LAPIN AMK
Lapland University of Applied Sciences

EDUCATIONAL GAME FOR FORESTRY

Creating Unity 3D Game

Äkräs Mini-Game Development

Solomon Finnan

Bachelor's Thesis
Lapland University of Applied Sciences
Degree Programme in Information Technology
Bachelor of Engineering

2018

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

School of Technology, Communication and Transport
Degree Programme in Information Technology
Bachelor of Engineering

Author	Finnan Solomon	Year	2018
Supervisor	Erkki Mattila		
Commissioned by	pLAB		
Title of Thesis	Educational Game for Forestry		
Number of pages	28		

This project was part of the Äkräs mini-game development project. It aimed at developing an enjoyable educational game for the children and the young society of Finland. The objective was also to create and raise awareness of the forest sector and the various business opportunities it offers.

The Unity game engine was used for developing the game, and it was another objective to learn the process of building a complete game using Unity. Moreover, the spiral software development model was used during the development process, and GitLab was used as a version controlling system.

The end result of this project was a forest themed fully playable demo version game. There will be further development in the future for the game to be released in the Äkräs mini-game website. Furthermore, this thesis described the implementation process in detail that can be used as a resource for a Unity engine based development.

Key words Unity, game, 3D, WebGL

CONTENTS

LIST OF FIGURES

FOREWORD

I would like to thank pLAB for giving me the opportunity to work with them and be part of a big project. Specifically, I would like to thank Tony and Elina for assisting me throughout this project concerning the programming issues and the thesis related matters. I would also like to thank the Äkras mini-game team for providing great resources for the project and their help and guidance whenever in need. Finally, I want to thank my family and friends for always being supportive.

SYMBOLS AND ABBREVIATIONS

GUI	Graphical User Interface
NPC	Non Player Character
DLL	Dynamic Link Library
GUI	Graphical User Interface
VR	Virtual Reality
AR	Augmented Reality
HTML	Hyper Text Markup Language
WebGL	Web Graphics Library
API	Application Programming Interface
AI	Artificial Intelligence

1 INTRODUCTION

The video game industry has been growing fast with the expanding number of users and the use of new innovations on gaming platforms starting from smartphones to Wii u console, virtual reality console, PlayStation, Xbox and other devices. In Finland, the game industry has certainly risen after the two successful companies over the past five years pushing a 100-million-euro industry to two billion. Rovio Entertainment (the creator of Angry Birds) and Supercell (the creator of Hay Day and Clash of Clans) are both based in Finland. These companies are examples of Finnish startup companies that grew into large game publishing companies. There are several opportunities and potential career paths for game developers in Finland. This is one of the reasons for choosing the thesis topic in addition to the author’s personal interest the game development field. (Tekes views 2016.)

This thesis produces a unity3D web game. It is commissioned by pLAB at the Lapland University of Applied Sciences. The company works in game developments, real-time integrated 3D visualization environments, and other IT solutions (Lapland UAS 2017). The project was done by taking part on an ongoing game project from pLAB, the Ärkäs mini-game development project. This project involves seven team members. A GitLab workspace was created and used to manage the project and develop in a team. This thesis presents the principles and methods of using game development technologies in the process of building a 3D game using the Unity game engine. Furthermore, in the thesis the process of implementation, profiling and optimization, and a useful development practice will be described.

The Ärkäs Mini-game development project plans to reach the children and young people at the age of 7 to 21 and their families in Lapland via different forestry based games. These games are being developed to promote the wellbeing and living of the children and young people by increasing their awareness of the forest sector and wildlife education. This creates opportunities, since it is believed that it is important for young people to learn the importance of forests and to see forests as a multifaceted source of income.

These project exploits today's technologies, methods and social media channels to implement interesting games.

The primary beneficiaries of this project are young people who are aware of the opportunities for the forest sector as a livelihood. Game enthusiasts will receive a newly built game. Schools and colleges have access to a teaching - learning environment for young people. Forestry operators are provided with a tool to introduce the forest sector to young people. Businesses will have the opportunity to introduce themselves to the forest. The forest sector also benefits from all these in general.

