IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/45SE/2025/4534925/
Vol. 2025 (22), Article ID 4534925

Principles of Creating Attractive Arcade Games with Original Examples*

Emil MĄKA and Joanna WIŚNIEWSKA

Institute of Information Systems, Faculty of Cybernetics,
Military University of Technology, Gen. S. Kaliskiego 2, 00-908 Warsaw, Poland

Correspondence should be addressed to: Emil MĄKA, maka.emil.98@gmail.com

* Presented at the 45th IBIMA International Conference, 25-26 June 2025, Cordoba, Spain

Abstract

In this work, we present some best practices in the development of arcade games. Creating computer games may be a profitable business. Additionally, it can be a job that is done with passion and commitment. We find that it is worth discussing about some recommended practices in this field. We believe that it is good to treat game development as a project in which every detail must be considered – such as the target audience, the game's objectives, rules, and graphic style. People who create computer games should also be aware of the basic animation principles to strive for perfection of their products and make the game aesthetic and eye-pleasing. In this paper, we also introduce the rules for creating game storylines and characterize some useful tool for programists to cope with working on such a large and multi-faceted project as creating your own computer game. Finally, we illustrate presented theory with an example of created arcade game.

Keywords: computer games, animation theory, graphical user interface, Unity

Introduction

Designing and implementing arcade-style video games is both a creative and technical challenge. These games, known for their fast-paced gameplay, simple mechanics, and engaging loops, demand a unique approach to game development. Unlike narrative-driven or strategy games, arcade games focus on immediate player feedback, tight controls, and increasing levels of difficulty that keep players coming back for more.

The process begins with a strong core gameplay concept, often centered around a single mechanic such as jumping, dodging, or shooting. This mechanic must be polished to perfection, as it forms the foundation of the player's experience. Visual and audio design also play a crucial role, enhancing responsiveness and providing essential cues to the player.

This article explores the key stages of arcade game development, from concept to implementation. It outlines design principles, technical considerations, and best practices used in creating compelling, replayable experiences. Whether you're an aspiring indie developer or a hobbyist, understanding these fundamentals is essential to crafting successful arcade games.

Fundamental Principles of Animation

Animation is the illusion of movement achieved by rapidly changing images that are slightly different from each other. Starting with the definition, the etymology of the term animation comes from two words: "anima" (soul) and "animo" (I animate). We can interpret these words as "to animate", "to bring to life". Walt Disney's studio had a huge influence on the development of animation theory. The work "Disney animation: The Illusion of Life" by Thomas et al (1981) contains 12 basic principles of animation, considered by some in later years to be the

Cite this Article as: Emil MĄKA and Joanna WIŚNIEWSKA, Vol. 2025 (22) "Principles of Creating Attractive Arcade Games with Original Examples " Communications of International Proceedings, Vol. 2025 (22), Article ID 4534925, https://doi.org/10.5171/2025.4534925

"animation bible" or "the twelve commandments of the animator". Considering the importance of the above-mentioned principles and the full reliance on them during the creation of the project, we consider it important to list them and briefly characterize them:

- Squashing and stretching – a key principle of giving a character/object energy, weight, and speed by stretching and squashing some animation frames to excess.
- Anticipation) – key frames of animation in which the animated character prepares and somehow suggests to the observer what will happen next.
- Staging – achieved through the appropriate frame, scenography, play of light or colors, points that we should follow when watching the animation. It is thanks to the staging that we get the essence of the frame that attracts the viewer's attention.
- There are two different approaches to drawing animations: straight ahead action (involves drawing frames one after another, animating the whole thing smoothly from the starting point; in this approach the most desired effect is movement, so it is best suited for animating dynamic scenes) or pose to pose (first, we create two key frames for the character's movement, and then fill in the frames between them to get from one key point to another).
- Follow through and overlapping action – a technique that helps to render movement more realistically and that gives the feeling that the laws of physics are acting on the animated character.
- Timing – determines where frames should be placed on the timeline so that the animation is consistent, fluid and, above all, in accordance with the laws of physics.
- Slow in, slow out – this principle is closely related to timing. The movement of objects in reality, such as the human body, animals, vehicles, etc., needs time to accelerate or decelerate. Therefore, at the beginning or end of the action, more animation frames are drawn to create the effect of acceleration and deceleration while maintaining synchronization and a constant number of FPS.
- Arcs – using the arc principle in animation, we achieve the effect in which the animated object moves in a natural way. Therefore, it is necessary for each subsequent frame of the animation to be drawn along a previously determined parabola.
- Secondary actions – adding secondary action to the main action allows the scene to be brought to life and supports the main action. The important thing is that the secondary actions do not distract the viewer, but only emphasize the main action.
- Exaggeration – by using exaggeration you can highlight a character's traits in a fun, unrealistic way. This is a great way for an animator to increase the appeal of a character and reinforce the narrative.
- Solid drawing – the principle of good drawing means that the animator, when designing and sketching an object, should preserve its three-dimensionality. The artist should understand and preserve anatomical proportions, physical interactions, the behavior of light and shadows, etc.
- Appeal – although the name may be misleading, the charm of a character does not mean that he is attractive. The character should draw the viewer to him. Such effects can be achieved in various ways, for example, for good characters it is a good habit to design them with a symmetrical, child-like head.

Computer Games

A computer game is a program, together with the data attached to it (music, graphics, sound), the main purpose of which is to provide the user with entertainment by requiring the player to solve logical or skill tasks. These tasks vary depending on the genre and may consist, for example, of eliminating virtual opponents or competing with artificial intelligence or other players (multiplayer game).

“Game” by Britannica (2025) is generally defined as „a universal form of recreation generally including any activity engaged in for diversion or amusement and often establishing a situation that involves a contest or rivalry. Card games are the games most commonly played by adults. Children’s games include a wide variety of amusements and pastimes primarily for children”.

Based on the above-mentioned definition, we can distinguish three main components of the game: players (who want to play the game), rules (define the limitations of the game), goals (arouse competition and the desire to win).

The main aspects of the game are:

- Challenge – the rules define the game, and consequently, its goal. When players decide to play the game, they accept and follow the rules that govern it. It is the rules, or obstacles placed on the way to the goal of the game, that motivate and encourage the player to continue playing.

- Conflict – an opponent who may be a fictional character or a process taking place in the game, makes it difficult for the player to achieve their goals.
- Gameplay – the rules of the game are presented and communicated to the player in a specific, precise and clearly defined manner.

Another definition of games, by Rollings et al (2003), emphasizes that a game requires a goal, because it is essential to every game. It gives meaning to the game and engages the player to continue. The goal of the game is defined by the rules. It is conventional, because the game designers can define it in any way they want. It must also be nontrivial, because the game should contain some elements of challenge. Even in a purely random game, such as craps or roulette, players must learn to know their chances and place bets in such a way as to obtain the greatest possible benefit. Additionally, the rules define the meaning of actions and events encountered by the user during the game. They are consistent with the goal of the game, so they allow the player to deduce what course of action will be most beneficial to achieving it. Examples of rules are: gameplay, order of play, goal (or goals) of the game, end condition, meta-rules.

Pretending is an inherent element in many game theory accounts. Pretending, or the "magic circle" by Huizinga (1938), is defined as the creation of an imaginary reality in the mind that separates the actions that are important in the game from those that are important in the real world. This is the division between the world invented by the developers and reality. The imaginary reality created in the mind of the participant allows for a more vivid sense of the sensations and emotions flowing from the game. The player, on the other hand, by staying inside the magic circle agrees to define a temporary, artificial meaning for the situations and events occurring in the game.

Gaming is a form of entertainment in which the participant can interactively make various decisions and gain different experiences. Gaming is distinguished by the freedom of actions taken and the ability to choose how these actions can be performed. By requiring constant interaction between the player and the computer or other electronic device included in the hardware platforms, the participant has a direct influence on the course of events. However, the decisions made by the player are limited by the rules, which is why the game requires the use of cunning, logical thinking, intelligence or manual dexterity.

