the data further complicates the analytical procedures. Uncertainty in data might arise from diverse sources such as stochastic variations or linguistic complexities. Such uncertainties can introduce unusual, rare, or unfamiliar data, adversely influencing data analysis across various fields and applications.

In handling uncertain or infrequent data instances, the common practice is to classify them as **outliers** or **exceptions**. Hawkins defines an outlier as “an observation that deviates significantly from other observations, raising suspicions that it was generated by a different mechanism” [1]. This characterization underscores the novelty and rarity of outliers, emphasizing the need to pay heed to their divergence from typical data.

The visibility of outliers lies in their distinctiveness from the majority of similar or common data points. The accurate identification of outliers can offer crucial insights into numerous domains, from signaling network intrusions and fraudulent financial activities to swift changes in medical equipment parameters reflecting critical health conditions [2, 3, 4]. Their misidentification, however, can obscure the true essence of the analyzed data, complicating and potentially distorting overall findings.

2. Literature review

The state of knowledge, based on an analysis of publications, shows that finding and detecting outliers in data using fuzzy logic methods is one of the approaches that provides satisfactory results. Outlier detection in complex networks is critical in various domains, such as social, communication and biological networks, where traditional methods often struggle with complexity and dynamics [5, 6, 7]. To address these challenges, innovative approaches using advanced mathematical frameworks have been proposed. In computer security, Harish B. S. and Kumar S. V. Aruna (2017) used modified fuzzy clustering for effective outlier detection, which was validated by comparative experiments [8]. Furthermore, Garg and Batra (2019) presented an ensemble technique that combines multiple outlier detection methods to improve effectiveness and reduce false alarms [9].

In software-defined networks (SDN), Novaes et al. (2020) proposed the use of long short-term memory (LSTM) and fuzzy logic for outlier detection and mitigation, demonstrating superior performance in different SDN environments [10]. For cyber outlier detection, Mungara et al. (2020) used fuzzy neural networks and achieved effective results across different datasets [11].

Moniruzzaman and Hossain (2013) provided a comprehensive evaluation of NoSQL databases for big data analytics, highlighting the classification, characteristics and comparison of these databases [12]. In fuzzy regression, Gładysz (2010) presented a method for detecting outliers based on membership degree and distance analysis, which proved to be effective on different datasets [13].

Yuan et al. (2019, 2020) made significant contributions to outlier detection with methods based on weighted density fuzzy rough sets and multi-fuzzy granules, respectively, both of which showed promising results in various applications [14, 15]. Liu and Deng (2016) introduced a local information-based method for outlier detection in uncertain data, demonstrating high effectiveness [16].

For hybrid features, Yuan et al. (2021) developed an adaptive approach using fuzzy information entropy for outlier detection, which was validated on different datasets [17]. In network security, Hamamoto et al. (2018) combined genetic algorithms with fuzzy logic for effective network outlier detection [18].

Couso et al. (2019) reviewed the applications of fuzzy sets in data analysis, from statistical foundations to machine learning techniques, providing a comprehensive overview of their practical use [19]. Kiersztyn (2022) proposed a fuzzy rule-based outlier detector, enhancing the understanding and development of fuzzy rule-based methods [20].

Zheng et al. (2015) addressed graph similarity search in large graph databases with efficient algorithms, contributing to the field of graph data mining [21]. Mazarbhuiya and Shenify (2023) introduced a fuzzy rough set-based classification method for outlier detection, which has been validated in various contexts [22].

Yu and Wu (2012) applied data mining techniques and fuzzy logic for outlier intrusion detection, while Saybani et al. (2011) used similar approaches for sensor failure detection and prediction in refineries, both demonstrating practical applications of these methods [23, 24].

Overall, these studies highlight the significant advances and diverse approaches to outlier detection in various fields, emphasising the potential and effectiveness of combining fuzzy logic, neural networks and advanced data mining techniques.

3. An outlier in terms of fuzzy rules definition

The main purpose of this paper is to propose approaches to detect and recognize outliers. In this section, we present the structure of fuzzy rules and definitions on which we base our scientific research.

The general construction of an IF-THEN rule based on any dataset using Type 1 Fuzzy Sets takes the canonical form:

$$\text{IF } d_i \text{ is } A_k \text{ THEN } d_i \text{ is } B_k \tag{1}$$

where A_k and B_k represent labels of Type 1 Fuzzy Sets. Such rules play a vital role in fuzzy logic-based systems, enabling the transformation of linguistic terms into actionable rules for decision-making or system control.

Once a dataset is specified, the antecedents and consequents are determined, and all possible combinations of rules R_k can be created. Their number depends on the number of object attributes analysed and the number of labels assigned to the fuzzy sets defined for each attribute. It is up to the user/expert to determine the words of all antecedents and consequents in the rules. These rules can then be used in various fields, such as decision support systems, control systems, or pattern recognition.

An example of a fuzzy rule is presented below:

IF the complaint is submitted in the middle of spring
THEN in an average time CFPB sends a complaint

Let $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, $N \in \mathbb{N}$, be a finite non-empty set of objects. Let $\mathcal{R} = \{R_1, R_2, \ldots, R_K\}$, $K \in \mathbb{N}$, be a set of fuzzy rules IF d_i is A_k THEN d_i is B_k, $i =$

$1, 2, \ldots, N$, and A_k, B_k are the antecedent and the consequent of R_k, respectively, represented by fuzzy sets in $\mathcal{D}$ (so R contains traditional IF-THEN rules).

For a given $k \leq K$, *the degree of outlier of* R_k is defined [25, 26]:

$$O(R_k) = \begin{cases} \min\{\max\{T, 1 - T\}, 1 - C\}, & T > 0 \\ 0, & T = 0, \end{cases} \tag{2}$$

where T is *the degree of truth* (aka *conditional and unqualified proposal* [27]) evaluated as:

$$T = \frac{\sum_{i=1}^{N} \mu_{A_k \to B_k}(d_i)}{\sum_{i=1}^{N} \mu_{A_k}(d_i)}, \tag{3}$$

for $A_k \to B_k$ – a fuzzy implication, e.g. t-norm, and C – *the degree of sufficient coverage* [28], determines if rule is activated for sufficiently large number of $d \in \mathcal{D}$ objects:

$$C = f(r_c), \tag{4}$$

where r_c is the coverage ratio:

$$r_c = \frac{1}{N} \sum_{i=1}^{N} t_i \tag{5}$$

with $t(i)$ computed as:

$$t_i = \begin{cases} 1, & \text{if } \mu_{A_K}(d_i) > 0 \text{ and } \mu_{B_K}(d_i) > 0 \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

Finally, the S-shape function $f\colon [0, 1] \to [0, 1]$

$$f(r) = \begin{cases} 0, & r < r_1 \\ g(r) & r_1 \leq r \leq r_2 \\ 1 & r > r_2 \end{cases} \tag{7}$$

where $0 \leq r_1 < r_2 \leq 1$ and g is a non-decreasing and continuous function on $[r_1, r_2]$.

Definition 1 (An outlier in terms of fuzzy rules) *Let* $\kappa \in (0, 1]$. *An object* $d_i \in D$, $i = 1, 2, \ldots, N$ *is* an outlier *if it activates any rule* R_k, $k = 1, 2, \ldots K$, *such that*

$$O(R_k) \geq \kappa \tag{8}$$

4. S-shape function

So far, outlier detection has been performed based on S-shape function proposed by Mendel in [29]. This function is presented in Equation (7). Tests based on the above S-shape function were presented in the articles: [30], [31], and [32].

Within this article, it was decided to conduct research based on another non-decreasing function. This function is presented in Figure 1 and is represented by the Equation (7).

For new proposed S-shape function (Equation (7) r_1 is 0.02 and r_2 is 0.18.

For the S-shape function proposed by Mendel, r_1 was 0.03 and r_2 was 0.17. So the range was increased to verify how this would affect outlier detection.

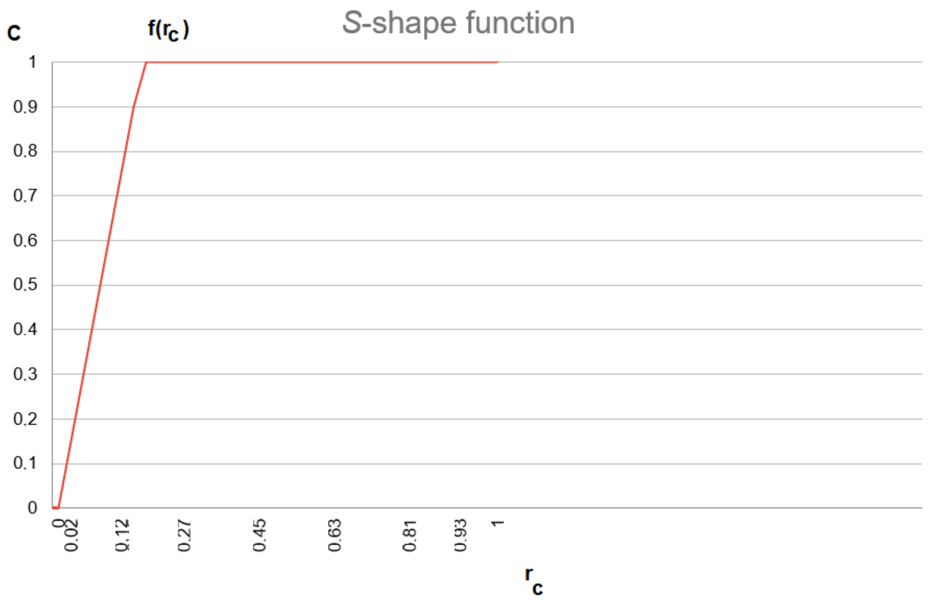


Figure 1. A graphical representation of new S-shape function.

5. Examples

Now we will show you how looks detection outliers in practice. Here, we exemplify fuzzy rules and implications as follows: let A be a fuzzy set representing linguistic label *summer* in $X = \{1, 2, \ldots, 366\}$ – numbers of days in a year in which the complaint is submitted, with μ_A given:

$$\mu_A(x) = \begin{cases} \frac{x-123}{47}, & x \in [123, 170] \\ \frac{-x+217}{47}, & x \in [170, 217] \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Let B represent label *middle county* in $Y = [5, 70]$ – per capita income in county (representing in thousands) from which come of a person who submits a complaint with $\mu_B(y)$:

$$\mu_B(y) = \begin{cases} \frac{y-35}{9}, & y \in [35, 45] \\ \frac{-y+54}{9}, & y \in [45, 54] \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Let C represent label *average time* in $Z = [0, 30]$ – numbers of days between receiving and sending the complaint to the company by CFPB (Consumer Financial Protection Bureau), with $\mu_C(z)$:

$$\mu_C(z) = \begin{cases} \frac{z-2}{4}, & z \in [2, 6] \\ \frac{-z+10}{4}, & z \in [6, 10] \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Hence the sample fuzzy rule is:

$$\begin{aligned} &\text{IF complaint is submitted in summer} \\ &\text{AND submitter comes from middle county (per capita income)} \\ &\text{THEN in average time CFPB sends complaint} \end{aligned} \tag{12}$$

For example, a complaint is submitted on May 4 (the 124th day of the year) and the submitting person comes from the county where per capita income is 53 thousand \$. Hence, $\mu_{A_K}(124) \simeq 0.02$, $\mu_{B_K}(53) \simeq 0.11$, so via the product implication, the value of the rule is $\simeq 0.0022$
Next, for each fuzzy rule, perform calculations according to the formulas (1)-(7). The newly proposed function shown in Figure 1 was used during the tests. The implication used in the calculation is Product. Additionally was decided that fuzzy rules are deemed as outliers when the degree of sufficient coverage $C \geq 0.1$ and the degree of outlier $O(R_k) \geq 0.94$. For fuzzy rule: IF the complaint is submitted in the middle of spring AND the submitter comes from a rich county (median household) THEN in an average time CFPB sends a complaint $O(R_k) = 0.94$ and $C = 0$.

6. Results and comments

The research was conducted on a database containing 40083 objects. We performed test 1500 fuzzy rules. 2 unique rules were found, which translate into 5 unique objects. An example outlier found is presented below:

- IF the complaint is submitted in the middle of spring AND the submitter comes from a rich county (median household) THEN in an average time CFPB sends a complaint

As a result, we obtained the following two unique fuzzy rules D_{out1}, and D_{out2}:

- 85. IF the complaint is submitted in the middle of spring AND the submitter comes from a rich county (median household) THEN in an average time CFPB sends a complaint [$O(R_k) = 0.94$, and $C = 0$]
- 121. IF the complaint is submitted in early winter AND the submitter comes from a rich county (median household) THEN in an average time CFPB sends a complaint [$O(R_k) = 0.94$, and $C = 0$]

It is associated with 5 objects. IDs of the objects are: $D_{out} = \{805\,340, 801\,371, 372\,521, 43\,196, 773\,246\}$.

The definition introduced allows detection and, more importantly, recognition of specific objects (which are outliers). We detected 2 outliers - 5 objects. The method has been tested on real data available in the data set, where outliers may indicate incorrect data input, or indeed, anomalous data properties.

The use of the new S-shape function shown in Figure 1 did not significantly affect the outcome of outlier detection.

7. Conclusions

Fuzzy sets allow the definition of linguistic variables using natural or "quasi-natural" language. The use of fuzzy sets in the presented author's method for detecting and recognizing outliers made it possible, therefore, to define and present the parameters and the obtained results in a natural way. Based on the research work, we can conclude that the use of fuzzy logic, IF-THEN fuzzy rules effectively and easily detect and recognize outliers in databases.
Appropriate treatment of outliers can allow us to prevent possible threats in databases. The outlier detection method can be applied, for example, to a database that has medical data, so that we can detect sick patients who have specific symptoms that are different from those of people with the same disease [33], [34], [35].
The outlier detection research within this article was extended to include the use of another new S-shape function. Applying the function shown in Figure reffig did not significantly change the results. In the future, it is worth considering using an even wider range of functions in which it grows and narrowing the aforementioned range.
The research work helped expand knowledge in such areas and fields of computer science as databases, artificial intelligence, and data analysis.

References

[1] Hawkins, D. M. *Identification of outliers*, volume 11. Springer, 1980.

[2] Aggarwal, C. C. and Yu, P. S. Outlier detection for high dimensional data. In *ACM Sigmod Record*, volume 30, pages 37–46. ACM, 2001.

[3] Breunig, M. M., Kriegel, H. P., and Sander, J. Density-based clustering of spatial data with noise. In *KDD '01 Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2001.

[4] Song, X., Wu, Q. J., and Jermaine, C. Conditional anomaly detection. In *SIGMOD '07 Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. 2007.

[5] Wang, J.-F. et al. Anomaly detection of complex networks based on intuitionistic fuzzy set ensemble. *Chinese Physics Letters*, 35(5):058901, 2018.

[6] Suresh, S. and Kannan, K. S. Identifying outliers in fuzzy time series. *Journal of Modern Applied Statistical Methods*, 10(2):710–717, 2011.

[7] Cateni, S., Colla, V., and Vannucci, M. A fuzzy logic-based method for outliers detection. pages 605–610. 2007.

[8] Harish, B. S. and Aruna kumar, S. V. Anomaly based intrusion detection using modified fuzzy clustering. *International Journal of Interactive Multimedia and Artificial Intelligence*, In Press:1, 2017. doi:10.9781/ijimai.2017.05.002.

[9] Garg, S. and Batra, S. A novel ensembled technique for anomaly detection: Ensembled anomaly detection technique. *International Journal of Communication Systems*, 30:e3248, 2016. doi:10.1002/dac.3248.

[10] Novaes, M. P., Carvalho, L. F., Lloret, J., and Proença, M. L. J. Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. *IEEE Access*, 2020. doi:10.1109/ACCESS.2020.2992044.

[11] Mungara, K. K. et al. Detection of cyber anomaly using fuzzy neural networks. *Journal of Engineering Sciences*, 2020.

[12] Moniruzzaman, A. B. M. and Hossain, S. A. Nosql database: New era of databases for big data analytics - classification, characteristics and comparison. *International Journal of Database Theory and Application*, 6(4), 2013.

[13] Gładysz, B. A method for detecting outliers in fuzzy regression. *Operations Research and Decisions*, (2):25–39, 2010.

[14] Yuan, Z., Chen, B., Liu, J., Chen, H., Peng, D., and Li, P. Anomaly detection based on weighted fuzzy-rough density. *Appl. Soft Comput.*, 134(C), 2023. ISSN 1568-4946. doi:10.1016/j.asoc.2023.109995. URL `https://doi.org/10.1016/j.asoc.2023.109995`.

[15] Yuan, Z., Chen, H., Luo, C., and Peng, D. Mfgad: Multi-fuzzy granules anomaly detection. *Information Fusion*, 95:17–25, 2023. ISSN 1566-2535. doi:https://doi.org/10.1016/j.inffus.2023.02.007. URL `https://www.sciencedirect.com/science/article/pii/S1566253523000490`.

[16] Liu, J. and Deng, H. Outlier detection on uncertain data based on local information. *Knowledge-Based Systems*, 51:60–71, 2013. ISSN 0950-7051. doi:https://doi.org/10.1016/j.knosys.2013.07.005. URL `https://www.sciencedirect.com/science/article/pii/S0950705113002074`.

[17] Yuan, Z., Chen, H., Li, T., and Liu, J. Fuzzy information entropy-based adaptive approach for hybrid feature outlier detection. *Fuzzy Sets and Systems*, 421:1–28, 2021. URL `https://www.elsevier.com/locate/fss`.

[18] Hamamoto, A. H., Carvalho, L. F., Sampaio, L. D. H., Abrão, T., and Proença Jr., M. L. Network anomaly detection system using genetic algorithm and fuzzy logic. *Expert Systems With Applications*, 92:390–402, 2018. URL `https://www.elsevier.com/locate/eswa`.

[19] Couso, I., Borgelt, C., Hüllermeier, E., and Kruse, R. Fuzzy sets in data analysis: From statistical foundations to machine learning. *IEEE Computational Intelligence Magazine*, 14(1):31–58, 2019. doi:10.1109/MCI.2018.2881642.

[20] Kiersztyn, K. and Kiersztyn, A. Fuzzy rule-based outlier detector. In *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–7. IEEE, 2022. doi:10.1109/FUZZ-IEEE55066.2022.9882567.

[21] Zheng, W., Zou, L., Lian, X., Wang, D., and Zhao, D. Efficient graph similarity search over large graph databases. *IEEE Transactions on Knowledge and Data Engineering*, 27(4), 2015. doi:10.1109/TKDE.2014.2330821.

[22] Mazarbhuiya, F. A. and Shenify, M. An intuitionistic fuzzy-rough set-based classification for anomaly detection. *Applied Sciences*, 13(9):5578, 2023. doi:10.3390/app13095578.

[23] Yu, Y. and Wu, H. Anomaly intrusion detection based upon data mining techniques and fuzzy logic. In *2012 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, Seoul, Korea, 2012.

[24] Saybani, M. R., Wah, T. Y., Amini, A., and Aghabozorgi, S. R. Anomaly detection and prediction of sensors faults in a refinery using data mining techniques and fuzzy logic. *Scientific Research and Essays*, 6(27):5685–5695, 2011. doi:10.5897/SRE11.333.

[25] Kosko, B. Fuzziness vs. probability. *International Journal of General Systems*, 17:11–240, 1990.

[26] van den Berg, J., Kaymak, U., and van den Bergh, W.-M. Fuzzy classification using probability-based rule weighting. *in Proc. IEEE Int'l Conf. on Fuzzy Systems*, pages 991–996, 2002.

[27] Klir, G. J. and Yuan, B. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Upper Saddle River, NJ:Prentice-Hall.

[28] Wu, D., Mendel, J., and Joo, J. Linguistic summarization using if-then rules. In *IEEE International Conference on Fuzzy Systems*, pages 1–8. 2010.

[29] Wu D., J. J., Mendel J. Linguistic summarization using if-then rules. *IEEE International Conference on Fuzzy Systems*, 2010.

[30] Niewiadomski A., Duraj A., and Bartczak M. Outliers recognition via linguistic aggregation of graph databases. *Applied Sciences. Modeling a Pre-Touch Reaction Distance around Socially Touchable Upper Body Parts of a Robot*, 11(16), MDPI, 2021.

[31] Niewiadomski A., Kacprowicz M., and Bartczak M. Outliers detection in graph-represented databases using fuzzy rules. *Pacific Asia Conference on Information Systems, PACIS 2021*, Dubai, Saudi Arabia, 2001.

[32] Kacprowicz M., Bartczak M., and Niewiadomski A. Detection and recognition of outliers by the use of if-then rules. *Conference PP-RAI'2022 - 3th Polish Conference On Artificial Intelligence*, Gdynia, Poland, 2022.

[33] Jääskelä J. *Anomaly-Based Insider Threat Detection with Expert Feedback and Descriptions*. Master's thesis, University of Oulu, Finland, 2020.

[34] Zachariah T. *Master thesis. A Mathematician's Guide to Fuzzy Logic with Applications in Fuzzy Additive Systems. Master of Science in Mathematical Sciences*. Middle Tennessee State University, 2021.

[35] Duraj A. and Szczepaniak P. S. *Wykrywanie wyjątków metodami analizy danych*. Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2023.

A System of Interaction with Non-Player Characters in a cRPG Game Based On Interpersonal Relationships

Krzysztof Kocot[0009–0007–2320–5286]**, Damian Pęszor**[0000–0002–3754–2212]

Silesian University of Technology
Department of Graphics, Computer Vision and Digital Systems
Akademicka 16, 44-100 Gliwice, Poland
jan.kowalski@example.com
damian.peszor@polsl.pl

Abstract. *The system of interaction with non-player characters is a pivotal component in multiple computer game genres, reflecting the dynamics of inter-character relationships. Specifically, within the cRPG genre, where there is a marked emphasis on deep immersion, this element assumes heightened significance.*

In real-world scenarios, human relationships are distinguished by deep intricacies, which encompass a spectrum of emotions, motivations, and conflicts. As such, it would be desirable to introduce a system capable of replicating this complexity, providing players with richer experiences. The primary aim of this study was the development of a system that complies with this specification.
Keywords: *computer games, non-player characters interaction system, artificial intelligence*

1. Introduction

One of the noticeable trends in the computer games industry is the pursuit of the most accurate environment reproduction. This phenomenon is reflected in both increasingly realistic graphics and advanced systems of behavior for Non-Player Characters (NPCs). Interaction systems have moved from simple conversations to extensive dialogue options, the selection of which in real time affects the gameplay, e.g. by blocking or unlocking available behaviors, tasks, and story paths. Non-player characters are no longer objects placed in space whose influence on the hero is strictly defined, but they started to move, react to the player's behavior, communicate with each other, or simulate such emotions as joy, sadness or fear. This is especially important in the case of computer role playing games (cRPG), a genre recognized and defined in early video game classifications, such as [1, 2]. The genre of cRPG is very heterogeneous, spanning all possible positions in 13 dimensions that define the game as described by [3] and all 17 dimensions stated by [4], more accurately described as the focus of a gameplay facet by [5]. The origins of cRPG games can be traced to traditional role-playing games (RPGs), e.g. Dungeons & Dragons, played in the real world between players taking on the roles of characters they have created. The genre is bound by a heavy focus on interaction with NPCs, a staple of cRPGs that distinguishes it from hack'n'slash and adventure games. This interaction can have a real impact on the gameplay, e.g. by unlocking other dialogues or tasks. Therefore, it seems reasonable to assume that a system that maps interpersonal relationships in the game will find its application in games of this type created in the future. This paper proposes a system of interaction with independent characters modeled on interpersonal relationships in a cRPG game. The basic assumptions of the designed dialogue system are the connections between characters, represented in the form of a set of opposing feelings, e.g. distrust-trust. Additionally, characters are described by a set of opposing states of well-being, e.g., anger-controlled, which influences the real-time change in attitudes between characters, both player-controlled and NPCs. The remainder of this paper is constructed as follows; Section 2 describes the current state of the art, while Section 3 refines the problem statement. Section 4 defines the methods and materials used, which leads to Section 5 containing the results of the proposed system. Finally, the discussion and conclusions are provided in Section 6.

2. State of the art

The surge in the popularity of computer games has positively impacted the number of studies and scientific articles related to games themselves, the methodologies used in their creation and their implications on the environment. A review of the literature reveals several proposed solutions to the problem discussed.

A system described by researchers at the University of Nottingham [6] is based on integrated models of emotion, personality, and interpersonal relationships. The personality component is grounded in the widely recognized OCEAN model detailed in the literature [7]. This model establishes a baseline emotional threshold for a character, which can dynamically fluctuate depending on the relationships between characters. These relationships are represented through a two-dimensional coordinate system, where one axis represents familiarity and the other affinity. Furthermore, the proposed mechanism encompasses seven distinct emotions. Each emotion can be triggered by specific in-game events. Upon activation, they alter the emotional state of a character for a certain duration. The intensity of the emotion depends on the nature of the triggering event, its significance, and the personality of the character affected. A notable feature is the masking of a character's emotions based on, for instance, relationships or cultural norms. After a certain period, the invoked emotion reverts to its baseline state unless reactivated.

Researchers from Pierre and Marie Curie University in France have presented another perspective on the issue of social relationships in games [8]. Their approach is predicated on four variables representing the disposition of independent characters towards one another. These are: affinity and dominance, represented by a value range of $[-1, 1]$, and familiarity and solidarity, represented by a value range of $[0, 1]$. The emotional state of each agent is depicted in pairs of emotions; joy-distress, hope-fear, admiration-reproach, pride-shame. Each pair is defined by a value range of $[-1, 1]$, allowing emotions to be expressed in the form of a vector $V \in ([-1, 1])^4$. In the proposed model, three distinct vectors are discerned:

- Emotion elicited by an event, the magnitude of which depends on the event and the character's personality.
- The emotional state of the character, influenced by past experiences.
- Emotions displayed by the character, which might differ from the character's actual emotional state because they may choose to project something different.

Each of the variables representing a character's emotional state has a list of emotions that influence it either positively or negatively, with the degree of change influenced by the intensity of the emotion. In the absence of events, the relationships are assumed to remain unchanged.

In another study, researchers from Jagiellonian University [9] explicitly state that the most convenient method to depict the complexity of the real or virtual world is the graph model. The universal system of nodes and edges serves as an instrument for describing the majority of relationships, hierarchies, and structures. A hierarchical graph facilitates the management of intricate entities, such as a group composed of several characters. The nodes are interconnected with directed edges; however, two edges connecting the same elements but in opposing directions can be perceived as a symmetric relationship. A node, in conjunction with all nodes to which it is connected, as well as its attributes and labels, is referred to as an instance and allows for the interpretation of the game's state. The authors use multi-layered graph to represent the game's entire logic. The graph incorporates interconnected layers of narrative, locations, characters, and objects. A crucial premise here is the interdependence of the layers, with the narrative being particularly significant. The primary source of modifications to the game's graph results from actions undertaken by the player. An example of such an action might be picking up an object. In this scenario, the object, represented as a node, would be detached from its current location and attached to the character that collected it.

While addressing the topic of relationships in games, it is pertinent to mention “The Sims” gameseries published by Electronic Arts. While games of this series can be categorized under the simulation genre, they are among the most recognized market representatives addressing this issue, making them worthy of analysis. “The Sims 4” offers a comprehensive spectrum of both positive and negative emotions experienced by the characters, known as Sims. Each of these emotions influences a Sim’s development and relationships, impacting areas like career, athletic abilities, or interactions. The game features more than 100 distinct interactions that can occur between characters. The creators have categorized these into subsets of friendly, romantic, mean, and neutral interactions. Moreover, special interactions are available that require meeting specific criteria. Some of these, like the first kiss, can only occur once for each Sim.

In another context, there’s a Japanese cRPG game “Xenoblade Chronicles”, released by Monolith Soft that employs a system named “Affinity”. This system, among other things, manages relationships between key characters, delineating their dispositions towards one another. Three subcategories of Affinity are discerned. The first exists between members of the player’s group. These relationships are quantified by points that can fluctuate due to shared battles or task completions. Upon surpassing specific point thresholds, the relationship ascends in its level, spanning only neutral to exceptionally positive relationships. The next relationship type involves non-player characters. When one non-player character mentions another during a conversation, they become connected, and this connection is reflected on a graph. Unlike the former relationship type, this variant ranges from very negative to very positive. The final category is the relationship between the player’s group and available locations, essentially representing the group’s reputation within a region. This reputation can change based on dialogues and actions performed, influencing accessible tasks and items within the game’s trading mechanism.

Seamount has released a first-person exploration game under the name “Thousand Threads”. A fundamental premise of the game is an expansive open world through which the player traverses, assuming the role of a postman. Throughout the game, players deliver letters, gather resources, hunt, and most importantly become acquainted with characters and locations. Discovering the latter is intricately linked to conversations with the former; through dialogues, players ascertain where to locate people and places of interest. An unconventional interaction mechanic with NPCs has been designed, wherein NPCs retain memory of in-game events. Each NPC maintains a dynamic list of information related to someone’s location and behavior. As the player explores the world, they can engage in interactions with characters that can be legal, like delivering a letter, or illicit, such as assault or theft. In the case of the latter, if the player’s misdeeds are witnessed, there is a risk that news will spread among NPCs, potentially resulting in consequences like retribution. In addition, in-game interactions also occur among NPCs themselves.

For example, one character might aim to attack another to seize a valuable item.

An educational experiment in the form of the game, "Kobun", was conceptualized with Virtual Reality utilization in mind. The game features a sole non-player character named Kobun, who can be commanded verbally by the player. Additionally, each interaction is dependent on an emotional component that encompasses emotions, mood, and personality. Emotions are short-lived, but powerful feelings. Mood represents long-term sentiment that exhibits subtle fluctuations over time. Both emotions and mood influence one another, yet both are modulated by personality, which defines the non-player character and remains constant. Additionally, if a player issues a command resulting in harm to Kobun, the non-player character might be reluctant to cooperate promptly in subsequent commands, recalling the prior event.

3. Problem statement

The fundamental premise of the project entails the establishment of a system of relations with non-player characters, designed to simulate, in a simplified manner, the relationships among a group of individuals in the real world. Upon evaluating the examples provided in both the scientific literature and the existing implementations, one can delineate the pivotal components of the proposed system. In the context of the issue under discussion, it is pertinent to enumerate four mechanisms.

A relationship mechanism manifested as a graph that details the connections between characters. These connections should be defined by three integral components: disposition, trust, and respect, modeled in terms of antithetical sentiments such as disdain-respect. For improved clarity, these should be quantified within the range $\in [-100, 100]$ and are deemed symmetric. To facilitate players in consciously manipulating these relationships, a visualization of this graph, reminiscent of the format depicted in Xenoblade Chronicles, is deemed necessary. Differentiating player-to-NPC relationships as more intricate and NPC-to-NPC as simpler, encompassing only disposition, might be warranted to streamline the player's strategic gameplay experience.

Another integral facet of the framework should be the mood, again defined by three values within the range [-100,100], indicating the levels of happiness, composure and restfulness, respectively. Furthermore, each character should possess a personality delineated by an arbitrary number of traits from a set that includes the following: emotional, melancholic, emotionally unstable, composed, anxious, chronically fatigued, amicable, introverted, gullible, distrustful, arrogant. Both these components should significantly influence perceived emotions. Each emotion experienced should have its basis in an in-game event and should lead to a shift in relationships. However, the influence of each emotion should attenuate over time.

A conspicuous absence from the existing solutions cited is the mutual influence of relationships on each other. Such dependencies are ubiquitously observed in human interpersonal relationships. If individual A and B share a close bond and individual C wrongs B, A might harbor prejudice against C despite unfamiliarity. Likewise, the matter extends to disregarding the impact an event can engender without altering relationships. Scenarios may arise where an event's significance is trivial enough that its effect on relationships is bypassed, or the rapport between character A and character B is tenuous enough that the event again leaves the relationship unchanged. Introducing such nuances would undoubtedly infuse novelty and enable players to encounter a unique experience.

It is evident that an accurate simulation of such a complex phenomenon as social interaction in a community is virtually unfeasible given the current state of technological development. Consequently, the designed system significantly simplifies the relationships and is predicated on very superficial and straightforward principles:

- The enemy of one's enemy is his friend.
- The enemy of one's friend is his enemy.
- The friend of one's enemy is his enemy.
- One's friend's friend is his friend.
- Negative well-being has a negative impact on the emotions one feels.
- Positive well-being has a positive impact on the emotions that one experiences.
- Personality influences the emotions one feels.
- Emotions are conveyed mainly during conversation.
- The same feeling carries more weight if it is the first impression.

The foundational premise of the envisaged system is to mirror human relationships between the player and NPCs, as well as among the NPCs themselves, followed by modulation of these relationships through pertinent interactions, such as dialogues or the execution of specific favors. A paramount challenge lies in the meticulous calibration of the system to ensure that the system not only fulfills its predetermined objectives, but also manifests a level of transparency that empowers players to consciously manipulate the available set of relationships. On one hand, players should perceive a semblance of engagement with entities that harbor distinct feelings towards them, feelings that are susceptible to alteration and that influence the manner in which the character interacts with the player. For instance,

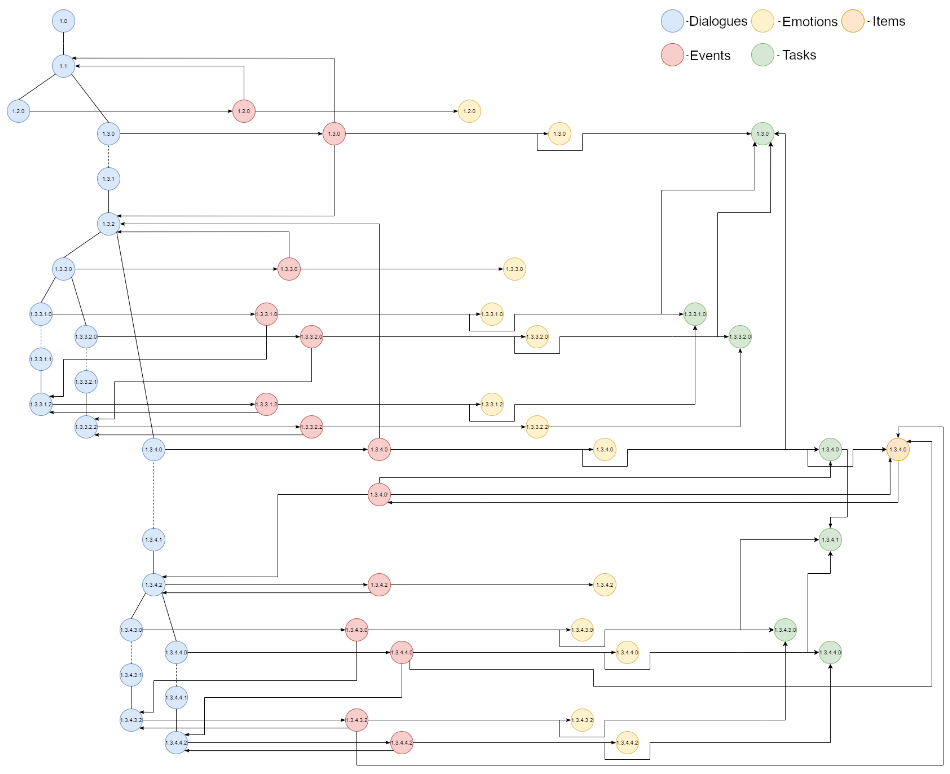


Figure 1. A graph including all the elements and connections between them included in the sample task available in the test application.

a character with whom the player has nurtured a close, trust-based relationship may confide a secret or entrust a significant task to the player. Conversely, the system must epitomize clarity to the extent that, following a brief introduction, players can accurately anticipate the relational shifts engendered by their actions.

4. Materials and methods

4.1. Graph of Connections Among Objects

Given the intricate graph that connects various data structures available within the system, the dependencies among them have been illustrated based on an exemplary task implemented within the application, as shown in Fig. 1.

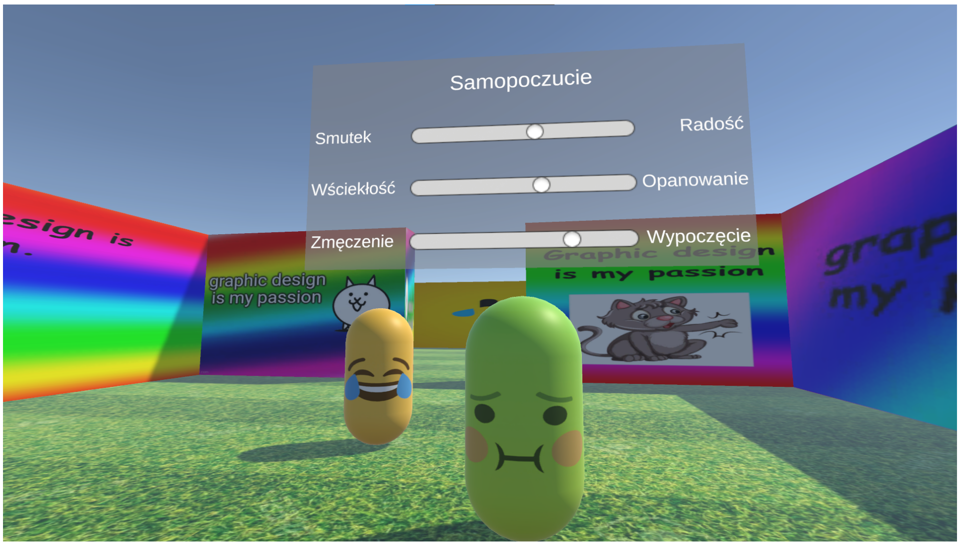


Figure 2. A screenshot elucidating the detailed affective state of an non-player character.

4.2. Emotion Perception Mechanism

The mechanism for perceiving emotions, one of the main tenets of the system, is based on asynchronicity. In case of test application, implemented as a Unity engine's coroutine. Coroutines facilitate the temporal distribution of tasks. Upon invocation, a coroutine executes its segment of the program, subsequently relinquishing control back to the point of invocation. This cycle is repeated per frame until the entire coroutine program has been executed.

Each emotion comprises a list of changes in disposition, affective state, and self-perception of autonomous characters, which in the demostrative game are visualized as in Fig. 2. Upon the elicitation of an emotion, a coroutine is initiated for each change, with its lifecycle dependent on the intensity of the change and its duration, as illustrated in Fig. 3. The depicted process is isolated from the influence of any other factors, such as affective state, personality, or other emotions.

1. In each frame, the value of the primary variable (e.g., level of trust towards the player) is altered by a value $\Delta_z = I \cdot \Delta_t$.

2. In each frame, the value of the correlated variable responsible for attenuation is modified by a value $\Delta_{zt} = \frac{1}{2}I \cdot \Delta_t$.

3. The impact of both values is sustained for a period of time $t = IDk$.

4. In each frame, the value of the variable responsible for attenuation is altered by a value $\Delta_{zt} = -\frac{1}{4}I \cdot \Delta_t$.

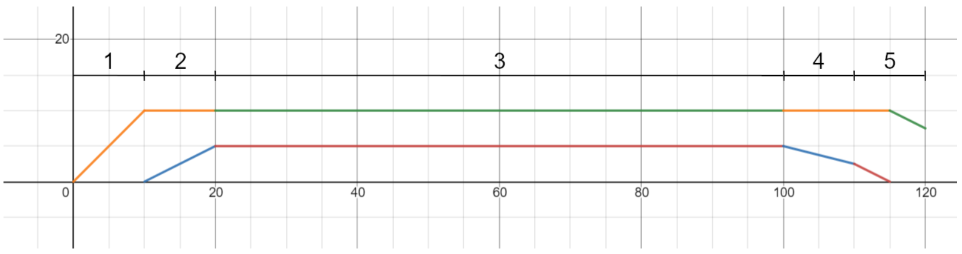


Figure 3. The progression of the influence of emotions on a variable is delineated with parameters $I = 1$, $D = 10$, $k = 8$.

5. In each frame, the value of the primary variable is modified by a value $\Delta_z = -\frac{1}{2}I \cdot \Delta_t$ taking into account the remaining attenuation,

where:
Δ_z - Alteration of the primary variable per frame. For instance, a modification in the trust variable in each frame.
Δ_{zt} - Alteration of the correlated variable responsible for attenuation per frame. For example, a modification in the variable responsible for trust attenuation in each frame.
I - Intensity of the Primary Value Modification.
D - Duration of modification.
Δ_t - Time since the last frame.
k - Attenuation multiplier specified as a system parameter

4.3. Computation of Perceived Emotions

The calculation of perceived emotions is carried out utilizing a universal formula, as presented in Eq. 1:

$$\Delta_{zm} = W_r \cdot W_s \cdot W_o \tag{1}$$

where:
Δ_{zm} - Value of the elicited change.
W_r - Influence of other relationships on perceived change.
W_s - Impact of affective state on perceived change.
W_o - Influence of personality on perceived change.