This project is generally limited by the developer’s programming skill and knowledge. Besides, a level design and an additional feature implementation depend on the progress of the project according to the schedule. The project mainly focuses on completing a fully playable one level game.

2 GAME PROGRAMMING

2.1 Unity Engine

Unity Game Engine is among the world's leading tools used to develop video games, animations, and simulations. This game engine provides a feature to develop a project and have the option to deploy it onto multiple platforms. According to (Unity technologies 2017c), Unity currently has “25+ platforms across mobile, desktop, console, TV, VR, AR and the Web”. Figure 1 below presents all the platforms that Unity engine supports.

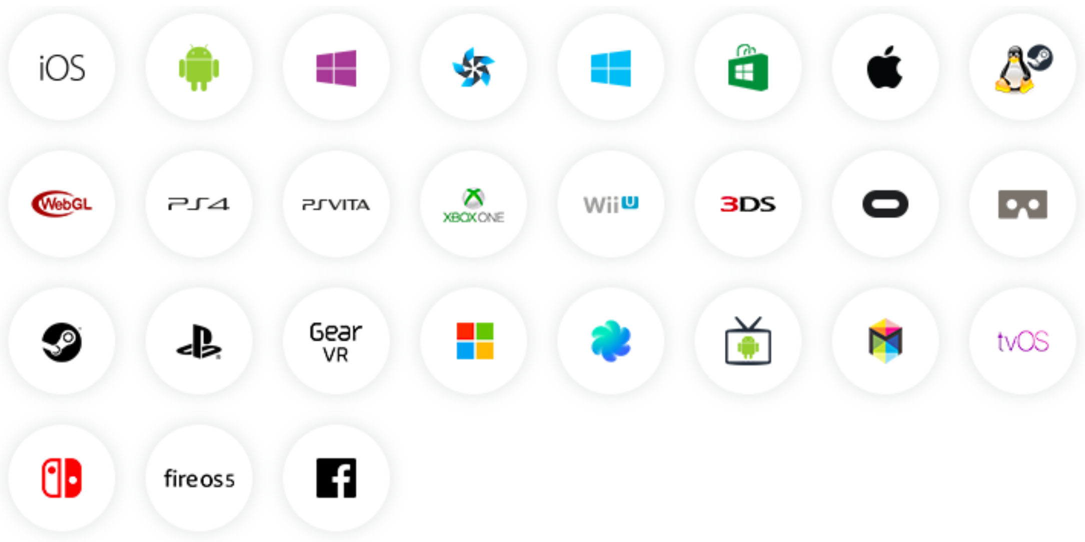


Figure 1. Unity Platforms (Unity technologies 2017c)

Unity offers a great Asset store with a wide variety of assets available to assist users with Unity development. There are many types of free and paid assets gathered in the asset store altogether. These tools range from simple to complex and pretentious models, characters, materials, audios, particle effects, animations and rigging tools, game environments, GUI builders, plug-ins and many more. The Asset store minimizes the time and effort spent on developing assets repeatedly, by providing ready-made assets for everyone. It also helps to speed the development by saving the time from working on solutions on which developers are not experts in, by using the suitable assets from the store. (Unity Technologies 2017a.)

Other factors which make Unity a great game engine are the tools and file formats it supports and the documentation and overall support from Unity technologies. Unity supports importing 3D models and files from the most commonly used 3D applications such as Blender, 3ds Max, Maya, Cinema 4D, Modo, Lightwave, Cheetah3D and SketchUp, making it a great engine to use for game development. Unity has a well-organized documentation, user manual and scripting API with example codes which serves users as a main reference. Besides, many online tutorials are available on the Internet both from Unity Technologies and individuals with unity expertise. (Pluralsight 2015.)