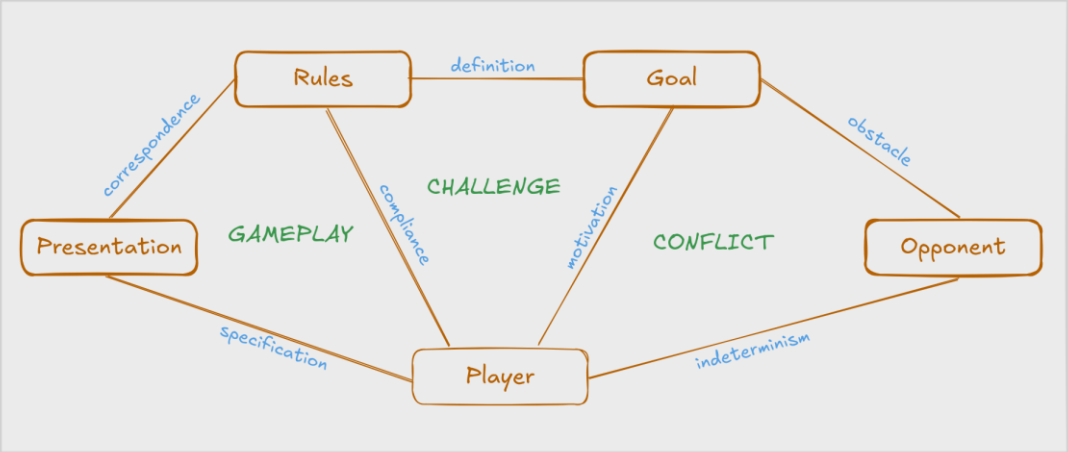

Fig. 1. Components, dependencies and aspects of the game

Arcade games are a subgenre of a computer game in which the successful completion of the tasks requires the player to have physical skills and hand-eye coordination. There is no room for strategic thinking in games of this type. However, the speed of reaction and the use of appropriate techniques available in the game are important in order to most effectively overcome an obstacle or defeat an opponent. Often, such techniques are based on performing appropriate sequences of movements on the controller called combo attacks. These are improved attack or defense abilities that the player can use, giving a much greater advantage in the game compared to basic attacks.

In most arcade games, the player controls a single avatar as the protagonist. The avatar has the ability to move and interact with other objects in the game world. During the game, the player has the opportunity to develop the character by finding power-ups or through a permanent character development system, e.g. skill tree, statistics development.

The described game genre has seen its implementations using two- and three-dimensional graphics. Over the years, productions have been created presenting the game world from different perspectives, but in the canon of 2D games, the side view has been preserved.

Computer Game as a Project

Nowadays, games play an incredibly important role for us. Although not everyone is aware of it, games develop our imagination, spatial vision, hand-eye coordination and reflexes. The examples given here are just some of the many advantages that playing on a computer screen brings. With the growing popularity of e-learning, entities responsible for creating such courses are increasingly reaching for teaching through play, i.e. games. This proves that the demand for games and their derivatives is very high. It is therefore extremely important for potential game producers to break through the market with their product.

It is impossible to build a durable, solid and beautiful building without a qualified architect and, consequently, a detailed construction project. The above example illustrates how important the first stage of the game design process is for the game – the conceptual stage.

The basic issue at this stage of game production is creating a design document. This step, often skipped by amateurs, is very important for the stability of the entire project. In fact, everything we find in the game should be well analyzed and designed in advance. When creating such a document, we should first focus on describing our own vision of the game, i.e. who it is addressed to, the technology of the project and a description of the world and game mechanics. By breaking down each idea into its fundamental components, we can develop and describe every element of the game in detail. Thanks to such a document, we are able to evaluate the game initially.

The next step is to write a game script. It focuses on interactions with other characters in the world, the plot, and placing the main character in a catchy and engaging time and location for the recipient.

All that remains is to plan and design the use of the objects, buildings and items found in the game. This will significantly speed up production during the next stage of creation – implementation.