Herein, the predominant factor is the influence of other directly connected relationships. This influence determines whether the new change is positive or negative, while also defining its baseline intensity, as illustrated in Eq. 2:

$$W_z = \frac{100 + y}{100} \cdot \Delta_x \cdot j \tag{2}$$

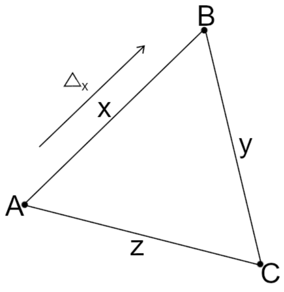


Figure 4. Relationship dependency among three non-player characters

where:
W_z - Influence of relationships on alteration of the disposition of Character C towards Character A, as shown in Fig. 4.
y - Disposition of Character C towards Character B, visible in Fig. 4.
Δ_x - Change in disposition from Character A towards Character B, illustrated in Fig. 4.
j - Coefficient of attenuation in relationships during their transmission.

The impact of personality on perceived emotions is a composite of the influences of its components, as described in Eq. 2, for which various functions have been employed to either amplify or mitigate potential changes.

$$W_s = \frac{1}{100} \cdot (100 + W_h + W_c + W_f) \tag{3}$$

where:
W_s - Influence of personality on perceived emotions.
W_h - Influence of the *happiness* variable.
W_c - Influence of the *calmness* variable.
W_f - Influence of the *freshness* variable.

In each of these, two elements can be distinguished: the calculation of the sign and the value of the change. The conditions that the sign should satisfy are presented in Table 1.

The computed sign functions are as follows:

For the ‘happiness’ and ‘calmness’ variables, refer to Equation 4.

$$sign(z \cdot \Delta_n) \tag{4}$$

wherein:
z - Value of the variable.
Δ_n - Potential change in disposition.

For the ‘freshness’ variable, refer to Equation 5.

$$-sign(\Delta_n) \tag{5}$$

Table 1. Influence of constituent variables of affective state on the intensity of perceived emotions, where Δ_n - represents potential change in disposition, W - denotes the sign of the variable's impact on change intensity, h - *happiness* variable, c - *calmness* variable, f - *freshness* variable.

Happiness variable			Calmness variable			Freshness variable		
Δ_n	h	W	Δ_n	c	W	Δ_n	f	W
≥ 0	≥ 0	+	≥ 0	≥ 0	+	≥ 0	≥ 0	-
≥ 0	< 0	-	≥ 0	< 0	-	≥ 0	< 0	-
< 0	< 0	+	< 0	< 0	+	< 0	< 0	+
< 0	≥ 0	-	< 0	≥ 0	-	< 0	≥ 0	+

wherein:
Δ_n - Potential change in disposition.

Although identifying a function that meets the requirements for the sign is relatively straightforward, devising a single composite function to calculate the value of change is unfeasible without employing appropriate algorithms. One approach addressing this issue is polynomial approximation, which allows for the approximation of any function using a polynomial. The accuracy of the approximation is contingent upon the distribution of input values; however, for those selected in the project, the results obtained are satisfactory. The polynomial approximation through the least-squares method was utilized in the project to calculate the polynomial.

The progressions of the desired function change are presented in Figures 5, 6, and 7:

For the *happiness* variable, Fig. 5.

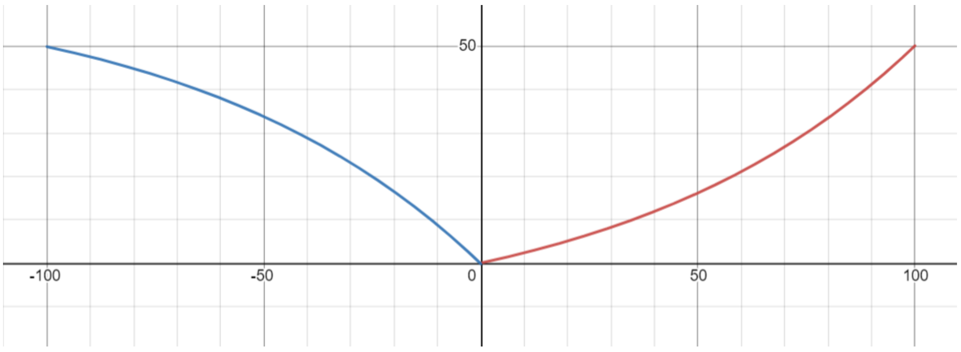


Figure 5. Desired progression of the change function for the *happiness* variable.

For the *calmness* variable, Fig. 6.

For the *freshness* variable, Fig. 7.

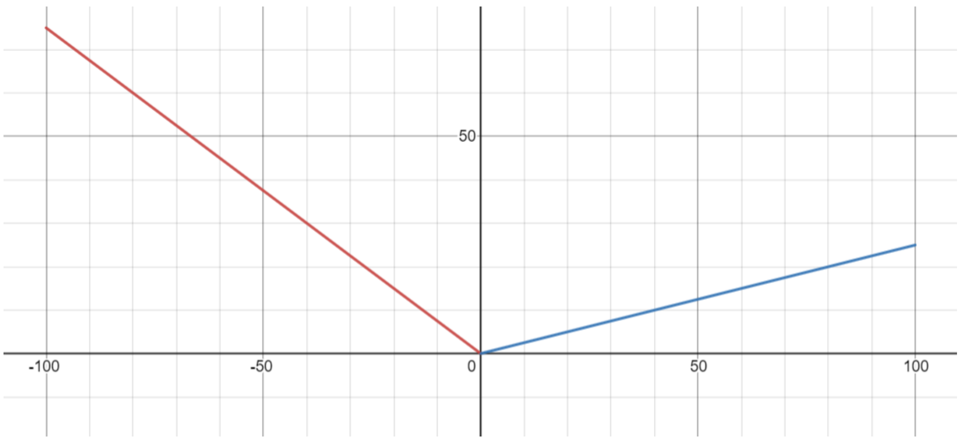


Figure 6. Desired progression of the change function for the *calmness* variable.

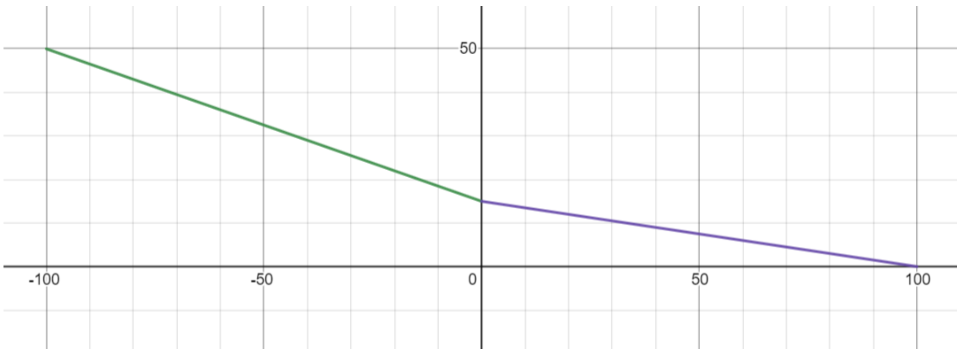


Figure 7. Desired progression of the change function for the *freshness* variable.

The values based on which the most satisfactory approximation was achieved using the polynomial approximation are outlined in Table 2. In the case of the *freshness* variable, due to its progression closely resembling a linear function, the addition of subsequent values only deteriorated the result obtained.

The functions derived from the application of the polynomial approximation are presented below. For the polynomial obtained for the *happiness* and *calmness* variables, a minor adjustment was needed to ensure the neutrality of the impact of the function for $x = 0$, as expressed in Equation 6:

$$f(x) = W - W(0) \cdot \frac{100 - |x|}{100} \tag{6}$$

where:
W - Resultant Polynomial.

The introduction of the aforementioned correction resulted in the derivation of

Table 2. Values upon which the polynomial approximation for the constituent variables of affective state was computed.

Happiness variable		Calmness variable		Freshness variable	
x	$f(x)$	x	$f(x)$	x	$f(x)$
-100	50	-100	75	-100	50
-75	43,8	-75	56.25	0	15
-50	34,4	-50	37,5	100	0
-25	20,5	-25	18,75		
0	0	0	0		
25	6,1	25	6.25		
50	15,7	50	12,5		
75	29,5	75	18,75		
100	50	100	25		

negative values for certain intervals of the polynomial, which required the addition of another correction in the form of Eq. 7.

$$f'(x) = max(0, f(x)) \tag{7}$$

where:
$f(x)$ - Function obtained after the addition of the first correction.

As a consequence of the implemented modifications, the altered functions and their progressions are represented as follows:

For the *happiness* variable, refer to Eq. 8 and Fig. 8.

$$f(x) = max(0, -ax^5 - bx^4 + cx^3 + dx^2 - ex + f - 5.97 \cdot \frac{(100 - |x|)}{100}) \tag{8}$$

where $a = 0.0000000015$, $b = 0.0000003364$, $c = 0.0000453305$, $d = 0.0077141259$, $e = 0.2990191142$, and $f = 5.9731934732$.

For the *calmness* variable, refer to Eq. 9 and Fig. 9

$$f(x) = max(0, -ax^4 + cx^2 - dx + e - 5.245 \cdot \frac{(100 - |x|)}{100}) \tag{9}$$

where $a = 0.000000373$, $c = 0.0081585082$, $d = 0.25$ and $e = 5.2447552448$

For the *freshness* variable, refer to Eq. 10 and Fig. 10

$$f(x) = \frac{1}{1000}x^2 - \frac{1}{4}x + 15 \tag{10}$$

where $a = 0.001$ and $b = 0.25$

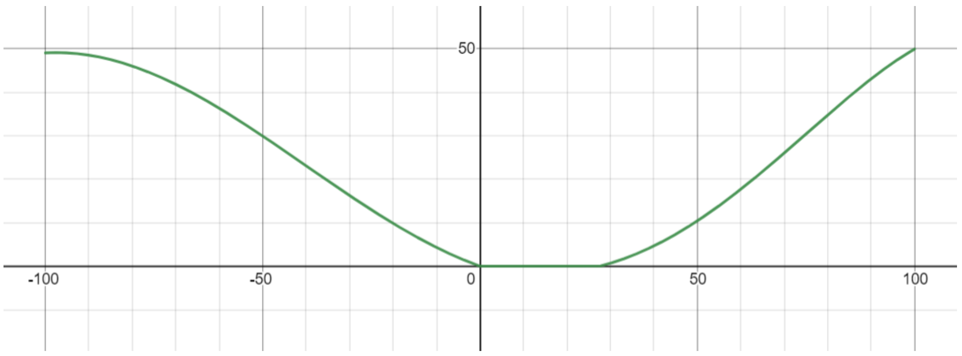


Figure 8. Resultant progression of the change function for the *happiness* variable.

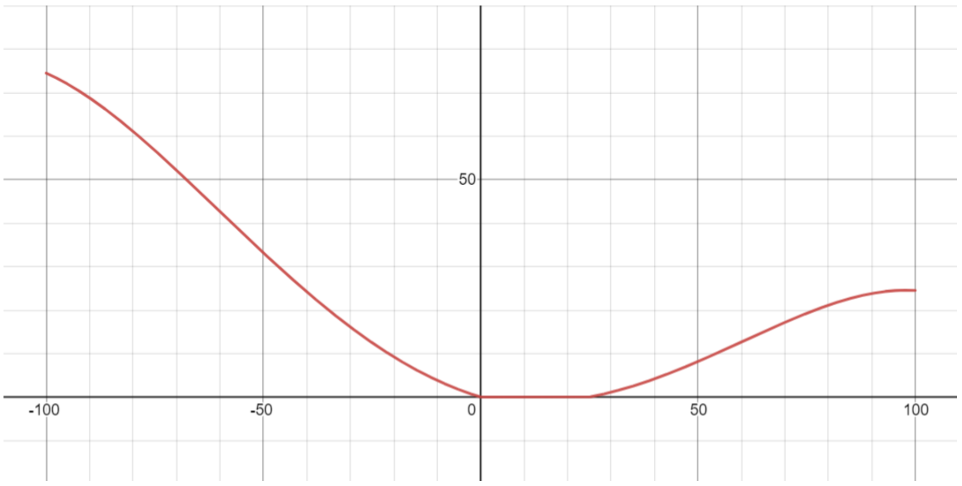


Figure 9. Resultant progression of the change function for the *calmness* variable.

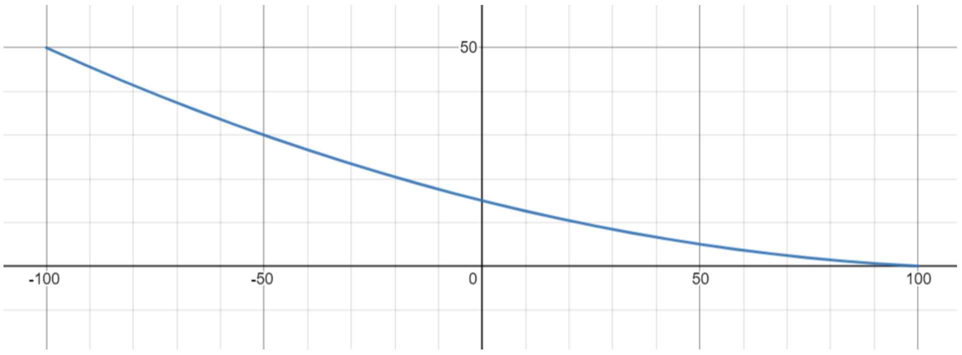


Figure 10. Resultant progression of the change function for the *freshness* variable.

Table 3. Influence of personality traits on perceived emotions, where the variables *happiness*, *calmness*, and *freshness* pertain to the character's affective state; the variables *like*, *trust*, and *respect* relate to the character's disposition towards the player; and the variable *weight* refers to the disposition of independent characters towards each other.

Trait	Changed value	Condition	Influence
Emotional	*happiness, calmness*	None	+20%
Melancholy	*happiness*	20% chance	$-sign(\Delta) \cdot 20\%$
Emotionally unstable	*happiness*	20% chance	+∨- 20%
Possessed	*calmness*	$\Delta < 0$	-20%
Nervous	*calmness*	$\Delta < 0$	+20%
Chronically tired	*freshness*	None	$-sign(\Delta) \cdot 20\%$
Friendly	*like/weight*	None	+20%
Secretive	*like/weight*	None	−20%
Easily	*trust*	$\Delta > 0$	+20%
Distrustful	*trust*	None	$-sign(\Delta) \cdot 20\%$
Hubby	*respect*	$\Delta > 0$	-20%

The final factor influencing perceived emotions is personality, which, once again, is a composite of the influences of all personality traits. This influence, along with the principles upon which it operates, is delineated in Table 3.

5. Results

5.1. Available Tasks

To test the developed system, two tasks have been implemented within the demonstration application. The first task is considerably more elaborate, comprising five distinct conclusions, each of which elicits different emotions along the progression path, as illustratively depicted in Figure 11.

The second task, which has a singular conclusion, becomes accessible when the character named *Jack* exhibits a minimum disposition value of 40 towards the player, concurrently with a trust value reaching a minimum of 1.

Furthermore, the player has two dialogue options at his disposal with the character *Jack*, which are designed to facilitate the test of emotion transmission across the relationship matrix. These options are accessible in the dialogues titled *Test emotion transmission* and *Test negative emotion transmission*, and trigger extremely positive and extremely negative emotions, respectively.

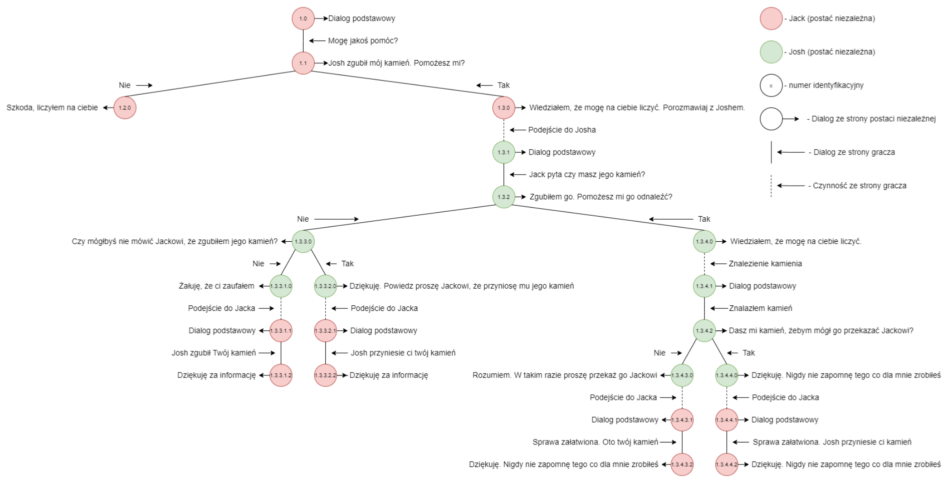

1.0
Dialog podstawowy
Mogę jakoś pomóc?
1.1
Josh zgubił mój kamień. Pomożesz mi?
Nie
Tak
Szkoda, liczyłem na ciebie
1.2.0
1.3.0
Wiedziałem, że mogę na ciebie liczyć. Porozmawiaj z Joshem.
Podejście do Josha
1.3.1
Dialog podstawowy
Jack pyta czy masz jego kamień?
1.3.2
Zgubiłem go. Pomożesz mi go odnaleźć?
Nie
Tak
Czy mógłbyś nie mówić Jackowi, że zgubiłem jego kamień?
1.3.3.0
1.3.4.0
Wiedziałem, że mogę na ciebie liczyć.
Nie
Tak
Żałuję, że ci zaufałem
1.3.3.1.0
1.3.3.2.0
Dziękuję. Powiedz proszę Jackowi, że przyniosę mu jego kamień
Znalezienie kamienia
1.3.4.1
Dialog podstawowy
Znalazłem kamień
Podejście do Jacka
Podejście do Jacka
Dialog podstawowy
1.3.3.1.1
1.3.3.2.1
Dialog podstawowy
1.3.4.2
Dasz mi kamień, żebym mógł go przekazać Jackowi?
Josh zgubił Twój kamień
Josh przyniesie ci twój kamień
Nie
Tak
Dziękuję za informację
1.3.3.1.2
1.3.3.2.2
Dziękuję za informację
Rozumiem. W takim razie proszę przekaż go Jackowi
1.3.4.3.0
1.3.4.4.0
Dziękuję. Nigdy nie zapomnę tego co dla mnie zrobiłeś
Podejście do Jacka
Podejście do Jacka
Dialog podstawowy
1.3.4.3.1
1.3.4.4.1
Dialog podstawowy
Sprawa załatwiona. Oto twój kamień
Sprawa załatwiona. Josh przyniesie ci kamień
Dziękuję. Nigdy nie zapomnę tego co dla mnie zrobiłeś
1.3.4.3.2
1.3.4.4.2
Dziękuję. Nigdy nie zapomnę tego co dla mnie zrobiłeś
- Jack (postać niezależna)
- Josh (postać niezależna)
x
- numer identyfikacyjny
- Dialog ze strony postaci niezależnej
- Dialog ze strony gracza
- Czynność ze strony gracza

Figure 11. A graph illustrating an exemplary task, complete with abbreviated versions of dialogues and pathways leading to all possible conclusions.

5.2. Relationship changes

The functionality of the designed system, in particular its primary component, the mechanism to sense and transmit emotions, can be elucidated through specific scenarios. For a lucid representation of the changes, it is imperative to adopt the most neutral relationship state, with all weights set to zero, as the initial condition depicted in Fig. 12. In the demonstrative game, details related to disposition, such as trust and respect, are visible, as illustrated in Fig. 13. When an emotion is elicited in a NPC, it modifies its orientation towards the player. Subsequently, this emotion becomes associated with the entity. If this entity identifies another interconnected entity, it will initiate a dialogue, as exemplified in Fig. 14, during which the initially perceived emotion is conveyed according to established guidelines. This cycle persists until the emotion traverses the entire network of connections, or the magnitude of the induced change diminishes sufficiently to surpass a predefined attenuation threshold.

Assuming the baseline relationship state mentioned above and eliciting an extremely positive emotion in a NPC, the outcome is illustrated in Fig. 15. In this scenario, all relationships experience enhancement. However, the farther away the initial emotion source, the more attenuated the introduced change becomes. According to the assumption that the impact of emotions on relationships diminishes over time, the final version of the connection network in the scenario discussed, following the dissipation of the emotion, is depicted in Fig. 16. By introducing an extremely negative emotion into the baseline network, the values obtained would be identical but with an inverse sign.

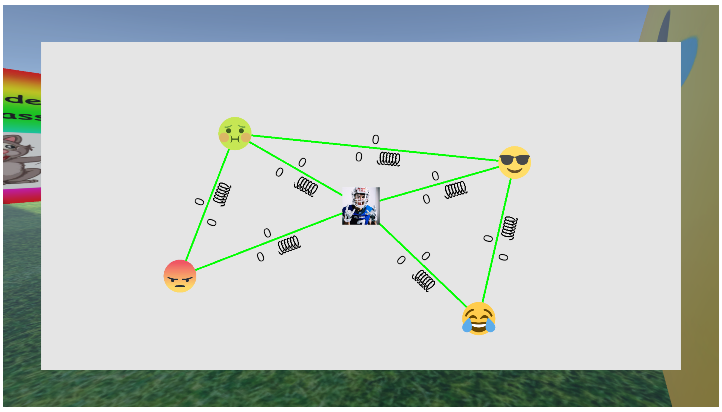


Figure 12. A screenshot delineating the foundational level of relationships within the game.

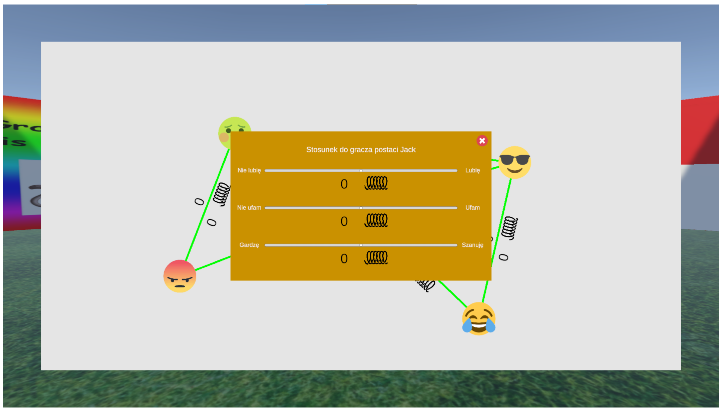


Figure 13. A screenshot elucidating the detailed disposition of an non-player character towards the player, as visualized from the perspective of the relationship graph.

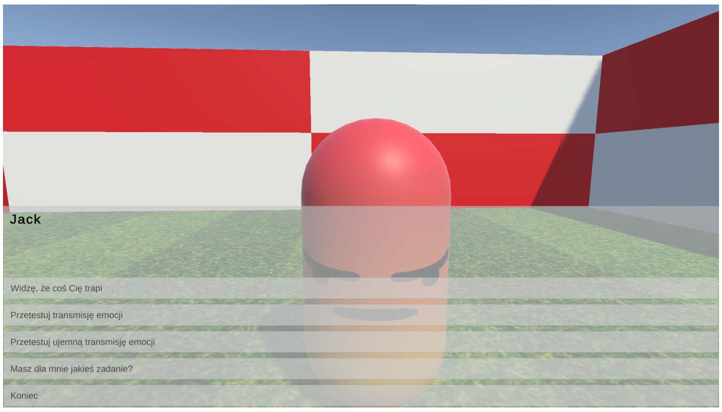

Jack
Widzę, że coś Cię trapi
Przetestuj transmisję emocji
Przetestuj ujemną transmisję emocji
Masz dla mnie jakieś zadanie?
Koniec

Figure 14. A screenshot exemplifying the initiation of dialogue with an non-player character.

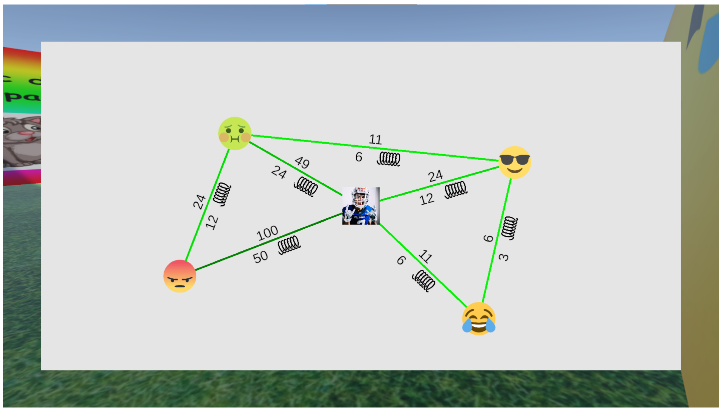


Figure 15. A screenshot presenting the state of the relationship matrix subsequent to the traversal of an exemplary emotion.

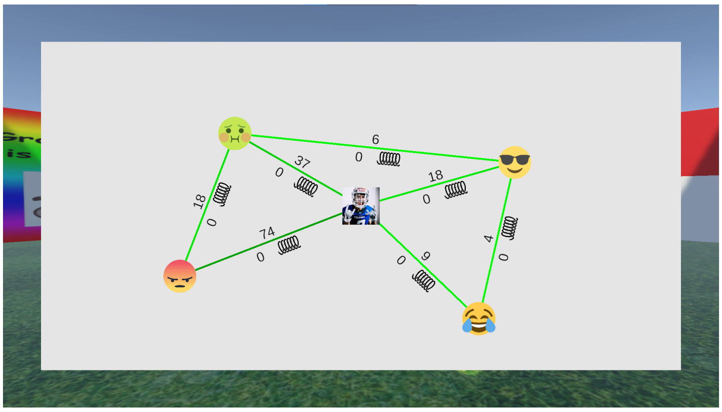


Figure 16. A screenshot illustrating the state of the relationship matrix following the attenuation of the influence exerted by an elicited emotion.

By implementing a minor modification to the baseline relationship state, characterized by a negative disposition of one independent character towards the player, as depicted in Fig. 17, one can observe an altered system behavior. This behavior entails a modification of the transmitted emotion upon reaching a negatively predisposed character. This scenario is illustrated in Fig. 18.

6. Conclusions

The primary objective of this research endeavor was to design a system that augments typical interactions with NPCs by incorporating a semblance of human factors, such as a rudimentary simulation of emotion perception and guidance. This stipulation was realized by establishing several principles that, on the one hand, considerably simplify human behaviors, but on the other, mirror certain dependencies that influence our perception of the surrounding environment.

An avenue for the system's potential development is the optimization of the implemented solutions in terms of readability and execution time. Worthwhile considerations might include the extension of the system with novel functionalities such as compliments and insults, dynamic creation of new relationships, a system that facilitates the design of new narrative elements, or the incorporation of multiplayer gameplay components. Despite numerous prospects for further development, the paramount objective should remain to identify an arena where the entire mechanism can harness its full potential, thereby substantiating its significance. A quintessential example would be the integration of the entire system into a larger project with an intricate narrative trajectory, complemented by professional graphical and auditory presentation, as these elements considerably influence gameplay perceptions. Only under such circumstances can a more precise determination of

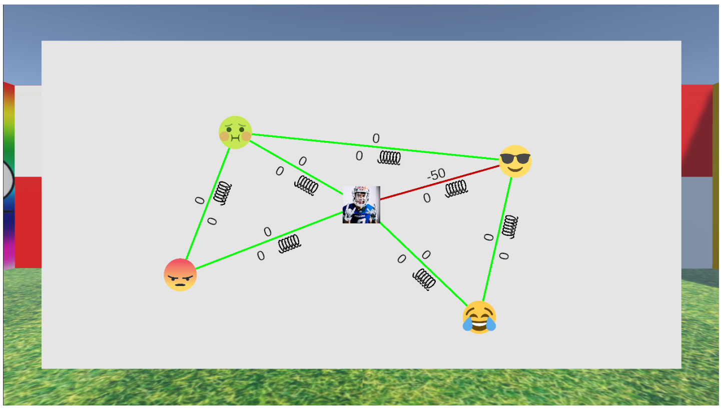


Figure 17. A screenshot illustrating the baseline state of the relationship matrix, taking into account the negative disposition of one independent character towards the player.

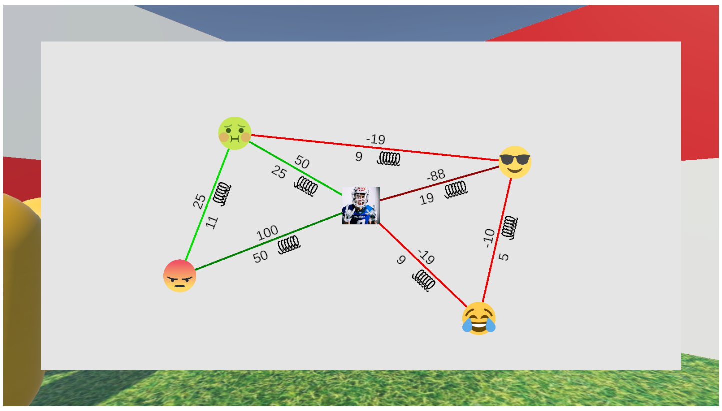


Figure 18. A screenshot depicting the state of the relationship matrix following the propagation of an extremely positive emotion, considering the negative disposition of one independent character towards the player.

the intuitiveness of the chosen solutions and the system's potential to captivate a player be made.

References

[1] Wolf, M. J. 6 genre and the video game. In *The medium of the video game*, pages 113–134. University of Texas Press, 2002.

[2] Apperley, T. H. Genre and game studies: Toward a critical approach to video game genres. *Simulation & gaming*, 37(1):6–23, 2006.

[3] Aarseth, E., Smedstad, S. M., and Sunnanå, L. A multidimensional typology of games. In *DiGRA Conference*. 2003.

[4] Elverdam, C. and Aarseth, E. Game classification and game design: Construction through critical analysis. *Games and culture*, 2(1):3–22, 2007.

[5] Lee, J. H., Karlova, N., Clarke, R. I., Thornton, K., and Perti, A. Facet analysis of video game genres. *IConference 2014 Proceedings*, 2014.

[6] Chowanda, A., Blanchfield, P., Flintham, M., and Valstar, M. Computational models of emotion, personality, and social relationships for interactions in games. In *The 2016 international conference on autonomous agents & multi-agent systems*. 2016.

[7] Saucier, G. and Goldberg, L. R. *The language of personality: Lexical perspectives on the five-factor model*. Guilford, New York, 1996.

[8] Ochs, M., Sabouret, N., and Corruble, V. Modeling the dynamics of non-player characters' social relations in video games. In *AIIDE*. 2008.

[9] Grabska-Gradzińska, I., Porębski, B., Palacz, W., and Grabska, E. Towards a graph-based model of computer games. *Advances in Information Technologies and Communication*, 2012.

The Impact of Content and Composition of Procedural Environment Generation Elements on Player Satisfaction

Bartosz Krukowski, Aneta Wiśniewska[0000−0002−8746−0047], **Rafał Szrajber**[0000−0003−2777−0251]

Lodz University of Technology
Institute of Information Technology
al. Politechniki 8, 93-590 Łódź, Poland
aneta.wisniewska@dokt.p.lodz.pl
rafal.szrajber@p.lodz.pl

Abstract. *Using procedural generated content, developers could not only speed up their work with the design process, but also create unique games with different experiences for every player. This paper describes chosen existing solutions for procedural level generation. Create tools for procedurally generated maps and conduct user research in terms of the dependence of satisfaction and behavior on the amount of details on the map. The effect of the work is a tool for generating underground cave-like levels for FPS games. The study found that the number of objects generated on maps affects player satisfaction.*
Keywords: *computer games, procedural generation, game enviroment*

1. Introduction

Nowadays, both computer game developers and gamers are demanding more and more from new productions. Developers have long been looking for ways to speed up the the game development process. Procedurally created content in games not only shortens development time, but also helps prototype gameplay elements, makes the game often more replayable, and allows the developers themselves to enjoy the game without knowing what to expect during gameplay. A special category of procedurally created content is level generation. Developers of such tools face two challenges. The first - algorithmic, so that the map can be completed, the player can not fall out of it, so that the map layout is interesting to explore. The second - aesthetic, so that the generated maps are nice enough so that the player can not distinguish the level designed and created manually by the developers from the one generated by the computer.The following work tries to answer how the amount of these details can affect the satisfaction of the player and his behavior in the virtual environment.

2. Related works

Procedurally generated levels [1] are increasingly being used in game production because of the reduction in time and effort required in creating levels. However, it comes with some challenges that need to be overcome when creating the tool - the generator [2].

Procedural generation of content is tested by, among other things, by comparing levels created by level designers and procedurally generated ones [3]. Testing is done on various levels - playability or satisfaction. In this work we focus on the latter aspect. The authors [4] in their work compared player satisfaction. The study showed that participants evaluated the version of the game created by the algorithm more positively. They justified this by the fact that the environments were more realistic and aesthetically pleasing. In contrast, in paper [5], the authors did not detect a significant difference in the behavior of players passing levels of the game they created and levels created by the generator. Likewise, the authors of the paper [6] were unable to conclusively demonstrate which version of the levels overall was better. They did, however, note that procedural generation performed better in certain parts of the overall levels.

In the present work, several versions of levels created using procedural generation will be compared. They will be compared in terms of how much satisfaction they give the player.

3. Methodology

The main objective of this work is to create a tool that allows the procedural creation of maps for a computer game, and then study the behavior and satisfaction of users' exploration depending on the amount of detail in the generated levels.

Drawing from selected game examples, the separation of the creation of the room layout from its internal appearance was applied, two algorithms were created to change the room outline to a more interesting one, and a chance for loops in the levels was provided. The implemented process of procedural level generation is divided into five stages. The generator consists of the following stages. The entire generation process took place on a map consisting of square boxes of fixed width.

3.1. Randomize room sizes and positions

The first step is to randomly select room sizes and positions based on a given number of rooms and a limited size. Additional parameters checked in the generation are whether the rooms are created at the right distance from other rooms. With these conditions, the user gets more control over the density of rooms. If on the preset space it was not possible to create the selected number of rooms, due to choosing too many rooms or due to a very sub-optimal arrangement of rooms, the process is repeated.

3.2. Randomization of the appearance of rooms

Each room created is subjected to modification based on the selected room shaping method, the room changes its shape from rectangular to resemble a more irregular form associated with caves. Each room field is then given a height drawn using Perlin noise. This represents the distance of the ceiling from the floor at a given location on the map. Depending on the set level of detail on the map, the room is assigned a random position of decorative objects, such as chests or tents. Depending on the set difficulty level of the map, the room is assigned a specific number of positions with enemies - bonfires.

3.2.1. Box Method

The first of the algorithms used in the tool is the original Box method. It relies on the use of 3x3 squares to fill a given room space. Initially, the entire two-dimensional array representing the room is filled with elements that the player cannot walk on - a solid block. Next, the center of the room is selected and the 3x3 area in the array around it is replaced with a value corresponding to the area on which the player can move - the floor. Then a random spot is selected from the created area and again the 3x3 area around it is filled with the floor. The above steps are repeated a preset number of times. The number of these repetitions depends on the size of the room. An example use of the method is shown in Figure. 1.

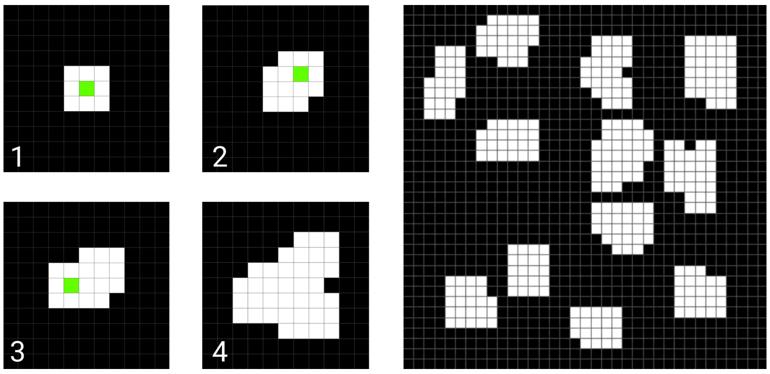


Figure 1. Left: The process of creating a sample room is shown. The green color shows the randomly selected space in the already created area. The last (4th) tile shows the final result of a fully generated room. Right: An example generated map after the second step of the generator using the Box method.

3.2.2. Random Walker

The second method is the Random Walker method [7]. A central point in the room is chosen. This point is the starting place for the algorithm, which visits random fields in its neighborhood in a set number of steps. The visited field becomes a part of the room - such fields will be available for the player to move around. If the algorithm encounters a field already occupied by a room, it can move to it without losing a step from the allocated pool of steps. It is worth noting that when creating rooms using Random Walker, there is a chance for empty spaces in the room to appear. This is an additional advantage, which makes the rooms look more interesting in the game, since there is a column in place of the empty space. An example use of the method is shown in Figure. 2.

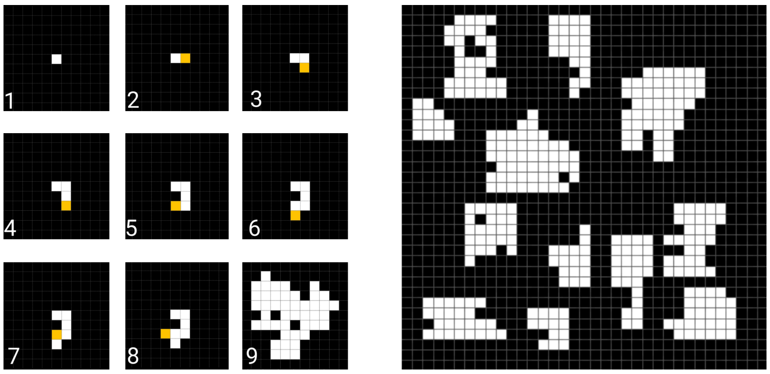


Figure 2. Left: The process of creating a sample room using the Random Walker. Orange color presents the last move of the algorithm. The last (9) tile shows the final result of a fully generated room. Right: An example of the generated map under the second step of the generator using the Random Walker method.

3.3. Connecting rooms by corridors

The generated rooms go into a list, which is sorted by the x and y coordinates of the rooms. Rooms close to the 0,0 point (lower left corner) are closer to the beginning of the list. The room at the beginning of the list is marked as the starting room, and the room at the end of the list is marked as the final room, which the player must reach. The next step is to connect the rooms sequentially in pairs using corridors. The creation of a corridor is done by drawing two points containing one room and the other. The point from the first room is the one from which the algorithm will want to reach the second through a series of commands. The algorithm randomly tries to align the x and y position of point one with point two. Through proper sorting, as well as the random creation of corridors, there are intersections of corridors on the map, and thus natural loops are created.

3.4. Placement of structures: walls, floors and ceilings

Based on the pre-generated positions of rooms and corridors, the fields that build the skeleton of the level are set on the stage in the right place and rotation. The size of the field is also parameterized, so both very claustrophobic rooms and corridors and very large spatial arrangements of caves can be created easily and conveniently.

Each field has a grid of vertices, the number of which can be changed, and the grid itself is subjected to Perlin noise in three axes. Thanks to this treatment, each field is shaped differently and looks like a crude cave wall. Three options for such noise have been implemented in the program, each responsible for a different type of field. For example, the floor tiles have the least distortion in the Z-axis, and the ceiling tiles have the most distortion. In the game, this means that the floor is flatter and the ceilings are more irregular.

3.5. Placement of ornament objects

Based on the generated rooms and the information contained in them about the placement of the items, as well as the selected level of the number of decorations, detail objects are placed on the map. These include interactive objects such as life potions, enemies or ammunition barrels, but also all the smaller decorations on the walls or ceilings, such as stones, plants or crystals. An example of the generator's final results depending on the level of detail can be seen in Figure. 3.

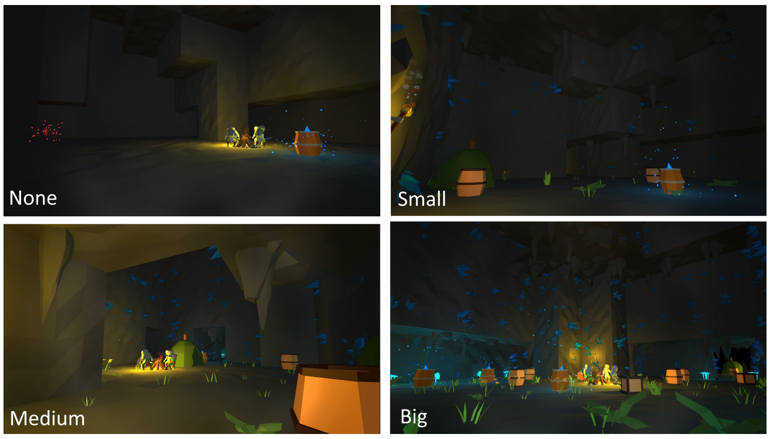


Figure 3. Example rooms created with the level of the amount of ornaments set successively to None, Small, Medium and Big.