The Unreal engine is a main competitor of the Unity engine and it is also widely used for game developments. Both of these engines have their own strengths and shortcomings, but there were comparisons made between the two engines before choosing Unity for this project. Unreal engine 4 uses the programming language C++ and Unity engine uses C#. Due to the developer's prior experience and preference, C# was more preferred. The other comparison made was concerning the Asset store, Unity has a wide variety of numerous assets compared to Unreal engine. This was advantageous hence it is time taking to develop everything for a game, and asset stores will efficiently be used throughout the development process. On the other hand, Unreal engine offers a high level graphics compared to Unity. However, high graphics quality was not required for this game. Due to this and the above mentioned factors Unity was found to be a better choice for this project. The other main reason Unity was chosen for this project was for its cross platform development feature. Even though Unreal engine has the same feature, it supports 15 platforms whereas Unity engine supports 27 platforms. The project members' skills and experience using the game engine was also an additional factor. (Unreal Engine 2017; ValueCoders 2017.)

2.2 Modeling

Modelling characters is a very essential part of game developments or animation productions. There are number of cases where games and animations initiate from a pre-made models. A good model design hatches ideas in a game or an animation developer’s mind and results in new productions in the fields.

Developers do not necessarily need to model the desired characters from a scratch. There are a number of free and paid character creation tools available on the Internet to create realistic 3D models. 3D modelling tools like MakeHuman, DAZ Studio, Adobe Fuse CC, iClone and others minimise a lot of time and energy spent on modelling. (Yusuf 2017.)

For instance, MakeHuman is a software that is used to make 3D virtual human characters easily from a GUI (Graphical User Interface) with different controllers. It uses main parameters such as gender, age, height, weight and ethnicity and other minor parameters consisting of details of the model, all to be controlled by using sliders in the GUI. All the changes made to the model is viewed on the screen simultaneously. It helps the users figure out what they want by navigating through the various controls. (MakeHuman 2016.)

During this thesis project, the author joined the Äkräs mini game development project. This project delivers interesting mini games that are designed to educate players and create a good perception about the forest. The team consisted of game artists and there were many designed models and game objects that were already made and available before the game development began. Therefore, no modelling work was done during this project. In the beginning, experiments were done by placing the different game objects/models inside the Unity scene editor and trying out animations, giving the developer new ideas to implement on the game.

2.3 Scripting

Unity offers two programming languages natively, C# and UnityScript. Besides these, other .NET languages which can compile a compatible DLL can be used with Unity. In this thesis project, the programming language used was C# for many reasons. Mainly because Unity Documentation focuses and provides more examples on C# compared to UnityScript. C# also has many tutorials and helps on the Internet compared to UnityScript. (Unity Technologies 2017b.)

The main drawback of using the UnityScript is the fact that it is only made for Unity engine and cannot be applied elsewhere. Whereas C# is an industry-standard language and gaining scripting skills from the language can be useful to develop on many other platforms. In addition, Unity is planning to eliminate UnityScript in the future because the language has low number of users (14.6% of all Unity 5.6 projects have at least one UnityScript file but only 0.8% of all projects use 100% UnityScript). Unity is planning to make scripting upgrades and believes it will be a waste of time to develop updates for UnityScript when it has such low number of users. (Fine 2017.)

2.4 WebGL Platform

This project aims to publish the game on two platforms, mainly on the Web and on Windows as a PC standalone game. WebGL is a JavaScript API that conveys 2D and 3D graphics to the web. It is based on the popular 3D graphics rendering standard OpenGL. WebGL is supported by HTML technologies, which enables 2D and 3D graphics contents to be rendered using the HTML canvas tag element in the web browser. Therefore eliminating the need to use any third party plug-ins unlike other technologies such as Flash and Silverlight. (Arora 2014.)

Unity offers a WebGL build to deliver games on web browsers. WebGL is cross-browser and it is supported in most of the modern browsers, Google Chrome 9+, Mozilla Firefox 4+, Apple Safari 5.1+, Opera 12+ and MS Internet Explorer 11+. Besides the importance of browser’s support, the hardware of the player’s device should also support WebGL features. For example, a device need to have the proper graphics hardware for WebGL features. Commonly, problems with graphics hardware can be solved by updating graphics driver. (MDN web docs mozilla 2017.)