Once the game's genre and its initial concept – the idea describing how we plan to deliver entertainment through gameplay – have been defined, it's necessary to consider who will enjoy the product and to whom we are dedicating it. It often happens that video game producers first identify their audience – the "target market" – and only then think about a game concept that will sell easily.

In a game that reflects reality, the player must pretend. They pretend to believe in the game world, their personification, and the situations that arise. The player plays a certain role in such games. However, it is up to the designer to define it. The player can be a soldier on the battlefield, a scientist, a dancer, or an athlete. The key is to describe the player's role in detail and introduce them to the game world. Such actions enhance the experience and enjoyment of the game.

Abstract video games often have arbitrary rules, so the player rarely has a pre-determined opinion about what they will be able to do in the game. On the other hand, games that reflect reality take place in a world that the player is somewhat familiar with. They are approached with certain ambitions and expectations. Games that reflect reality are about fulfilling dreams of achievement, conquest, power, and creation.

Once you have a rough idea for your game, it's worth stopping here and thinking about how to fulfill the player's dreams and what elements will generate the most excitement. It's also worth thinking about the challenges and decisions that will arise during the game.

When a few weeks of design and bug fixing are over, it is recommended to move on to prototyping. It is best to create small, specific prototypes of specific mechanic applications - a prototype of controls, a prototype of shooting mechanics, a prototype of missions, dialogues, etc. With such skeletons, we have a broader picture of what should be removed, what needs improvement and what can remain unchanged. It is important to stick to the main concept of the game and not introduce major changes at this stage of production, because it will disrupt the entire project.

Prototyping allows us to identify which features work and which don't, enabling us to progress confidently to the next milestones. So we implement subsequent elements of the game in accordance with the design documentation. After implementing each element we are obliged to conduct tests in order to eliminate errors or correct them if necessary. If the process was successful, we return to adding more fragments of the game. In fact, we repeat this action until we exhaust the assumptions that we described in the documentation.

During this stage of creation, a series of tests checking the consistency of the entire software separate us from the final version. It is necessary to examine how the elements work together and eliminate any graphical errors. For larger productions and those that use large amounts of computer resources, it is worth paying attention to improving performance.

Useful Tools

The arcade game, presented in this work, was implemented in the C# programming language. In this section, we describe tools what were useful during game development.

Unity (see: Unity (2025)) is an advanced editor designed for creating two-dimensional and three-dimensional computer games, with its own graphics engine, sound engine and physics simulator. It allows you to create applications on web browsers, personal computers, video game consoles and mobile devices. It also offers many tools for editing and creating animations, which is why it is chosen by many developers. The editor provides many additional functions that can be purchased in the Asset Store (see: Asset Store (2025)) – a built-in online store offering the use of paid or free components such as textures or scripts. The engine allows you to write your own scripts in C#.

Universal Render Pipeline (see: URP (2025)) is the name of an extension in Unity that allows for greater control over rendering and allows for the quick and easy creation of optimized graphics on multiple platforms. With it, you can use lights in 2D games and other effects in post-processing. The module is still being developed and improved with new features.

Shader graph (see: Shader Graph (2025)) is a module in Unity that allows you to easily create a shader (i.e. a short computer program executed on the graphics card, which in graphics describes the properties of individual pixels and vertices) without having to write code, but by connecting elements and building a graph. The tool allows for a real-time preview of connected nodes, which significantly speeds up work. Shaders were used in the project to create a material, a resource in Unity that specifies how the surface should be rendered. The material contains a reference to textures, color shades and to the shader, which specifies how each pixel of the object to which the material was added should be rendered, taking into account, for example, light.

Aseprite (see: Aseprite (2025)) is an open-source application. It is designed to create 2D graphics in the style of Pixel Art. It has many helpful and advanced tools for creating and editing animation frames, which - after displaying them in the right sequence and time - create an animation.