4. Procedure

The creation of a tool for the procedural generation of maps with the possibility of partial tracking of the player's behavior and movement made it possible to conduct research toward the influence of the amount of detail on the map on the subject's satisfaction with the game, as well as his behavior [8]. The created tool was combined with an FPS game.

Players had a live updating minimap where they could see their current position, the layout of the rooms, and the endpoint of the level. Respondents were introduced to the rules of the game through panels with information about the game's objective, controls and additional rules. Each participant had a total of four maps to complete. The tutorial map - was the level where players were introduced to the basics of the game. The next three maps, generated procedurally, but with a fixed level of decoration. On Map1 - the Small level, on Map2 - the Medium level, on Map3- the Big level. Each map had a fixed size of 7 rooms.

After passing all the maps, the subjects were to submit a generated file with a record of their behavior, as well as complete a survey about the level they liked best (the tutorial map was not considered). During gameplay, the player's behavior was measured with the following values: time to traverse the map, percentage time standing still, percentage time walking, percentage time running, distance traveled, and the shortest possible path on the map from start to exit. A total of 21 participants took part in the study.

5. Results

For a better understanding of the value of the distance traveled, it is worth adding that the average value of the shortest path from each map is 214 units. In addition, each map had the same number of opponents.

Analyzing the collected data in Table. 1, some conclusions can be drawn. Players, in addition to the three maps on which statistics were counted, had an additional introductory level on which they could get used to the mechanics of gameplay - running, shooting, fighting. Despite this, one can notice a significant drop in transition time and distance traveled after the Map1. This could mean that players then became more familiar with the game and moved through it with even more freedom. If the amount of detail did not affect the player's behavior, one would expect even shorter transition times and distance traveled on the Map3, but this is not the case. We observe an increase in these factors and, in addition, a small increase in the percentage of standing time. Thus, we can conclude that the amount of detail has a significant impact on transition times, distance traveled and overall player behavior.

Observing the parameters, one can come to the observation that players on the

Map3 - the most decorated - spent much more time and walked a much greater distance because the map was the most attractive to them. These conclusions are confirmed by the results of the survey showed in Table. 2, in which the Map3 won by a large margin of 61.9%, as the one that players liked the most.

Table 1. A table showing the results collected during the trials. All values are averages. The values for standing, walking and running are given as percentages from the total time to pass the level.

	Map1 - Small	**Map2 - Medium**	**Map3 - Big**
Time[s]	105.37	74.55	89.77
Standing[%]	17.76	14.28	15.42
Walking[%]	43.26	46.02	38.59
Running[%]	36.38	37.70	42.07
Distance[units]	765.82	525.32	630.81

Table 2. A table showing the percentage breakdown of which map respondents indicated they liked best.

Map1 - Small	**Map2 - Medium**	**Map3 - Big**
9.5%	28.6%	61.9%

6. Discussion

The purpose of the work was to create a tool to procedurally generate levels for a game in the FPS genre, and then test users on the impact of the amount of detail on their game satisfaction and behavior.

The generation was divided into 5 stages, where each stage is separate from the previous one which allows testing and designing a particular stage without worrying about violating the others. The use of two algorithms that change the shape of the rooms made it possible to create unique levels with natural-looking underground rooms.

The results of tests and surveys allow us to conclude that the amount of detail has a significant impact on the player's behavior in the game.

The implementation of the project led to the following conclusions. The separation of the map layout from its appearance greatly facilitates the work of the programmer and designer, as it allows to work on different levels at the same time without conflicting with each other. Generating map and room layouts without using predefined shapes positively affects the look and uniqueness of the level. The amount of details of the environment has a significant impact on the behavior of the player on the procedural map.The biggest change can be observed in the transition time and distance traveled in the case of a map that visually pleases him.

7. Conclusions

The implementation of the present work has contributed to the knowledge, understanding and expansion of the knowledge of procedurally generated content. The presented results could be a prelude to a more in-depth study on how ambient details affect player behavior. An example extension of the work could be the study of how the player's play style is affected by an insufficient amount of raw material, such as ammunition.

The tool has many possibilities for development, examples would be: Adding height to rooms (as in irregular ceiling heights) making the map potentially more interesting in terms of both visuals and gameplay. Adding more wall and ceiling decorations would make these items even less repetitive and unique. Adding more items in the room that allow for interaction. Adding a greater variety of enemies, such as those that attack from a distance. Adding to the tool the ability to create corridors that are dead ends which would make the level even more believable in its objectives.

Acknowledgment

This article has been completed while the second author was the Doctoral Candidate in the Interdisciplinary Doctoral School at the Lodz University of Technology, Poland.

References

[1] Shaker, N., Togelius, J., and Nelson, M. J. Procedural content generation in games. 2016.

[2] Togelius, J., Champandard, A. J., Lanzi, P. L., Mateas, M., Paiva, A., Preuss, M., and Stanley, K. O. Procedural content generation: Goals, challenges and actionable steps. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[3] Rodrigues, L., Bonidia, R., and Brancher, J. Procedural versus human level generation: Two sides of the same coin? *International Journal of Human-Computer Studies*, 141:102465, 2020.

[4] Korn, O., Blatz, M., Rees, A., Schaal, J., Schwind, V., and Görlich, D. Procedural content generation for game props? a study on the effects on user experience. *Computers in Entertainment (CIE)*, 15(2):1–15, 2017.

[5] Rodrigues, L. and Brancher, J. Procedurally generating a digital math game's levels: Does it impact players' in-game behavior? *Entertainment Computing*, 32:100325, 2019.

[6] Connor, A. M., Greig, T. J., and Kruse, J. Evaluating the impact of procedurally generated content on game immersion. *The Computer Games Journal*, 6:209–225, 2017.

[7] Xia, F., Liu, J., Nie, H., Fu, Y., Wan, L., and Kong, X. Random walks: A review of algorithms and applications. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(2):95–107, 2019.

[8] Desurvire, H. and El-Nasr, M. S. Methods for game user research: studying player behavior to enhance game design. *IEEE computer graphics and applications*, 33(4):82–87, 2013.

First Edition of Summative Module for Network Infrastructure and Applications: The Ideas, the Reality and the Conclusions

Marcin Kwapisz[0000−0001−5128−9009],
Michał Karbowańczyk[0000−0003−3875−1101],
Mateusz Smoliński[0000−0002−3890−5943]

Lodz University of Technology
Institute of Information Technology
al. Politechniki 8, 93-590 Łódź, Poland
marcin.kwapisz@p.lodz.pl
michal.karbowanczyk@p.lodz.pl
mateusz.smolinski@p.lodz.pl

Abstract. *In this paper we present our concept and practical experience taken from the process of implementing Summative Module course into existing structure of the Network Infrastructure and Applications specialty programme. We try to determine if the way we have taken led both us - teachers and our students to the satisfactory finish point, that is, the new module meets its conceptual goals and provides students with broader team project experience. The conclusions are drawn from the day-to-day issues that were raised during the course, measurable factors i.e. the data on what voluntary activities were taken by students within the project, and various forms of students' feedback. Finally our general considerations regarding future editions of the course are shown.*

Keywords: *summative course, team projects, learning by doing, programme design*

1. Introduction

The Summative Module (referred to as SM) is a brand new obligatory element of the programme of each of the first-cycle studies at Lodz University of Technology. The aim of this course is to implement a team project by students, which would cover as broad as possible the competences acquired by students in the course of their studies. From a methodological perspective the SM should complete the Kolb's cycle[1], moreover, it should meet the Criterion I.5.d from ABET Criteria for Accrediting Engineering Programs[2]:

> culminating major engineering design experience that 1) incorporates appropriate engineering standards and multiple constraints, and 2) is based on the knowledge and skills acquired in earlier course work

According to the SM course curriculum[3] it establishes the following requirements:

- The indicated problem includes issues related to the field of study and verification of the achievement of learning outcomes planned for a given study programme.
- Project implementation includes elements of project management, basics of software life cycle analysis and elements of standardization.
- Project implementation includes the phase of defining the problem and searching for information, as well as project implementation.
- Students work in groups of 3-5 people under the supervision of a teacher or academic teachers and/or with the participation of other teachers or practitioners - depending on the requirements of a given project. Teamwork includes project meetings and seminars with and without the participation of a supervisor.
- The substantive issues are adapted to current trends and technologies, in particular issues that are socially important or constitute technological challenges or constitute real problems that students encounter in their close environment.

2. Deployment of SM in the NIA specialty programme

From the perspective of the Network Infrastructure and Applications (referred to as NIA) specialty programme, implementing SM was both relatively easy and relatively difficult. This phenomenon is raised by the fact that the specialty programme was created from the very beginning with the aim of interconnecting the courses, so that during the sixth semester, a team project could be carried out as part of the Network Database Systems (referred to as NDS) course.

1. Infrastructures of Development and Production Environments (IDPE) (V)
2. Fundamentals of Network Applications (FNA) (V)
3. Introduction to Mobile Systems (IMS) (V)
4. Network Database Systems (NDS) (VI)
5. Principles of Internet Systems' Security (PoISS) (VI)
6. Basics of Virtual Systems (BVS) (VI)
7. Technologies of Network Components (TNC) (VI)

These courses have been designed so that after completing them, the student will be able to:

- design and build infrastructure to run a scalable and distributed network IT system. This infrastructure can be physical, virtual or containerized
- design, build and implement a distributed network IT system in a given infrastructure, using a relational or non-relational database for its operation. According to the assumptions developed by companies such as Amazon [4][5], the database cannot be accessed directly but must be made available via a network API, e.g. REST API [6]
- design, build and implement an application client of a distributed IT system, which can be a standard web application running in a browser or an application for mobile devices. We should not forget about IoT systems, which do not need a user interface for their operation, but must communicate with a distributed network IT system operating in the network that provides a communication interface for them. IoT systems are distributed systems by nature.

As one can see on Figure 1., before SM introduction the NDS course played two roles. First, it was a continuation of the Web application / REST service development path, introducing advanced data processing aspects such as object-relational mapping [7][8], concurrent execution and online transaction processing[9][10]. Then, as a team software development project, it required students to utilize their experience in the fields of Continuous Integration / Continuous Delivery[11] and

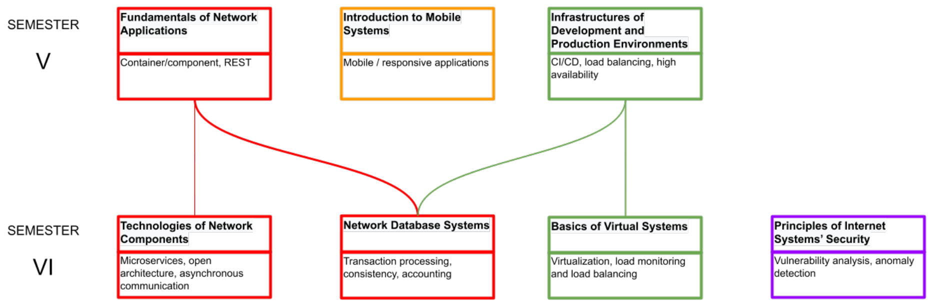

SEMESTER
V
SEMESTER
VI
Fundamentals of Network Applications
Container/component, REST
Introduction to Mobile Systems
Mobile / responsive applications
Infrastructures of Development and Production Environments
CI/CD, load balancing, high availability
Technologies of Network Components
Microservices, open architecture, asynchronous communication
Network Database Systems
Transaction processing, consistency, accounting
Basics of Virtual Systems
Virtualization, load monitoring and load balancing
Principles of Internet Systems' Security
Vulnerability analysis, anomaly detection

Figure 1. Structure of NIA programme before SM introduction. Line thickness indicates assumed relationship strength.

project management systems such as Atlassian[12] Jira project management, Bitbucket repositore and Confluence collaboration software, gained during the previous semester. This, hovewer, didn't come with no cost. The NDS course was widely considered to be very exhaustive for the participants, with real workload significantly exceeding the nominal amount (150-180 hours) resulting from the assigned ECTS points (6).

Since we saw no point in either destroying a proven structure or doubling the student workload, which was already considered high, we decided to leave NDS course along with new SM course. NDS project scope was reduced to its core described above, ceasing other aspects to the SM; in other words, NDS project became the core subsystem of the SM project. In addition, the project implementation scheme used so far has been extended to include the following components:

1. The scope of the project's functionality (domain model) was determined by the lecturers participating in SM as project stakeholders (before this the domain model was proposed and defined by the students). This change was implemented on the basis of conclusions drawn from the implementation of the Lodka Wolontariatu project by students of the previous edition of NIA.

2. A broad variety of extensions, both functional and technological, was proposed to be chosen by team mebmers. The extensions are optional, however, leaving only core expectations to be met results in barely satisfactory grade.

3. The teams were required to configure and maintain deployment infrastructure, so that the system is publicly accesible. This, in conjunction with ex-

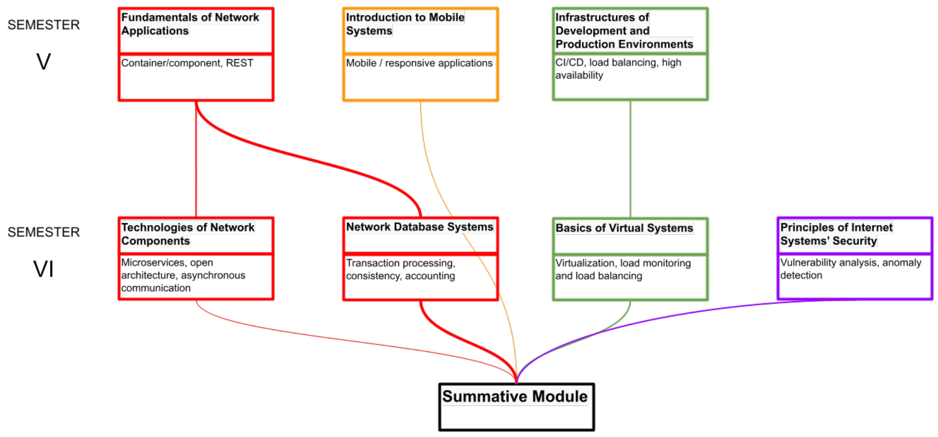

SEMESTER V
Fundamentals of Network Applications
Container/component, REST
Introduction to Mobile Systems
Mobile / responsive applications
Infrastructures of Development and Production Environments
CI/CD, load balancing, high availability
SEMESTER VI
Technologies of Network Components
Microservices, open architecture, asynchronous communication
Network Database Systems
Transaction processing, consistency, accounting
Basics of Virtual Systems
Virtualization, load monitoring and load balancing
Principles of Internet Systems' Security
Vulnerability analysis, anomaly detection
Summative Module

Figure 2. Structure of NIA programme after SM introduction. Line thickness indicates assumed relationship strength.

tensions described above, allowed SM to cover the whole scope of NIA courses and fulfill its summarizing task, as depicted in Figure 2.

The teachers, as project stakeholders, took on the role of Product Owner[13]. The responsibility of the PO was to present his vision of the project, make key design decisions and influence the directions of its development, and encourage students to experiment with various solutions to problems that occurred during implementation. The teachers proposed project topics to meet the requirements of the course:

> "in particular issues that are socially important or constitute technological challenges or constitute real problems that students encounter in their close environment"

- Online pharmacy (M. Karbowańczyk)
- Online furniture store (P. Milczarski)
- Management system for an amateur volleyball league (A. Hłobaż)
- Water budget in a multi-family building (M. Smoliński)
- Heat settlement for premises in many buildings (M. Smoliński)

- eCSO for members of the housing community (M. Smoliński)

The extensions were proposed by the heads of courses belonging to the fifth and sixth semester NIA programme. A total of 20 extensions were created, varying in topic and difficulty, from which students could choose up to 5. Completing the extension allowed for an increase in the final grade.

3. Project stages

3.1. Stage 1

This stage was devoted exclusively to the preparation of the project - agreeing its domain model and scope of functionality with the Product Owner. The students positively assessed the new experience of having to understand a business model that was not known a priori. However, it turned out that two significant issues affected this stage. The complexity of the domain models was too diverse between individual projects, which in the later stages resulted in undesirable differences in the workload of students. Moreover, it was a mistake to assign more than one project to one lecturer, which from the students' perspective resulted in difficult cooperation compared to other lecturers. Both issues were clearly pointed at in students' feedback.

3.2. Stage 2

This stage was devoted to implementing functionality related to application accounts, authentication and access control. In this respect, the requirements of individual projects were unified. The implementation of this stage required mastering the entire dedicated technology stack and the configuration of both the development and production environments. In addition, at least REST service integration tests covering whole implemented functionality were necessary to define and perform. This stage was unanimously indicated by the students as the most difficult in terms of acquiring new competences.

3.2.1. Stage 2 - extensions

All extensions available at Stage 2, along with their number of choices, are presented in Table 1.

Table 1. Extensions - Stage 2

Extension	**Number of choices**
Notification about account activation/unblocking/blocking via a message sent to the e-mail address assigned to the user account	6
Providing of container-based testing (development) infrastructure for the project	5
Mocking test components providing an alternative test configuration	3
Using the recaptcha mechanism when registering an account and changing your account details (including changing one's password)	
Two-phase authentication using one-time codes delivered via email	2
Implementation of optional authentication via an external OAuth service	1
Configuring your own authentication OAuth server, e.g. Keycloak	0
Dynamically assigned level of access to the user account after meeting certain conditions based on account activity statistics, which provides additional functionalities	0

3.2.2. Stage 2 - conclusions

As the requirements set in stage 2 were identical for all projects, the above-mentioned differentiation of domain models did not have a major impact on the uniformity of the student workload. However, the statistics regarding the selection of extensions (Table 1.) at this stage allow us to draw two conclusions. First of all, the available extensions were used by students to a very small extent - about 1/3 of the expected number. Moreover, the popularity of extensions related to automated tests is interesting, which is a significant difference compared to the low interest in this topic noted in other courses, even though it is a widely recognized industry standard[14]. This phenomenon is probably related to the fact that SM implementation is students' first experience of real work in a project team. This indirectly proves the validity of introducing the requirement for students of all engineering fields of study to complete team projects.

3.3. Stage 3

Stage 3 was intended for the implementation of functionality related to the individual topic of the project - its domain model. In principle, this stage should be relatively simple in terms of technological problems, which should be fully solved at stage 2. Therefore, at this stage, a number of extensions related to the system architecture, alternative implementation models, communication models, UX/UI extensions or security tests of the entire system were proposed.

3.3.1. Stage 3 - extensions

All extensions available at Stage 2, along with their number of choices, are presented in Table 2.

3.3.2. Stage 3 - conclusions

What is striking about the statistics for stage 3 is the minimum number of extensions selected (3). This number is so small that there is no point in analyzing the popularity of individual extensions. The ad hoc discussion conducted after the discovery of this phenomenon, as well as the comments in the survey conducted after the project's completion, clearly indicate that the reason is not the students' lack of interest in extensions as such (they were considered interesting), but a simple lack of time caused by the excessive burden of implementing the functionality.

Table 2. Extensions - Stage 3

Extension	**Number of choices**
Mobile application covering external user functionality	1
Building a continuous integration/continuous deployment infrastructure	1
The application's use of images or other binary resources stored in a dedicated storage space and provided by the server with static resources	1
Storing information about completed (key) activities performed by an external user in a separate database in order to present statistical data, use of microservices and asynchronous communication	1
Extending the development infrastructure with load balancing/redundancy of the application server	0
System performance tests	0
Functional tests (user interface in SPA application)	0
Use of HATEOAS for automatic adaptation of the REST client in the SPA application to REST API changes	0
Actively notifying the user about a selected event regarding his account (e.g. blocking/unblocking)(for SPA applications)	0
Allowing a user to subscribe to active notifications sent to all subscribers by internal user	0
Conducting a final security audit	0
Expanding the mobile application with inclusivity for people with low vision	0
Enabling anonymization of an external user's personal data and transferring his (key) activity history to a separate database	0
Provide automatic, dynamic translation of the selected text attribute of an object	0

This weakens the role of SM as a project covering competences from all courses of the NIA specialty programme (for example, the extension consisting in performing penetration tests in accordance with the methodology introduced in the PoISS course was not implemented at all) and should be eliminated for subsequent editions of SM.

4. Post-course survey

As a standard at the NIA specialty, after each semester, a survey is conducted to examine students' opinion. In the case of the SM survey, 21 students out of 39 enrolled participated in it. Taking into account the average level of participation in systemic surveys at TUL, the level of participation should be considered high. Thanks to this, it was possible to avoid the common problem of a biased survey, in which the participants are almost exclusively students who are extremely dissatisfied with their participation in the course. The survey results are presented in Tables 3. and 4.

Two general conclusions can be drawn from the numerical assessments presented in the tables. The reception of the SM course by students is generally positive, most grades are at least 4 on a scale of 1-5. This allows us to believe that the general concept of the course and its implementation are appropriate. However, the low assessment of the connection between the SM course and some other NIA specialty courses (IMS, BVS, PoISS) raises concerns. This means that SM only partially fulfilled its role presented in Figure 2. This result is not surprising, as a similar conclusion was drawn based on small number of extensions selected by students in stage 3.
In the light of the conclusions drawn earlier, it is not surprising that most of the students' open responses concerned workload and time optimization. A suggestion often addressed to teachers is, first of all, unification, and secondly, limiting the complexity of domain models of implemented projects in order to leave more time for interesting technological extensions. This aspect was also reflected in suggestions for next generations of students, where great emphasis was placed on systematicity and organization of teamwork. The students also paid attention to difficulties they met while establishing team cooperation practices, also in psychological aspects, and suggested that mentoring support from experienced team leaders operating in software development businesses would be very valuable. Another suggestion, also related to optimizing time use, is to limit the volume of doc-

Table 3. Survey results

Assess individual aspects of the course in the context of your previous experience and your career prospects Scale 1-5:	**Results**
Do you consider the time devoted to the course to be well spent?	4.0
Have you previously been interested in the course matter?	4.3
Did completing the course expand your knowledge and skills?	4.3
How do you assess the usefulness of the acquired knowledge and skills in your future professional career?	4.0
How do you assess the relationship between the burden of completing the course and the usefulness of the acquired knowledge and skills?	3.6
Do you think the volume of the material is large? (from 1: there should be definitely more material to 5: there should be definitely less material)	3.5
Reflect on your course grade	**Results**
Do you think the grade you received was fair?	4.7
Are you satisfied with the grade you obtained?	4.5
Do you think that the grade obtained is the result of a substantive assessment of your competences?	4.2
Assess whether completing the course allowed you to gain new competences / experience in technical aspects	**Results**
Programming (general)	3.9
REST services development	4.0
SPA applications programming	4.4
Authentication and access control	4.4
Transaction processing, consistency guarantees	4.5
Deployment environment	4.0
Testing	3.7
Assess whether completing the course allowed you to gain new competences / experience in context of teamwork	**Results**
Using teamwork tools	4.3
Division of workload within a team	3.9
Agreeing on common standards, conventions, solutions	4.0
Mutual verification	4.0
Solving non-trivial problems cooperatively	4.0

Table 4. Survey results continued

Assess whether completing the course allowed you to gain new competences / experience in personal aspects	**Results**
Task and time planning	4.0
Keeping commitments made	4.0
Communication with the team	4.2
Application of agreed standards, conventions	4.0
Submitting to substantive evaluation of your own work	4.1

Reflect on how the project was prepared by teachers	**Results**
Was the content of the tasks understandable?	3.8
Were the assessment rules understandable?	4.0
Were any additional materials (instructions, etc.) sufficient to complete the tasks?	3.7
Has the infrastructure (including services and software) necessary to carry out the tasks been provided?	4.5

Reflect on how the project was run	**Results**
Was the teacher prepared to conduct the project?	4.4
Was the class time effectively used by the teacher?	3.8
Was the teacher open to discussions?	4.7
Did the teacher provide assistance in completing the tasks?	4.6
Was the assessment of the tasks fair compared to other assessments in the group?	4.8

Assess whether and how the content of the course is related to the content of other courses. Scale from 1 - undesirable connections (e.g. repetition) to 5 - desirable connections (e.g. continuation, supplementation)	**Results**
Fundamentals of Network Applications (FNA)	4.5
Infrastructures of Development and Production Environments (IDPE)	3.7
Non-Relational Databases (NRDB) (note: this course was attended by all NIA students by choice)	3.8
Introduction to Mobile Systems (IMS)	2.6
Network Database Systems (NDS)	4.7
Technologies of Network Components (TNC)	3.7
Basics of Virtual Systems (BVS)	2.4
Principles of Internet Systems' Security (PoISS)	2.7

umentation produced as part of the project. Moreover, the students indicated that they would be more satisfied with the project if they had the freedom to choose the technology stack, which in the current SM practice is strictly defined as one of the main requirements.

5. Conclusions and future plans

The results of SM projects and course evaluation surveys indicate that the assumed general concept of this course and its location in the NIA specialty programme turned out to be correct, so it will be continued in subsequent iterations. This does not mean, however, that we do not see any scope for future improvement.

The most serious problem seems to be the ineffective use of students' potential by overburdening them with the implementation of too extensive domain models. This phenomenon was recorded from various perspectives: day-to-day observations, statistics on the choice of technological extensions, or directly from students' responses in the survey. Also from the perspective of learning outcomes, implementing a large number of similar use cases does not constitute any added value. Therefore, it is necessary to direct the potential of students towards deepening their competences through the implementation of many technological extensions. A related problem is the variation in the complexity of the domain model between projects conducted by different teachers and, at the same time, different workload and, consequently, availability of teachers due to the different number of projects conducted. This has led to unequal working conditions for individual teams of students, which is in stark contrast to the principles we try to follow in our work. Therefore, for subsequent editions, we must prepare more carefully to act as product owners.

We also find various proposals submitted by students worth consideration. For example, mentoring support in organizing teamwork could be a valuable experience for students and we intend to provide it in cooperation with Lodz ICT Cluster[15]. This does not mean, however, that every student suggestion will be uncritically taken into account. For example, significantly reducing the scope of project documentation would certainly be attractive from the students' point of view, as it would noticeably reduce the workload of the project. However, the goal of SM and classes in general is not per se to create software, but to achieve the learning outcomes assumed for the course, and the SM project documentation produced by students is to certify this. Similarly, the postulate of allowing free choice of technology stack for project implementation seems attractive at first glance due to the additional experience associated with the need for students to make this choice. However, our experience in project implementation indicates that the diversity of the execution environment, for example at the level of the programming language, inevitably leads to undesirable differences in the level of difficulty and workload of projects, and in extreme cases even to the - difficult to predict - inability to achieve certain learning outcomes. Moreover, excessive focus on the specific features and problems associated with specific programming languages is inevitably going to obscure universal concepts that should be learned by doing[16]. Therefore, any allowance for variation in the runtime environment must be preceded by careful consideration and introduced gradually.

Acknowledgment

The authors would like to thank HTD Polska and Idemia Poland for their support in the course of the project.

References

[1] Kolb, D. A., Boyatzis, R. E., and Mainemelis, C. Experiential learning theory: Previous research and new directions, in in perspectives on thinking, learning and cognitive styles. In *Perspectives on Thinking, Learning, and Cognitive Styles*. Routledge, 2001.

[2] ABET. Criteria for accrediting engineering programs, 2022 – 2023, 2022. `https://www.abet.org/wp-content/uploads/2022/01/2022-23-EAC-Criteria.pdf` [Accessed: September 2023].

[3] Lodz University of Technology, ECTS Course Catalogue, 2023. `https://programy.p.lodz.pl/ectslabel-web/` [Accessed: September 2023].

[4] Ward, T. Enabling data persistence in microservices. *AWS*, 2021.

[5] Marietti, A. The API Mandate: How a mythical memo from Jeff Bezos changed software forever, 2022. `https://konghq.com/blog/enterprise/api-mandate` [Accessed: September 2023].

[6] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Ph.D. thesis, University of California, 2000.

[7] The Eclipse Foundation - Jakarta Persistence Team. Jakarta Persistence 3.1, 2022. `https://jakarta.ee/specifications/persistence/3.1/jakarta-persistence-spec-3.1.pdf` [Accessed: September 2023].

[8] Martin Lorenz, G. H. and Rudolph, J.-P. Object-relational mapping revised - a guideline review and consolidation. In *Proceedings of the 11th International Joint Conference on Software Technologies (ICSOFT 2016*, volume 1, pages 157–168. SCITEPRESS, 2016.

[9] Bernstein, P. A. and Newcomer, E. *Principles of Transaction Processing, 2nd Edition*. Morgan Kaufmann, 2009.

[10] El-Sayed, N., Sun, Z., Sun, K., and Mayerhofer, R. OLTP in real life: A large-scale study of database behavior in modern online retail. In *2021 29th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 1–8. 2021.

[11] Shahin, M., Ali Babar, M., and Zhu, L. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access*, PP, 2017. doi:10.1109/ACCESS.2017.2685629.

[12] Atlassian. Powerful products for every team, 2023. `https://www.atlassian.com/software` [Accessed: September 2023].

[13] Schwaber, K. and Sutherland, J. The definitive guide to scrum: The rules of the game, 2020. `https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf` [Accessed: September 2023].

[14] Myers, G. J., Badgett, T., Thomas, T. M., and Sandler, C. *The art of software testing*, volume 2. Wiley Online Library, 2004.

[15] Lodz ICT Cluster website. `https://ictcluster.pl/` [Accessed: September 2023].

[16] Reese, H. W. The learning-by-doing principle. In *Behavioral Development Bulletin*, volume 17, pages 1–19. Joseph D. Cautilli, 2011.

Steering Behaviors for Vehicle Movement in Racing Computer Games

Adrian Majzer, Dominik Szajerman[0000−0002−4316−5310]

Lodz University of Technology
Institute of Information Technology
Politechniki 8, 93-590 Łódź, Poland
dominik.szajerman@p.lodz.pl

Abstract. *The goal of this work was to design the method for simulating the movement of racing vehicles in 3D space using steering behavior algorithms. The resulting way of moving vehicles should be suitable for later use in any racing game as artificial intelligence for opponents. Some well-known bahaviors were adapted: seek, path following, collision avoidance, and new ones were proposed: cornering braking, blocking, overtaking, and reacting to stucks. Moreover, a method of combining them was proposed based on the identification of mutually exclusive behaviors. The resulting agents were tested by proposing a physical model of the vehicle and its steering method using a set of parameters: throttle, steer, brake, and hand brake.*

The results show that the proposed method allows agents not only to correctly but also credibly drive vehicles in the game world.

Keywords: *steering bahaviors, vehicle modeling, car racing games*

1. Introduction

Driving algorithms for computer opponents in computer games have been developed since the beginning of the computer games genre known as racing games. The basic problems known here are keeping the vehicle on the track, controlling speed, and avoiding collisions. Over time, methods for solving particular problems become more and more complex. There are new aspects of driving a vehicle, its physical properties, weather factors, etc.

Steering behaviors are general algorithms that relate to the movement of entities in a computer game world. Vehicles are one of the types of entities whose physical movement cannot be translated directly into the movement of, for example, a human character. For this reason, specific steering behavior solutions for vehicle movement can be addressed.

In this work several simple steering behaviors have been adapted for racing games. In addition, new behaviors specifically useful in such games have been proposed, as well as a way of combining them.

2. Related Work

One of the solutions similar to the one proposed in this work is described in [1]. The article describes a racing game called Racer, in which the player, driving one of the cars, must compete in a race with AI-controlled vehicles. The vehicles are controlled along a path consisting of points whose positions are defined in 3D space. The AI in Racer also tries to appropriately select the speed by accelerating and braking, and also avoids collisions with other game objects by using a cuboid-shaped collision block that surrounds the controlled vehicle.

The work [2] shows a solution specific to racing games. At the same time, it is very generalized, which means that the steering behaviors retain their stateless character, and at the same time the solution scales well when introducing new ones. In this solution, however, it is necessary to determine whether a given behavior generally attracts or repels the agent. Also, creating the “interest maps” and “danger maps” used there may result in a large computational overhead compared to the basic solution.

An interesting continuation of this work is [3], where the problem of selecting parameters depending on the environment is solved thanks to the evolutionary algorithm. There are three categories represented by fitness functions: Target, Danger, and Path. Agents were characterized by high robustness in various variants of the game environment.

Steering behaviors [4] allow for quick and relatively easy implementation of agent movement in the world of a computer game. The implementation is based on simple calculations on vectors and stateless objects representing individual behaviors. The most basic behaviors include: seek, flee, pursuit, evasion, offset pursuit, arrival, obstacle avoidance, wandering, path following etc. It is also possible to create flock behavior and combine behaviors in various ways.

3. Method

3.1. Physical basis of vehicle movement

A vehicle model currently standard in computer games was used, which is set in motion thanks to the rotation of the wheels to which torques are applied. Additionally, two more factors acting on the car were taken into account in the vehicle modeling: aerodynamic drag and downforce.

The force of air resistance is given by the formula:

$$F_x = c_x \frac{v^2 d}{2} S \tag{1}$$

where: F_x - drag force $[N]$, $c_x = 0.34$ - aerodynamic drag coefficient for the chosen vehicle, v - vehicle speed $[m/s]$, $d = 1.185 kg/m^3$ - air density in which the vehicle is moving, $S = 1.99m^2$ - frontal area for the chosen vehicle [5].

The second additional force is the downforce. Here, an approximation is used. Its value is the product of the constant downforce value (one of the parameters of the car model) and the length of the current vehicle velocity vector. The resulting downforce depends on the speed of the car. Although this solution is very simple, it is sufficient for the needs of the physical model being created, and adding this force significantly improved the stability of the vehicles.

The engine of the car is modeled using a torque curve. And the vehicles use an automatic transmission in which gears are changed after reaching specific engine speeds. The automatic transmission works in such a way that when the revs drop too low, the gear is changed to a lower gear, and when the revs are high enough, the gear is changed to a higher one.

3.2. Adapted behaviors

One of the basic steering behaviors is seeking. It consists in directing the current velocity vector of the controlled vehicle to its target. The algorithm accepts information about the vehicle's position and the target's position as input. At the output, the algorithm returns the difference between the desired speed vector and the current vehicle speed vector, i.e. the force vector that allows the object to bend the motion path towards its target.

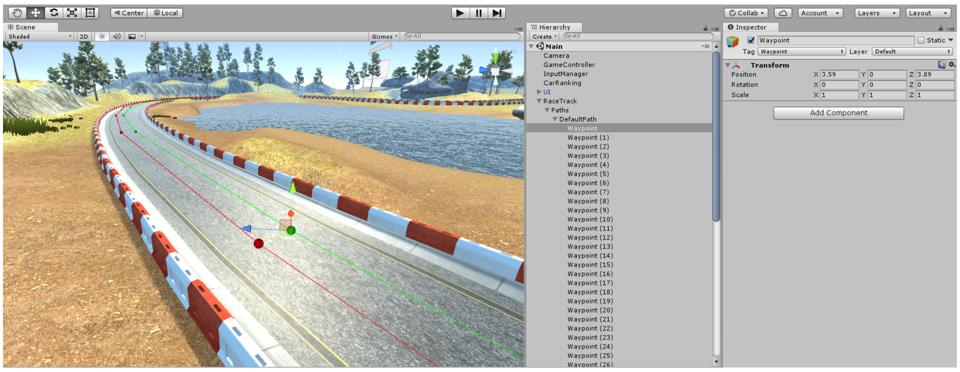
In our method, the search behavior for a car is reduced to positioning the front wheels towards the target at which the vehicle should be directed.

In order for the car to be able to move towards the target, and not just turn the wheels towards it, the level of the gas pedal must be set. In the case of the seek behavior, the controlled object would move at maximum speed all the time. This would mean that pressing the gas pedal would simply set it to 1, the maximum value. However, a driver cannot keep the pedal pressed all the time, as this would sooner or later result in him going off the path. Therefore, the acceleration is controlled together with the braking behavior, which is presented later.

Path following is one of the most important steering behaviors used in racing games. It involves moving vehicles along a previously defined path. It should be noted, however, that the controlled vehicles do not have to strictly stick to the designated path, but only move while remaining at a designated distance from its center.

A path can be represented in various ways. A simple way to represent a path is a polyline, i.e. a line composed of interconnected segments. In other words, it is simply a set of N waypoints, from which subsequent pairs of points p_i and p_{i+1} (where $i = 0, 1, ..., N - 1$) state sections forming a path. A path can be open when it has a beginning and an end, or closed when its beginning is also its end. Proces of path editing is shown in Figure 1.

In each frame, the behavior algorithm checks whether the vehicle is currently



Waypoint Static
Tag Waypoint Layer Default
Transform
Position X 3.59 Y 0 Z 3.89
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Figure 1. Path editing using the Unity engine editor (own work[1])

controlled by a *collision avoidance* behavior or a *blocking* behavior that precludes control by a path following behavior. If other behaviors do not interfere, the car's wheels are directed to the currently selected point on the path using the *seek* behavior. The next step is to check whether the distance between the car and the currently selected point is smaller than the previously defined distance allowing it to change the destination to the next one. This distance is the only parameter for path following behavior.

Obstacle avoidance behavior, as the name suggests, involves avoiding collisions with other game objects that are in the path of the controlled vehicle.

Detection can use raycasts, i.e. projection of rays. It involves checking whether a ray released from a specific point in a specific direction in space will hit an object. The length of the rays should depend on the speed at which the vehicle is moving. In the case of a large object such as a car, one beam detecting obstacles is not enough and several of them are needed.

For the needs of our algorithm, 7 rays are created. The arrangement of all the rays can be seen in Figure 2. These are 2 side rays (purple), 2 front lateral obliques (orange), 2 front lateral (blue) and 1 front central (red).

If an object is hit by a ray, an information about the direction of the ray and the intensity of the impact is considered. For the lateral and front lateral oblique rays, this value is smaller than for the others. The obstacle detected by them is not perfectly located in front of the vehicle and can only hit it from the side, so less wheel rotation is needed than in the case when the obstacle is detected in front of the vehicle. Additionally, if the object detected by the ray is a vehicle, the *overtaking* behavior is initiated, which is described next.

The central front ray is projected at the very end because it is different from the others. It cannot be used to simply determine which direction the car should turn to avoid an obstacle. Therefore, if the obstacle is detected only by the central ray,

[1]Presented scene contains free asset from Unity Asset Store "Lake Race Track" by NIANDREI.

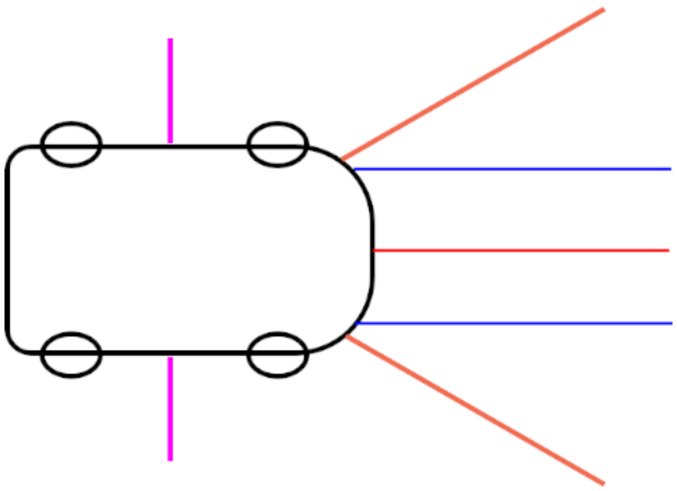


Figure 2. Visualization of rays emitted from a car to detect obstacles (own work)

or the impact intensities on both sides cancel out, the vector normal to the surface of the hit object is checked. On its basis, the turning direction is set. Moreover, if the obstacle is centrally in front of the car and the speed of the car is too high, it is probably impossible to avoid it without braking. Therefore, when the car's speed exceeds the set avoidance speed, *braking* is initiated.

After the above computations, it is known what to avoid and how to avoid it. All that remains is to check whether avoidance is possible, i.e. whether the vehicle's direction of movement is not too far away from the direction of the path. Otherwise, collision avoidance behavior could cause the car to go off the path because the car does not steer to the next point on the path while avoiding a collision. If the angle between the path direction vector and the vehicle movement direction vector is greater than the maximum deflection angle (which is a parameter of collision avoidance behavior), the obstacle will not be avoided because it is more important to stay on the track (*path following* behavior).

It is worth noting that collision avoidance behavior is not perfect. This means that not in every case the controlled car will be able to avoid a collision with an obstacle. However, also in real life, the driver is not always able to react quickly enough to avoid a collision. Therefore, this effect is acceptable and even desirable from the point of view of agents in a computer game.