2.5 The Spiral Model

The spiral model of software development was used in this development process. This model gives the freedom to iterate, and focuses on developing and refining the original design by improving and testing repeatedly until the desired goal is reached. Throughout this game development, there were many changes implemented to the original idea of the game, therefore this model was found to be an appropriate approach. As can be seen from Figure 2 below, a development begins at the center of the figure and moves following the swirling pattern in clockwise direction. Iterating through the four divisions, the risks can be analyzed regularly and the new prototypes can be built alleviating those risks. After testing the prototypes, the next plan is designed on the basis of the previous findings and observations. (Schell 2015.)

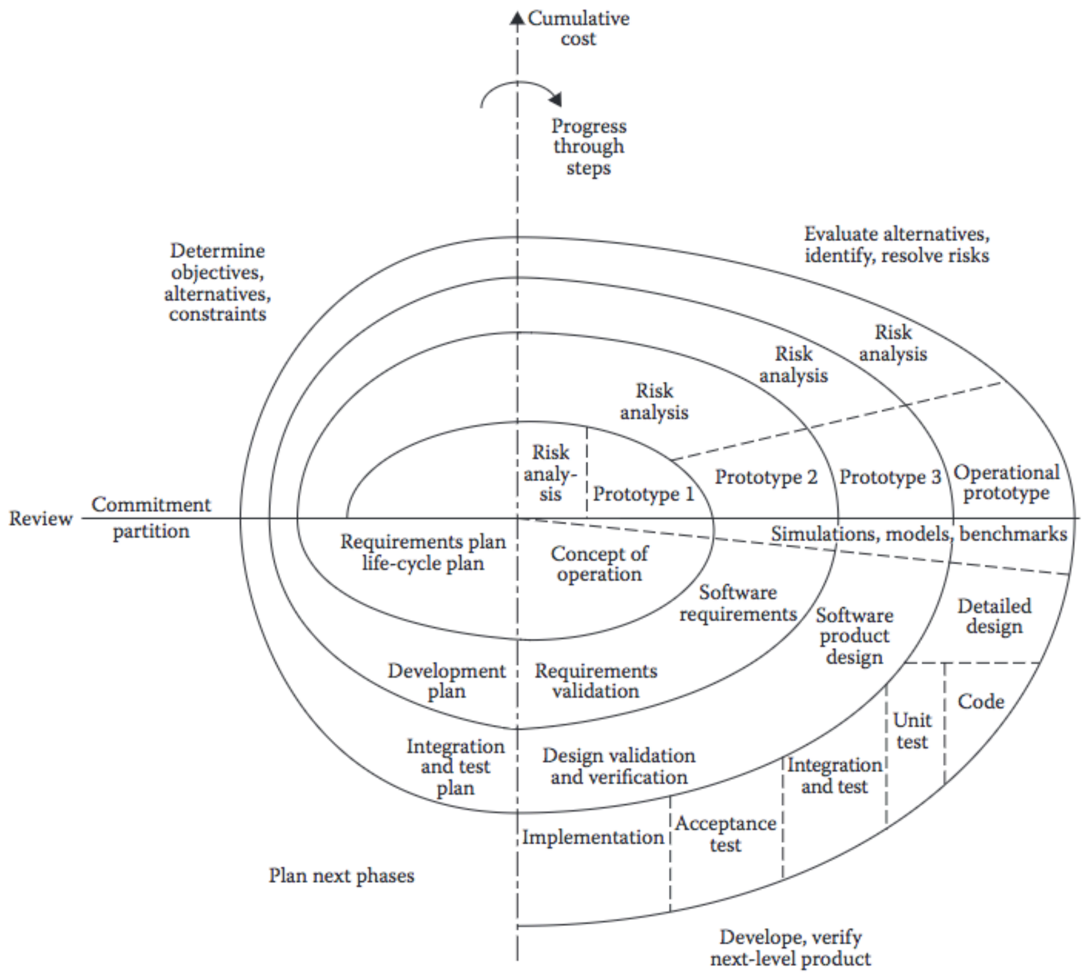


Figure 2. Spiral Model (Schell 2015)

3 MINI GAME DEVELOPMENT PROCESS

3.1 Overview of the Project

The fact that human beings especially youngsters are interested in a relaxed and fun way of learning or doing activities leads to the main concept behind this thesis project, Which is the possibility to create an interesting game in order to teach and raise awareness of the young people. The youngsters will have enjoyable educational games as a result of the Äkräs mini-game development. This thesis project is part of the mini-game development and delivers a forest themed 3D game, which will benefit the youngsters and the forest sector.