Pixel art (see: Pixel Art (2025)) is a method of creating raster graphics, in which images are drawn and edited at the pixel level using appropriate graphics programs. Pixel art is characterized by a unique visual style, where all pixels, or appropriately small squares filled with color, create a coherent whole as an image. The visual effect is very similar to a mosaic. This technique was the forerunner of the first 2D games, and was therefore used in such productions as Mario Bros. or Pac-man. It sets the canon of games in the 90s. The fact is that despite the rapid development of 3D games with increasingly better and more realistic graphics, games in the pixel art style still enjoy wide interest among players.

Microsoft Visual Studio (see: MS Visual Studio (2025)) is a programming environment from the .NET family of languages, including C#. It is Unity's preferred tool for editing scripts and debugging. It allows you to download an extension necessary to work with the aforementioned game engine. It allows you to quickly and efficiently rebuild code, maintaining syntax and clean code principles.

Game's Implementation

To create animation, sprites are necessary, i.e. two-dimensional raster images. Sprites graphically represent objects in the game world, which can move independently of the background on the screen. They are most often used as objects and characters in 2D games. They can be static or animated - in fact, they themselves, when displayed in the right combination, create an animation, as will be shown below.

To draw sprites for the engineering work, the previously mentioned Aseprite program was used, which has a number of functions necessary for effectively drawing successive frames of animation.

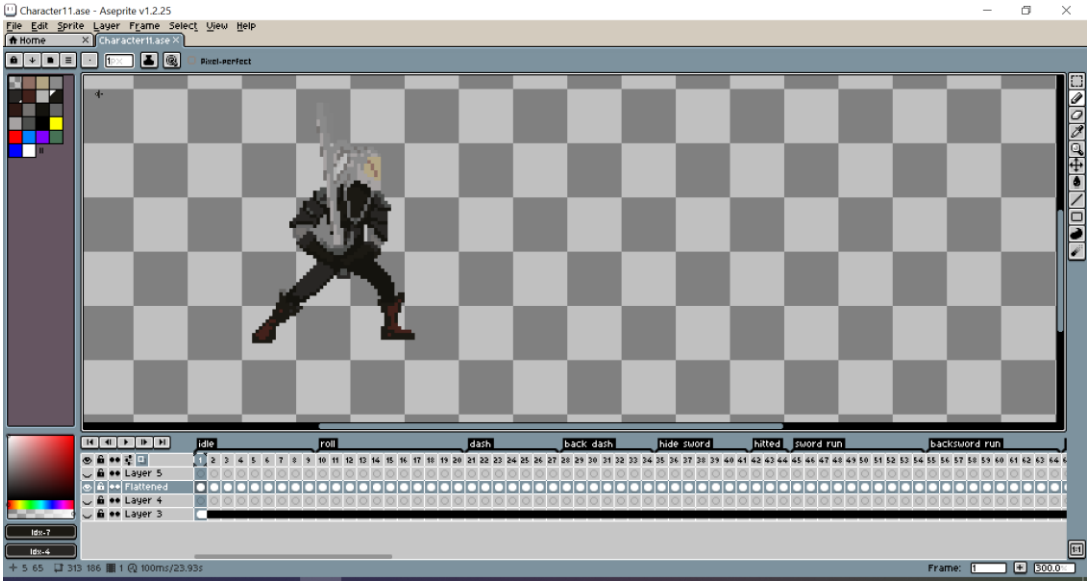

Fig. 2. Sprite in Aseprite

For the main character, 250 sprites were drawn, consisting of 21 different animations. These include: running with a drawn sword, running with a sword on his back, fast and strong attacks, death and others.

In order to make it easier for a computer program to load the textures necessary for animation, it is common to save sprites – subsequent frames of one animation – to one graphic file as a texture atlas, i.e. an image with many textures in one main image. This file is much larger, duplicated by the number of rows with the height of one sprite and similarly the number of columns with the width. This practice allows for significant savings in computer resources.

After finishing creating individual sprites in Aseprite, you should export the drawn animation frames to a single image file with the .png extension. The built-in tool in the program is helpful in this, which automatically divides sprites into rows and columns. A file prepared in this way can be easily imported into the game engine. In Unity, the sprite sheet editing tool is very handy and intuitive. It allows you to divide the image by the number of rows and columns, by the size of one sprite or by automatic selection of sizes. After dividing the texture, we get individual sprites ready to use in any configuration.