3.3. Behaviors introduced

There is little information about the behavior of **cornering braking** in commercial games. This is probably due to the fact that this is not a universal behavior, unlike the algorithms discussed earlier. All basic control behavior algorithms can be used to control any character. However, the braking behavior only applies to vehicles. However, it is very important because they cannot drive on a race path at high speed without braking at all.

The algorithm involves comparing the difference between the car's direction and the direction of the race track (i.e. the path it follows). Then, based on this difference, the appropriate braking force of the car is selected. The greater the difference, the more the vehicle must be braked, and if the difference is small or zero, the vehicle can accelerate freely.

A corner angle is determined between the vector in the direction in which the path leads and the vehicle's velocity vector (the velocity vector has the direction of motion of the car). The spinning angle is calculated. It is the length of the vehicle's angular velocity vector multiplied by a factor determining the degree of caution to be exercised when the vehicle's angular velocity increases. An increase in angular speed means a loss of grip, which leads to oversteering, i.e. the rear wheels "running away" to the outside of the corner. The required caution factor is then calculated. Thanks to the inverse lerp method, it is possible to calculate the linear parameter t, which gives an interpolated value from the given range $[a, b]$. Here, the lower end of the range is 0, the upper one, depending on whether the vehicle is *overtaking*, one of the angles that are parameters of braking behavior. The value is the maximum of the previously calculated corner angle and spinning angle. The greater the value of the selected angle, the greater care must be taken. Having calculated the required caution factor, it is possible to calculate the desired speed at which the vehicle should move in order to stay on the path. The speed is sought between the maximum speed of the car and the maximum speed multiplied by the cautious speed factor (this parameter determines what percentage of the maximum speed can be the minimum desired speed). The value of this coefficient depends on whether the car is *overtaking* or not. Based on the value of the desired car speed, the agent can decide whether to use the gas pedal or the brake.

Blocking behavior, like braking, is a specialized behavior that applies only to racing vehicles. It consists in the fact that after detecting an opponent approaching from behind, the vehicle is controlled in such a way as to make it difficult for the opponent to overtake or to completely block it. The opponent detection itself can be implemented in a very similar way to obstacle detection in the previously discussed *obstacle avoidance* behavior. Rays are projected from the rear of the vehicle and only checked whether they hit another car. Only 4 beams are enough to detect vehicles behind the controlled car: 2 rear side beams and 2 rear side oblique beams. Contrary to the case of avoiding a detected vehicle, instead of steering in the opposite direction, the car is steered towards the detected car to block its path.

Overtaking behavior also only applies to racing cars. It involves detecting a potential opponent who can be overtaken and then trying to overtake him. In car racing, overtaking most often occurs immediately after cornering, when the driver is able to take advantage of the speed advantage over his opponent. For this behavior, special paths are used only for overtaking. The approach is to create an additional path along the normal one which the vehicle will follow when the behavior is active. Additionally, when overtaking, the vehicle's caution is reduced, thanks to which it can move a bit more riskily, but also gains the opportunity to overtake its rival.

The overtaking behavior has already been partially mentioned in the description of other behaviors. Overtaking is a behavior that influences the behavior of *path following* behavior, *braking* behavior on corners and uses *obstacle detection*. The first stage of overtaking is detecting the opponent who is in front of the controlled car. In order not to perform another ray casting, which is quite an expensive operation, ray casting results from *collision avoidance* behavior is used. If one of the rays hits another car, the agent starts overtaking it. The current path is then switched to the overtaking path.

Overtaking behavior influences other behaviors. In the case of *path following* behavior, this is merely a change in the path the vehicle follows. The overtaking path contains exactly the same number of points as the regular path and they are placed at similar distances. The point to which the car is currently heading on the overtaking path will be very close to the point with the same index on the regular path. However, the effect on *braking* behavior is that the caution that the vehicle must exercise to stay on the path is reduced. Specifically, when a car is overtaking, different parameters are used in braking behavior than during normal driving. The first parameter is the maximum caution angle used to determine the required caution coefficient. For overtaking, it is greater and therefore allows larger deviations of the driving direction from the direction of the path and angular speed, and therefore less caution. The second parameter is the speed caution factor, which is the percentage of maximum speed taken into account when determining the desired speed. It is also larger and therefore allows for greater desired speed. Combined, changing path and allowing the vehicle to be less cautious allows it to corner faster, allowing it to overtake its target. During overtaking, the distance between the controlled car and its target to be overtaken is checked. If this distance is greater than specified by a set parameter, the overtaking behavior is finished. This applies both to situations when the car overtook its target and moved away to a safe distance, but also when its target fled to the same distance. Ending also switches the path back to normal.

Reacting to stucks is a kind of additional behavior that protects the controlled car from getting into conditions, which prevent it from continuing the race. There are two possible reactions. The first is to start reversing. If the car's speed is below a certain level for the time specified by the parameter, the car starts to reverse. When reversing, the behavior that controls the car's wheels turns them in the opposite direction, which allows it to avoid obstacles that required reversing. The second reaction takes place after a longer time. The car is then immediately teleported to the track.

3.4. Combining steering behaviors

Each of the algorithms described here concerns individual behaviors related to vehicle movement. When creating game AI, single behaviors are very rarely used. Usually, several of them are implemented depending on the game needs. However, it is very important that these behaviors do not cancel each other out. For this purpose, various methods of combining them are used. However, not all behaviors are necessarily mutually exclusive, some can work seamlessly together.

One of the methods to solve the problem of mutually exclusive algorithms is to switch active behaviors. It involves checking the state of the world and reacting to changes occurring in it. Choosing which behaviors to be active involves decision-making, which is a layer above the movement layer in the game AI model [6]. Such approach was used in this work. This is due to the fact that the implemented behaviors also take into account the restrictions related to the car's movement. Therefore, at the output they do not return forces, but the values of variables controlling the car, such as pressing the gas pedal, brake pedal or the level of steering wheel rotation. Control behavior algorithms in their basic version are much more universal, which is why they can be combined in many different ways. Despite using this solution, the application does not use any advanced decision-making techniques. It is only decided which behaviors cannot be active at the same time, so that their operation does not interfere with each other. When the situation changes, behaviors are switched accordingly.

Path following can only be active when neither *blocking* behavior nor *collision avoidance* behavior is in effect. However, both behaviors that preclude *path following* behavior may be interrupted if the vehicle's direction of motion deviates too much from the direction of the path. The *braking* behavior is active at all times. The *blocking* behavior cannot only be active when *overtaking*. Table 1 shows which behaviors are mutually exclusive and which are not.

Table 1. Mutual exclusion of behaviors. "yes" means they can cooperate, otherwise there is "no".

behavior	path following	braking	collision avoidance	blocking	overtaking
path following	-	yes	no	no	yes
braking	yes	-	yes	yes	yes
collision avoidance	no	yes	-	yes	yes
blocking	no	yes	yes	-	no
overyesing	yes	yes	yes	no	-

4. Experiment

4.1. Vehicle control

To test an agent using our method, we had to provide a way for the agent to drive the vehicle. The car is driven by setting 4 parameters:

- *throttle* means the level of pressing the gas pedal and is a number in the range [0, 1] – The higher its value, the more torque will be transferred to the vehicle's wheels.
- *steer*, with values in the range $[-1, 1]$ is used to control the steering wheel, and thus the car's wheels – the value -1 means the maximum turning of the steering wheel to the left, the value 1 to the right and 0 is the default steering wheel position in which the wheels are not turned,
- *brake* variable with values in the range [0, 1] determines the level of pressing the brake pedal – the higher its value, the more the car wheels will be braked,
- *handbrake* variable is similar to brake, but is used to brake only the rear wheels.

The brake is taken into account first, if the variable controlling the brake pedal has a value greater than 0 and the car's speed exceeds 1 km/h and it is not currently in reverse, braking is performed by resetting the torque on the wheels and making the braking force dependent on the value of the brake variable. Otherwise, the car stops braking and starts moving backwards, transmitting negative torque to the wheels depending on the value of the brake variable. However, if it is not the brake pedal that is pressed but the gas pedal is pressed, i.e. the value of the throttle variable is greater than 0, braking is ended (the braking torque on the wheels is zeroed) and the previously calculated current torque from the engine is transferred to the wheels. At the very end, the steering wheel is turned. The maximum steering angle depends on the car's speed. In a real car, the driver does not make sudden movements of the steering wheel at high speeds, but in the case of a game where the input is taken from the keyboard, the wheels are turned almost immediately by the maximum amount. Therefore, the maximum steering angle is limited along with the speed. At the very end, braking is performed using the handbrake, i.e. the braking torque for the rear wheels of the vehicle is set depending on the value of the handbrake variable.

The AI controller works by triggering steering behaviors according to the following algorithm:

1. path following
2. throttle and brake

3. obstacle avoidance
4. blocking
5. overtaking
6. reacting to stucks

4.2. Tests

In order to verify the solution, a series of tests were carried out. Of course, also during the implementation of individual AI behaviors, their operation was checked on an ongoing basis and all parameters were adjusted so that the vehicle movement was reproduced as best as possible. However, only when all the behaviors were ready with selected parameters, it was possible to check how the AI cars behave under the influence of all the algorithms working together. For this purpose, a several of 5 races over a distance of 5 laps were performed for AI vehicles only, and then the same tests were repeated with the human player. Each race involved 4 identical vehicles, which always started from the same starting positions.

5. Results

The test results, i.e. the total travel times along with the positions occupied by the cars, are summarized in Tables 2 and 3. They present a typical series of races that can be run using agents implementing the proposed method.

Table 2. Total run times and car positions in a 5-lap race in which only AI vehicles took part.

	vehicle							
	AI1		AI2		AI3		AI4	
race	time	position	time	position	time	position	time	position
1	3:03.02	3	2:47.38	1	3:13.10	4	2:50.82	2
2	3:01.86	1	3:16.07	4	3:07.77	3	3:05.54	2
3	2:46.87	1	2:51.27	2	3:05.00	4	2:51.65	3
4	2:48.09	2	2:46.71	1	3:07.08	4	3:05.68	3
5	3:41.57	4	2:48.97	1	3:02.16	3	2:50.97	2

Based on the data collected in Table 2, it can be clearly seen that even when AI vehicles raced without the human player's participation, they occupied different positions and finished the race at different times. This means that the implemented control behaviors have a real impact on how the AI car will drive in the race. As mentioned earlier, the vehicles started from the same starting positions each time,

Table 3. Total run times and car positions in the 5-lap race in which AI vehicles and the human player took part.

	vehicle							
	Human		AI2		AI3		AI4	
race	time	position	time	position	time	position	time	position
1	3:01.34	3	2:48.50	1	2:49.70	2	3:02.10	4
2	3:06.78	3	3:14.37	4	3:05.23	2	2:54.39	1
3	2:48.39	2	3:04.48	4	3:03.35	3	2:47.64	1
4	3:05.76	3	3:02.62	1	3:05.42	2	3:18.72	4
5	2:44.72	1	2:48.16	2	3:00.65	3	3:01.76	4

so their times and positions depended solely on the course of the race. It can also be noted that out of 5 races, only AI1 and AI2 vehicles managed to take first place. This is most likely due to the starting positions of the cars, as AI3 and AI4 started from behind AI1 and AI2. Usually, the difference in times achieved by subsequent cars at the finish line ranges from a few to even several dozen seconds. However, there may also be a situation like in race 3, in which the AI2 vehicle was faster than AI4 by $0.38s$. In addition to collecting data on times and positions during the races, vehicle movement was also observed. It was noticeable that the decisive factor is the very beginning of the race, i.e. the first 3 laps, where the leaders actually emerge. Despite this, there were situations of fierce competition until the very end of the race, as in the previously described case of race 3.

During races with the participation of the human player, the observations are similar. This time, as can be seen in Table 3, even the AI4 vehicle, which started from behind the player and the AI2 vehicle, managed to get the first position twice. This was due to the fact that the player managed to overturn the AI2 vehicle by himself, losing control of the car, thanks to which the AI4 vehicle was able to overtake it. It can also be noticed that only in the 3rd race the player managed to take second place, and first place only in the 5th time. It can be seen the player learning process here, which is normal for video games. During each game, the player learns how to play a given game to achieve the best results.

To sum up, the solution verification was successful. When racing, AI cars used all the implemented control behaviors. Sometimes, as a result of overtaking, during which the AI vehicle is less careful, they hit barriers, which caused them to lose speed. Nevertheless, even in this situation, the vehicles managed to get back into the race. Situations where agents make mistakes are desirable from the point of view of a computer game because they increase credibility and do not cause frustration for the player.

6. Conclusions

This work consisted of adapting and developing steering behaviors, and proposing a method of combining them, that thay are suitable for controlling agents that are a car drivers at a race. Steering behaviors were used to generate input data for the created car controller. They were:

- path following (including seek behavior),
- corner braking and throttle steering,
- obstacle avoidance,
- blocking,
- overtaking.

They were combined together by switching of active behaviors. All implemented algorithms have been thoroughly tested. Controlled vehicles have proven to be a good representation of the real drivers racing on a racetrack, including the fact that they can make mistakes. This gives a much more credible and expected effect for the player than if they were perfect. This solution can be successfully used to create a racing game.

References

[1] Chan, M. T., Chan, C. W., and Gelowitz, C. Development of a car racing simulator game using artificial intelligence techniques. *International Journal of Computer Games Technology*, 2015:1–6, 2015. ISSN 1687-7055. doi: 10.1155/2015/839721.

[2] Fray, A. Context behaviours know how to share, 2013. URL `https://andrewfray.wordpress.com/2013/03/26/context-behaviours-know-how-to-share/`.

[3] Dockhorn, A., Kirst, M., Mostaghim, S., Wieczorek, M., and Zille, H. Evolutionary algorithm for parameter optimization of context-steering agents. *IEEE Transactions on Games*, 15(1):26–35, 2023. ISSN 2475-1510. doi: 10.1109/tg.2022.3157247.

[4] Reynolds, C. Steering behaviors for autonomous characters. 2002.

[5] Rodae Catalogue, 2023. URL `www.automobile-catalog.com/car/2007/254060/audi_r8_4_2_fsi_quattro.html`.

[6] Millington, I. *Artificial Intelligence for Games*. Morgan Kaufmann Publishers, 2006.

Quantum Simulation of the Excited State Decay in Two Dimensions

Marcin Ostrowski[0000−0001−8985−8123]

Lodz University of Technology
Institute of Information Technology
al. Politechniki 8, 93-590 Łódź, Poland
marcin.ostrowski@p.lodz.pl

Abstract. *In this paper we propose a quantum algorithm simulating photon free propagation and interaction between photon and atom in the two-dimensional cavity. We examine decay of the atom excited state with the emission of photon and photon-atom scattering. The paper studies the properties of the proposed algorithm and then compares the obtained results with the theoretical predictions.*

Keywords: *quantum computations, quantum simulations, excited state decay*

1. Introduction

In the near future, quantum calculations can make a major contribution to the development of informatics [1]. Nowadays, some institutions claim to have a quantum computer and offer its computing power. Probably wider access to these machines will become possible soon. Therefore, it is worth examining what new possibilities are offered by quantum computing.

For many years we know Shor [2] and Grover [3] algorithms which are of lower computational complexity than their best classical counterparts. Nowadays, a quantum computer can be used to solve a variety of problems, such as quantum image and signal processing. Another promising application of quantum computer are quantum simulations [4, 5, 6], i.e. the computer modeling of physical quantum systems. It gives the possibility of effective modeling quantum processes, which is not possible using classical computers [7]. Quantum computers can simulate a wide variety of quantum systems, including fermionic lattice models [8, 9], quantum chemistry [10, 11], field theories [12, 13] and quantum particles [14, 15].

As is well known, simulations of quantum systems performed with the use of conventional computers are not effective. This means that for classical computer the memory resources and time required to simulation grow exponentially with the size of quantum system. In the case of a quantum computer, the situation is different. The relationship between the size of quantum computer (register) and the size of the simulated quantum system is linear. Therefore, a very important task is to find the appropriate algorithms that can properly simulate complex quantum systems and non-trivial interactions between them. This is a difficult issue, because most of the interesting quantum systems is feasible in infinitely-dimensional Hilbert spaces. In such situations, we can use the technique of sampling the wave function and build an algorithm based on the Quantum Fourier Transform. This case was tested in [16, 17, 18, 19], which examined the free Schrödinger particle and the harmonic oscillator. In our previous works we used this method for rectangular potentials (like thresholds and wells). This provides the ability to examine other interesting processes, such as the tunnel effect [20] and scattering of the Schrödinger particle[21].

Another important issue is the simulation of quantum optics processes such as: photon emission or absorption by atom or molecule. This requires simulating the electromagnetic quantum field, which is the system with an infinite number of freedom degrees. In this case, we can replace a continuous band of energy levels with its discrete counterpart. We use this method in the current publication and in [21, 22]. In the above-mentioned works, we have simulated photon in the one-dimensional cavity. This case, due to its simplicity (uniformly distributed energy levels), allows to simulate the free evolution of photon with a simple algorithm. In order to simulate 2^{n_q} energy levels we need only n_q one-qubit gates, where n_q is the number of qubits. In this work we examine photon in the two-dimensional cavity. This case requires a more complex algorithm because each energy level is simulated in a separate quantum gate. However, this situation is much more realistic. Moreover, a similar method can be used to construct a 3-dimensional algorithm.

The theoretical approach to the problem of unstable quantum systems decay can be found in [23]. Works of other authors also focus on the simulation of excited states decay. Models based on cavity QED are particularly tested. For example, processes such as: beta decay of the helium atom [24] and decay of a two-level atom in a crystal [25] are examined. In contrast to the cited works, we examine the problem on purely algorithmic grounds, using abstract model of quantum gates. We abstract completely from specified physical implementation.

In order to simulate a quantum register, we used environment written in C++ language based on direct matrix multiplication. We also performed several tests of the algorithm with the use of IBM Qiskit SDK environment [26]. In both cases we obtained the same results. Python (Qiskit) code of the algorithm main blocks can be found in Appendix D.

2. Description of the simulated system

Let us consider a complex quantum system which is composed of two parts A and F. Subsystem A (atom) has two energy levels: level $|0\rangle_A$ with energy equal to zero (the ground state) and level $|1\rangle_A$ with energy E_A (the excited state). We identify subsystem F with photon (without spin) trapped inside a two-dimensional cavity of length a (with periodic boundary conditions). Stationary states of subsystem F are denoted by $|n_x, n_y\rangle_F$, where $n_x, n_y \in \mathcal{Z}$. For these states, components of the momentum vector are given by Eq. (15) and energy is given by Eq. (18).

The following operator is chosen as the Hamiltonian of interaction between subsystems A and F:

$$\hat{H}_{int} = \sum_{n_x,n_y} (g_{n_x n_y} \hat{a}^\dagger \hat{b}_{n_x n_y} + g^*_{n_x n_y} \hat{a} \hat{b}^\dagger_{n_x n_y}), \tag{1}$$

where $\hat{a}$ is an operator decreasing energy of subsystem A ($\hat{a}|1\rangle_A = |0\rangle_A$) and $\hat{b}_{n_x n_y}$ are annihilation operators of subsystem F ($\hat{b}_{n_x n_y}|n_x, n_y\rangle_F = |0\rangle_F$). Complex parameters $g_{n_x n_y}$ are coupling constants given by Eq. (25). The summation in Eq. (1) is done over all integer values of n_x and n_y excluding $n_x = n_y = 0$ (there is no photon with zero momentum). In practice, due to the conditions imposed by the simulation, we consider bounded spectrum of energy $|n_x| < n_{max}$ and $|n_y| < n_{max}$. The Hamiltonian (1) describes transitions between states in the following form: $|0\rangle_A|n_x, n_y\rangle_F \leftrightarrow |1\rangle_A|0\rangle_F$, where $|0\rangle_F$ is the vacuum state.

The total Hamiltonian of the system AF has the form:

$$\hat{H} = E_A \hat{a}^\dagger \hat{a} + \sum_{n_x,n_y} E(n_x, n_y) \hat{b}^\dagger_{n_x n_y} \hat{b}_{n_x n_y} + \hat{H}_{int}, \tag{2}$$

where $E(n_x, n_y)$ are given by Eq. (18).

3. Description of the algorithm

In order to solve the Schrödinger equation for the Hamiltonian (2) we use the time evolution operator in the form:

$$\hat{U}(\mathrm{d}t) = \exp(-\mathrm{i}\hat{H}\mathrm{d}t/\hbar). \tag{3}$$

For $\mathrm{d}t \to 0$ operator given by Eq. (3) can be approximated as follows:

$$\hat{U}(dt) = \exp(-\mathrm{i}E_A \hat{a}^\dagger \hat{a} \mathrm{d}t/\hbar) \prod_{n_x,n_y} \exp\Big(-\mathrm{i}E(n_x, n_y) \hat{b}^\dagger_{n_x n_y} \hat{b}_{n_x n_y} \mathrm{d}t/\hbar\Big)$$
$$\prod_{n_x,n_y} \exp\Big(-\mathrm{i}(g_{n_x n_y} \hat{a}^\dagger \hat{b}_{n_x n_y} + g^*_{n_x n_y} \hat{a} \hat{b}^\dagger_{n_x n_y}) \mathrm{d}t/\hbar\Big). \tag{4}$$

The above equation is equivalent to using the first-order Lie-Trotter formula with Trotter error $O(t^2)$ [27]. The simulation for a large time t is obtained by dividing the evolution into n Trotter steps ($t = n \cdot \mathrm{d}t$).

The whole algorithm is shown in Fig. 1. The U_{ϕ_A} gate is the phase-shift gate simulating the free evolution of subsystem A (the first component from Eq. 4). Phase shift angle for the U_{ϕ_A} gate is equal to $\phi_A = E_A \hbar^{-1} dt$. The UF block implements the photon free evolution (the second component from Eq. 4) while the R block implements the atom-photon interaction (the third component from Eq. 4).

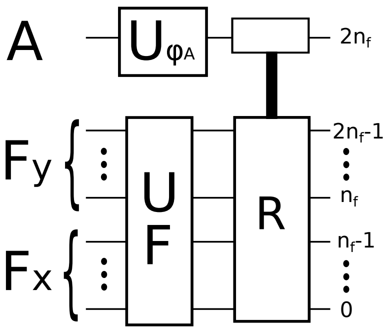


Figure 1. The scheme of the algorithm (for one time step)

3.1. The algorithm simulating propagation of free photon

State of photon is encoded in the momentum representation in two n_f-qubits subregisters (subregisters F_y and F_x in Fig. 1). In F_x and F_y subregisters states $|n_x\rangle_F$ and $|n_y\rangle_F$ are are encoded, respectively (see Appendix A). Computational base states of F_x and F_y subregisters we denote by $|j_x\rangle$ and $|j_y\rangle$, respectively, where $j_x, j_y = 0, 1 \ldots 2^{n_f} - 1$. The $j_x = j_y = 0$ state encodes the vacuum state (no photon in the cavity). Other states encode one-photon states ($|n_x, n_y\rangle_F$) according to the following rule:

$$n_x = \begin{cases} j_x & \text{for} \quad 0 \leq j_x < n_{max} \\ j_x - 2n_{max} & \text{for} \quad n_{max} \leq j_x \leq 2n_{max} - 1 \end{cases} \tag{5}$$

and

$$n_y = \begin{cases} j_y & \text{for} \quad 0 \leq j_y < n_{max} \\ j_y - 2n_{max} & \text{for} \quad n_{max} \leq j_y \leq 2n_{max} - 1, \end{cases} \tag{6}$$

where $n_{max} = 2^{n_f - 1}$.

The algorithm simulating the time evolution of free photon (the UF block from Fig. 1) is shown in Fig. 2. Each photon stationary state (each component of the sum in the second term from Eq. (2)) is simulated by a separate U_j gate, where

$$j = 2^{n_f} \cdot j_y + j_x. \tag{7}$$

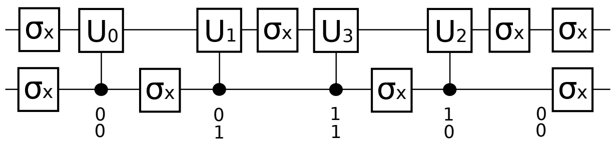


Figure 2. The algorithm simulating free propagation of photon (example for $n_f = 1$). The U_j gates are numbered by j parameter given by Eq. (7). The numbers below the U_j gates are binary representation of the j number.

The U_j gates are the phase-shift gates which operate according to the scheme:

$$|0\rangle \to |0\rangle, \qquad |1\rangle \to \exp(-\mathrm{i}\phi)|1\rangle, \tag{8}$$

where: $\phi = E(n_x, n_y)\, dt/\hbar$ and n_x, n_y are given by Eqs (5-6). The order of the U_j gates follows the Gray code [28] for the j number. As a result, the number of the NOT gates is reduced to a minimum. The NOT gates act on the qubits for which bit flip ($0 \leftrightarrow 1$) occurs in binary representation of the j number. For example, the σ_x gate between U_1 and U_3 gates (in Fig. 2) acts on the second qubit because $1 = 01_2$ and $3 = 11_2$.

3.2. The algorithm simulating interaction between atom and photon

The scheme of the algorithm simulating interaction between subsystems A and F (the R block from Fig. 1) is shown in Fig. 3.

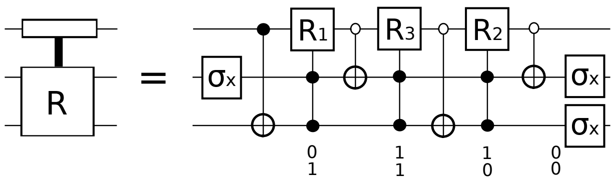


Figure 3. Implementation of the R block (example for $n_f = 1$) simulating interaction between atom (the highest qubit) and photon. The numbers below the R_j gates are binary representation of the j number.

Each component of the sum from Hamiltonian (1) (describing transition $|0\rangle_A|n_x, n_y\rangle_F \leftrightarrow |1\rangle_A|0\rangle_F$) is simulated by a separate R_j gate ($j = 2^{n_f} \cdot j_y + j_x$) which operate as follows:

$$|0\rangle \to \cos\phi_j|0\rangle + \mathrm{i}e^{+\mathrm{i}\eta_j}\sin\phi_j|1\rangle, \tag{9}$$

$$|1\rangle \to \cos\phi_j|1\rangle + \mathrm{i}e^{-\mathrm{i}\eta_j}\sin\phi_j|0\rangle, \tag{10}$$

where $\phi_j = |g_{n_x n_y}|dt/\hbar$, $\eta_j = \mathrm{Arg}(g_{n_x n_y})$ and $g_{n_x n_y}$ are given by Eq. (25). Calculation of the j numbers from (n_x, n_y) levels is done with the use of Eq. (7) and the transformation reverse to Eqs (5-6):

$$j_a = \begin{cases} n_a & \text{for} \quad n_a \geq 0, \\ n_a + 2n_{max} & \text{for} \quad n_a < 0, \end{cases} \tag{11}$$

where $a = x, y$.
The order of the R_j gates follows the Gray code [28] for the j number. As a result, the number of the CNOT gates is reduced to a minimum. The CNOT gates (with control qubit active in $|0\rangle$ state) act on the qubit for which bit flip $(0 \leftrightarrow 1)$ occurs in the j number. This rule is the same as the rule for NOT gates presented in the previous section.
The NOT (σ_x) and the CNOT gates preceding the R_j gete implement the transformation:

$$|0\rangle_A |j\rangle_F \rightarrow |0\rangle_A |1 \ldots 1\rangle_F, \tag{12}$$

$$|1\rangle_A |0\rangle_F \rightarrow |1\rangle_A |1 \ldots 1\rangle_F. \tag{13}$$

For example, (for $n_f = 1$) the first two gates from Fig. 3 (the σ_x and the CNOT gates) implement the transformation for $j = 1$:

$$|0\rangle_A |01\rangle_F \rightarrow |0\rangle_A |11\rangle_F,$$
$$|1\rangle_A |00\rangle_F \rightarrow |1\rangle_A |11\rangle_F.$$

If we attach another CNOT gate (the gate between the R_1 and the R_3 gates) we obtain the transformation for $j = 3$:

$$|0\rangle_A |11\rangle_F \rightarrow |0\rangle_A |11\rangle_F,$$
$$|1\rangle_A |00\rangle_F \rightarrow |1\rangle_A |11\rangle_F.$$

4. The simulation results

4.1. Free photon simulation

In the first part of the research, we simulate only photon (subsystem F). In this case, an $n_q = 12$ qubit register has been used. A square cavity with a side of $a = 30\mu$m with time step d$t = 3 \cdot 10^{-14}$s has been simulated. As an initial state of photon, we choose the Gaussian distribution given by Eq. (20). The simulation results of the two cases are presented:

- plane wave ($\langle x \rangle = -9\mu$m, $\langle y \rangle = 0\mu$m, $k = \frac{32}{3}\Delta k$, $\phi = 0$, $\mathrm{d}k_{\|} = 3\Delta k$, $\mathrm{d}k_{\perp} = 0.3\Delta k$), where Δk is given by Eq. (16). For this case, the results are shown in Figs 4-5.

- localized Gauss packet ($\langle x \rangle = \langle y \rangle = -6\mu$m, $k = \frac{32}{3}\Delta k$, $\phi = \pi/4$, $\mathrm{d}k_{\|} = \mathrm{d}k_{\perp} = 3\Delta k$). For this case, the results are shown in Figs 6-7.

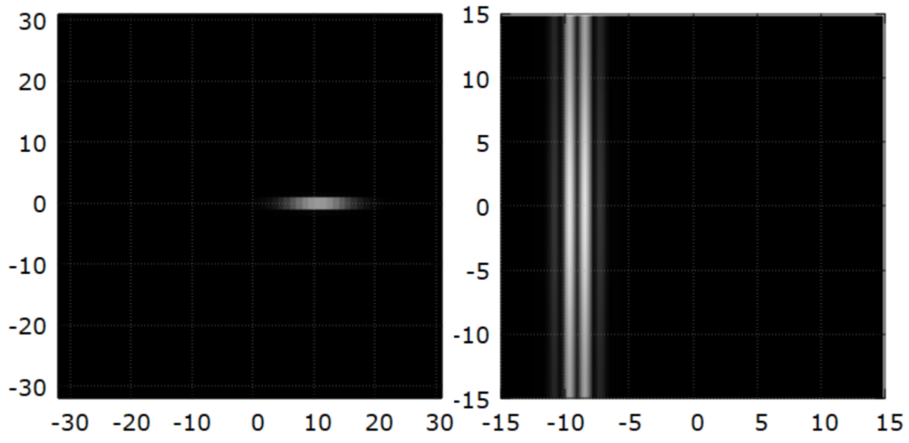


Figure 4. Simulation of plane wave propagation. The left plot shows the photon's initial state (for $t_0 = 0$s) in the momentum representation. In this case the numbers on the axes correspond to (n_x, n_y). The right plot shows the initial space distribution (for $t_0 = 0$s) of the electric field (Eq. 19). In this case, the numbers on the axes indicate the position in the cavity, measured in μm.

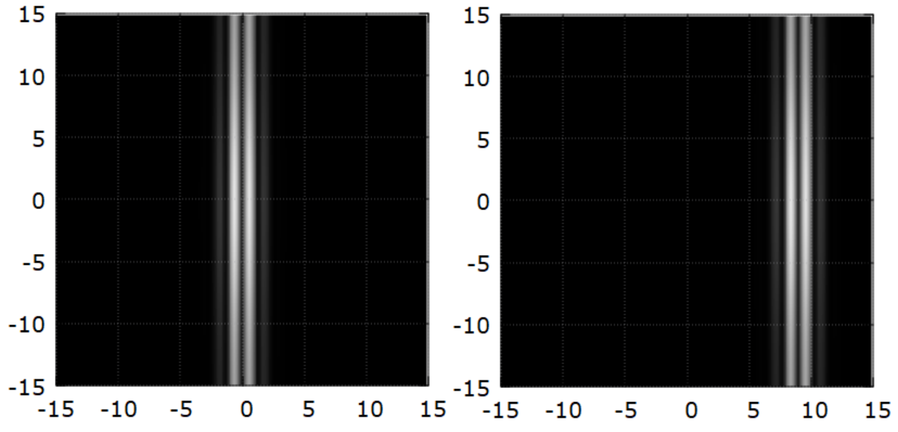


Figure 5. Simulation of plane wave propagation. The plots show space distribution of the electric field (Eq. 19) for $t_1 = 3 \cdot 10^{-14}$s (the left plot), and for $t_2 = 6 \cdot 10^{-14}$s (the right plot). The numbers on the axes indicate the position in the cavity, measured in μm.

4.2. Simulation of photon emission

In this case, an $n_q = 13$ qubit register has been used. A square cavity with a side of $a = 30\mu$m with time step $\mathrm{d}t = 5 \cdot 10^{-16}$s has been simulated. As an initial

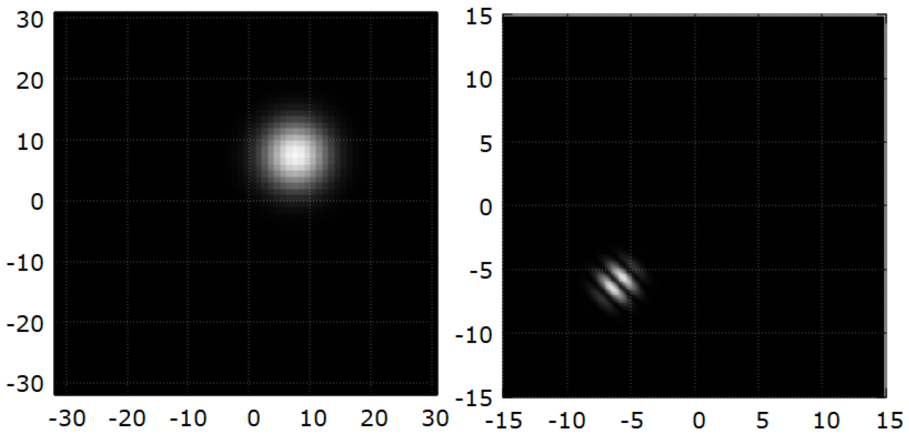


Figure 6. Simulation of the Gauss packet propagation. The left plot shows the photon's initial state (for $t_0 = 0\text{s}$) in the momentum representation. In this case, the numbers on the axes correspond to (n_x, n_y). The right plot shows the initial space distribution (for $t_0 = 0\text{s}$) of the electric field (Eq. 19). In this case, the numbers on the axes indicate the position in the cavity, measured in μm.

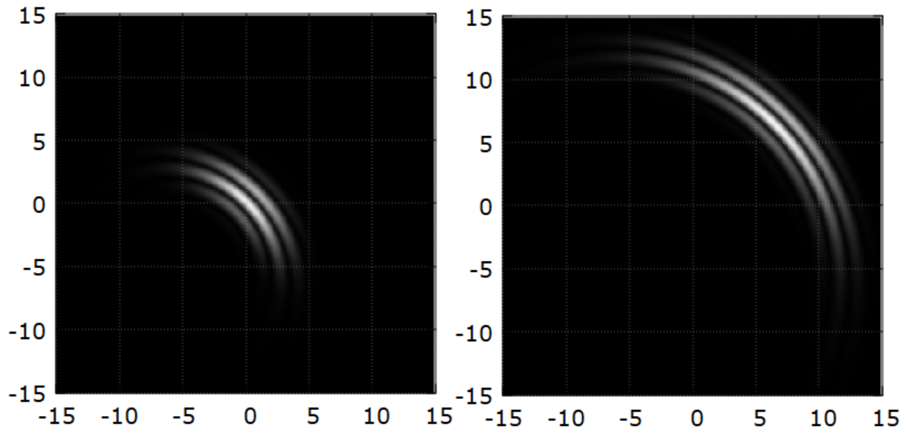


Figure 7. Simulation of the Gauss packet propagation. The plots show space distribution of the electric field (Eq. 19) for $t_1 = 3 \cdot 10^{-14}\text{s}$ (the left plot), and for $t_2 = 6 \cdot 10^{-14}\text{s}$ (the right plot). The numbers on the axes indicate the position in the cavity, measured in μm.

state of the system, we choose $|1\rangle_A |0\rangle_F$ (the atom in the excited state, subsystem F in the vacuum state). Energy of the excited state is equal to $E_A = 0.6\text{eV}$. The two cases has been examined:

1. for the decay constant $\Gamma = 5 \cdot 10^{-22}$J (see Eq. (24)). The atom is located in the center of the cavity ($x_a = y_a = 0$m). The results for $n = 92$ time steps are shown in Fig. 8.

2. for the decay constant $\Gamma = 7 \cdot 10^{-22}$J. The position of the atom in the cavity is $x_a = 5\mu$m $y_a = -5\mu$m. The results for $n = 60$ time steps are shown in Fig. 9.

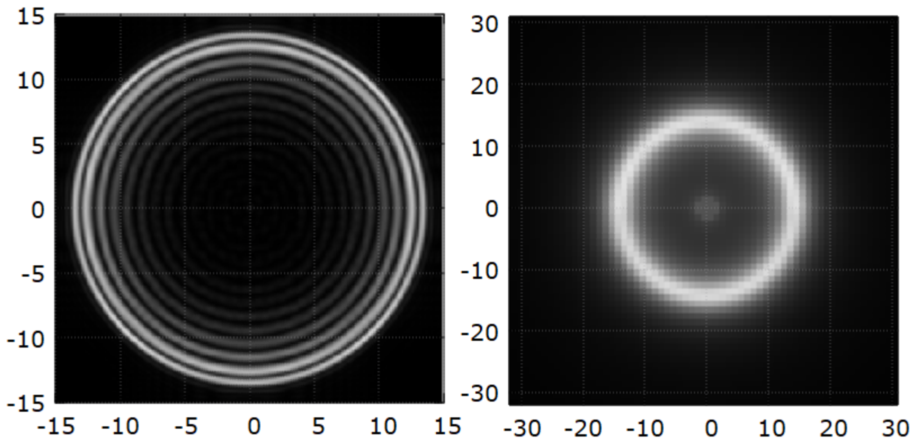


Figure 8. Simulation of photon emission (the case no. 1). The left plot shows the final space distribution of the electric field (Eq. 19) for $t = 4.6 \cdot 10^{-14}$s (the numbers on the axes indicate the position in the cavity, measured in μm). The atom at point $x_a = y_a = 0$ is not shown. The right plot shows the final photon state in the momentum representation (the numbers on the axes correspond to n_x, n_y).

The probability of finding the atom in the excited state as a function of time has also been examined. The results for three different values of Γ are shown in Fig. 10. In this case we also use an $n_q = 13$ qubit register. Time step has been reduced to $dt = 10^{-16}$s. Total number of time steps is equal to $n = 120$. Energy of the excited state is equal to $E_A = 0.6$eV.
In Fig. 10 in addition to the simulation results, we present the theoretical prediction given by the formula (e.g.[29]):

$$p_{thr}(t) = \exp\left(-\frac{2\pi}{\hbar}|\tilde{g}|^2 t\right), \tag{14}$$

where $\tilde{g} = \Gamma\sqrt{\frac{d}{dE}}$, $d = 2\pi E_A/dE$ is degeneration of energy levels and $dE = \frac{hc}{a}$ is distance between adjacent energy levels.

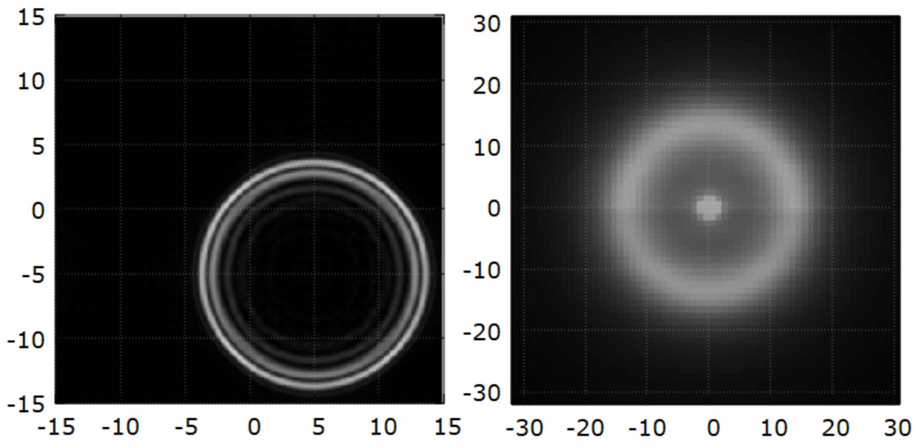


Figure 9. Simulation of photon emission (the case no. 2). The left plot shows the final space distribution of the electric field (Eq. 19) for $t = 3 \cdot 10^{-14}$s (the numbers on the axes indicate the position in the cavity, measured in μm). The atom at point $x_a = 5\mu$m $y_a = -5\mu$m is not shown. The right plot shows the final photon state in the momentum representation (the numbers on the axes correspond to n_x, n_y).