Theoretically, the idea of this thesis is to create a good game idea to implement it in such a way that it educates young people about the forests of Lapland and entertain them at the same time. The main requirements for this project were that the game should be appropriate for youngsters. It should be a teaching game or aims to raise awareness of the forest. This thesis should also briefly discuss the technologies used in making the game, mainly Unity game engine.

Unity is among the top developer engines with over 2 million registered developers. It is a free software that is used from traditional indie developers and startup companies to a well-known publishing companies. It is a powerful tool with a well scripted manual and documentation alongside example codes, free tutorials, online training, an active community and forums, and Asset store and galleries with many useful contents. There are also hundreds of tutorials on the Internet on unity programming which proves it as a reliable and easy tool to learn and develop games. (ValueCoders 2017.)

During the practical part of this project, the unity Scripting API was used as the main programming reference. The unity online community, different online forums and stack overflow website were of a great support to solve technical problems throughout the development. Besides these, YouTube videos from Unity and others were used to gain information on how to implement the raw ideas of the game and also to find solutions for many problems. Furthermore, Personal discussions and different suggestions from pLAB IT specialists were always helpful to improve and accelerate the project.

3.2 Implementation Process

The aim of this thesis project is to develop a complete unity game, and write the theoretical basis and explanations about the technologies that will be used in the practical work of producing the game. The main phases in this project consisted of the planning phase, designing phase, Implementation phase and testing phase.

The planning phase was basically designing and writing a project plan for this thesis. It began by creating a game idea and planning the basic concepts of the game. It defined all the major functions that needs to be accomplished in their order of sequence and set an estimated deadline and work amount to activities by breaking down the functions wisely. Risk analysis was also composed at this stage.

The designing phase was for the most part building the terrain. Experiments were done with the 3D models which were available for this project in order to create scenes depicting the game idea. The contents and rules of the game were also designed even if there were several changes while developing.

The Implementation phase was mainly programming and it took most of the time compared to the other stages. In this game, C# was used for coding. Having set the basic desired graphics in the unity editor window, the first task was to develop a prototype by implementing the core game idea. While prototyping, if the game idea needs adjustments, different tweaks can be applied in the editor scene to get new ideas and do better modifications. The prototype will give an idea of the end product or the full game’s appearance.

The phases in game development runs simultaneously with each other. There could be several design changes at the implementation phase but mainly all the programming is done here, developing the prototype into a full game. Character movements, player’s missions, enemy designs, game audios and more components were further developed. In addition to implementing the designed features, it was important to add more contents and features to make the game interesting. Meanwhile, the GUI components of this game such as menus, score, health bar and time counter were designed to be easy and user friendly.

Testing was done on different stages following the completion of main tasks and side by side with programming, starting from when the game was playable. Testing and fixing bugs and defects on early stages helped minimize flaws when finalizing the project. At the final testing, testing was done on target platforms, in this case different web browsers and windows machine. Simultaneously, Bugs were eliminated as much as possible to optimize the game's quality and performance.

4 IMPLEMENTATION

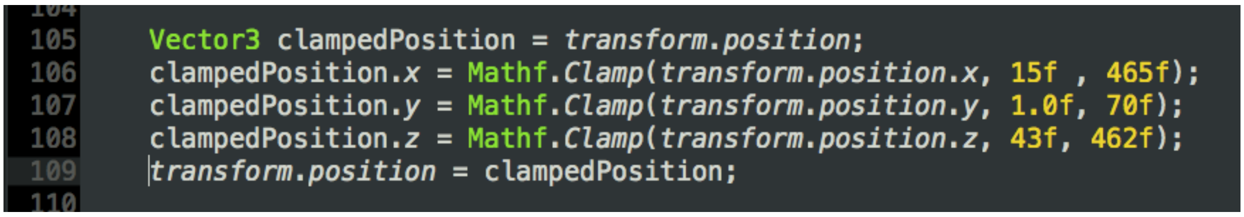
4.1 Character Movement

In this project, designing the character movements was originated from the various animations which were already designed for the models. For instance, the player character for this game is a bird known as Eurasian golden oriole. The model's gliding and flapping animations were used suitably with the character's movements. Whenever the bird is moving upwards the flapping animation is run, and the gliding animation is connected with moving downwards to create a more realistic flying. Moreover, Unity's rigid body and physical colliders were applied along with other related physics components to create a naturalistic environment. As for the enemy characters, movements were controlled through recorded animations and code, which will be further discussed on the next sub chapter.