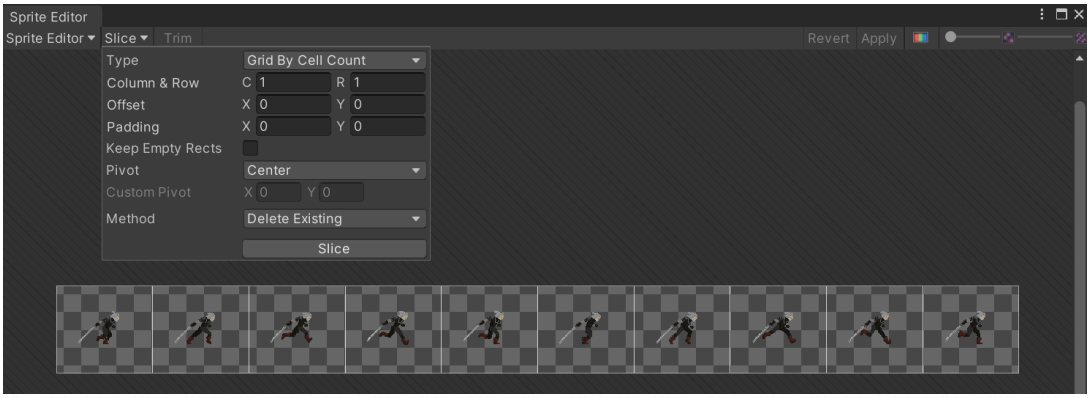

Fig. 3. Sprites' editor in Unity

Animations are an incredibly important part of a game's visuals. They help bring life and character to objects. They also provide a lot of information to the player: Am I running or walking? Am I jumping or rolling? Did an enemy damage me? Without clear visuals, the game would be much more difficult and frustrating. Animations help prevent this.

In Unity, by adding an Animator component to a Game Object, you can easily add and manage Object animations. An Animator requires a reference to the Animator Controller, which allows you to organize Animation Clips associated with Animation Transitions.

Animation Clips (see: Animation Clip (2025)) are the most important part of the animation system in Unity. They represent a single part of the character's movement, for example running animations. The timeline (Fig. 4), placed on the right side of the window, is divided according to the displayed animation frames. Bright dots placed on the timeline are keys, which correspond to the settings of a single element (e.g. a single sprite). The playback speed is modified by the Samples variable visible in the figure, which is responsible for how many animation frames will be played in one second.

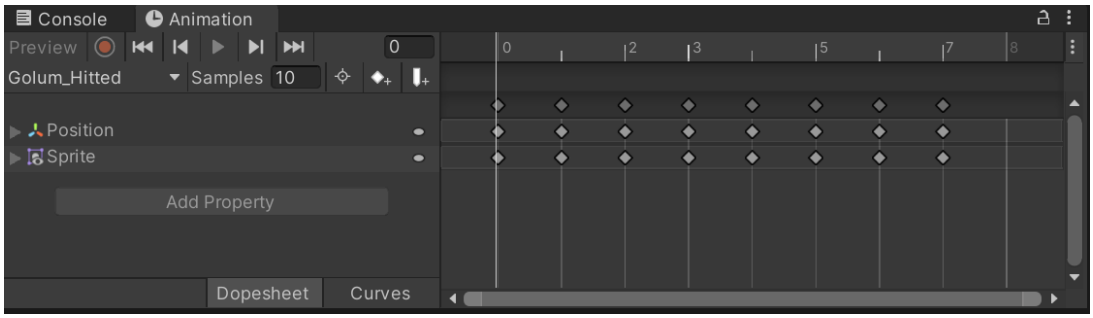

Fig. 4. Animation settings window with timeline visible

In addition to playing sprites, animation settings offer many more conveniences. During the animation, it is possible to change any values contained in the components assigned to the animated object.