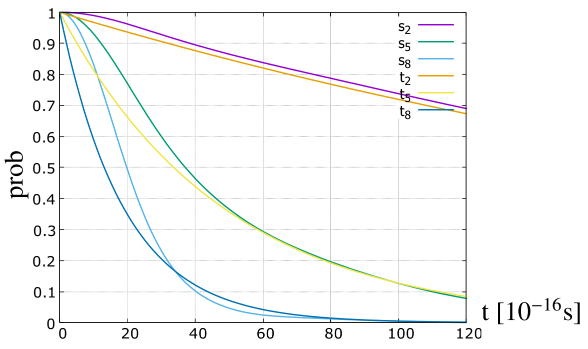


Figure 10. The probability of finding the system A in an excited state as a function of time (in 10^{-16}s units). The three cases are investigated: $\Gamma = 2 \cdot 10^{-22}$J (the curves denoted by "2"), $\Gamma = 5 \cdot 10^{-22}$J (the curves denoted by "5") and $\Gamma = 8 \cdot 10^{-22}$J (the curves denoted by "8"). The curves denoted by "s" are results of simulation while those denoted by "t" are results of theoretical predictions (given by Eq. (14)).

4.3. Simulation of photon scattering

In this case, an $n_q = 13$ qubit register also has been used. A square cavity with a side of $a = 30\mu$m with time step d$t = 5 \cdot 10^{-16}$s has been simulated. The total number of time steps is $n = 120$. As an initial state of the photon, we choose the Gaussian distribution shown in Fig. 6. The atom is located in the center of the cavity ($x_a = y_a = 0$m), and its initial state is the ground state $|0\rangle$. Energy of the excited state is equal to $E_A = 0.45$eV. The two cases has been examined: for $\Gamma = 10^{-21}$J (Fig. 11) and for $\Gamma = 5 \cdot 10^{-22}$J (Fig. 12).

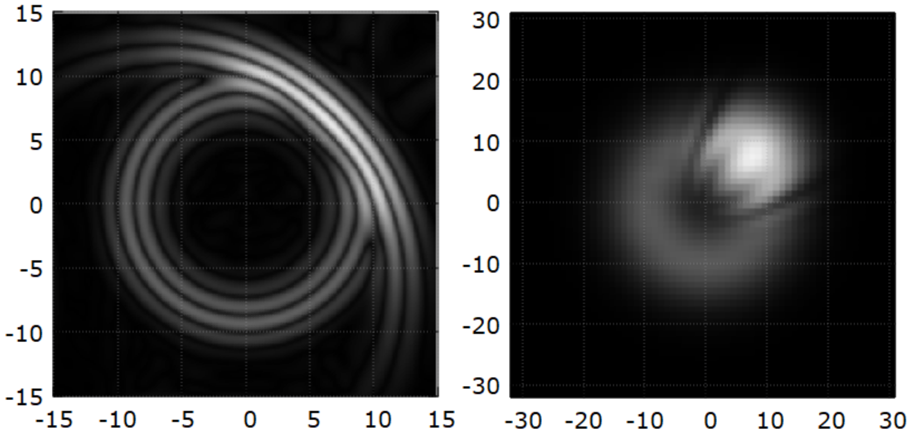


Figure 11. Simulation of the photon scattering on the atom in the ground state for $\Gamma = 10^{-21}$J. The left plot shows the final space distribution of the electric field (Eq. 19) for $t = 6 \cdot 10^{-14}$s (the numbers on the axes indicate the position in the cavity, measured in μm). The atom at point $x_a = y_a = 0$ is not shown. The right plot shows the final photon state in the momentum representation (the numbers on the axes correspond to n_x, n_y).

5. Conclusions

- The evidence from this study suggests that even for $n_q = 13$ qubits it is possible to obtain satisfying results. However, the algorithm is scalable - an increase in the number of qubits means a higher sampling density of photon spectrum and, consequently, more accurate results.

- As is well known, a typical lifetime of atom excited states is in the order of 10^{-8}s (excluding metastable states). This time is much longer in comparison to time of the photon propagation in the cavity (the order of 10^{-14}s). Therefore, the time scale of deexcitation process has been reduced by several orders of magnitude. This is due to the limited size of the cavity - for longer simulation time the photon returns to the atom and is reabsorbed. However, the length a of the cavity can be increased. Enlargement of the F_x and F_y registers by one qubit increases the length of the cavity to $2a$ (at the

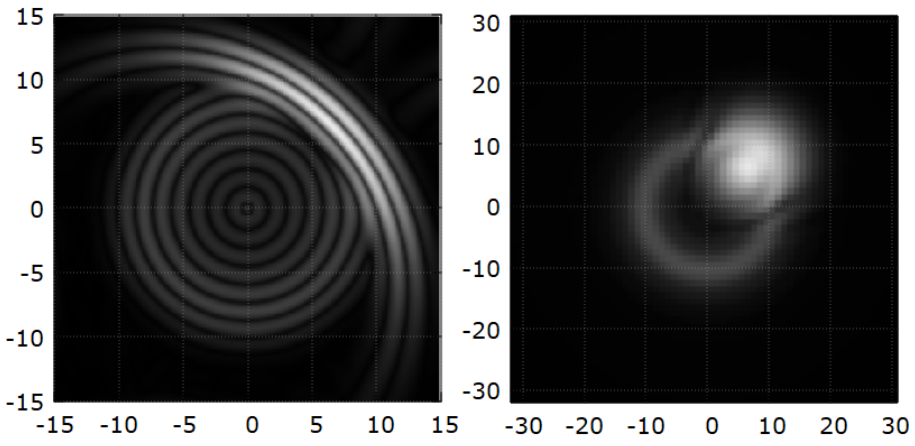


Figure 12. Simulation of the photon scattering on the atom in the ground state for $\Gamma = 5 \cdot 10^{-22}$J. The left plot shows the final space distribution of the electric field (Eq. 19) for $t = 6 \cdot 10^{-14}$s (the numbers on the axes indicate the position in the cavity, measured in μm). The atom at point $x_a = y_a = 0$ is not shown. The right plot shows the final photon state in the momentum representation (the numbers on the axes correspond to n_x, n_y).

same density of energy levels). Increasing the simulation time by a factor of 10^6 requires the addition of $\log_2 10^6 \approx 20$ qubits to both subregisters. (The number of qubits can be reduced by exclusion form the simulation levels with energy significantly different from E_A. We have successfully used this method in the previous works for a one-dimensional case.)

- The algorithm presented here has been used to simulate relatively simple processes and the obtained results can be predicted theoretically. Thanks to this, we can examine correctness of the obtained results. However, an extended version of the algorithm can be used in the future to study photon scattering phenomena on more complex shields, such as chemical molecules or elementary particles. This will probably become possible when sufficiently large quantum computers appear.

- In this work a spinless photon is simulated. As a result, the simulation time has been reduced. However, the algorithm is easy to extend. It is enough to add a single qubit storing information about spin of the photon. The free propagation algorithm from Fig. 2 does not require modification. The interaction algorithm from Fig. 3 requires doubling the number of R_j gates and the interaction coefficients from Eq. (25) should depend on the variable s, where $s = 0, 1$.

- In order to perform the algorithm on a quantum computer, initial Gaussian state of the particle F must be entered to the quantum register. In this work we do not present the appropriate algorithm. A simple algorithm for inputting these types of states into the quantum register has been proposed in [30].
- Transition from the momentum to the position representation given by the formula (19) is made outside the quantum register.

Appendix A. Photon in a two-dimensional cavity

- the single-photon base states in the momentum representation can be denoted as follows: $|n_x, n_y\rangle_F$, where $n_x, n_y \in \mathcal{Z}$,
- the $|0, 0\rangle_F = |0\rangle_F$ state corresponds to the lack of photon (the vacuum state),
- in the square cavity with a side equal to a components of the wave vector can be written in the following form:

$$k_x = \Delta k\, n_x, \qquad k_y = \Delta k\, n_y, \tag{15}$$

where

$$\Delta k = \frac{2\pi}{a}. \tag{16}$$

- length of the wave vector:

$$k = |\vec{k}| = \Delta k \sqrt{n_x^2 + n_y^2} \tag{17}$$

- energy of photon:

$$E(n_x, n_y) = \hbar c k = \frac{\mathrm{hc}}{a} \sqrt{n_x^2 + n_y^2} \tag{18}$$

- electric field related with photon (for the two-dimensional case) is equal to:

$$E_z(x, y) = \mathrm{i}\sqrt{\frac{\hbar}{\epsilon_0 V}} \sum_{n_x,n_y} \sqrt{\frac{\omega(\vec{k})}{2}}\left(-\psi^*_{n_x,n_y}\mathrm{e}^{-\mathrm{i}\vec{k}\vec{r}} + \psi_{n_x,n_y}\mathrm{e}^{\mathrm{i}\vec{k}\vec{r}}\right) =$$

$$= -\sqrt{\frac{\hbar}{\epsilon_0 V}} \sum_{n_x,n_y} \sqrt{2\omega(\vec{k})}\left(\mathrm{im}\{\psi_{n_x,n_y}\} \cos \vec{k}\vec{r} + \mathrm{re}\{\psi_{n_x,n_y}\} \sin \vec{k}\vec{r}\right), \tag{19}$$

where $\vec{r} = [x, y]$, $\vec{k} = [k_x, k_y] = [\frac{2\pi}{a} n_x, \frac{2\pi}{a} n_y]$ and $\omega(\vec{k}) = c|\vec{k}|$.

Appendix B. The Gaussian state of photon in two dimensions

The two-dimensional Gaussian state of photon in the momentum representation can be written as follows:

$$\psi(k_x, k_y) = \mathrm{c}\exp\left(-\frac{(k_{||}-k)^2}{4\mathrm{d}k_{||}} - \frac{k_\perp^2}{4\mathrm{d}k_\perp} - \mathrm{i}(k_x\langle x\rangle + k_y\langle y\rangle)\right), \qquad (20)$$

where: $k_{||}$, $k_\perp$ are components of the wave vector parallel and perpendicular to average velocity vector, respectively. The transformation between ($k_{||}$, $k_\perp$) and (k_x, k_y) components is given by:

$$k_{||} = k_x\cos\phi + k_y\sin\phi, \qquad k_\perp = -k_x\sin\phi + k_y\cos\phi. \qquad (21)$$

The reverse transformation can be written as follows:

$$k_x = k_{||}\cos\phi - k_\perp\sin\phi, \qquad k_y = k_{||}\sin\phi + k_\perp\cos\phi. \qquad (22)$$

Hence, the initial photon state is defined by the following parameters:

- $\langle x\rangle$, $\langle y\rangle$ - initial position of the packet center,
- k, ϕ - the photon wave vector (amplitude and direction),
- $\mathrm{d}k_{||}$, $\mathrm{d}k_\perp$ - the standard deviation of the wave vector (parallel and perpendicular to average velocity vector).

Appendix C. The atom-photon interaction

Using the dipole approximation (e.g.[29]) we can write the atom-electromagnetic field interaction coefficients in the following form:

$$g_{n_x n_y} = \langle 0, \psi_s|\hat{H}_{int}|\psi_f, 1\rangle = -\frac{\mathrm{ie}}{m_e}\mathrm{e}^{i\vec{k}\vec{r}_a}\sqrt{\frac{\hbar^3}{8\epsilon_0 V\omega(\vec{k})}}\int \mathrm{d}^3\vec{r}\,\psi_s^*(\vec{r})\vec{\epsilon}\circ\vec{\nabla}\psi_f(\vec{r}), \qquad (23)$$

where $\hat{H}_{int}$ is the interaction Hamiltonian between the atom and the field, $|0\rangle$ and $|1\rangle$ are the vacuum state and single-photon state (for mode n), respectively. ψ_s and ψ_f are initial and final electron wave functions, respectively. The $\vec{r}_a = [x_a, y_a]$ vector is position of the atom in the cavity.
Let us define coefficients:

$$\Gamma = \frac{\mathrm{e}}{m_e}\sqrt{\frac{\hbar^3}{8\epsilon_0 V\omega_A}}\int \mathrm{d}^3\vec{r}\,\psi_s^*(\vec{r})\vec{\epsilon}\circ\vec{\nabla}\psi_f(\vec{r}), \qquad (24)$$

where $\omega_A = E_A/\hbar$ and E_A is energy of the atom excited state. Using Eq. (24) we can write the interaction coefficients in the following form:

$$g_{n_x n_y} = -\mathrm{i}\Gamma\sqrt{\frac{\omega_A}{\omega(\vec{k})}}\mathrm{e}^{i\vec{k}\vec{r}_a}. \tag{25}$$

Appendix D. The simulation code in Qiskit

Code Listing 1. The free evolution algorithm (for one time step)

```
from qiskit.circuit.library.standard_gates import RYGate
from qiskit.circuit.library.standard_gates import PhaseGate
qc = QuantumCircuit(nq)

def mojGray(a):
    return a ^ (a >> 1)

bramkiC = list() # list of control gates
for i in range(0, nq, 1):
    bramkiC.append(i)

#free evolution of A:
qc.p(phiA, nq-1)

#free evolution of F:
for i in range(nq - 1):
    qc.x(i)
for i in range(na):
    j = mojGray(i)
    zmiana = j ^ mojGray((i + 1) % na)
    qc.append(PhaseGate(phi0[j]).control(nq-1), bramkiC)
    qc.x(int(math.log2(zmiana)))
for i in range(nq - 1):
    qc.x(i)
```

Code Listing 2. The interaction algorithm (for one time step)

```
#A-F interaction:
for i in range(1, nq-1,1):
    qc.x(i)
qc.cx(nq-1,0)
for i in range(1, na, 1):
    j = mojGray(i)
    qc.append(RYGate(phiTT[j]).control(nq-1), bramkiC)
    zmiana = j ^ mojGray((i + 1) % na)
    qc.cx(nq-1, int(math.log2(zmiana))) # ACNOT
    qc.x(int(math.log2(zmiana)))  # ACNOT
for i in range(0, nq-1, 1):
    qc.x(i)
```

References

[1] Feynman, R. P. Simulating physics with computers. *Internat. J. Theor. Phys.*, 21:467–488, 1982.

[2] Shor, P. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, SFCS-94. IEEE Comput. Soc. Press. doi:10.1109/sfcs.1994.365700.

[3] Grover, L. K. From schrodinger equation to the quantum search algorithm. *Am. J. Phys.*, 69:769–777, 2001.

[4] Lloyd, S. Universal quantum simulators. *Science*, 273:5278, 1996.

[5] Schaetz, T., Monroe, C., and Esslinger, T. Focus on quantum simulation. *New Journal of Physics*, 15:085009, 2013.

[6] Lanyon, B. P. Universal digital quantum simulation with tapped ions. 2011.

[7] Childs, A., Maslov, D., Nam, Y., Ross, N., and Su, Y. Toward the first quantum simulation with quantum speedup. *PNAS*, 115(38), 2018.

[8] Wecker, D. Solving strongly correlated electron models on a quantum computer. *Phys Rev A*, 92:062318, 2015.

[9] Kokail, C., Maier, C., and van Bijnen, R. Self-verifying variational quantum simulation of lattice models. *Nature*, 569, 2019.

[10] Wecker, D., Bauer, B., Clark, B., Hastings, M., and Troyer, M. Gate count estimates for performing quantum chemistry on small quantum computers. *Phys Rev A*, 90:022305, 2014.

[11] Hempel, C., Maier, C., and Romero, J. Quantum chemistry calculations on a trapped-ion quantum simulator. *Phys. Rev. X*, 8:031022, 2018.

[12] Jordan, S., Lee, K., and Preskill, J. Quantum algorithms for quantum field theories. *Science*, 336:1130âĂŞ1133, 2012.

[13] Sinha, S. and Russer, P. Quantum computing algorithm for electromagnetic field simulation. *Quant. Inf. Proc.*, 9:385–404, 2010.

[14] Garg, N., Parthasarathy, H., and Upadhyay, D. Real-time simulation of h-p noisy schrödinger equation and belavkin filter. *Quant. Inf. Proc.*, 16(121), 2017.

[15] Yepez, J., Vahala, G., and Vahala, L. *Quant. Inf. Proc.*, 4(6), 2006. doi: 10.1007/s11128-005-0008-8.

[16] Wiesner, S. Simulation of many-body quantum systems by a quantum computer. 1996.

[17] Zalka, C. Efficient simulation of quantum system by quantum computers. *Fortschr. Phys.*, 46:877–879, 1998.

[18] Strini, G. Error sensitivity of a quantum simulator i: a first example. *Fortschr. Phys.*, 50:171–183, 2002.

[19] Benenti, G. and Strini, G. Quantum simulation of the single-particle schrödinger equation. 2007.

[20] Ostrowski, M. Quantum simulation of the tunnel effect. *Bulletin of the Polish Academy of Sciences Technical Sciences*, 63(2):379–383, 2015.

[21] Ostrowski, M. Simulation of the schrödinger particle nonelastic scattering with emission of photon in the quantum register. *Bull. Pol. Ac.: Tech.*, 68(5), 2020.

[22] Ostrowski, M. Simulation of photon emission by an excited atom in the quantum register. In *TEWI 2021 (Technology, Education, Knowledge, Innovation)*, pages 139–153. Lodz University of Technology, 2021.

[23] Anastopoulos, C. Decays of unstable quantum systems. *Int J Theor Phys*, 58:890–930, 2019.

[24] Vaintraub, S., Blaum, K., Hass, M., Heber, O., Aviv, O., Rappaport, M., Dhal, A., Mador, I., and Wolf, A. Simulations of β-decay of ^{6}he in an electrostatic ion trap. 2014.

[25] Buzek, V., Drobny, G., Kim, M., Havukainen, M., and Knight, P. Numerical simulations of atomic decay in cavities and material media. *Phys.Rev.A*, 60(1):582–592, 1999.

[26] URL `https://qiskit.org/`.

[27] Childs, A., Su, Y., Tran, M., Wiebe, N., and Zhu, S. Theory of trotter error with commutator scaling. *Phys. Rev. X*, 11:011020, 2021. URL `https://doi.org/10.1103/PhysRevX.11.011020`.

[28] URL `https://en.wikipedia.org/wiki/Gray_code`.

[29] Haken, H. *Light: Waves, photons, atoms*. North-Holland Pub. Co., 1985.

[30] Ostrowski, M. Loading initial data into the quantum register. In *XV International Conference “System Modelling and Control”, Łódź, Poland, 23-24 September 2013*. 2013.

Enemy Machine Learning-Based System in a cRPG-based Game

Mateusz Płonka[0009–0003–1008–0446]**, Damian Pęszor**[0000–0002–3754–2212]

Silesian University of Technology
Department of Graphics, Computer Vision and Digital Systems
Akademicka 16, 44-100 Gliwice, Poland
mateuszplonka.tg@gmail.com
damian.peszor@polsl.pl

Abstract. *The article presents the design and implementation of an antagonist's AI system for a cRPG game using machine learning. By integrating Behavior Trees with reinforcement learning, game immersion is ensured while facing the AI-driven opponent. The implemented system offers tools for both training and gameplay. The AI agent autonomously selects actions, guided by environmental cues and state assessments. Its decision-making, such as choosing attacks aligned with the opponent's defenses and using supporting actions when truly needed, underscores its rationality, taking into account action costs.*
Keywords: *computer games, artificial intelligence, machine learning, agent*

1. Introduction

Over the years, computerization has become widespread, leading to an increase in user requirements. The ongoing problem is to meet the current demand, which requires a system capable of analyzing information, solving problems, and making independent decisions. The solution to this problem is the rapidly developing field of Artificial Intelligence (AI) [1]. Its main task is to conduct research towards the creation of devices and computer programs that will be able to conduct a thorough analysis of data and skillfully draw conclusions from them while creating the illusion of contact with a rational being. The development of AI systems is particularly important in the computer game industry. Users, expecting the level of challenge to increase over time spent in the game, began to emphasize systems in which opposing Non-Player Characters (NPCs) would learn the players' fighting techniques. The use of Artificial Intelligence also had a huge impact on increasing the level of immersion in the world presented in games. Giving the illusion of consciousness to the machine allowed players to feel surrounded by thinking characters, rather than based on fixed algorithms and dialogue texts of

NPCs. The genre that specifically is focused on NPCs is computer Role Playing Games, which has many varieties, of which it is worth mentioning action cRPG (acRPG), because the enemy systems included in this subgenre are closest to the mechanism described in this work. This category, is a subgenre of Role-Playing games as defined by [2], a combination of Action and Role-playing genres as defined by [3], or a combination of different foci of the same gameplay facet by [4]. While cRPGs in general encompass the entirety of dimensions proposed in [5] and [6], acRPG is characterised by realtime Pace and mimetic Representation. Thus, acRPG emphasizes both the plot and the experience of interacting with the presented world, as well as the pace of the game and quick rational decision-making. The use of AI allows for an illusory representation of the human consciousness of NPCs, increasing the sense of immersion of the presented world and posing more difficult challenges to the player by giving enemies the ability to think logically, make decisions, or draw conclusions, e.g., from fights or battles. This paper proposes a acRPG-based framework for a machine learning (ML) system which adapts character's behavior and skills to the surrounding environment, situation, or the player's level of training, while taking into account the level of experience acquired while fighting the user or another enemy. As the game progresses, the enemy must develop the ability to predict, make rational decisions, and perform actions that are a better option for a given problem. The system learns through subsequent approaches to interacting with the player, developing combat skills, and responding to attacks, ultimately putting the player in front of an increasingly greater challenge. The proposed framework is based on a Behavior Tree (BT) that covers possible scenarios and Reinforcement Learning (RL) for real-time combat control. The paper is constructed as follows: The 2 section introduces the problem and presents a literature analysis, while the 3 section describes the methods used in the proposed solution. The 4 section presents the outcomes obtained during the implementation of the system, the structure of the AI, and the results of the best training. Finally, Section 5 concludes the article with an additional presentation of potential paths for further development.

2. State of the art

The problem of implementing AI in games is the subject of many books and scientific publications. An example is an article discussing the use of Tree Pruning technology as a new way to develop strategy game algorithms, published by Greer, K. [7]. The study involved testing existing chess algorithms for the most human-like behavior, while maintaining the highest possible performance when determining the next move. The article mentions the Chessmaps Heuristic algorithm created in 1998 as a positional evaluator for heuristic searches. The goal of the program was to train a neural network based on the recorded games of both ordinary players and chess masters. The relationship model created in this way defined the relationship between the fields a player controls and the fields to which the player moves his pawns. This created input data for teaching the neural network that controls the game. This solution quickly proved too limited to create the illusion of a human game. For this reason, tests using a combination of several heuristic algorithms began to be carried out, and these showed a significant acceleration in the process of finding the best response. The solution to this problem was the use of data compression techniques, such as Tree Pruning, which allowed adding and removing sequences in real time, representing the given steps as movement on the board without depending on the position of the pawn, and shortening the search path by logically eliminating groups of solutions.

AI technology is also an integral part of strategy games, as shown in [8]. The authors present innovative methods for AI deduction in large-scale strategy games using RL. Games of this type are distinguished by their high complexity, automated actions of opponents based on a wide set of rules. The ability of the enemy to decide, adapt, and learn is widely recognized as one of the most important features of this genre. Moreover, rule-based systems lack adaptability which, combined with the complexity of the mechanism of strategy games, prompted the authors to research in this area. The conclusions of the study showed the potential of RL technology, which is a branch of ML, to satisfy in a reasonable amount of time, thus surpassing the techniques used so far. This technology is characterized by the absence of a supervisor, is based on the principle of trial and error, and determines a fixed target without having to directly describe the agent's behavior. It is useful in situations where behavioral strategies are unknown. As such, it is quite a promising approach for large-scale strategy games. Research was carried out by initially training the agent on a hand-made model created based on rules and tactical actions. The system was then allowed to learn by playing on a pre-prepared AI, and was eventually put into Self-Play mode, in which it independently refined its strategies by running new simulations. One of the tests conducted was a simulation of a complex battle scenario between the French and Russian armies. Using the previously mentioned techniques, a system was constructed to control decision making at the high level of the French army, issuing strategic orders at the lower level, while the existing AI controls the Russian army and follows French orders flowing down the hierarchy from the level controlled by the learning AI. The commander's decisions were enforced in a fully distributed manner, allowing for a variety of agent coordination techniques.

The development in the field of AI in computer games is described accurately in [9]. The article lists numerous examples of the application of AI, as well as the purpose of research in this area and the problems they solve. Among the examples is the use of complex computer game systems to simulate experimental environments. Current game engines are reaching a level of complexity comparable to that of the real world, enabling simulations that focus on cognitive issues without the added burden of using physical sensors and motor systems, such as those used in robotics. This allows more attention and time to be devoted to research and its results, rather than adapting the environment and selecting an appropriate research group. Research to develop AI in games itself is also presented, identifying AI as the most future path for the development of the industry. Graphics have now reached a level where visually stunning games are the norm rather than the exception, causing game developers to seek innovation other than excellent graphics. This has inevitably contributed to the popularization of research in fields related to the topic of AI, laying the foundation for the key functionalities of games being developed in the future.

It is worth mentioning examples of games that depend on AI. One such example is *ECHO*, a sci-fi game by the Ultra Ultra studio. The game features an unusual system for training enemies' AI based on the actions the player performs. The movements, behaviors, and reactions of the opponents are consistently developed as the game continues through the use of ML. This has a drastic effect on the difficulty level - the player is forced to think through his every move, as using any tactic only makes the enemy stronger. The limitations of the technology used are cleverly covered by the storyline: the network collects information about the player's movements only when the light is on. When darkness sets in, the application has the ability to expand the neural network, allowing the user to perform any interaction, arguing that it blinds opponents. When the map lights up again, the game uses the updated network model and starts collecting new information. This procedure allowed the network to be updated less frequently, increasing the game's performance.

Another example is the first-person horror game *Alien: Isolation* produced in 2014 by The Creative Assembly studio. The main system worth noting is the multilevel AI of the enemy moving around the map. The AI of the enemy has been separated into two layers:

1. **Macro AI (Director)** - an external layer, which is not directly connected to the antagonist, but collects information about the current state of the game and player. Examples of variables taken into account in the analysis are: the distance of the enemy to the player, the time spent in close proximity to the player or the time of visibility on the motion detector. Based on the parameters, providing only the probability of a player in a given location, intuitive behavior is simulated. This information is then processed by the lower AI layer (Micro AI).

2. **Micro AI** - the lower layer implemented in the form of a work system and a Decision Tree. It is responsible for allocating tasks based on a priority queue. The way of execution and response to given inputs are implemented by the Decision Tree model. The enemy has more than 100 nodes corre-

sponding to different responses tied together by branches of the tree, creating a logical path to get the desired action at any given time.

Some of the paths available to the model are initially unreachable. Unlocking is done by consistently teaching the agent with data from the outer layer, mainly based on the user's behavior, contributing to an increase in difficulty as the game progresses.

Watch Dogs: Legion by the Ubisoft studio features an interesting use of AI. The characters moving around the map are completely controlled by AI and can be recruited by the player. The NPCs have their own apartment, job, interests, and relationships with other NPCs existing in the game, which can affect game difficulty. Side tasks are also generated depending on the team currently in place, its internal relationships, or the characteristics of its members - occupations, needs, and interests. The use of AI has advanced in this production to the extent that it is the user who decides on the future creation of the world. Each character can become a partner or remain an NPC adhering to his routine. The choice is up to the player.

Table 1 compares the approach of developers to the topic of using ML in the games they create. Collated are the games described earlier, but also other popular games using AI.

Table 1. Table comparing the features of game implementations

	Alien: Isolation	**Bioshock**	**ECHO**	**The Division**	**Watch Dogs Legion**
AI learns from player	YES	YES	YES	NO	NO
AI affects the storyline	NO	YES	NO	NO	YES
Model update	NO	NO	YES	NO	YES
Use of Behavior Tree	YES	YES	YES	YES	NO

Analyzing the examples of AI applications found in the academic literature and existing commercial solutions, the first observation is undoubtedly the enormity and variety of ways in which this technology is applied. It can be concluded that depending on the problem at hand, AI takes on a different, distinctive appearance.

In the context of the problem discussed, two main algorithms should be specified. The first is a system for deciding the behavior of the enemy, based on the observation of the current situation and numerous state machine parameters. In the case of this problem, it is important to note the particular pattern of construction of AI in titles that use this field of IT to create antagonistic characters. These very often consist of complex Behavioral Trees, one such example is previously mentioned *Alien: Isolation*, which implements a Behavior Tree as the main structure to manage the enemy behavior.

The second problem under discussion is the creation of an AI system capable of making decisions on responses related to combat and defense. The main premise of the model would be to develop its skills based on observed events, maximizing the accuracy of its choices with each successive battle. Unlike the previously presented standard, in games that use AI to create environments, human relationships or make complex decisions, developers most often choose solutions involving complex heuristic algorithms, dynamic sequences of movements (Tree Pruning), Machine Learning (reinforcement learning) or linking several solutions.

3. Materials and methods

3.1. Problem statement in the context of cRPG

The main problem of the paper is to create an antagonist agent system in a cRPG game that has the ability to develop its skills based on observed events in real-time. Encounters with the opponent are based on the development of the enemy's AI by fighting against a second instance of the agent or the player himself. By choosing the active learning mode, the user can actively participate in the development of the opponent's AI, teaching it more unpredictable tactics. Regardless of the mode of play, the main goal is to defeat the opponent in the shortest amount of time.

The use of ML technology in cRPGs does not have a significant impact on control systems or gameplay patterns. However, the technology allows one to increase the impassiveness and difficulty level of the game over time. ML can not only be used to develop the AI of opponents but also of any NPC.

In the proposed framework, extensive state machine combined with two types of ML solutions is used. The first is a **Behavioural Tree** used to make decisions and take actions based on the current state of the environment surrounding the NPC. Examples of reactions include following the player when he comes within sight of the agent or fleeing to the safest possible place when wounded. The model also selects its reactions depending on the situation, moving freely around the map in the absence of an opponent nearby. The second type of ML is based on RL. The agent is provided with numerous observations about both its current state and the surrounding environment, as well as the value of the opponent's or player's state machine. On the basis of the information provided, the model makes a decision regarding the selection of one of the actions made available to it. The actions performed are then evaluated, and the sum of the separated points in a given episode is used to optimize decision making in later battles. An additional mechanism that defines the logic of the rules of the above AI modules is the combat rules system that defines the interrelationship of the relevant actions and their effect on the NPC.

3.2. Main concepts of the framework

The common idea of the four modes available in the application that demonstrates the framework is to perform a simulation of a magic duel. Confrontation can take different forms, depending on two main criteria. The first determines the level of user intervention in the actual duel - an active player who is the model's opponent and a passive observer of the duel between two AIs. The second, on the other hand, clarifies the question of whether AI will develop its network of connections based on the received signals or merely use a previously learned model. Based on the combination of the above-discussed decision parameters, four modes were created, providing access to each of the possible combinations of the ML process and gameplay guidance. These combinations of modes are shown in Table 2.

Table 2. Table of combinations of mode criteria

	Active player (opponent)	**Passive player (observer)**
Expanding the model by teaching with reinforcement	Training with a player	Training of two AI
Gameplay with a created model	Game with a player	Game of two AI

The cycle of the application, regardless of mode selection, includes a series of magic duels discussed above. The primary goal of each confrontation is the strategic use of spells to reduce the opponent's life points to 0. When one side achieves the desired goal, it wins and the duel arena is reset to its initial settings, initiating a new battle. A particular duel involves 10 iterations, each lasting 5 minutes. At the end of each iteration, the game's parameters are reset to their initial values, and participants are redeployed in the arena. In a situation where the goal is not achieved by either side in a given time, the round is considered to end with a draw.

A key element of the application is the confrontation mechanism, based on interactions between three elements, these being: **Fire**, **Water** and **Ice**. Each element dominates the other, while being vulnerable to the third. The system of interaction between the elements is analogous to the rules of the game Paper-Stone-Scissors. For example, a user protected by a fire shield is completely insensitive to ice attacks, although he becomes more vulnerable to water attacks, which have the potential to neutralize the protection used.

3.2.1. Game with a player

The application includes a mode that allows the user to run a game against a trained AI model. GUI provides the user with information on the current-level status, the value of the opponent's parameters, and the value of its own state machine. The user has access to information such as the current health, mana, strength, and duration of the shield and healing spell, as well as the number of battles won since the last launch. It also has actions to perform which are moving the character, rotating the camera, accelerating, performing a jump, projectile, or shield spell.

Each use of a spell involves a subtraction of mana points with a value depending on the complexity of the spell. After its execution, the system is paused for $1s$ which is signaled by charging the element symbol. Attempting to invoke a spell in the absence of mana or an unregenerated spell system involves an error message displayed in the upper left corner. The goal of the game is to defeat the opponent by using offensive spells according to the rules of elemental logic and the rational use of protection and healing spells. A screenshot of the gameplay between the bot and the player is shown in Fig. 1.

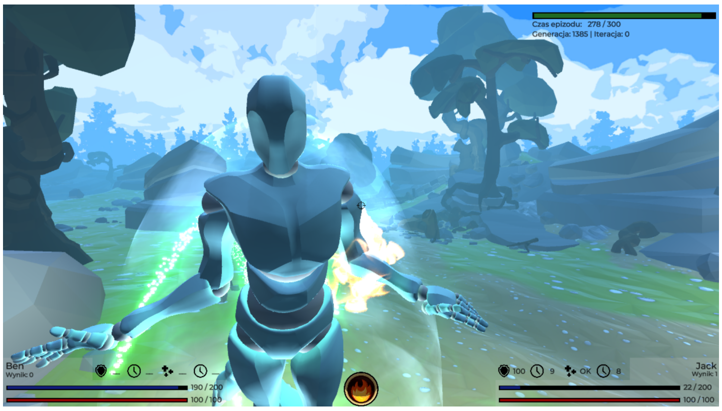


Figure 1. Screenshot - Game with player

3.2.2. Training with a player

The mode allows the user to perform a battle during which the AI learns consistently from the player's actions while developing its model. The gameplay and character controls are identical to the previous mode. An example screenshot of the learning process on the training map is presented in Figure 2.

The training modes are distinguished by an additional console window, displayed along with the application process and containing real-time generated, statistical data on the current learning progress - the mean and standard deviation of the reward points assigned by the RL system.

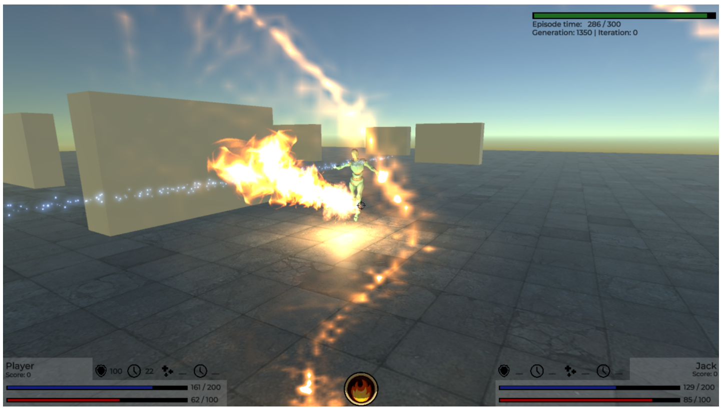


Figure 2. Screenshot - Training with player

3.2.3. Game of two AI

This is a duel mode between two NPCs, which are hostile to each other and use a previously trained model. During the gameplay of this mode, the user plays the role of an observer, with no active influence on the course of the duel. The observer is free to modify the transformation of the camera. An additional option for the observer is the functionality of locking the movement and rotation system, allowing the viewer's view to be set up as a static camera. The game process between two agents is presented in Fig. 3.

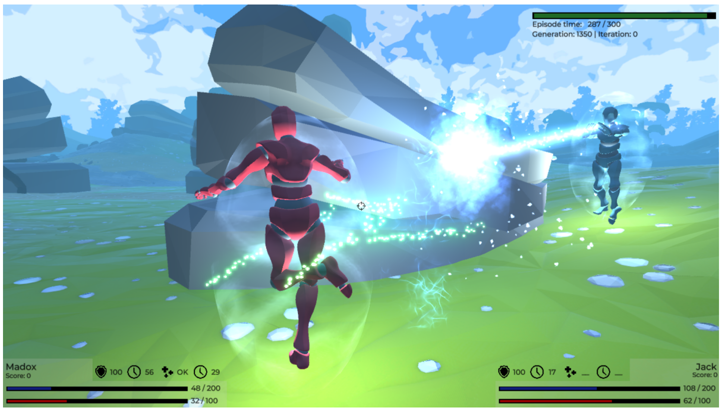


Figure 3. Screenshot - Game of two AI

3.2.4. Training of two AI

AI training mode using the self-play gameplay mode, in which the AI makes conclusions based on a fight with a second instance of the model. The mechanism allows the model to evolve without user intervention, additionally carrying out the development process at a 10x accelerated rate. The observations of both agents improve the final model, with no logical connection to each other during the battle. A screenshot of the learning of two agents is presented in Figure 4.

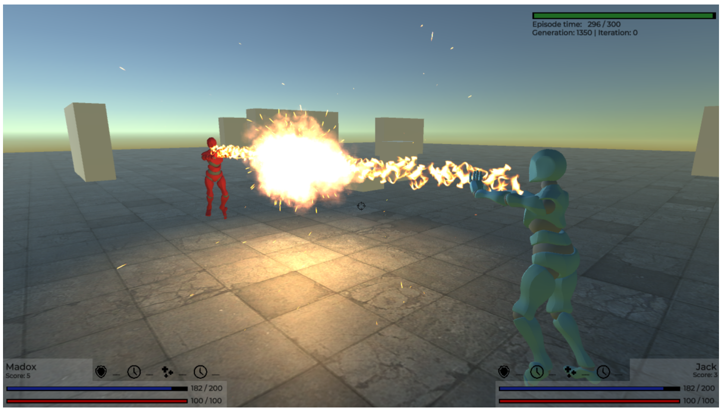


Figure 4. Screenshot - Training of two AI

The mode in question has functionalities analogous to the other modes. The user plays the role of an observer and, with the help of a system of position locking and camera rotation, has the option of minimizing the window while the learning is carried out. The mode also generates statistical data in real time.

3.3. Algorithms

Two types of ML were used to develop the decision making and response skills of the agents:

3.3.1. Behavioral Tree

The algorithm used for decision making and action execution is based on the current state of the environment surrounding the NPC. This mechanism is grounded on nodes representing a given state or action, interconnected through logical expressions. The procedure to identify the most appropriate action for a given situation commences from the root node. Subsequently, information is conveyed to lower nodes, which analyze the state of passed parameters or represent basic logical operations. The only exception to this rule are the leaf nodes, which represent a specific action.

Tree elements can perform simple operations, such as checking the state of a variable, and more complex tasks, like terrain analysis in search of an optimal refuge location. Typical responses include following a player when he is within the visual range of the model or fleeing to the safest possible location when health levels are critically low.

Upon completion of its operation, a given node can return three states: *Success*, *In Progress*, or *Failure*. These states are subsequently analyzed by the previously mentioned Boolean logic nodes and conveyed towards the leaf determining the action to be executed. System optimization is as vital as the quality of the information returned, which is why the BT algorithm has been implemented alongside a mechanism minimizing tree search. This mechanism is designed to skip searching for tree nodes that are not critical, for instance, classification, thus increasing the quality of the returned information and the time it takes to search for it.

3.3.2. Reinforcement learning

The second type of ML used in this paper is reinforcement learning. In this approach, a number of observations about both the current state of the adversary representing the model and the surrounding environment, as well as the value of the adversary's state machine, are input into the model. On the basis of these inputs, the model decides which action to take from the available options. Such actions include choosing the appropriate spells - distance, area, or healing - or refraining from any action. Each decision made is evaluated, and the total points earned in an episode serve as the basis for optimizing decision-making in subsequent battles. The training process, which involves replaying specific episodes of the battle, is repeated many times to construct an implicit mathematical model, developed on the basis of the points earned. A graphical representation of the system cycle proposed in [10] is shown in Fig. 5.

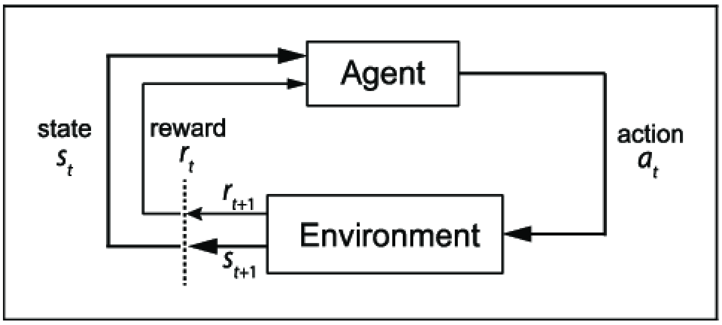


Figure 5. Reinforcement learning - algorithm cycle [10]

The goal of this operation is to train the model to make autonomous decisions, thus creating the illusion of behavior closely mirroring human responses. Examples of such actions include using a spell that is the best counter to the enemy's current shield, using an expensive healing spell only when health is critically low, or using a shield that counteracts frequently used enemy techniques.