Animation Event (see: Animation Event (2025)) allows calling a function from any script assigned to an animated Game Object without a parameter or with one parameter while playing a selected frame on the timeline. This is an incredibly useful property in the animation system, which was used in the project, among other things, to deal damage to characters in the game. Let's assume a situation in which the player uses a certain attack to deal damage to the opponent. Then the method responsible for starting the animation is called. However, due to the duration of the animation, it is not advisable to immediately take away the opponent's health points.

The screenshot in Fig. 5 shows the individual frames that make up the animation. It is desirable to take away the opponent's health points only in the frame marked with a green arrow, because earlier, during the animation playback, the opponent could have gotten ahead of us and dealt damage first. Then the attack animation would be interrupted and the animation of receiving damage would immediately start. By using Animation Event, we are sure that the method responsible for taking the Opponent object and taking away the health points from this object will be called at the right moment.

Fig. 4. Animation frames showing the selected attack

The individual layers (see: Fig. 5) in the game were added as separate Game Objects. After adding a simple script to each of them, it is possible to achieve the parallax effect (an illusion that gives depth, three-dimensionality) by changing the position of these objects depending on the camera position. Accordingly, distant objects, such as mountains, move quickly relative to the camera. On the other hand, the closer ones - trees and bushes - move slower, giving the impression of depth to the entire background.

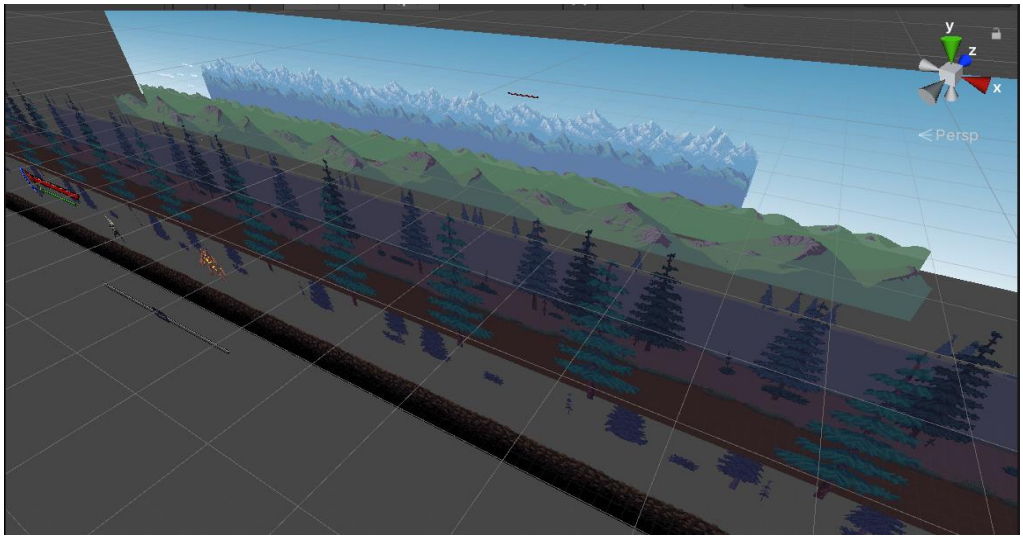

Fig. 5. Screenshot from the Unity editor showing the background layers

This script also handles background looping: since the level is finite, it repositions off-screen layers ahead of the player to create the illusion of an endless environment. Object pooling comes to the rescue, which moves a given layer to the player's vicinity so that the layer is always in the camera's range.

When designing a user interface, emphasis should be placed on readability and simplicity, so that the entire layout is intuitive and convenient for the player. To briefly describe the User Interface (UI), it is a part of a computer program, application or computer game responsible for communication with the user. The most common form of interface is the GUI (Graphical User Interface), i.e. a graphical representation of the UI that allows the player to interact with the application through components, e.g. Button or TextField.

Heads-up display (HUD) is an integral part of most computer games, containing additional information about the main character. For the needs of the application, such elements as the player's health bar, the opponent's health bar, bars of various resources can be designed. An example of a health bar, energy points and stamina is visible in Fig. 6.