The model receives observations on the state machines of both agents and the parameters of the ongoing game. Numeric values are normalized, while state-description observations are passed in the OneHot system format, assigning one bit for each possible state, allowing only one active state at a time.

Actions are determined by the model, specifying the choice and type of spell used. The first action is "Spell Type," specifying the type of spell selected, while the second, "Element Used," selects the element-dependent spell variant.

The reward system is divided into subsections, each of which relates to a segment of the application. Reward values are assigned by different systems, some related to level management, while others are related to the agent's character parameters. Rewards for an episode, basic combat, use of a shield in combat, healing, and area spells are all part of the reward system, each of which has specific values assigned to different scenarios and actions taken during the game.

3.4. Functional requirements

In the *Training with the player* and *Game with the player* modes, the user acts as a character, actively engaging in interactions with an experienced or constantly learning AI model, following identical game rules and having analogous action capabilities to the opponent. Functions available to the player include moving in two axes on inclined surfaces up to 45 degrees, jumping with a height equal to half the length of the character model, accelerated movement up to 150% of the base speed, and choosing one of three elements (Fire, Water, Ice) as the current type of attack and defense. In addition, the player has access to spells: elemental projectile, shield, healing, and area attack.

The other modes *Training two AIs* and *Gaming two AIs* allow the user to observe the operation of the system as an observer who can move freely in three axes in the battle scene space, but without the ability to interfere with the battle and without physical representation in the game world. The observer's functions include free movement without the constraints of gravity or collisions with physical objects, a temporary motion lock that mimics a static camera, a training record, and a safe exit from the application with an automatic training record.

3.4.1. Reliability

The system implements the function of replaying a new duel, using existing objects on the stage. This process is initiated after the duel is over, with the activation of a mechanism that resets the stats values to defaults and allocates a new safe space on the map for the next fight. At the same time, the system performs a verification of the safety and availability of selected points in space for the player or NPC. This mechanism is used when allocating a new location after a completed duel and identifying the enemy's current point of interest in wandering mode. Duels, in both training and gameplay modes, are divided into five-minute iterations, after which the location on the map is randomly changed using the safe location mechanism. This system provides a solution to situations in which no participant shows interest in a rival. Additionally, to prevent endless practice sessions, the duel ends in a draw after ten iterations. The application also creates automatic backups of the training model file after every 100,000 decisions made by non-player characters, protecting the data from loss due to destruction of the model during a training session.

3.4.2. Performance

The system automatically eliminates redundant objects from the battle scene, reducing the load on the CPU and GPU. The application features a built-in training map with a minimalist design, which minimizes the load on the graphics card during prolonged training sessions.

3.4.3. Usability

The application comes with an external starter program to facilitate user interaction with the application and access to its functions. The user interface (UI) design has been optimized to communicate relevant information in an understandable and accessible form. An error communication system informs users of potential problems in both the launcher and game or training modes. The application's window and UI are responsive, automatically scaling to the size of the screen in use, and the ability to lock training mode and minimize the game window makes it easier to use the computer during a training session.

3.4.4. Security

The system's security include a mechanism that blocks access to applications in the event of a critical error. The system informs the user about the cause of the problem and allows the user to safely close the program. Measures are also in place to prevent invalid data from being entered in the parameters of the startup program, and in the absence of a Python environment of version 3.6 or later installed, the application can be run in game mode using a previously trained model provided in the presentation application or trained on another device.

4. Results

The research resulted in a model of an agent capable of autonomous action selection based on information from the environment, the agent's data machine, and the adversary. All this is based on the mechanics of RL. The actions taken by the agent exhibit a high degree of rationality, which is manifested, among other things, in the selection of spells that respond to the characteristics of the opponent's shield and the use of healing or sphere spells in situations of real need, taking into account their mana cost.

Analysis of the resulting file confirms the effectiveness of the training. Fig. 6 illustrates the relationship between the value of the reward earned by the agent during successive battles. The upward trend is clearly marked, as confirmed by the red trend line placed in the aforementioned figure. The lowest reward value, at −283.45, was observed in the battle number 188, while the highest value, 19.86, was recorded in the battle 2205. The observed decrease in value in the last 200 battles may suggest a potential overfitting of the model.

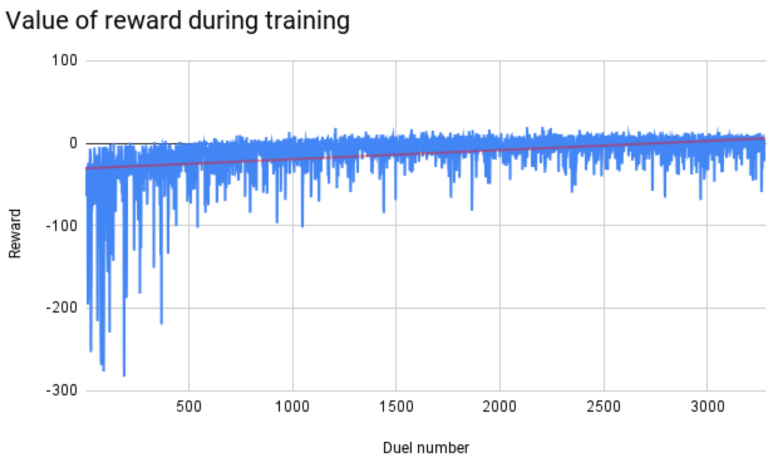


Figure 6. Value of reward during training

A reduction in the duration of the duel was observed as AI skills improved. Fig. 7 shows the relationship between the number of matches and the duration. The trend is clearly decreasing. The duration of the duels in the analyzed range ranged from 6 to 1563 seconds.

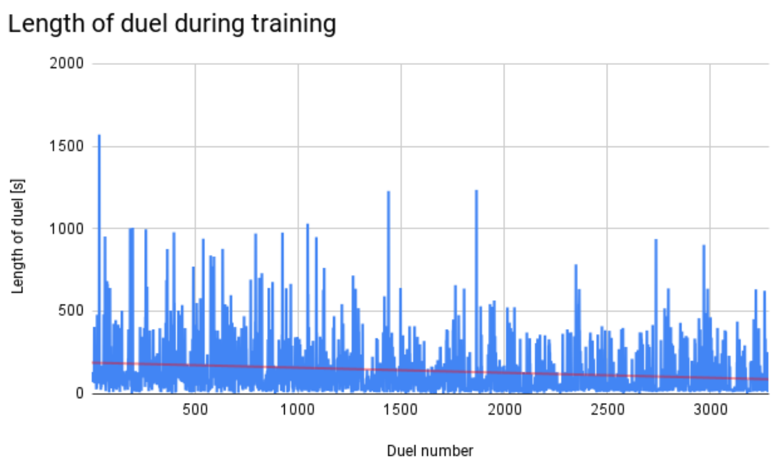


Figure 7. Length of duel during training

5. Conclusions

The main objective of this article was to design and implement the AI system of the antagonist in a cRPG game using ML. The result of the work on the described project was a complex NPC system equipped with two correlated AI systems. Combining the Behavioral Tree with Reinforcement Learning enabled maintaining immersion of the game with the artificial opponent, thus allowing the agent to make its own decisions based on previous experiences. The final implementation also had several tools to enable training and gameplay with the developed AI. The result of the work is a model of an agent capable of autonomous selection of actions based on information from the environment and its own and the enemy's condition. The actions taken by the agent show a high degree of rationality in the context of the problems they face. This is manifested in the selection of spells that respond to the characteristics of the opponent's shield and the use of healing or sphere spells in situations of real need, taking into account their mana cost.

The developed system has several potential avenues for further development. One of them is to expand the existing system with new functionalities, combat methods, or motor skills. The direction would allow the project to develop algorithmically and analytically, allowing for more complex research related to AI. An example analysis could be a test of the advantage between a smaller number of well-trained skills and a larger number of newly learned actions. A second possible development path is to create a new AI agent with different skills and actions relative to the current opponent. This would allow the development of observations on the mutual behavior of the two different models of RL against each other. Enforcing a hostile mindset would allow to study the process of adapting to the attacks of a new environment, having experience from the previous one.

References

[1] Goebel, R., Mackworth, A., and Poole, D. *Computational Intelligence a logical approach*. Oxford University Press, Nowy Jork, 1998. ISBN 9780195685725.

[2] Wolf, M. 6 genre and the video game. In *The medium of the video game*. University of Texas Press, 2002.

[3] Apperley, T. Genre and game studies: Toward a critical approach to video game genres. *Simulation gaming*, 37(1):6–23, 2006.

[4] Lee, J. H., Karlova, N., Clarke, R. I., Thornton, K., and Perti, A. Facet analysis of video game genres. *IConference 2014 Proceedings*, 2014.

[5] Aarseth, E., Smedstad, S. M., and Sunnanå, L. A multidimensional typology of games. In *DiGRA Conference*. 2003.

[6] Elverdam, C. and Aarseth, E. Game classification and game design: Construction through critical analysis. *Games and culture*, 2(1):3–22, 2007.

[7] Greer, K. Tree pruning for new search techniques in computer games. *Advances in Artificial Intelligence*, 2012.

[8] Madeira, C., Corruble, V., and Ramalho, G. Designing a reinforcement learning-based adaptive ai for large-scale strategy games. In *Proceedings of Second Artificial Intelligence and Interactive Digital Entertainment Conference*. Université Pierre et Marie Curie AND Universidade Federal de Pernambuco (UFPE), Paris, France AND Recife, PE Brazil, 2006.

[9] Fairclough, C., Fagan, M., Namee, B. M., and Cunningham, P. Research directions for ai in computer games. Technical report, Department of Computer Science, Trinity College Dublin, Dublin 2, Ireland, 2001.

[10] Galatzer-Levy, I., Ruggles, K. V., and Chen, Z. Relevance of machine learning to the study of stress pathology, recovery, and resilience. *Chronic Stress*, 2018.

Team Project: Łódka Wolontariatu - A Successful Example of University and Enterprise Cooperation

Mateusz Smoliński[1[0000–0002–3890–5943]], **Krzysztof Brzeziński**[2],
Marcin Kwapisz[1[0000–0001–5128–9009]],
Michał Karbowańczyk[1[0000–0003–3875–1101]],
Tomasz Ciszewski, Szymon Depcik, Adam Kapuściński, Daniel Modrzejewski, Paulina Papiernik, Damian Szczeciński, Kamila Topolska, Adrian Wojtasik, Radosław Zyzik

[1]*Lodz University of Technology*
Institute of Information Technology
al. Politechniki 8, 93-590 Łódź, Poland
mateusz.smolinski@p.lodz.pl
[2]*Accenture Polska*
University Business Park
Wólczańska 178, 90-530 Łódź, Poland
krzysztof.brzezinski@accenture.com

Abstract. *In this paper we describe the motivations, development process and the results of Łódka Wolontariatu project. The project goal was to address the issue of growing interest in volunteering activities in our city. The solution was to develop a multi-user information system, which provides City Hall of Łódź with support in coordinating many volunteering activities between city officials, organizations, volunteers and people in need of support. Development of such a complex system raised as many opportunities as challenges; it required careful preparation and continuous engagement from every project participant. This paper describes the way we have chosen to fulfill both City Hall and Lodz University of Technology requirements together with Accenture as an enterprise partner and project initiator. Finally we summarize positive results of the cooperation, obtained by all the participants.*

Keywords: *university-enterprise cooperation, educational cooperation, volunteering support*

1. Introduction

The positive impact of cooperation between universities and enterprises on economic development is obvious. This is evidenced by the existence of various institutions supporting initiatives in this area. For example, from our perspective - employees of Lodz University of Technology - we can mention organizations such as:

- At the national level: The National Center for Research and Development[1], Polish Agency for Enterprise Development[2]
- At the regional enterprise level: Lodz ICT Cluster[3]
- At the university level: severe cooperation support programs and dedicated organizational units[4]

Although cooperation between universities and business is generally associated with research and development activities, especially with the commercialization of scientific achievements, there is also a worth exploiting potential placed in the context of education. The forms of cooperation possible in such a partnership include students' participation in internships/apprenticeships organized by enterprises, as well as the implementation of projects commissioned by enterprises in the form of diploma theses[5][6]. These types of activities allow students to get an experience that is for various reasons impossible to achieve within a standard study cycle, such as teamwork in large-scale projects or the need to meet the business requirements of a real stakeholder. From the methodological perspective we can perceive that as the possibility to complete the Kolb's learning cycle[7]. Moreover, joint implementation of the project allows participants from both sides to build positive relationships, which may open the prospect of future long-term cooperation.

The Łódka Wolontariatu (further referred to as LW) project is a response to the growing interest in volunteering activities both on the part of residents - beneficiaries of volunteering, as well as volunteers and non-governmental organizations providing volunteering. The growing scale of the need for coordination of volunteer activities by the City Hall meant that the existing management methods turned out to be insufficient[8]. Coordinating volunteer activities over a large area requires ensuring efficient communication between the organizers of volunteering activities and volunteers. Volunteers choose offers of volunteer activities in which they want to participate. It is also important to facilitate communication between officials and residents, which allows for adapting volunteer offers to the current needs reported by residents. The LW project was created in pro publico bono partnership between the City Hall (as the stakeholder), IT companies and Łódź universities. In the design phase, students of the University of Lodz, Commerzbank and Fujitsu were involved in the work, while the main part of the project, that is, the part on which this paper is focused, was carried out by Accenture and students of the Network Infrastructure and Applications specialization at the Institute of Information Technology, FTIMS faculty, Lodz University of Technology in years 2022-2023.

2. Project assumptions and design

The LW system supports announcement of volunteer offers by various organizations (both governmental and non-governmental). Volunteers can apply for published offers, but the system checks the volunteers' applications and prevents the volunteer from registration in volunteering activities that overlap in time line. It is worth noting that in the LW system it is possible to create an account for minor volunteers who are over 13 years of age with the consent of their legal guardians.

Access to the functionality offered by the system requires a web browser and account registration. In LW system role-based access control was used. Depending on the access level assigned to the user account, the LW system offers different functionality for authenticated user. The following access levels for user accounts are distinguished in LW system:

- Administrator - the superior user of the system. Manages user accounts registered in the system; submitted offers; controls organizations
- Volunteer - access level representing users who want to help and take part in volunteering activities
- Citizen - a user who can report the needs, which can then be transformed into volunteer offers
- Organization member - access level assigned when the account is created by the owner of the organization. A user with this level has the ability to create and manage volunteering offers in their own organization
- Organization owner - access level representing the user who created the profile of a given organization. It also has an organization member level, extending its functionalities to include managing organization members
- Official - a user representing the City of Łódź Office. An account with this level can only be created by an administrator. His task in the system is to transform the needs of residents into volunteer offers
- Hotline - an access level that, like an official, can only be created by the administrator. Receives telephone reports from residents and enters them into the LW system
- Technical support - a user who helps other users in the system. Considers and solves all problems reported by other users of the LW system. The access level is also assigned only by the administrator

Two non-account related roles are also utilized. Anonymous (not authenticated) user is provided with a guest role with limited functionality. Moreover, a system role is utilized by some operations that are initiated by the LW system without any user action, allowing task automation.

3. Work distribution and organization

The organizational challenge that emerged in the conceptual phase was to determine the way in which the implementation of the project was to be integrated into the study program. As the scope of time and workload in the project did not allow for its completion within standard courses, the only available form was a diploma thesis. However, it should be remembered that the diploma thesis is, as a rule, an individual form, while the implementation of the project is by nature a form of teamwork. In order to eliminate the observed contradiction, it was decided to divide the expected functional scope of the project into nine modules:

1. Official module realized by Tomasz Ciszewski is providing functionalities for system administrator to manage the content available in the LW system[9]. The most important functionalities provided by the created module include: verification of organization profiles, verification of volunteering offers or creating news available for all users of the system.

2. Volunteer module realized by Szymon Depcik, which handles people interested in volunteering in Łódź city[10]. In this module main functionalities include: browsing and filtering volunteering offers, furthermore users with volunteer account have possibility to apply to volunteer offers. These users also have custom offer filter and list of skills represented by passed courses.

3. Volunteer attendance planning module realized by Kamila Topolska provides functionalities related to the time planning of volunteers' participation in volunteer offers by introducing the concept of a volunteer's calendar and invitations sent to a volunteer[11]. It is possible to plan participation in volunteering activities during the opening hours specified by the volunteer.

4. Citizen module realized by Daniel Modrzejewski and provides functionalities for reporting needs by citizens[12]. Then volunteering offers can be created based on reported citizens needs.

5. Account service module realized by Paulina Papiernik provides authorization and authentication in the LW system[13]. The module also provided mechanisms for editing user accounts, displaying user account statistics, and displaying and filtering the list of all user accounts in the LW system.

6. Non-governmental organization module realized by Damian Szczeciński provides management of the organizations and volunteering offers issued by

these organizations and volunteer applications submitted for specific volunteering offers[14]. In addition, it will provide the functionality of versioning volunteering offers and organizations in order to preserve the history of changes and eliminate the temporary unavailability of content for system's users.

7. Communication module realized by Adrian Wojtasik supports information exchange in the multi-user information system LW[15]. The purpose of the created module was to provide communication between users of the system and to deliver application programming interfaces which provided sending messages requiring user action, notifications, emails and managing conversations and messages assigned to reported issues.

8. Technical support and problem handling module realized by Radosław Zyzik is providing for LW system users information articles, which are divided into articles from the knowledge base and questions and answers[16]. The first type of articles contains information on volunteering, while the second type is the documentation of the system's operation. In addition, the functional module allows authenticated users of the system to report problems encountered in the LW system, which are managed by technical support staff.

9. Module for handling contracts and templates realized by Adam Kapuściński, this module provides streamline the management of volunteering contract templates by organizations offering volunteering, and to provide the ability to monitor signed contracts for both organizations and volunteers[17]. An additional functionality of the module is versioning of contract templates created by organization. This functionality ensures that system's users will always be able to verify the version of the contract for volunteering they have signed.

Each of the above-mentioned modules of the LW system contains over twenty use cases and required the extension of the web and network applications as well as the relational database that are part of the LW system. The scope of each of the modules was used to formulate the individual topic of the engineering thesis conducted by either Ph. D. Marcin Kwapisz or Ph. D. Mateusz Smoliński as the supervisor. Determination of the functional specifications, system architecture and technology stack selection, and implementing each module of LW system were carried out as part of an internship at Accenture under supervision of an experienced employee Krzysztof Brzeziński. He was also a co-supervisor for each engineering thesis and was responsible for compatibility, consistency and unification in modules of LW system. Finally, Ph. D. Michał Karbowańczyk was responsible for coordinating cooperation between our Institute of Information Technology and Accenture in the scope of LW project.

3.1. Organizational concerns

All system LW modules were implemented by the authors at the same time. A significant implementation problem was the dependencies between the functionalities that were implemented in various modules of the LW system. In many cases, module authors had to use code mocking in the modules they prepared. Students participating in the LW project used shared GIT code repositories, which facilitated the integration of their individually developed modules as part of their diploma theses. All elements of the LW system were deployed as separate Docker container images, which made it easier to run the LW system in a containerization environment[18].

It has also to be mentioned that the way of dividing the whole project into nine individual diploma theses imposed certain degree of risk on the students: tight relationships between the modules could lead to undesired dependencies. It would be very unsuccessful scenario if progress delay within one module had negative impact on another one and, in consequence, caused another student's thesis to be blocked from completion. This problem was considered with extraordinary attention during the conceptual phase; the modules were designed so that they are as loosely coupled as possible.

4. System architecture and technology

4.1. Implementation overview

The LW project is designed as the multi-user Internet system consisting of a SPA web application and stateless REST network application with a relational database as a data source. The SPA application is launched in the user's web browser, which uses server side rendering to generate GUI views. The network REST application runs in the Payara[19] application server environment, and the relational database operates in the PostgreSQL[20] database management system. The stateless REST application uses ORM to access the database, communication with the database utilizeses JDBC (Java DataBase Connectivity)[21] connections. The overall structure of the system is depicted in Figure 1. The multi-user LW system was designed according to three-layer architecture:

- data layer - for data storage a relational database embedded in the PostgreSQL relational database management system. Data structures in relational database structures were static. Automatic ORM (Object- Relational Mapping)[22] mapping mechanisms between the relational and object-oriented models were used. Data storage is possible through the use of the JPA (Java Persistence API), based on Hibernate library[23], which holds relationship between data perceived by the application as JPA entity objects and records in database tables[24][25].

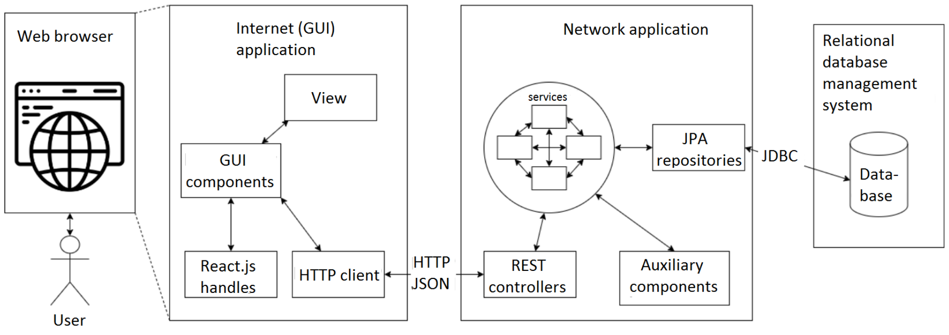


Figure 1. LW system architecture.

- business logic layer - was made in the form of a stateless network application, that was created with the REST (Representational State Transfer)[26, 27] architecture based on the Spring Boot application framework[28]. It provides a REST API web service through which it offers functionality for the web application. Due to the division of functionality into separate modules, one could assume that the microservices architecture would be the natural choice. However, the responsibility of individual modules is so closely related to each other that a monolithic architecture turned out to was a better choice. For database access network application uses JPA entity objects with Spring Data JPA with Java Persistence Query Language and Criteria API [29]. The network application uses three types of components to provide the required functionalities: controller, service and repository components. Controller components are responsible for providing API (Application Programming Interface) compliant with the REST standard. The service components are responsible for data processing in accordance with the business logic determined by the functional specification of the LW system. Repository components are responsible for access to data through the use of the ORM.

- presentation layer – was made in a form of a single page web application developed using the Next.js JavaScript/TypeScript library. The use of the Next.js application framework organized the structure of the web application project. Additionally, thanks to the hybrid SPA and SSR (Server Side Rendering) approach[30]. The use of SSR reduces the TTR (Time To Render) index of the website[31]. This technique also allows to improve SEO (Search Engine Optimization) results, making it easier to find the LW system website. The Next.js library reduces resource consumption on the client application side, which increases the application performance on older customer devices. This application provides a GUI graphical user interface that the system user can display in a modern web browser. The web application communicates with the business layer using the REST API and web sockets (i.e. communication module implementation). The use of the React.js library together with the Mantine library allowed to avoid code duplication and provide a uniform implementation of the graphical user interface. The Chart.js library provided data visualization in the form of charts

4.2. Security and data consistency

Spring Security was used to provide user authentication in LW system base on popular credentials: login and password[32]. Declarative access control was used in the components, which prevents the execution of a component method to which the user is not authorized in accordance with the access level assigned to his account. The LW multi-user system provides the ability to reset user passwords. From the administrative account level, it is possible to block or unblock selected user accounts. Furthermore the system provides accounting and event logging.

It is worth noting that once the user is authenticated, next requests from the web application to the REST web service use JWT (JSON Web Token)[33]. Such a token carries signed information in the JSON format, thanks to which the web application can authenticate and authorize the user sending the request to the network application. The use of tokens improves the use of the LW system by preventing the need for user authentication each time before sending a request to the network application.

Both web and internet applications provide strict verification of the correctness of data entered by the user in LW system. To ensure effective validation of data sent to REST controllers, annotations consistent with the JSR-303 Bean Validation API were used[34]. They are used to define validation rules for individual fields of data transfer objects.

The business logic layer uses transaction processing mechanisms with transaction boundaries managed by container, optimistic locks and ETag, which allow for maintaining data consistency and integrity and prevent unintentional data overwriting. The LW system includes errors handling that occur during its operation. If an error occurs while data processing, the system will log this error in the event log and the user will be notified about it by displaying an appropriate message.

5. Conclusions

The implementation of the LW project was successful thanks to the engagement and intense cooperation of all project participants. The division of the functionality of the LW system into nine modules allowed for their effective preparation as part of engineering thesis. Despite individually performed engineering thesis, the students jointly selected the architecture of the multi-user LW system and the technologies for implementing individual elements of the LW system as well as the method of integrating modules in individual elements of the LW system. During the development of the LW system, dependencies between functionalities belonging to different modules were successively resolved. Much care was taken to ensure the quality of the code, consistency in the preparation of LW system elements, and unification of the appearance of the graphical user interface. The entire LW system was created in accordance with modern standards for the production of multi-user Internet systems.

The implementation of the LW system was successful both in the application and diploma theses development aspects. Students gained the opportunity to apply the skills acquired in the course of their studies - in particular those related to the programming of network services, security of Internet systems, and transactional processing - to a real world project. As the most interesting new experience related to the implementation of the project, the students indicated the need to agree on the domain model of the system with the target stakeholder. This experience has already been included in the implementation of projects belonging to the study programme with the next year of students specializing in Network Infrastructure and Applications. All project participants described their experience as positive and the effects worth the effort.

References

[1] The National Centre for Research and Development. `https://www.gov.pl/web/ncbr/` [Accessed: September 2023].

[2] Polish Agency for Enterprise Development. `https://www.parp.gov.pl/` [Accessed: September 2023].

[3] Lodz ICT Cluster. `https://ictcluster.pl/` [Accessed: September 2023].

[4] Lodz University of Technology. Współpraca z biznesem. `https://p.lodz.pl/wspolpraca/wspolpraca-z-biznesem` [Accessed: September 2023].

[5] Geodecki, T. and Hausner, J. Współpraca uczelni z biznesem. polska na tle wybranych krajów Unii Europejskiej, 2023. `https://oees.pl/wp-content/uploads/2023/02/Wspolpraca-uczelni-z-biznesem_07.02.2023.pdf`.

[6] Bryła, P. Możliwości współpracy polskich uczelni wyższych ze sferą biznesu. *Studia Edukacyjne*, 31:95–112, 2014.

[7] Kolb, D. A., Boyatzis, R. E., and Mainemelis, C. Experiential learning theory: Previous research and new directions, in in perspectives on thinking,

learning and cognitive styles. In *Perspectives on Thinking, Learning, and Cognitive Styles*. Routledge, 2001.

[8] Łódź City Hall portal: Urban volunteering portal. `https://uml.lodz.pl/dla-mieszkancow/zdrowie/coronavirus-covid-19/wsparcie-wczasie-epiedmii/miejski-wolontariat/` [Accessed: September 2023].

[9] Ciszewski, T. Projekt Łódka Wolontariatu: moduł urzędnika, 2023.

[10] Depcik, S. Projekt Łódka Wolontariatu: moduł wolontariusza, 2023.

[11] Topolska, K. Projekt Łódka Wolontariatu: moduł planowania obecności wolontariusza, 2023.

[12] Modrzejewski, D. Projekt Łódka Wolontariatu: moduł obsługi mieszkańca, 2023.

[13] Papiernik, P. Projekt Łódka Wolontariatu: moduł obsługi kont, 2023.

[14] Szczeciński, D. Projekt Łódka Wolontariatu: moduł organizacji pozarządowej, 2023.

[15] Wojtasik, A. Projekt Łódka Wolontariatu: moduł komunikacyjny, 2023.

[16] Zyzik, R. Projekt Łódka Wolontariatu: moduł wsparcia technicznego i obsługi problemów, 2023.

[17] Kapuściński, A. Projekt Łódka Wolontariatu: moduł obsługi umów i wzorców, 2023.

[18] Poulton, N. *Docker Deep Dive*. 2023.

[19] Payara. Payara platform documentation. `https://www.payara.fish/learn/payara-platform-documentation/` [Accessed: September 2023].

[20] Fontaine, D. *The art of PostgreSQL*. 2022.

[21] Oracle. JDBC Developer's guide and reference, 2022. `https://docs.oracle.com/en/database/oracle/oracle-database/21/jjdbc/` [Accessed: September 2023].

[22] Lorenz, M., Hesse, G., and Rudolph, J.-P. Object-relational mapping revised - a guideline review and consolidation. In *Proceedings of the 11th International Joint Conference on Software Technologies (ICSOFT 2016*, volume 1, pages 157–168. SCITEPRESS, 2016.

[23] Ottinger, J. B. *Beginning Hibernate 6: Java Persistence from Beginner to Pro*. Apress, 2021.

[24] The Eclipse Foundation - Jakarta Persistence Team. Jakarta Persistence 3.1, 2022. `https://jakarta.ee/specifications/persistence/3.1/jakarta-persistence-spec-3.1.pdf` [Accessed: September 2023].

[25] Kurur, N. P. *Mastering Java Persistence API (JPA): Realize Java's Capabilities Spanning RDBMS, ORM, JDBC, Caching, Locking, Transaction Management, and JPQL*. BPB Publications, 2022.

[26] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Ph.D. thesis, University of California, 2000.

[27] Subramanian, H. and Raj, P. *Hands-On RESTful API Design Patterns and Best Practices: Design, develop, and deploy highly adaptable, scalable, and secure RESTful web APIs*. Packt Publishing, 2019.

[28] Sharma, S. *Modern API Development with Spring and Spring Boot*. Packt Publishing, 2021.

[29] Tudose, C. *Java Persistence with Spring Data and Hibernate*. Manning, 2022.

[30] Konshin, K. *Next.js Quick Start Guide. Server-side rendering done right*. Apress, 2018.

[31] Clarke, A. *SEO 2023: Learn search engine optimization with smart internet marketing strategies*. 2023.

[32] Spilca, L. *Spring Security in Action*. Manning, 2020.

[33] Jones, M., Bradley, J., and Sakimura, N. JSON Web Token, 2015. `https://datatracker.ietf.org/doc/html/rfc7519` [Accessed: September 2023].

[34] Morling, G. Jakarta Bean Validation 3.0, 2020. `https://jakarta.ee/specifications/bean-validation/3.0/jakarta-bean-validation-spec-3.0.pdf` [Accessed: September 2023].

A Method for Player Character Navigation in Three-Dimensional Computer Games

Dominik Szajerman$^{1[0000-0002-4316-5310]}$, **Jakub Łapiński**^{2}, **Radosław Bednarski**$^{2[0000-0003-0468-6401]}$

[1]*Stefan Batory Academy of Applied Sciences*
Batorego 64C, 96-100 Skierniewice, Poland
dszajerman@ansb.pl
[2]*Lodz University of Technology*
Institute of Information Technology
Wólczańska 215, 90-924 Łódź, Poland
radoslaw.bednarski@p.lodz.pl

Abstract. *This work contains a complete solution for navigation a player's character in 3D computer game world. The processing pipeline consists of several stages – algorithms originating from various domains. The world-perception stage uses a method known from the field of robotics. It consists in acquiring and analysing the image from the camera recording the depth of the scene. The proposed point cloud processing stage allows for classification of points and interpretation of the scene content. The next proposed stage serves to create a two-dimensional world map. The next step is an adapted pathfinding algorithm. The last stage – after finding the path – allows the agent to navigate along it.*

Additionally, a set of parameters for the components of the method algorithms has been proposed. These parameters affect both the way the method works and the processing times, which allows the designers to carefully select them for their own needs. An analysis was carried out showing the parameters selection.

To show how the method works, four scenarios for selecting parameters were presented and three game levels were designed to test the algorithms.
Keywords: *computer game testing agent navigation point clouds*

1. Introduction

Navigating a player's character in computer games is an unusual issue, because usually the person playing the game is responsible for it. It may be reasonable to navigate the character automatically, using an algorithm created for this purpose. This can be used, for example, in the process of automatic testing of computer games. When conducting tests, the testing program must be able to navigate the player's character to reach a designated location in an unknown area of the virtual game world. It is also important that the world is perceived by the algorithm in a way similar to the player's perception, i.e. by analysing the image from its perspective in order to obtain similar navigation possibilities [1]. This would allow to detect errors in the game that may result, for example, from poor level design. Computer games do not normally use algorithms that perceive the environment on the basis of image analysis. Much more research on this type of algorithms has been done in the field of robotics. They are used there to locate the robot in the surrounding environment and to recognize obstacles. Many of these solutions use three-dimensional point clouds that define the positions of the perceived elements of the environment.

This paper proposes a method that combines algorithms from various fields: robotics, point clouds, navigation and computer games to obtain a comprehensive operation that allows to mimic the player's way of perceiving the world and finding himself in various navigation tasks.

2. Related work

The complex issue of navigating a player's character in computer games can be broken down into the following smaller issues:

1. The agent's perception of the world around him and his location in the world.
2. Processing and storage of collected data in order to remember the visited area.
3. Determining the path to a specific target, taking into account the data on the observed obstacles.

The issue of perceiving the world by computer algorithms is related primarily to the field of robotics (simultaneous localization and mapping – SLAM). The most basic way to collect data is with a simple single camera. The first single-camera SLAM approaches focused on reconstructing the environment in the form of 3D models. These were the so-called Structure from Motion algorithms [2]. The MonoSLAM [3] algorithm solves the problem of non-operation in real-time that limits the previous solution. This was achieved by using fewer features extracted from each frame of the video. But these features are tracked more closely and precisely. The algorithm [4], in turn, was created to navigate the robot in corridors. It is based on recognizing the lines connecting the walls with the floor. In [5], an algorithm was proposed that enables obstacles to be avoided based on the image from the camera mounted on the robot and seeing from above. The requirement is that the entire route is visible. Many solutions dedicated to robot control are also based on additional sensors that allow to determine the distance from other objects. Most of these solutions [6] use a LIDAR (Light Detection and Ranging) laser distance meter or 3D scanners. The data obtained from this type of sensors is a point cloud in three-dimensional space, where the coordinates of the points are given relative to the position of the sensor. As a sensor generating a 3D point cloud, it can also be used a camera recording the scene depth [7].

In computer games, information about the surrounding can be obtained from the image displayed on the screen [8]. With 3D games, it is possible to use methods similar to those used to reconstruct the environment with a single camera. However, it can also be used other data obtained directly from the game engine via Application Programming Interface (e.g. Vizdoom [9]). The existing works used, inter alia, the camera depth buffer [10].

In [11], an algorithm was proposed that allows for wall detection in the case of clouds generated with a 3D scanner. The scanner works in a vertical plane. The points collected are projected onto a horizontal plane and only the point farthest from the sensor is selected. A map of the most distant points in each direction is created, i.e. the most likely boundary walls. In [12], the acquired points are projected onto a plane parallel to the floor. The classification of points is based on the specified height. Based on the assumption that the walls are vertical, high-density point clusters are formed in their place. Another aspect related to the processing of data from the point cloud is point segmentation, that is, dividing points into objects such as trees, ground or walls [13]. It was also used in robot navigation described in [14]. There, the points are divided into those that belong to the same surfaces. All aforementioned methods are aimed at obtaining a two-dimensional map that would be suitable for use in subsequent tasks, such as navigation.

3. Method

The following functional requirements have been specified that must be met by the algorithm by which the agent is controlled. The agent should perceive the world in a manner similar to how the player perceives it. The algorithm is expected to create a two-dimensional level map as the agent moves through the game world. The algorithm has to find a path to a given point in the game world. It has to navigate to the target using the generated world map. The algorithm at the beginning of the operation does not have a level map, it must create it during the operation.

The data processing pipeline is shown in Figure 1. At the beginning, the game world is seen with the help of an additional camera that records the depth of the scene and normal vectors in pixel. Then each of the collected pixels is processed into a 3D cloud point. In the next step, the point cloud is processed and the two-dimensional level map is updated based on it. Finally, a path to the target is deter-

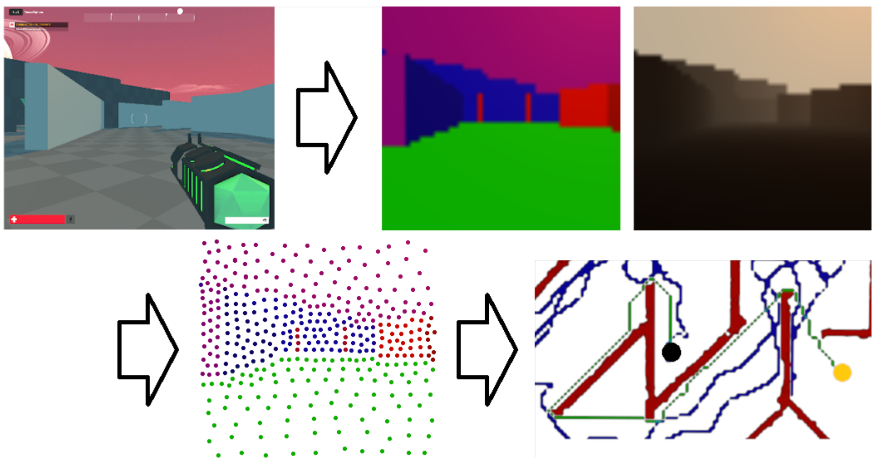


Figure 1: The pipeline: main camera image, colour-represented normal and depth vector recording from the main camera, 3D point cloud creation, obstacle map creation and navigation.

mined based on the map, which is then used to navigate the agent.

3.1. Perception of the world

The game world is perceived through image from the low-resolution camera. The camera is designed to record the depth of the scene and normal vectors determined in individual pixels. The acquired normal vectors are in the view (camera) coordinate system. Therefore, it is necessary to transform them by the matrix of transformation of camera coordinates to world coordinates. The obtained values – three normal vector coordinates and the depth are stored in the texture hereinafter referred to as "normal-depth texture" for the purpose of this method.

The *resolution* of this texture is a parameter of the algorithm. It has an exponential impact on the number of points obtained from each analysed image frame. In order to limit the amount of acquired data, they are not collected during each generated image frame. The parameter of the *interval between acquisitions* of the scene depth data has been introduced. It defines how often (e.g. every n frames) additional information is collected.

3.2. Point cloud processing

The next step is to process the information from each pixel of the *normal-depth texture* to a point described by world coordinates. Each pixel of this texture corresponds to a fragment of an object observed at a given point in the scene. The pixel depth is converted into the distance (z coordinate of the point) from the camera in game world units using near and far clipping plane distance. The remaining x and y coordinates of the point are determined on the basis of the pixel position in the texture and the texture resolution. The point calculated in this way is transformed to the world coordinate by inverted transformation matrix of the camera. All calculated points form a cloud.

The next step is the classification of points. The first step is to determine the height of the point in relation to the camera. Points above a certain height are classified as irrelevant and discarded, because obstacles above the agent, such as the ceilings, treetops, and lintels are irrelevant for navigation. Then, the slope of the terrain based on the normal vector at each point of the cloud is analysed. Points where the terrain is slightly sloping are considered passable, and those with a steeper slope are obstacles. It is enough to examine the angle between normal and the vertical vector. This value is compared with a certain threshold and on this basis the point is classified as either passable or an obstacle.

3.3. Generating a map of the terrain

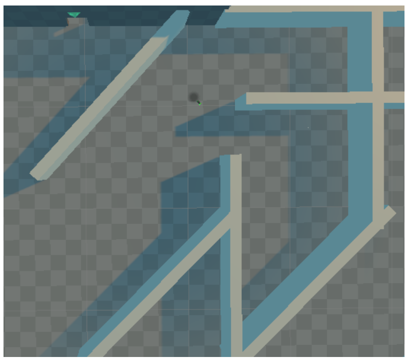
Based on the classified cloud points, a two-dimensional map is created showing the level of the game in a plan view. The first aspect of a map is the area it represents. The level map is a square with the side defined by the *map size* parameter. It has been implemented as a texture. An important parameter of it is the "map resolution". It is defined as the number of map pixels per one world unit. The *map resolution* is a parameter of the algorithm. The higher the *map resolution*, the more precisely the level is represented on it. Increasing the *map resolution* increases the computational complexity of the navigation process.

In the texture, obstacles are marked with the red component of the pixel color. Figure 2 shows a fragment of the level and the map generated on its basis.

3.4. Path navigation

The A^* [15] algorithm was used to determine the shortest path between the current position and the target. It is often found in agent navigation tasks in computer games. The algorithm creates a path to navigate through it.

The navigation algorithm consists of a part responsible for moving the agent to a target point in a straight line and a part responsible for determining the current point to which the agent is to follow. At the beginning of its operation, the algorithm takes the found path from the pathfinding component. The path update process is performed cyclically at predetermined time intervals, hereinafter called "decision intervals". It should be noted that each time a decision is made, a new path is calculated based on the updated obstacle map. After updating the path, the starting point of the path is set as the first target. During each frame it is checked if the point has been reached. Then, the next target is set, which is the next point on the path. The described procedure is repeated until the end of the path is reached.