Fig. 6. Player status bars

In arcade and, especially, fighting games the player has to launch attacks. The types of attack may be different, e.g. standard, fast, strong. Each type of attack has different tactical possibilities, differs in animation duration and damage dealt, which, combined with the use of skills, introduces variety and a multitude of ways the player can defeat the opponent.

Another important elements are special skills. In Fig. 7, we can see skills like barrier (neutralizes enemy's attack), shockwave (pushes away nearby objects and enemies), magic trap (immobilizes enemy), fireball (damages nearby enemies).

Fig. 7. Exemplary skills

The skills in the presented example were created using the ParticleSystem component, available in Unity. It allows you to generate "particles" as separate physical objects in an optimized way. The component has numerous functions for changing the behavior of these particles (speed, lifetime of a single particle, adding noise), changing graphics, collisions with other objects, etc. The visual effect of the skills was achieved by writing your own shaders using the ShaderGraph module.

Games in Unity are made up of Scenes, and everything contained in a scene is called a Game Object. It is a container that contains many implemented functionalities through the use of scripts and built-in components. Game Objects can contain other Game Objects in a parent-child relationship.

Rigidbody2D is a component that, when added to a Game Object, allows you to interact with the physics engine available in Unity. It allows you to simulate physical interactions in the game, e.g. applying a force to an object via a script. All operations related to simulating physics in the script should be used in the FixedUpdate method.

Collider2D is a component defining the shape of the Game Object used for physics simulations. The shape of the collider can be freely modified, but it is recommended to keep only approximate outlines of more complex shapes for performance reasons. The approximate shape of the Object is usually sufficient for collisions.

Each GameObject has a Transform component, which stores and controls its position, rotation, and scale in the scene.

Camera is a tool that captures and displays the game world to the player. Multiple cameras are allowed in a single scene. They can be set in any order and display only specific parts of the game world.

Conclusions

Implementing a computer game is a very time-consuming and demanding undertaking. Undoubtedly, the biggest challenge for a developer working alone was to combine and organize time for two monstrously large and essential areas of the project: creating graphics and proper implementation of the game. Without passion and love for computer games, it is difficult to implement such a project..

Working in the Unity game engine is very intuitive and pleasant. The Unity documentation was incredibly helpful during the implementation, as it describes in detail each element of the game engine.

A computer game project can never be considered finished. Constantly appearing new ideas for improvements and extensions do not allow me to abandon this project.

References

- Aseprite (2025), [Online], [Retrieved May 4, 2025], https://www.aseprite.org
- Asset Store (2025), [Online], [Retrieved May 4, 2025], https://assetstore.unity.com
- Animation Clip (2025), [Online], [Retrieved May 4, 2025], https://docs.unity3d.com/ScriptReference/AnimationClip.html
- Animation Event (2025), [Online], [Retrieved May 4, 2025], https://docs.unity3d.com/Manual/script-AnimationWindowEvent.html
- Britannica (2025), [Online], [Retrieved May 2, 2025], https://www.britannica.com/topic/game-recreation
- Huizinga, J. (1938) 'Homo ludens: Proeve Ener Bepaling Van Het Spelelement Der Cultuur', Groningen, Wolters-Noordhoff
- MS Visual Studio (2025), [Online], [Retrieved May 4, 2025], https://visualstudio.microsoft.com
- Pixel Art (2025), [Online], [Retrieved May 4, 2025], https://en.wikipedia.org/wiki/Pixel_art
- Rollings, A. and Adams, E. (2003) 'Andrew Rollings and Ernest Adams on Game Design', New Riders Pub.
- Shader Graph (2025), [Online], [Retrieved May 2, 2025], https://docs.unity3d.com/Manual/shader-graph.html
- Thomas, F. and Johnston, O. (1981) 'Disney Animation: The Illusion of Life', Abbeville Press, New York.
- Unity (2025), [Online], [Retrieved May 2, 2025], https://unity.com
- URP (2025), [Online], [Retrieved May 2, 2025], https://unity.com/features/srp/universal-render-pipeline