(a) Top view of a level fragment.

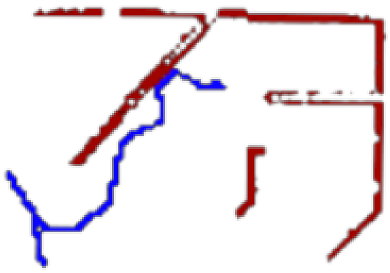

(b) A generated two-dimensional map of obstacles in a level fragment. Obstacles are marked in red and the path followed by the agent in blue.

Figure 2: Generating a map. Note that due to the path traveled by the agent, the camera did not see all the obstacles and therefore not all of them were included in the map at this stage of navigation.

The movement of an agent consists of translation and rotation. Translation takes place towards the target point at the speed determined on the basis of the "speed factor". Rotation is done in the direction of the predetermined speed so as to reduce the angle between the current and desired orientation.

4. Experiments

In order to test the method, a First Person Perspective (FPP) game [16] needed to be adapted. It provided basic mechanics such as first-person camera view, movement mechanics, and HUD (heads-up display). The following test values were used as defaults for the algorithm parameters described in the section 3:

- *normal-depth texture resolution*: 50×50 pixels,
- *interval between acquisitions*: 4 frames,
- *map resolution*: 2 pixels per world unit,
- *map size*: 100 world units,
- *decision interval*: 0.5 second,
- *speed factor*: 3×.

In order to measure the impact of the selected parameters of the algorithm on its course, the following data on the execution of the algorithms were collected:

1. Instantaneous (samples in each frame): pathfinding time (ms), distance left to target (world units), time elapsed since start (s), percentage of distance traveled.

2. Summary: *total time* (s) – it is the time measured from the moment on the starting point until reaching of the target; *path length* (world units) – it is measured from the starting point until the target is reached; *average agent speed* – quotient of path length and total time; *average time to calculate the path* (ms) – arithmetic mean of all pathifinding times.

For the purpose of the tests, three different levels were created.

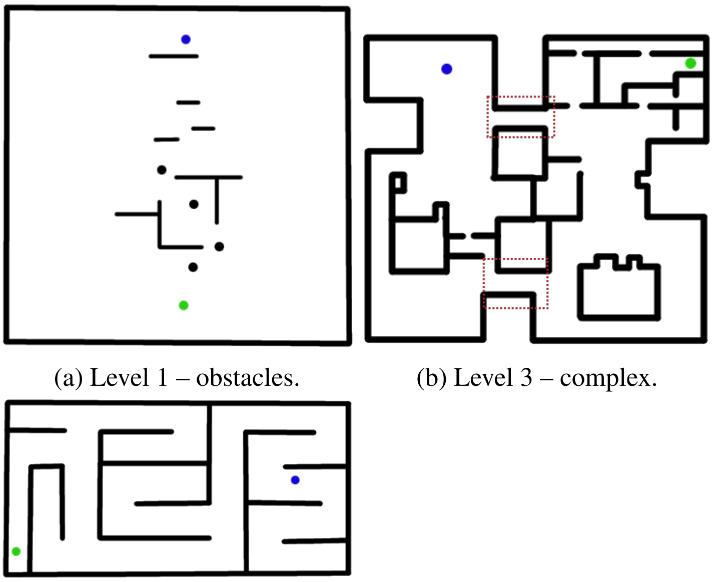
The first level was designed to test how the algorithm works against obstacles. Several obstacles have been placed along the way from the starting point to the target. These are obstacles such as walls or pillars. The exact locations of the obstacles are presented in Figure 3a (size: 100×100 world units).

The second level was designed to test how the algorithm deals with mazes. The map of the created level is shown in Figure 3c (100×50). The maze design was taken from [17] where it was used as an example to test various algorithms for pathfinding. The main difficulty in this level is that the algorithm must correctly remember which places are impossible to pass and which have not yet been explored.

The third level simulates the complexity of the level of the average computer game. The map of the created level is shown in Figure 3b (62×57). This is the most varied of the created levels. There are building-like obstacles on it. They are arranged in such a way that a labyrinth of narrow streets forms between them. Additionally, there are smaller obstacles simulating walls on the map. There is a maze in the upper right corner of the level map. It reflects the potential problems that can be encountered inside buildings. This is, for example, a large number of nooks and crannies and narrow passages. Another difficulty on the level is the difference in height between its two parts. The half of the level shown on the right side of the map is raised by 3 world units in relation to the left part. Height differences can be passed in two places using ramps. They are marked in Figure 3b with red dotted lines. One of them has low vertical clearance which may be perceived as an obstacle. Four experiments on the three game levels were conducted that test various aspects of the created algorithm:

1. The first experiment was to measure the impact of various obstacle map resolutions on the operation of the algorithm. For each level, 6 tests were performed, each with a different map resolution: 0.5, 1, 1.5, 2, 2.5, and 3.

2. The second experiment was to find out what is the optimal *speed factor* of the agent. For each level, 5 tests were performed, each with a different speed factor: 1, 3, 5, 8, and 13.


(a) Level 1 – obstacles.

(b) Level 3 – complex.

(c) Level 2 – maze.

Figure 3: Test levels. The starting points are the green dots, and the targets are the blue ones.

3. The third experiment was to check the impact of the change in the length of the *decision interval* on the operation of the algorithm. For each level, 4 tests were performed, each with a different interval between decisions: 0.25, 0.5, 1, and 2.

4. The fourth experiment measured the impact of *normal-depth texture resolution* on the algorithm performance. It was to check at what the lowest resolution the algorithm would still function properly at each level. Furthermore it was to check what is its impact on the decisions about choosing the fastest path. For each level, 3 tests were performed, each with a different resolution: 5×5, 10×10, and 50×50.

5. Results and discussion

In **the first experiment**, the impact of the *map resolution* on the operation of the algorithm was investigated. Table 1 on the left shows the results. In each of the test cases at level 1, the algorithm managed to reach the designated target. In level 2, with a map resolution of 0.5 pixels/unit, the algorithm failed to reach the target. In the remaining test cases, the algorithm managed to reach the set target. In level 3, with map resolutions of 0.5 and 1 pixel/unit, the algorithm failed to

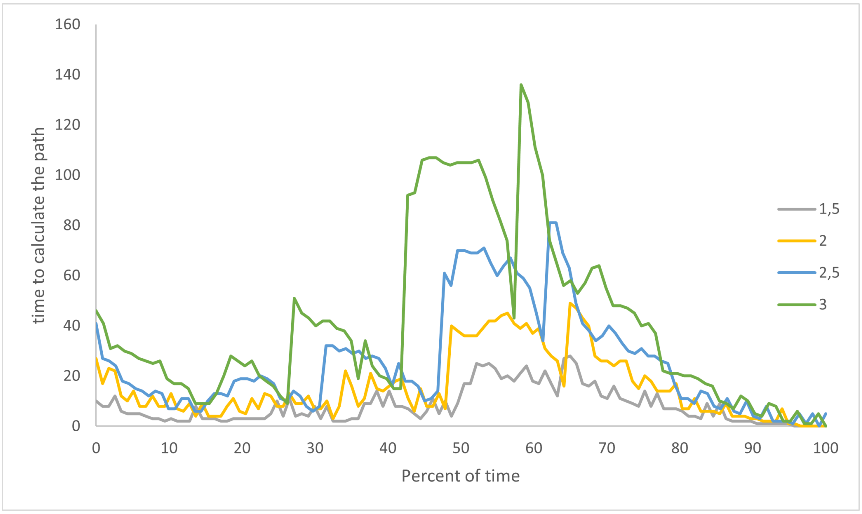


Figure 4: Experiment 1: Instantaneous time to calculate the path in level 3 for various map resolutions: 1.5, 2, 2.5, and 3 pixels/world unit.

reach the target. In the remaining test cases, the algorithm managed to reach the set target. In all cases, at all levels, a significant (exponential) increase in *average time to calculate the path* could be observed. When analysing changes in the *time to calculate the path* time during navigation (fig. 4), it can be noticed that as the end of the path approached, the path calculation time decreased. However, sometimes this time increased significantly. After analysing the algorithm's paths (fig. 5 right), it can be seen that the increases in calculation times occurred at times when the algorithm reached dead ends. When avoiding obstacles and passing the most diverse level 3, the *total times* always decreased with increasing map resolution. The exception was level 2 with a maze, where the *total time* achieved with the highest map resolution was the longest. This was due to such high *path calculation times* at this level at high resolutions that the *total time* was affected. At other levels, the *path calculation time* was not high enough to have a significant effect on the overall time to reach the target. The measured *path lengths* cases corresponded directly to the *total times*. On the other hand, the *average speeds of agent* in each case at a given level reached similar values. The collected data show that with complicated levels, the *map resolution* should be set to at least 1.5, but increasing its value to 2.5 would significantly improve the operation of the algorithm. However, this value should not be exceeded, as it would extend the required computation time, but would not significantly improve the path to compensate for the aforementioned longer computations.

In **the second experiment**, the impact of the *speed factor* on the operation of the algorithm was investigated. Table 1 on the right shows the results. In each of

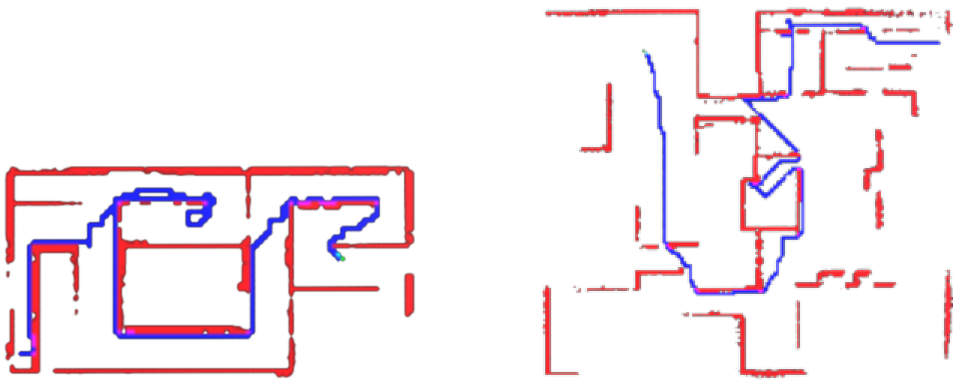


Figure 5: Experiment 1: Example path for level 2, map resolution 1.0 pixels/world unit (left) and level 3, map resolution 2.5 pixels/world unit (right). Note the different pixel thickness of the detected obstacles.

Table 1: Results for the experiment 1 (left) and 2 (right).

level	*map reso-lution*	*average time to calculate the path*	*total time*	*path length*	*average agent speed*
1	0.5	0.46	27.30	121.87	4.46
1	1	1.92	25.57	116.67	4.56
1	1.5	4.82	24.75	114.10	4.61
1	2	7.35	24.39	113.10	4.64
1	2.5	12.28	24.15	112.27	4.65
1	3	17.89	23.99	111.32	4.64
2	1	8.27	101.94	331.86	3.26
2	1.5	18.46	96.27	322.65	3.35
2	2	41.47	98.33	322.73	3.28
2	2.5	65.15	95.12	322.99	3.40
2	3	103.28	102.09	333.47	3.27
3	1.5	8.15	59.70	228.95	3.84
3	2	15.89	59.43	224.38	3.78
3	2.5	25.61	56.37	217.89	3.87
3	3	40.74	52.50	206.34	3.93

level	*speed factor*	*total time*	*path length*	*average agent speed*
1	1	73.57	114.44	1.56
1	3	25.38	115.97	4.57
1	5	15.96	116.50	7.30
1	8	11.15	116.64	10.46
1	13	8.66	118.24	13.65
2	1	258.43	301.87	1.17
2	3	104.39	345.06	3.31
2	5	44.53	236.66	5.31
2	8	32.01	239.74	7.49
2	13	34.41	257.43	7.48
3	1	169.17	225.40	1.33
3	3	58.42	224.27	3.84
3	5	41.09	240.16	5.84
3	8	32.45	249.41	7.69
3	13	40.48	308.83	7.63

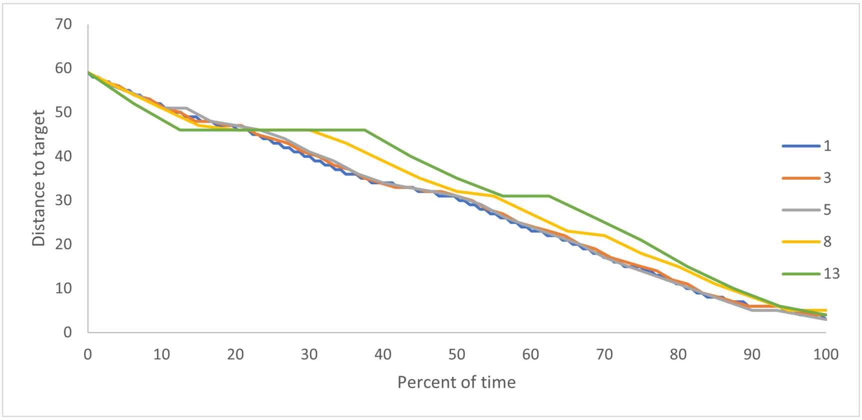


Figure 6: Experiment 2: Instantaneous distance to target in level 1 for various speed factors: 1, 3, 5, 8, and 13×.

the test cases at all levels, the algorithm managed to reach the designated target. In level 1, where the algorithm mainly dealt with avoiding obstacles, the obtained results were consistent with the expectations. As the *speed factor* increased, the *average speed of the agent* increased, and as a result, the *total time* decreased. With higher *speed factors*, the relationship becomes non-linear – the *total time* decrease is lower. Figure 6 shows the reason for this relationship. The chart for *speed factors* 8 and 13 are horizontal for a significant percentage of the time – i.e. the controlled agent did not approach the target for a significant part of the time. This was because it was then blocked by a wall, which can also be seen when analysing the traversed paths in Figure 7 (left). In level 2 the results are similar, but *path lenghts* became shorter. However, it should not be concluded that increasing the *speed factor* allows to find optimal routes faster. Rather, it should be noted that a change in this factor may result in the algorithm taking a different path to its target. In level 3, the relationship between the *speed factor* and the *average speed of the agent* was similar to that in the case of level 2. It should be noted, however, that at this level, the *total time* at the factor 13 was significantly increased compared to that at the factor 8. The algorithm in many places navigated the agent much further in "dead ends" than with lower speed ratios. The optimal *speed factor* (taking into account the *total time*) is 8. However, if the priority is to avoid collisions of the agent with the environment, the better choice would be to reduce it to 5.

In **the third experiment**, the impact of the *decision interval* on the operation of the algorithm was investigated. Table 2 on the left shows the results. In each of the test cases at all levels, the algorithm managed to reach the designated target. In levels 1 and 2, in each of the test cases the *total times* were similar. The fastest movement took place with the shortest *decisions intervals*, and the slowest with the longest. The path lengths were approximately the same in each case. On the

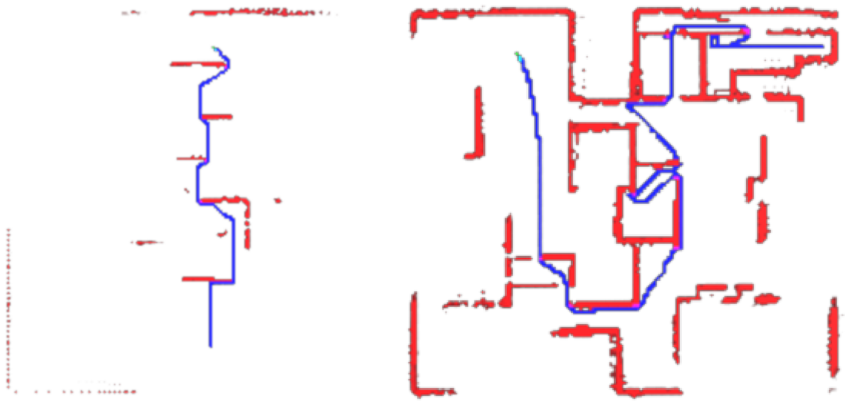


Figure 7: Experiment 2: Example path for level 1 and speed factor 13× (left), and the one for lever 3 and speed factor 1× (right).

other hand, the differences in the *total times* correspond to the difference in the *average speeds of the agents*. In level 3, there was a significant increase in the *total time* with the lengthening of the *decision interval*. The difference between the slowest and fastest pass was as high as 32%. The fastest one took place with an *interval* of 0.5 seconds. At this level, large differences in the *path lengths* were also observed. When the *decision interval* equal to 1s and 2s, there were moments in which the distance to the target increased slightly for a longer time compared to other passes (fig. 8), then it slightly decreased and this cycle repeated several times. At these moments, the path on the map (fig. 9 right) is not optimal and is arranged as if the algorithm did not find the path as quickly as in the case of shorter intervals between decisions.

In **the fourth experiment**, the impact of the *normal-depth texture resolution* on the operation of the algorithm was investigated. Table 2 on the right shows the results. In each of the test cases at all levels, the algorithm managed to reach the designated target. In Level 1, the results obtained for all test cases were similar. The differences between the *total times* did not exceed 4%, between the *path lengths* were up to 2.6%, and between the *average speeds* up to 1.7%. The best results have been obtained with the highest resolution. Analysing the paths in Figure 10, it can be seen that in the case of a resolution of 5×5, the path turned in front of the obstacles much closer than in the case of higher resolutions. This means that the algorithm later "noticed" obstacles in its path. The differences in the results between the test cases were, however, so small that it can be assumed that the 5×5 resolution is sufficient to avoid obstacles for the correct operation of the algorithm. However, the use of 10×10 resolution would allow to avoid obstacles from a greater distance. In level 2, the results obtained with the 5×5 and 10×10 *normal-depth texture resolutions* were significantly different from the

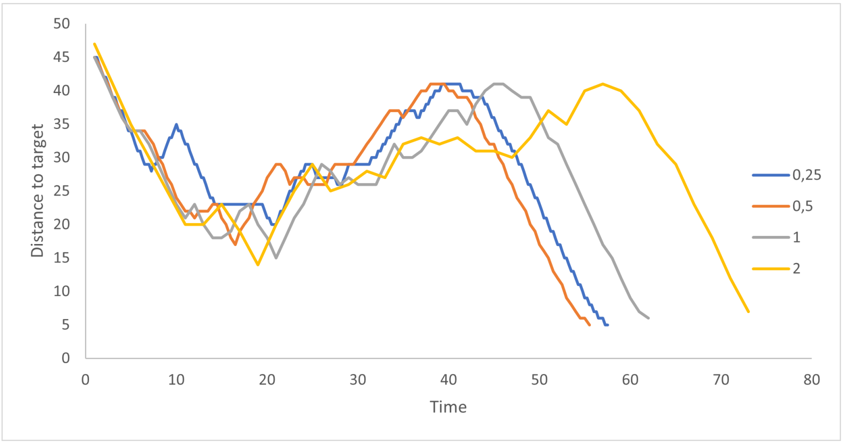


Figure 8: Experiment 3: Instantaneous distance to target in level 3 for various decision intervals: 0.25, 0.5, 1, and 2 seconds.

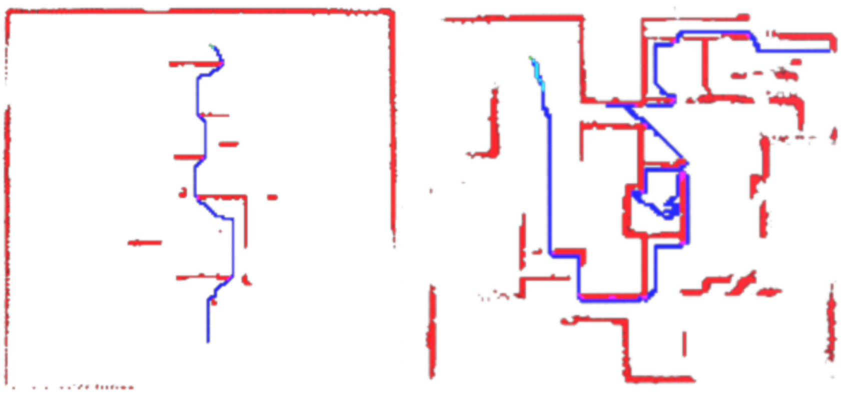
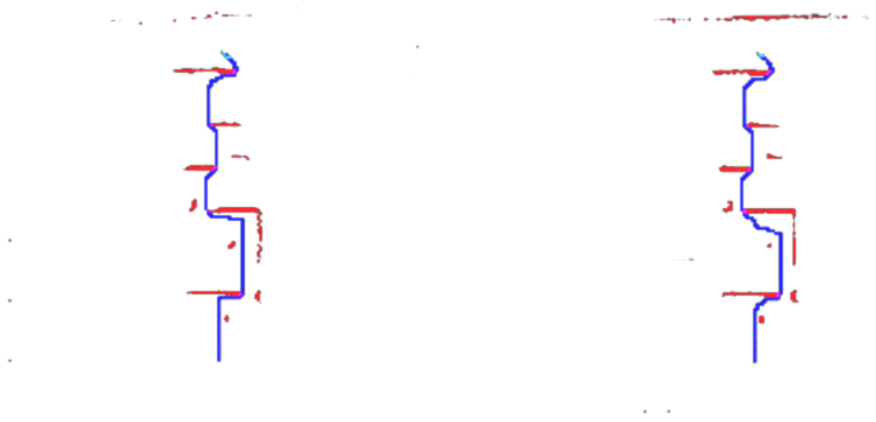


Figure 9: Experiment 3: Example path for level 1, decision interval 0.25s (left), and the one for level 3, decision interval 2s (right)


(a) 5×5. (b) 10×10.

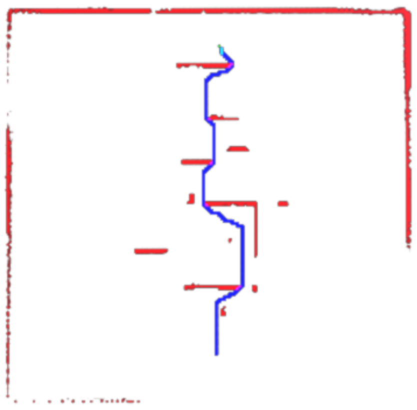

(c) 50×50.

Figure 10: Experiment 4: Map generated for level 1 and various *normal-depth texture resolutions* $n \times n$ pixels.

Table 2: Results for the experiment 3 (left) and 4 (right).

level	*decision interval*	*total time*	*path length*	*average agent speed*
1	0.25	24.89	115.24	4.63
1	0.5	25.35	115.87	4.57
1	1	25.14	113.63	4.52
1	2	26.12	114.34	4.38
2	0.25	100.35	341.51	3.40
2	0.5	103.46	343.44	3.32
2	1	109.66	339.01	3.09
2	2	111.07	344.40	3.10
3	0.25	57.68	224.97	3.90
3	0.5	56.23	217.56	3.87
3	1	63.22	234.32	3.71
3	2	76.08	254.21	3.34

level	*normal-depth texture resoultion*	*total time*	*path length*	*average agent speed*
1	5×5	25.56	116.36	4.55
1	10×10	24.78	114.23	4.61
1	50×50	24.48	113.34	4.63
2	5×5	64.01	227.78	3.56
2	10×10	65.97	230.01	3.49
2	50×50	103.82	340.82	3.28
3	5×5	114.08	383.54	3.36
3	10×10	67.50	252.45	3.74
3	50×50	58.97	222.46	3.77

results obtained with the 50×50 resolution. When analysing *path lenghts* it can be seen that it was caused by the choice of a different path to the target. In level 3, with similar results, additionally *total time* was definitely better for the highest resolution.

Level 3 tests in all experiments showed that a ramp with a low vertical clearance was mistaken for an obstacle. The algorithm never took the top-down path through it.

6. Conclusions

In this paper, a method was developed that allows the use of robotics algorithms and point clouds to navigate an agent representing a player's character in 3D computer games. The method of obtaining information about the environment has been adapted to the virtual world. A method of converting the acquired information into a point cloud was designed. Then, a method for generating a two-dimensional level map was developed. Finally, the algorithm for finding the path to the target was adapted and a way to navigate the agent along this path was designed.

A set of algorithm parameters has been proposed, which allow for the configuration of the operation of the whole method. Using *map resolution* affects obstacle handling and *time of calculating the path.* It should be matched to the complexity of the level and the density of obstacles. The *agent speed factor* parameter is used to adjust its precision of movement and degree of avoidance of obstacles. The *decision interval* in more complex levels, with a large number of small rooms and corridors, has a significant impact on the *total time* of achieving the target. The amount of data acquired is determined by the *resolution of the normal-depth texture*. For more extensive levels using a higher *normal-depth texture resolution* should be used.

Four various experiments were designed to check the operation of the algorithm. They also made it possible to determine the optimal aforementioned parameters of the algorithm. There are 3 levels designed for testing. They represented various environments that the algorithm can deal with. The maps generated on the basis of the visual surroundings well represented the tested levels and allowed for the correct calculation of paths to the target.

Future work could include an improvement towards the correct interpretation of ramps with low vertical clearance.

References

[1] Nowak, P. Virtual reality in investigation of human navigational skills. *Journal of Applied Computer Science*, 26(2):131–146, 2018.

[2] Fitzgibbon, A. W. and Zisserman, A. Automatic camera recovery for closed or open image sequences. In *Computer Vision — ECCV'98*, pages 311–326. Springer Berlin Heidelberg, 1998. doi:10.1007/bfb0055675.

[3] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007. doi:10.1109/tpami.2007.1049.

[4] Paolillo, A., Faragasso, A., Oriolo, G., and Vendittelli, M. Vision-based maze navigation for humanoid robots. *Autonomous Robots*, 41(2):293–309, 2016. doi:10.1007/s10514-015-9533-1.

[5] Abiyev, R. H., Arslan, M., Gunsel, I., and Cagman, A. Robot pathfinding using vision based obstacle detection. In *2017 3rd IEEE International Conference on Cybernetics (CYBCONF)*. IEEE, 2017. doi:10.1109/cybconf.2017.7985805.

[6] Shan, T. and Englot, B. LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018. doi:10.1109/iros.2018.8594299.

[7] Biswas, J. and Veloso, M. Depth camera based indoor mobile robot localization and navigation. In *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012. doi:10.1109/icra.2012.6224766.

[8] Kozłowski, K., Korytkowski, M., and Szajerman, D. Visual analysis of computer game output video stream for gameplay metrics. In *Lecture Notes in Computer Science*, pages 538–552. Springer International Publishing, 2020. doi:10.1007/978-3-030-50426-7_40.

[9] Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. Vizdoom: A doom-based ai research platform for visual reinforcement learning. 2016.

[10] Bhatti, S., Desmaison, A., Miksik, O., Nardelli, N., Siddharth, N., and Torr, P. H. S. Playing doom with slam-augmented deep reinforcement learning. 2016.

[11] Wulf, O., Arras, K., Christensen, H., and Wagner, B. 2d mapping of cluttered indoor environments by means of 3d perception. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. IEEE, 2004. doi:10.1109/robot.2004.1308934.

[12] Hirt, C., Zank, M., and Kunz, A. Real-time wall outline extraction for redirected walking. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*. ACM, 2017. doi:10.1145/3139131.3143416.

[13] Unnikrishnan, R. and Hebert, M. Robust extraction of multiple structures from non-uniformly sampled data. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*. IEEE, 2003. doi:10.1109/iros.2003.1248828.

[14] Wulf, O., Brenneke, C., and Wagner, B. Colored 2d maps for robot navigation with 3d sensor data. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. IEEE, 2004. doi:10.1109/iros.2004.1389864.

[15] Dechter, R. and Pearl, J. Generalized best-first search strategies and the optimality of a$*$. *Journal of the ACM*, 32(3):505–536, 1985. doi:10.1145/3828.3830.

[16] Unity. Fps microgame. URL `https://learn.unity.com/project/fps-template`.

[17] Terzimehic, T., Silajdzic, S., Vajnberger, V., Velagic, J., and Osmic, N. Path finding simulator for mobile robot navigation. In *2011 XXIII International Symposium on Information, Communication and Automation Technologies*. IEEE, 2011. doi:10.1109/icat.2011.6102086.

Enhancing Video Game Experience through Microexpression Recognition: A Deep Learning Approach

Julia Szymańska, Przemysław Zdrzalik, Witold Pietrzak, Mariusz Postół[0000–0002–9669–0565]

Lodz University of Technology
Institute of Information Technology
al. Politechniki 8, 93-590 Łódź, Poland
piotr.napieralski@p.lodz.pl

Abstract. *In the realm of interactive media, understanding subtle facial cues such as microexpressions can significantly enhance user experience by introducing more immersive and emotionally responsive environments. This article presents a comprehensive study on the detection and recognition of microexpressions from video streams using advanced machine learning techniques, focusing particularly on its application in video gaming.*
Keywords: *computer games, artificial intelligence, face recognition, neural networks, facial expressions*

1. Introduction

Interpersonal communication is a complex interplay of verbal and non-verbal cues, where non-verbal communication often conveys more information than the spoken words. Among various non-verbal cues, facial expressions are particularly telling as they can provide insights into a person's unspoken emotions and intentions. In recent years, the rapid advancement of video analysis technology and machine learning has enabled the development of systems capable of detecting subtle facial movements known as microexpressions.

Microexpressions are involuntary, brief, and subtle facial expressions that occur when a person either deliberately or unconsciously conceals a feeling. These expressions are typically of very short duration, lasting only about 1/25 to 1/15 of a second, but they are extremely telling, providing honest cues about a person's true emotional state [1]. Despite their brief appearance, the ability to detect and analyze these expressions accurately can be crucial in various applications, ranging from security and psychology to interactive entertainment and gaming.

In the context of video gaming, understanding and integrating the detection of microexpressions can significantly enhance player experience by enabling more natural and responsive interactions with non-player characters (NPCs). This integration can transform gameplay, making it more immersive and emotionally engaging. The emotional intelligence of NPCs could lead to more realistic and dynamically responsive gaming environments, where characters can react in real time to the player's subtle facial cues, adjusting the game's narrative or difficulty accordingly [2].

The technology for recognizing microexpressions in real-time, however, poses several challenges. The subtle nature of these expressions makes them difficult to capture with conventional facial recognition software, which typically identifies more pronounced emotional displays. To address these challenges, researchers have turned to sophisticated neural networks and deep learning techniques that have shown promise in capturing these minute expressions effectively [3, 4].

Our discussion will delve into the technological advancements that make real-time detection possible, the application of these technologies in video games, and the potential future developments that could further revolutionize the field. By harnessing the power of microexpression recognition, game developers can create richer, more engaging player experiences that leverage the subtle complexities of human emotion.

2. Detection and Recognition of Emotions Based on Microexpressions

Microexpressions are brief, involuntary facial expressions that reflect genuine emotions, despite an individual's attempt to mask their feelings. These expressions typically last only a fraction of a second and are universally considered crucial for revealing one's true state of mind. Their detection and accurate interpretation are invaluable across various domains such as psychological counseling, security, interpersonal communication, and interactive entertainment.

The automatic detection and analysis of microexpressions pose a significant challenge due to their subtlety and transient nature. Two primary approaches dominate the field:

- Machine Learning Models: Traditional algorithms, such as Support Vector Machines (SVM) and Random Forests, have been used to classify facial microexpressions by employing handcrafted features based on the Facial Action Coding System (FACS). These methods, while effective under controlled conditions, often require extensive manual annotation and struggle with variability in spontaneous expressions.

- Deep Learning Techniques: Recent advancements leverage Convolutional

Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to capture both spatial and temporal aspects of facial expressions. These models are trained on extensive datasets, allowing for more nuanced recognition capabilities that can adapt to the subtle dynamics of microexpressions.

Key challenges in this area include the scarcity of comprehensive and annotated datasets, the high similarity between different microexpressions, and the need for real-time processing capabilities. Techniques such as data augmentation, transfer learning from related facial recognition tasks, and the development of more efficient network architectures are being explored to overcome these hurdles.

The practical applications of effective microexpression recognition systems are vast:

- Security: Enhancing surveillance systems to detect signs of deceit or distress.

- Healthcare: Assisting clinicians in diagnosing mental health conditions by detecting inconsistencies between expressed emotions and reported feelings.

- Interactive Media: Improving user engagement in video games and virtual reality by adjusting content in real-time based on the player's emotional responses.

Looking forward, the integration of multimodal information, such as physiological signals and contextual data, promises to enrich emotion recognition systems. Furthermore, addressing ethical issues related to privacy and the consent of monitored individuals remains a crucial concern as these technologies advance and become more integrated into everyday life.

In summary, while the field of emotion detection from microexpressions is fraught with technical challenges, its potential to enrich human-machine interaction and enhance understanding of human emotional expression is immense.

3. Comparison of Existing Emotion Recognition Software

The development and deployment of emotion recognition technologies have been accelerated by advances in artificial intelligence, particularly through the application of deep learning. Several software solutions now exist that can analyze facial expressions and infer emotional states with varying degrees of accuracy and real-time performance. This section reviews and compares some of the prominent emotion recognition software platforms, focusing on their methodology, accuracy, and practical applications.

3.1. Microsoft Azure Emotion API

Microsoft Azure Emotion API leverages state-of-the-art machine learning algorithms to analyze facial expressions and identify a range of emotions. It provides a robust framework for detecting and classifying emotions such as anger, contempt, disgust, fear, happiness, sadness, surprise, and a neutral state. The API processes images and video streams to return an analysis of detected faces, including the spatial coordinates of facial landmarks and the intensity of expressed emotions.

The Azure Emotion API is designed for scalability and ease of integration, supporting various real-time applications. It offers detailed facial attributes analysis, including age, gender, pose, smile, facial hair, and makeup detection, which enhances its utility in diverse scenarios beyond emotion recognition, such as digital marketing and user interaction studies.

3.2. iMotions Software

iMotions is a comprehensive biometric research platform that integrates several sensors and software for advanced emotional and cognitive research. It combines facial expression analysis with eye tracking, EEG, GSR, and other physiological sensors to provide a holistic view of the user's emotional and cognitive state. iMotions software identifies seven basic emotions and provides metrics on intensity and certainty, making it particularly valuable for detailed scientific research and usability testing.

The software's ability to synchronize data from multiple sources makes it unique in scenarios where a deep understanding of human responses is required, such as in psychological studies, market research, and neurogame development.

3.3. Amazon Rekognition

Amazon Rekognition provides powerful image and video analysis capabilities and includes targeted features for facial analysis and emotion recognition. It can detect, analyze, and compare faces for a wide variety of user verification, people counting, and public safety use cases. In terms of emotion recognition, Rekognition identifies various facial expressions and emotions by analyzing the facial landmarks and applying machine learning models that predict the emotional state conveyed by a face in an image.

Like Microsoft Azure, Amazon Rekognition is designed to be highly scalable, processing millions of faces daily for various applications, including user verification systems and demographic information gathering.

3.4. Affectiva

Affectiva is an emotion measurement technology company that grew out of MIT's Media Lab and has pioneered emotion AI for emotion recognition. Their software analyzes facial expressions and emotions through advanced machine learning techniques developed over extensive consumer emotion data. Affectiva's technology is widely used in media testing, automotive safety, and mental health monitoring.

3.5. Comparison Summary

The table below summarizes the key features and typical use cases of the aforementioned emotion recognition software:

Software	**Key Features**	**Typical Use Cases**
Azure Emotion	Facial landmarks, emotion intensity, demographic analysis	Advertising, security, user interaction analysis
iMotions	Multi-sensor integration, emotion intensity	Scientific research, usability testing
Amazon Rekognition	Facial analysis, scalable image processing	User verification, demographic analysis
Affectiva	Extensive emotion data, real-time analysis	Media testing, automotive safety, health monitoring

Table 1. Comparison of Emotion Recognition Software

In conclusion, while all these platforms offer robust emotion recognition capabilities, the choice of software often depends on the specific requirements of the project, such as the need for real-time processing, integration with other biometric sensors, or the scalability of emotion analysis across different demographics and settings.

4. Developed Method for Emotion Detection Based on Microexpressions

Microexpressions are primarily defined by the position and pattern of facial muscle movements; for instance, a slight lifting of the corners of the mouth may indicate happiness. Therefore, the key to effectively recognizing microexpressions is to learn the positions and patterns of muscle movements rather than analyzing entire video frames. In this thesis, the MMNet method, proposed by Hanting Li in 2022 [5], is based on the aforementioned information from video frames and has achieved the highest effectiveness. Given the numerous advantages of the MMNet neural network articulated in the article, this specific method has been employed to create an algorithm that recognizes emotions based on facial microexpressions.

4.1. Problem Formulation

Existing methods for recognizing facial microexpressions primarily rely on deep learning, which improves the accuracy of classification. However, when approaches that use original video frames with sample identity information or manually extracted features, such as optical flow or LBP, are employed, there is a risk that the neural network will learn features related to identity, which do not significantly impact facial microexpressions. Therefore, the key aspects in recognizing microexpressions are the position and pattern of muscle movement on the face.

To address this issue and make more efficient use of the information from expression frames, we propose using a dual-branch MER (Micro-expression Recognition) paradigm. In this methodology, the main branch of the network is specifically designed to extract essential features regarding facial muscle movement patterns, which are crucial in recognizing microexpressions, while a sub-branch is used to extract facial position embeddings from the initial low-resolution frame. The initial frame is a key element containing primary information about the position of the facial muscles but does not include any expression information.

Adopting this dual-branch architecture allows avoiding the conflict of sample identity information with key muscle movement features. Thus, the main branch focuses on learning the microexpression movement pattern, independent of the identity information represented by the sub-branch. This approach allows for more effective and precise recognition of facial microexpressions, as the neural network focuses on the key features of microexpressions without interference from identity information.

4.2. Implementation of MMNet

The MMNet network, proposed by Hanting Li and colleagues in 2022 [5], is an advanced neural network developed for recognizing microexpressions (Motion-guided Network for Microexpression Recognition). It uses a dual-branch paradigm that allows extracting two key features of microexpressions: muscle movement patterns and the position of muscle movements on the face. The entire network consists of two branches: the main branch and the sub-branch. The main branch is tasked with extracting muscle movement pattern features by calculating the difference between the "onset" frame (initial) and the "apex" frame (peak). This approach allows the network to focus on the key aspects of muscle movement that are significant in recognizing microexpressions. On the other hand, the sub-branch deals solely with the position of muscle movements on the face, using only the "onset" frame for this purpose. This allows separating the position information from the movement features, minimizing the impact of sample identity information on the classification process. To further enhance the effectiveness of the MMNet network, a Continuous Attention (CA) block was used to learn muscle movement patterns on the face. Additionally, a Position Calibration (PC) module, based on the Vision Transformer (ViT), was used to add face position information and appropriately map the movement patterns to specific areas of the face. This allows the network to more precisely recognize microexpressions while reducing the impact of identity information on the samples. The main branch extracts features of the movement pattern, with the input parameter being the difference between the "onset" and "apex" frames. Meanwhile, the sub-branch generates face position embeddings from the initial low-resolution frame as input. Then, the position embeddings are added to the movement pattern features, allowing for more accurate mapping of the movement patterns to specific areas of the face.

5. Summary and Implications for Computer Games

The outcome of the research described above is an analysis of the newly implemented neural network named MMNet, which allows for the classification of emotions from microexpressions in video images. The network was trained using the CASME II database [6, 7, 8], along with a validation technique known as LOSO [9].

The accuracy of classification for both the test and training sets achieved very high results, with most networks exceeding 90%.

The ability to classify emotions from microexpressions makes the MMNet neural network a useful tool in emotion assessment programs in video images. Expanding such programs to include microexpression analysis can contribute to increased accuracy of emotion recognition and identification, which were previously difficult to detect. The implementation of this network and its use in recognizing emotions in video images may find applications in fields such as sociology, psychology, and business. Thanks to the MMNet network, precise recognition of emotions in viewers while watching advertisements or films is possible, which affects better content matching to the audience and increases effectiveness. On the other hand, research on microexpressions can significantly contribute to the diagnosis of emotional disorders.

Furthermore, the integration of MMNet into video games could revolutionize player engagement and narrative development. By accurately recognizing player emotions through microexpressions, game developers can tailor experiences in real-time, adjusting difficulty, storytelling, and interaction based on the player's emotional state. This could lead to more immersive and responsive gaming environments where players feel truly understood and dynamically engaged by the game mechanics.

Ultimately, the analyzed MMNet neural network represents a valuable tool for recognizing emotions based on microexpressions in video images, opening up many perspectives in the field of science and business, and particularly in the development of interactive computer games.

Acknowledgment

The authors would like to express their profound gratitude to Kuba Walczak for his invaluable assistance and insights throughout the research process. His expertise and guidance were instrumental in shaping the direction and execution of this study.

References

[1] Ekman, P. Lie catching and microexpressions. *The Philosophy of Deception*, pages 118–133, 2009.

[2] Gülkesen, K. H., Isleyen, F., Cinemre, B., Samur, M. K., Sen Kaya, S., and Zayim, N. A web-based game for teaching facial expressions to schizophrenic patients. *Appl Clin Inform*, 8(3):719–730, 2017. doi:10.4338/ACI-2016-10-RA-0172.

[3] li Tian, Y., Kanade, T., and Cohn, J. F. Recognizing action units for facial expression analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 23, pages 97–115. IEEE, 2001.

[4] Martinez, A. M. Automatic analysis of facial expressions: A survey. *Computer Vision and Image Understanding*, 127:1–16, 2017.

[5] Li, H., Sui, M., Zhu, Z., and Zhao, F. Mmnet: Muscle motion-guided network for micro-expression recognition. *CoRR*, abs/2201.05297, 2022. URL `https://arxiv.org/abs/2201.05297`.

[6] Yan, W.-J., Wu, Q., Li, Y.-J., Wang, S.-J., and Fu, X. Casme database: A dataset of spontaneous micro-expressions collected from neutralized faces. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pages 1–7. 2013.

[7] Quan, F., Wang, S., Yan, W.-J., and Fu, X. Cas(me)2 : A database for spontaneous macro-expression and micro-expression spotting and recognition. In *Interacción*. 2016.

[8] Yan, W.-J., Li, X., Wang, S.-J., Zhao, G., Li, Y.-J., Chen, Y.-H., and Fu, X. Casme ii: An improved spontaneous micro-expression database and the baseline evaluation. *PloS one*, 9(1), 2014.

[9] Kunjan, S., Grummett, T. S., and Pope, K. J. The necessity of leave one subject out (loso) cross validation for eeg disease diagnosis. In *Lecture Notes in Computer Science*, 12960. 2021.

Retopology of Three-Dimensional Models Using Hierarchical Clustering

Aleksandra Szymczak[0009–0001–3090–1710], **Damian Pęszor**[0000–0002–3754–2212]

Silesian University of Technology
Department of Graphics, Computer Vision and Digital Systems
Akademicka 16, 44-100 Gliwice, Poland
a.szymczak202@gmail.com
damian.peszor@polsl.pl

Abstract. *Retopology is an optimization process dedicated to simplifying mesh models that are based on triangles or quadrilaterals, thus improving the quality of simulations in which objects perform a movement process induced by skeletal animations or a physics system. The described project aims to develop an accessible tool that allows one to perform the model retopology process using hierarchical clustering methods. In addition, it was implemented as a package which extends the functionalities of the Unity 3D engine.*
Keywords: *computer games, artificial intelligence*

1. Introduction

When in 1950, work began on the first computers, there was a need to create solutions enabling the generation and processing of images and data visualization using digital systems. The first such attempts were made using printing devices and teletypewriters, creating primitive drawings. This initiated the computer science discipline of Computer Graphics, which was expected to be used in many fields. However, until the 1980s, due to its computational and memory requirements as well as its cost, it was practically used only by research or government centres. Its popularization in society was made possible only by the spread of personal computers [1]. In the era of computerization, computer graphics is identified not only with the visualization of data and information but also with a medium enabling accessible user interaction with technology. For this reason, it has become one of the most common Information Technology (IT) fields, accustoming society to digital visualization that has the features or appearance of the real world. A natural result of the popularization of digitally generated graphics was a sharp increase in user requirements for its implementation. A step in this direction was the implementation of Three-Dimensional Graphics technology, which allows generating and interacting with the digital perspective projection of objects. As technology further improves, both in terms of rendering devices, as well as algorithms used and methods of generating visualization, the quality improves as well.

Digital visualization of spatial objects is possible by using polygon mesh, which is a representation of the boundary planes of a three-dimensional model. These surfaces are described as a set of compounds of polygons characterized by coplanar vertices and collinear edges. Most often, meshes are built based on triangles or quadrilaterals, also called quads. However, polygonal mesh models are often transformed into triangular meshes, which significantly speeds up the image rendering process. This determines the position of the vertices forming a triangle in space, which, regardless of their position, will always be in one plane, which is not the case with quadrilaterals [2].

The vertices of polygons are one of the most important elements of the mesh because they not only store information about their position in 3D space, but also data about the color, normal vector and coordinates on the texture. Connected by edges, they form polygons whose mesh can mathematically be an undirected graph, with additional properties of geometry, shape, and topology [3]. Differences in the representation of meshes depend on the applications and features of objects, but the variety of available operations on them is common. The most popular of them are regularized Boolean operations, smoothing the surface by subdivision or simplifying their appearance based on the level of detail (LOD) [4].

Interactive 3D graphics rely on polygonal models due to their simplicity. The piecewise linear shape approximation implemented by polygon meshes is an ideal medium to perform a regular and simple rendering process by applying algorithms detrmining the visibility and color of pixels by interpolating the polygon surface. Due to the accessibility of rendering algorithms, accelerators have become widespread, allowing this process to be performed in any environment. Polygon meshes can represent almost all model shapes with some accuracy, so they were perfect as a common denominator [4].

The aim of this work is to design an algorithm that enables the retopology process of a three-dimensional model mesh. The idea behind this paper is to use hierarchical clustering as part of the multiresolution representation of the model. The designed algorithm was implemented and tested using Unity 3D engine. The rest of this paper is structured as follows. Section 2 introduces the problem while section 3 describes the methods used for the proposed solution. Section 4 presents the results obtained during the experiments performed. Finally, section 5 concludes the article with an additional discussion of its influence.

2. Problem statement

The key idea of the presented project is the design of a 3D mesh retopology system based the hierarchical structure obtained using hierarchical clustering, allowing the grouping of the vertices forming it based on their position data in space. The aim of retopology is to produce a new mesh with a reduced number of vertices and the triangles they form, while maintaining the special features of the model's appearance.

The concept of retopology originates from the field of 3D computer graphics. Retopology is an optimisation process dealing with the simplification of mesh models based on triangles or quadrilaterals [5]. The process makes it possible to optimize meshes generated either manually or automatically by tools for modeling three-dimensional objects. The retopology of models also improves the quality of simulations in which objects perform a motion process induced by skeletal animations or a physics system.

Retopology is a popular process, often part of model-making tools. Above all, the method enables the creation of objects with high resolution in an accessible way, streamlining the mesh optimisation process. Such an object has a reduced number of vertices forming it by maintaining a baseline level of detail. The operation enables the generation of a model that runs smoothly with animation, due to the reduced cost of rendering the object. Retopology is also useful for cleaning up 3D scans that are often incomplete or need to be optimised before use. Another use of this process is to generate mesh variants with different levels of detail for the LOD system. Creating 3D models using retopology has a significant impact on speeding up the optimisation process. At the same time, much better results are achieved than with manual mesh simplification. However, the end result of the retopology process is dependent on the knowledge of its methods.

A key aspect of the algorithm is a hierarchical clustering system. This process takes data on the position in space of the vertices that make up the mesh and then divides them into clusters. Each time, a complete hierarchy of vertices is created, it can be visualized through a dendogram. The clustering boundary parameter defined by the user is used to determine when the hierarchy level is cut off and to read the clusters thus created. Based on these, new vertices are created to form the three-dimensional grid of the model.

The results of hierarchical clustering are dependent on how the distance between different objects is calculated, i.e. the metric and linkage criterion used. These parameters have a significant impact on the mathematics behind the process of finding and defining links between vertices of the mesh. At the same time, they result in smaller or larger changes to the resulting hierarchy. Different parameters define the aggressiveness of the retopology level, which is closely related to the appearance of the final generated model mesh, whether defined by a distance threshold or the number of clusters.

3. Materials and methods

The architecture of the package was based on two solutions implemented using the C# language and Python. The C# was used to perform operations on the model mesh, for the user interface, and to supervise communication between applications. Python language was used to perform hierarchical clustering using specialized programming libraries. The designed algorithm allows for parameterization by selecting a percentage value that describes the level of simplification of the mesh. The hierarchical clustering system depends on the following parameters:

- **Distance threshold** - the maximum distance between clusters, this in fact determines the number of clusters, if it is not defined explicitly.
- **Desired number of clusters** - Defines the number of clusters. If this number is reached, no further mergin of clusters is performed.
- **Distance metric** - the user's choice of the metric to be used when calculating the distance between the data to create their similarity table.
- **Linkage type** - A linkage criterion that will determine how the clusters will be represented in the distance calculation.

3.1. Hierarchical clustering

The key factor in the proposed method is finding the part of the mesh that can be represented as a single vertex. Hierarchical clustering is used to find such connections between vertices of the model based on a linkage matrix between each possible pair of vertices representing a given distance selected in the *Linkage* parameter, similarly to [6]. The process of clustering the data into clusters itself is performed using the agglomerative clustering approach.

During the implementation of the hierarchical clustering process, different clustering methods were tested. The approaches differed not only in terms of the algorithm, but also in terms of the implementation itself. In the end, it was decided to base hierarchical clustering on a constraint-based method. This approach enabled a retopology system for 3D models that simplifies the mesh without significant loss of detail even for high levels.

The critical component of the proposed method is a suitable hierarchical clustering method. The Bayesian Hierarchical Clustering (BHC) approach seemed to be the most promising among the analyzed ones [8]. For most agglomerative clustering methods, the data representations are built as binary trees, where the leaves correspond to the data points and the internal nodes to the clusters formed. In contrast, an extension of the BHC concept has trees, called rose trees in the literature, having an arbitrary number of branches at each internal node. This development is the Bayesian Rose Trees (BRT) method [7]. Searching such structures is much

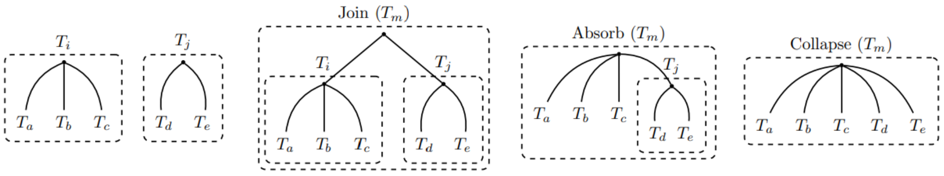


Figure 1. Rearranging the action of the rose tree sub-trees altogether [7]

more difficult algorithmically, so three basic methods of merging their subtrees are proposed in [7]: *join* which creates a new parent node, *absorb* which is merging without creating a new node, and *collapse* which removes a node. The discussed methods of rose-tree subtree merging operations are shown in Figure 1. The BRT clustering method implemented in the project was based on Guilherme Caponetto's solution [9], adapting the algorithm to the objectives of the work. Unfortunately, the clustering process was high complexity due to the computational power needed to create the input data linkage matrix. The results of mesh simplification based on the BRT results were unsatisfactory, as the retopologised grids formed deformed or incomplete grids. However, this was probably influenced by the type of data being clustered assumed in the paper. In theory, clustering triangles as individual planes rather than as vertices would have been a more promising approach for this method. In the end, the clustering method using binary trees was rejected in favor of a constraint-based method.

3.2. Simplification of the model mesh

The proposed algorithm to reduce the number of vertices and, at the same time, the triangles they form. At its core, simplification is based on removing a group of vertices and in their place creating a new one with an averaged position.The problem proved to be complex due to the nature of the triangles' storage of information about the vertices they form. These are stored in an array in a given order, according to the right-hand rule. This allows information to be obtained about the direction of the normal vector of the plane, which always falls on it at right angles. It determines the side of the plane to be encapsulated, i.e. to generate the colour. Disturbing this order would be associated with the destruction of the model structure. The designed algorithm solves this problem through two assumptions:

1. Triangles in which at least two vertices are grouped into one cluster are removed.
2. In the case of triangles in which only one vertex is removed, it is replaced by a new vertex that maintains the existing order.

The simplification operation is performed by reconstructing the cluster hierarchy

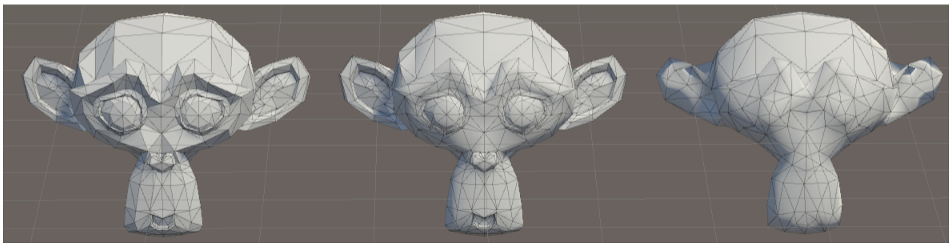


Figure 2. Demonstration of system operation on a monkey model

according to the data passed through the clustering process. A list of the topmost clusters in the hierarchy is passed to the algorithm for simplification. The aim of this operation is to transform the mesh so that it is constructed only from the vertices whose indices have been passed as a result of the clustering process.

4. Results

The main problem associated with the test of the clustering algorithm is the verification of the hierarchical structure created. For this purpose, the dendrograms were used to visualize the hierarchies created.

4.1. Experimental results

Fig. 2 shows an example of how the implemented retopology system works for the 3D model of Suzanne the monkey. The object on the left is the original model built from almost two thousand vertices and just under a thousand triangles. The subsequent example models are retopologised meshes, simplified, in turn, by two and ten times the number of vertices. For the central object, the number of vertices was around a thousand, and the number of triangles was unchanged. For the mesh, whose level of detail was reduced by a factor of ten, its number of triangles was reduced by a factor of two, and the number of vertices building it was reduced by almost nine times. Regardless of the level chosen, the model did not lose its shape resembling its base version, only the amount of its detail was reduced in line with the work.

Only at extreme levels of simplification does the model have a perturbed topology. This is illustrated in Fig. 3, where the rabbit model has been simplified sequentially so that it has only one thousand and one hundred vertices. The original object was made up of two and a half thousand vertices and almost five thousand triangles. Their number was reduced to less than two hundred with only one hundred vertices. With such a large simplification, the resulting model lost its properties as its mesh became deformed and incomplete. However, such an effect is understandable with such a high level of simplification.

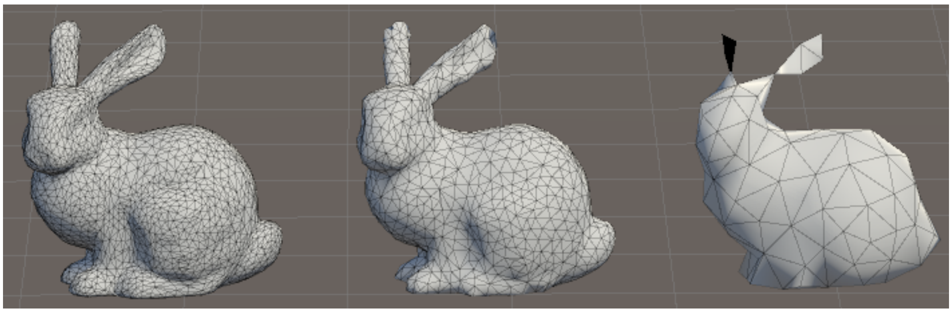


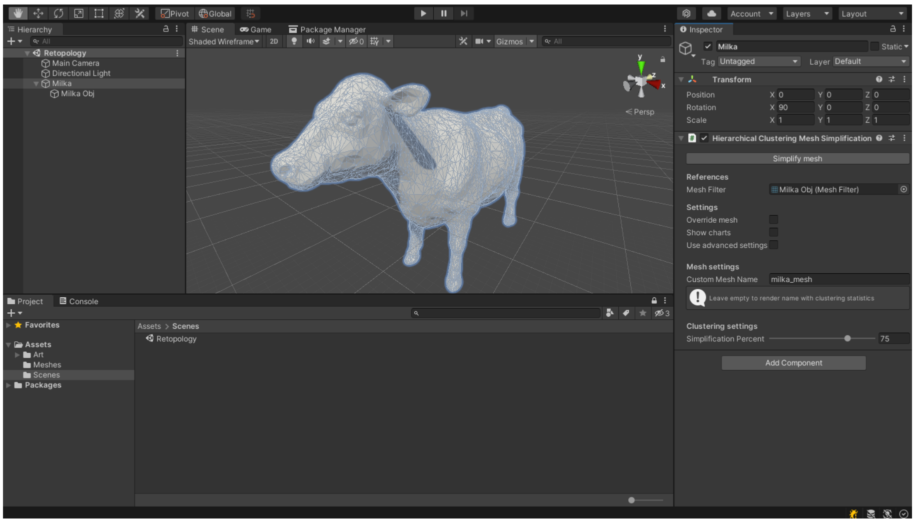
Figure 3. Demonstration of system operation on a rabbit model

4.1.1. Complexity of the retopology process

In addition, a study was conducted on the speed of the retopology process for models of varying complexity. In the course of these, it was concluded that the main aspect affecting the implemented system is the input data relation matrix creation function. Due to the rather high computational complexity class of the system, a safeguard was added to return information to the console regarding the amount of Random Access Memory (RAM) needed in the case of a model composed of too many vertices. Table 1 illustrates the study of the complexity of the retopology process based on the recursively simplified mesh of the cow model, shown in Fig. 4. The original mesh was built with 57,870 vertices and 26,234 triangles. The concept of the study was based on documenting the running time of the system in question. To this end, the model was recursively retopologised to produce an initial new mesh at a simplification level of 90%. The set percentage meant that, with each iteration, a mesh with the number of vertices that was 90% of the retopologised model was obtained. The level of simplification of a given iteration relative to the original model was estimated by the number of vertices obtained relative to the initial number. In this way, it was possible to study not only the simplification process itself, but also the clustering, as the number of data to be clustered changed in each iteration.

In Table 1 presented here, some records have been omitted for clarity; however, Fig. 5, which is a graph of the dependence of the retopology process time against the number of vertices in the simplified model, includes the results of each iteration of the study.

The data obtained allowed observation of the system operation time with respect to the number of vertices undergoing the clustering process, which resembles two juxtaposed linear functions. This relationship is illustrated using Fig. 5.



Milka — Static
Tag: Untagged — Layer: Default

Transform
Position: X 0, Y 0, Z 0
Rotation: X 90, Y 0, Z 0
Scale: X 1, Y 1, Z 1

Hierarchical Clustering Mesh Simplification
Simplify mesh

References
Mesh Filter: Milka Obj (Mesh Filter)

Settings
Override mesh: ☐
Show charts: ☐
Use advanced settings: ☐

Mesh settings
Custom Mesh Name: milka_mesh
Leave empty to render name with clustering statistics

Clustering settings
Simplification Percent: 75

Add Component

Figure 4. Preparation of the cow model for simplification

Table 1. Table comparing programme times for the selected cow model

Number of vertices relative to the original model	Number of grid elements for a given iteration		Duration of the retopology process [s]
	Vertices	Triangles	
100%	57 870	26 234	-
90%	52 085	26 234	1 325
81%	46 874	26 234	800
73%	42 186	26 234	350
66%	37 967	26 234	148
59%	34 170	26 234	108
48%	27 697	26 234	67
39%	22 418	26 234	49
28%	16 342	26 234	30
23%	13 236	26 234	25
21%	11 912	23 842	23
19%	10 720	21 456	19
15%	8 683	17 378	13
10%	5 695	11 416	6
4%	2 448	4 940	2

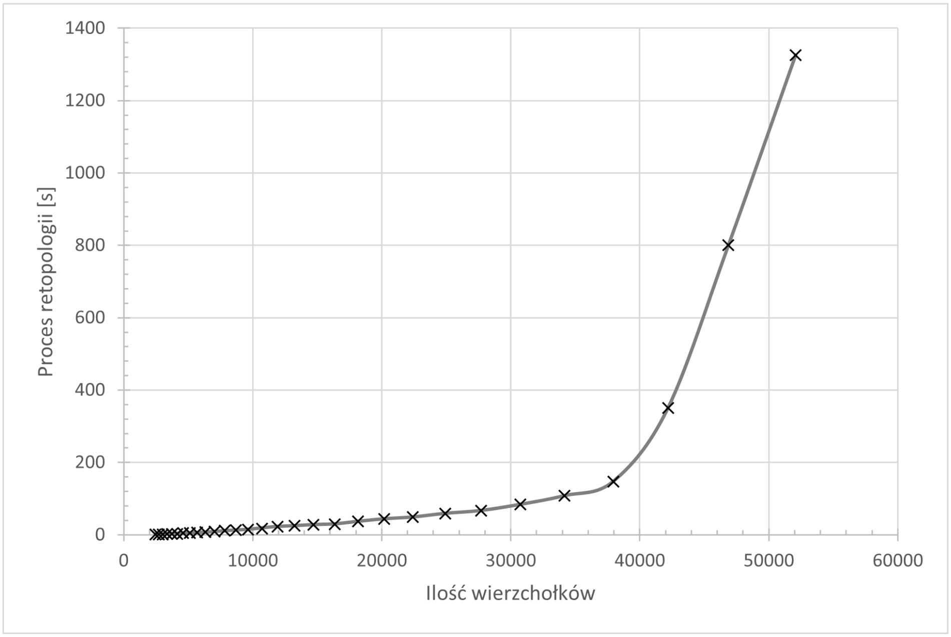


Figure 5. Plot of the duration of the retopology process against the number of vertices of the model being simplified

5. Conclusions

The main objective of this study was to design and implement a retopologisation system for a three-dimensional model using hierarchical clustering. The realized package, which is an extension to the Unity 3D engine, has several potential avenues for further development. The most interesting of these is the further development of the existing hierarchical clustering system with a view to reducing the complexity associated with the creation of link matrices. This direction would allow the project to advance algorithmically and analytically by enabling retopologies of more complex and intricate models. For example, for a given vertex, the matrix would collate the links to vertices in its immediate vicinity. Another possible development option is to rebuild the specialized hierarchical clustering method in terms of a retopology of three-dimensional meshes. In theory, this procedure could have a decisive impact on achieving the best possible vertex clustering algorithm. A final option is to extend the package with a LOD clustering system. This would significantly facilitate the work of graphic designers and 3D modellers and, at the same time, the usability of the package.

Acknowledgment

We would like to express our deepest gratitude to a few individuals who played a pivotal role in the research journey of the first author.

We cannot forget to mention the partner of the first author, who became her personal *rubber duck* - always ready to listen to complaints about algorithmic problems and inexplicable errors in the solution. Your patience and support have been immeasurable.

Lastly, we wish to express profound gratitude to the parents of the first author, who have consistently supported growth and education over the years. Thanks to them, the first author learned to strive for excellence and continuous development, always striving for more. Your unwavering faith in the abilities of the first author has been a motivation and guiding light.

References

[1] Foley, J., van Dam, A., Feiner, S., Hughes, J., and Phillips, R. *Introduction to Computer Graphics*. Addison-Wesley Publishing Company, Massachusetts, USA, 1996. ISBN 0-201-60921-5.

[2] Vranic, D. *3D model retrieval*. Ph.D. thesis, University of Leipzig, 2004.

[3] Agoston, M. *Computer Graphics and Geometric Modelling - Implementation & Algorithms*. Springer-Verlag, London, Great Britan, 2005. ISBN 978-1-84628-108-2.

[4] Luebke, D., Reddy, M., Cohen, J., Varshney, A., Watson, B., and Huebner, R. *Level of Detail for 3D Graphics*. Morgan Kaufmann Publishers, San Francisco, USA, 2003. ISBN 1-55860-838-9.

[5] Rossoni, M., Gonizzi Barsanti, S., Colombo, G., and Guidi, G. Retopology and simplification of reality-based models for finite element analysis. *Computer-Aided Design and Applications*, 17:525–546, 2019.

[6] Low, K.-L. and Tan, T.-S. Model simplification using vertex-clustering. pages 75–82, 188. 1997.

[7] Blundell, C., Teh, Y., and Heller, K. Bayesian rose trees. *arXiv preprint*, 1203.3468, 2012.

[8] Heller, K. and Ghahramani, Z. Bayesian hierarchical clustering. In *Proceedings of the 22nd international conference on Machine learning*, pages 297–304. 2005.

[9] Caponetto, G. Bayesian hierarchical clustering in python, 2021. URL `https://github.com/caponetto/bayesian-hierarchical-clustering.git`.

Deep Learning for Saliency Map Generation

Przemysław Zdrzalik, Julia Szymańska,
Witold Pietrzak, Jędrzej Mońko

Lodz University of Technology
Institute of Information Technology
al. Politechniki 8, 93-590 Łódź, Poland
piotr.napieralski@p.lodz.pl

Abstract. *This paper explores advanced techniques in generating saliency maps using deep learning. Saliency maps are instrumental in understanding visual attention mechanisms in both humans and models, enabling enhanced interpretations of image regions impacting neural network decisions in tasks like object detection and classification.*
Keywords: *computer games, deep learning, saliency maps, computer vision*

1. Introduction

The rapid advancements in image processing technologies have magnified the importance of understanding human visual perception. Saliency maps serve as an effective tool for analyzing fundamental visual elements that our brains prioritize and represent. Particularly in the realm of video games, saliency maps have emerged as a crucial technology for enhancing user engagement and gameplay dynamics.

In video games, understanding where a player looks and what attracts their attention can significantly impact game design and user experience. Saliency maps are used to predict these visual attention points, thereby informing designers on how to craft more immersive and visually engaging gaming environments. The emergence of virtual reality (VR) technologies, especially head-mounted displays (HMDs), presents new challenges and opportunities for understanding and employing visual attention mechanisms in digital environments. Saliency maps, which highlight regions in visual scenes that attract human attention, are increasingly vital in optimizing user experience and interface design in immersive VR settings. These maps have profound implications in various applications, notably in video games, where understanding and directing player attention can significantly enhance gameplay and interactivity.

Celikcan et al. (2020) explore the application of visual saliency prediction in real-time rendered VR environments experienced through consumer HMDs [1].

Their research highlights how dynamic, computer-generated scenes influence visual attention and demonstrates that adjustments in navigation speed can affect the visual saliency of elements within these immersive environments. The study reveals that depth cues become less significant as navigation speeds decrease, whereas 2D image features gain prominence. This nuanced understanding of saliency in VR can guide developers in creating more engaging and visually intuitive VR experiences.

Additionally, the work of Koulieris et al. (2014) introduces an automated high-level saliency predictor that significantly benefits game balancing and scene design [2]. By incorporating cognitive and perceptual hypotheses into their model, they provide a method to adjust game levels automatically based on object saliency. This approach not only aids in creating challenging and engaging game environments but also ensures that important game elements remain within the player's focal attention, thereby enhancing the overall gaming experience.

These studies underscore the importance of saliency maps in understanding and engineering user interactions within complex, visually rich environments. By integrating insights from these research efforts, this paper explores advanced deep learning techniques to generate and utilize saliency maps, aiming to improve the design and functionality of interactive systems, particularly in VR and gaming applications.

This paper explores advanced deep learning techniques that enhance the generation of saliency maps and discusses their potential applications in the gaming industry, among others.

2. Background

2.1. Deep Learning in Computer Vision

Deep learning continues to revolutionize the field of computer vision by enabling the analysis of complex patterns from extensive datasets. Convolutional Neural Networks (CNNs) remain central to modern image analysis techniques, influencing a broad array of applications from autonomous driving to interactive gaming environments. In the gaming sector, these advancements facilitate more than just enhanced graphics; they enable real-time scene rendering and the creation of interactive environments that dynamically respond to player actions.

In the context of video games, CNNs and other deep learning models play a pivotal role not only in enhancing graphical fidelity but also in optimizing gameplay mechanics through saliency mapping and behavior prediction. For example, Bako et al. (2021) describe the application of deep learning for variance reduction in Monte Carlo rendering, a method crucial for producing photorealistic images in games and VR applications. Their research demonstrates how machine learning can be integrated at different stages of the rendering pipeline to enhance efficiency and reduce computational costs [3].

Furthermore, the integration of machine learning into game development tools has transformed how game environments are created and interacted with. Lanham (2018) discusses the use of Unity Machine Learning agents, which allow developers to incorporate complex ML algorithms like deep reinforcement learning to train intelligent behaviors within games. Such technologies enable games to adapt to players' strategies, improving game balance and player engagement [4].

These innovative applications highlight the synergy between deep learning and computer vision in gaming, where AI-driven tools not only enhance visual realism but also contribute to the interactive and adaptive nature of modern video games. This intersection of deep learning and gaming exemplifies how visual data can be used to enrich user interfaces and create immersive, responsive gaming experiences.

2.2. The Concept of Saliency Maps

Saliency maps are computational tools used to visualize the parts of images that are most engaging to a model, highlighting areas that significantly influence computational assessments. These maps identify features in an image that attract the most attention from a neural network, such as distinctive colors, intense contrasts, or unique textures, making them particularly useful in various applications from image segmentation to enhancing user interface design.

One intriguing application of saliency maps is in the analysis of visual attention across different media, including movies. Rogalska and Napieralski (2018) explore the use of visual attention saliency maps specifically for movie retrospection. Their research focuses on how viewers' attention is distributed across different elements of a film, such as characters, objects, and settings [5]. By analyzing these saliency maps, filmmakers and researchers can understand which aspects of a scene draw more viewer attention and are thus more likely to influence the audience's emotional and cognitive responses.

The study by Rogalska and Napieralski demonstrates that saliency maps can be a powerful tool for content creators to assess and optimize the visual impact of their work. For instance, in movie production, directors and editors might use saliency maps to decide where to place key visual elements or how to time scene changes to align with viewer attention peaks. This methodology not only enhances the storytelling effectiveness but also ensures that important narrative elements are visually emphasized, improving the overall viewer engagement.

Furthermore, the implications of such studies extend beyond film analysis. They contribute to the broader field of computational neuroscience and cognitive psychology by providing insights into the mechanisms of human attention. Understanding these patterns helps in designing better interfaces, advertisements, and educational materials that are aligned with natural viewing tendencies and preferences.

In summary, saliency maps serve as a bridge between raw visual data and human perceptual insights, offering a quantifiable measure of visual importance that is invaluable for both academic research and practical applications in media and user-centered design.

3. Methodology

3.1. Generating Saliency Maps

To generate saliency maps, the model processes the input image through several convolutional layers, pooling layers, and sometimes fully connected layers at the end. The final output is a saliency map that highlights the regions of the original image that are likely to attract the most attention from a human viewer.

The process can be represented by the following equation:

$$S = \sum_{i=1}^{N} w_i \cdot \mathrm{ReLU}(C_i) \tag{1}$$

In Equation 1, S represents the saliency map, N is the number of channels in the last convolutional layer, w_i are weights that linearly combine the activation maps, C_i are the activation maps from the last convolutional layer, and ReLU is applied to ensure that only positive features contribute to the final saliency map.

The generated saliency maps can be used in various applications, including enhancing image processing systems, improving object detection models, and even in areas like marketing to analyze visual content effectiveness. Additionally, the training of such models often involves optimization techniques that minimize the difference between the predicted saliency map and ground truth maps, typically annotated by human observers.

$$L = \frac{1}{M} \sum_{j=1}^{M} (S_j - Y_j)^2 \tag{2}$$

Equation 2 describes a mean squared error loss function where L is the loss, M is the number of training examples, S_j is the saliency map generated by the model for the j-th example, and Y_j is the corresponding ground truth saliency map.

3.2. Dataset and Training

The MSRA10K dataset [6] is one of the most widely used benchmarks for saliency detection research. It consists of 10,000 images, each annotated with pixel-level saliency labels. These images encompass a wide variety of scenes and objects, making it a comprehensive dataset that challenges the robustness and generalization ability of saliency detection models. The high diversity in the dataset includes natural scenes, animals, indoor, and urban landscapes, which are crucial for training models to perform well across different visual contexts [7, 8]. Before training, images are typically preprocessed to fit the input requirements of the neural network. This involves resizing images to a uniform dimension and normalizing the pixel values. The saliency maps are also resized accordingly to match the output dimensions expected by the network.

```
X_resized = resize(X_original, (width, height))
X_normalized = X_resized / 255.0
```

This pseudocode demonstrates the basic preprocessing steps: resizing the original image $X_{original}$ to a specified width and height, and normalizing the pixel values by dividing by 255 to scale them between 0 and 1. The training of the model involves feeding the preprocessed images into a convolutional neural network that predicts the saliency map for each image. The network architecture typically includes multiple layers of convolutions, activation functions, pooling layers, and may also include upsampling layers if the architecture is designed to output full-resolution saliency maps [9].

The objective during training is to minimize a loss function that measures the difference between the predicted saliency maps and the ground-truth annotations. A common choice for the loss function in saliency map generation is the binary cross-entropy loss, given the binary nature of the saliency annotations (salient or not salient).

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{3}$$

Where N is the number of pixels in each image, y_i is the ground truth saliency value of the pixel, and $\hat{y}_i$ is the predicted saliency value of the pixel. This loss function is effective for training saliency detection models as it directly penalizes the model for incorrectly predicting the saliency of each pixel.

Throughout training, the model's performance is validated using a separate subset of images from the MSRA10K dataset not seen during the training phase. Common metrics used to evaluate the performance of saliency models include Precision, Recall, F-Measure, and Mean Absolute Error (MAE). These metrics provide insights into how well the model can generalize to new data and its effectiveness in identifying salient regions accurately.

In summary, the use of the MSRA10K dataset provides a robust platform for developing and benchmarking saliency detection models. Through meticulous preprocessing, systematic training, and rigorous validation, models can be optimized to perform effectively across diverse visual content.

4. Results

4.1. Evaluation of the Algorithm

The effectiveness of the trained model was assessed using a suite of quality metrics including Precision, Sensitivity (Recall), F1-Score, True Positive Rate (TPR), and False Positive Rate (FPR). These metrics were calculated based on the comparison of thresholded saliency maps generated by the model against ground truth maps. Thresholding involved limiting the saliency values to a certain range before making comparisons.

The results of this evaluation were presented in detailed performance tables and visually through comparative illustrations of the saliency maps at various stages of model training.

4.2. Performance Metrics

The evaluation results are summarized in several tables, showing the performance metrics for the network trained initially on the MSRA10K dataset and later stages using combined datasets. These metrics provide insights into the model's performance and its development through successive training phases.

Table 1. Quality metrics for the network trained on the MSRA10K dataset

Metric	**Mean Value**	**Minimum Value**	**Maximum Value**
Precision	0.89	0.00	1.00
Recall	0.94	0.27	1.00
F1-Score	0.91	0.04	0.99
TPR	0.94	0.27	1.00
FPR	0.02	0.00	0.26

Table 2. Quality metrics for the network after subsequent dataset integrations

Metric	**Mean Value**	**Minimum Value**	**Maximum Value**
Precision	0.89	0.00	1.00
Recall	0.90	0.20	1.00
F1-Score	0.88	0.07	1.00
TPR	0.90	0.20	1.00
FPR	0.02	0.00	0.27

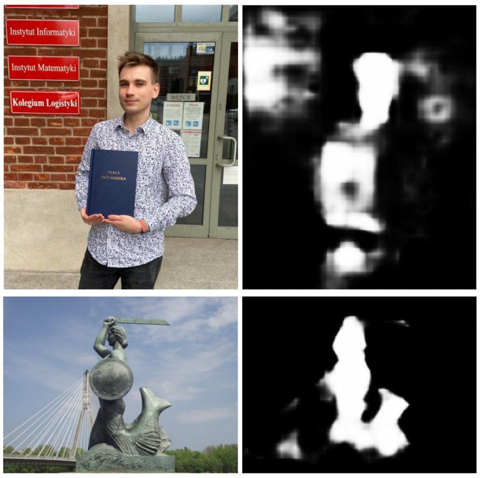


Figure 1. In order from left to right, original image, significance map significance map generated by the developed method

4.3. Visual Comparisons

The visual comparisons of saliency maps generated by the network at different stages of training were illustrated. Each illustration shows the original image, the benchmark saliency map, and the saliency map generated by the model. These visual comparisons provide a clear representation of the model's ability to approximate human visual attention across various learning phases.

The analysis of these results suggests that each of the fully convolutional networks was capable of generating saliency maps of satisfactory quality. Despite the incremental additions of datasets for training, there was no substantial decline in the quality of the generated saliency maps. All models maintained performance within expected ranges across all evaluated metrics, confirming the robustness and reliability of the trained models.

5. Conclusions

The research conducted has resulted in three models capable of effectively generating saliency maps for a broad range of images. The method presented for generating saliency maps using deep learning models has achieved promising results. This work was based on advanced neural network techniques, particularly using the FCN-8 model based on the VGG-16 architecture. The primary goal was to generate saliency maps that indicate which areas of an image attract human attention.

During the research, an analysis was conducted on various datasets representing scenes with varying degrees of complexity. The results demonstrate that the models are capable of generating saliency maps of satisfactory quality in terms of the adopted metrics. The created models are characterized by a low False Positive Rate (FPR), indicating that their significant pixel assignments can be treated with a high degree of certainty. High True Positive Rate (TPR) values suggest that most significant pixels were correctly detected.

No significant differences were observed in the capabilities of the three models after different phases of training when evaluating the quality of the maps they generated on the test set. The models have achieved promising results in generating saliency maps on diverse images. However, there are many opportunities for further development. Efforts should be made to create larger and more diverse training and testing datasets. Evolving the model architecture, considering a version based on VGG-19, and combining the method with other techniques could yield even better results. It would also be useful to create a test set containing more complex images to confirm or refute the conclusion that the results of each of the three models are very similar. These conclusions underline the importance of proper result analysis and further development of methods for generating saliency maps in the context of applications in user interface design and object feature analysis in computer vision.

5.1. Potential Applications in Video Game Design

The use of saliency maps can significantly enhance video game design, providing developers with tools to analyze and predict player attention and interaction. By integrating saliency map generation into game development, designers can create more engaging and immersive environments that guide the player's focus to specific elements or areas of the game scene. This technique can be particularly useful in dynamic game content, where adjusting the level of detail and complexity based on the player's focus can improve performance and player satisfaction.

Experimental applications in gaming could include adaptive difficulty adjustment, where the game's challenge level changes in response to the player's attention to critical game elements. Additionally, saliency maps can help in the layout and design of game interfaces, ensuring that important notifications or game stats are placed within areas that naturally attract the player's gaze. Future experiments could explore how different game genres, such as strategy games versus fast-paced shooters, may benefit differently from tailored saliency-based design interventions. These insights emphasize the potential of saliency maps not only as a research tool but also as a practical component in the next generation of game development.

Acknowledgment

The authors would like to express their profound gratitude to Kuba Walczak for his invaluable assistance and insights throughout the research process. His expertise and guidance were instrumental in shaping the direction and execution of this study.

References

[1] Celikcan, U., Askin, M. B., Albayrak, D., and Capin, T. K. Deep into visual saliency for immersive VR environments rendered in real-time. *Computers & Graphics*, 88:70–82, 2020. ISSN 0097-8493. doi:10.1016/j.cag.2020.03.006. URL `https://www.sciencedirect.com/science/article/pii/S0097849320300388`.

[2] Koulieris, G. A., Drettakis, G., Cunningham, D., and Mania, K. An automated high-level saliency predictor for smart game balancing. *ACM Transactions on Applied Perception*, 11(4):17:1–17:21, 2014. ISSN 1544-3558. doi:10.1145/2637479. URL `https://doi.org/10.1145/2637479`.

[3] Bako, S., Turk, M., Wang, W., Hollerer, T., and Meyer, M. *Deep Learning for Variance Reduction in Monte Carlo Rendering*. University of California, Santa Barbara, 2021. ISBN 9798460406913.

[4] Lanham, M. *Learn Unity ML-Agents Fundamentals of Unity Machine Learning: Incorporate new powerful ML algorithms such as Deep Reinforcement Learning for games*. Packt Publishing, 2018. ISBN 1789138132.

[5] Rogalska, A. and Napieralski, P. The visual attention saliency map for movie retrospection. *Open Physics*, 16(1):188–192, 2018. doi:doi:10.1515/phys-2018-0027. URL `https://doi.org/10.1515/phys-2018-0027`.

[6] Cheng, M.-M. Msra10k salient object database. `https://mmcheng.net/msra10k/`, 2020. URL `https://mmcheng.net/msra10k/`. Accessed: 2024-05-05.

[7] Hou, Q., Cheng, M.-M., Hu, X., Borji, A., Tu, Z., and Torr, P. Deeply supervised salient object detection with short connections. *IEEE TPAMI*, 41(4):815–828, 2019. doi:10.1109/TPAMI.2018.2815688.

[8] Hou, Q., Cheng, M.-M., Hu, X., Borji, A., Tu, Z., and Torr, P. Deeply supervised salient object detection with short connections. In *IEEE CVPR*, pages 3203–3212. 2017.

[9] Borji, A., Cheng, M.-M., Jiang, H., and Li, J. Salient object detection: A benchmark. *IEEE TIP*, 24(12):5706–5722, 2015. doi:10.1109/TIP.2015.2487833.