The movement function for the player character was simply written to use readings from the keyboard's arrow keys using the Input.GetAxis function, and add a multiplying force to them. Vertical keys control forward and backward translations, and horizontal keys control rotations. In this project, all of the flying characters are limited to perform rotation about the y-axis to keep the game simple, rotations about the x-axis and z-axis are set to freeze. Additionally, left and right Shift buttons were added to control the bird's upward and downward movement.

In this project it was essential to keep the player character and all the other game components with in the designed terrain environment. Initially, cube meshes were used as borders with colliders attached to them. While this works perfectly with other moving game objects, the player character bird was having a rough and unrealistic collisions at certain areas in the terrain. Therefore two additional functions were used to constrain the bird with in the game area and to improve the collision.

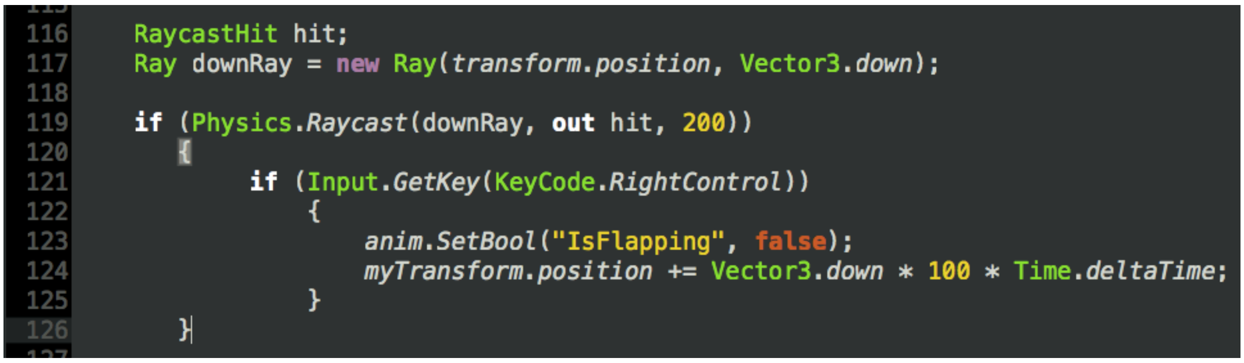
Clamping function was used to constraint the movement of the Bird in all the x, y and z directions. As it can be seen from Figure 3 below, minimum and maximum values were set for x, y and z positions. If the character tends to exceed these limits, it will maintain its prescribed clamp position.



```
Vector3 clampedPosition = transform.position;
clampedPosition.x = Mathf.Clamp(transform.position.x, 15f , 465f);
clampedPosition.y = Mathf.Clamp(transform.position.y, 1.0f, 70f);
clampedPosition.z = Mathf.Clamp(transform.position.z, 43f, 462f);
transform.position = clampedPosition;
```

Figure 3. Clamped Position

Despite the approach explained above, there were still issues related to collisions around uneven terrain compositions, especially with an increased speed of the player character. The Physics.Raycast function was then applied to detect and control collisions better. Figure 4 below presents the Raycast function implemented. Before reading the keyboard input for moving the bird downwards, the Raycast function casts a ray from the bird’s position, detects which game object it is facing or colliding with and then returns its value. This prevents the bird to move outside of the environment as the ray cast function will return false.



```
RaycastHit hit;
Ray downRay = new Ray(transform.position, Vector3.down);

if (Physics.Raycast(downRay, out hit, 200))
    {
        if (Input.GetKey(KeyCode.RightControl))
            {
                anim.SetBool("IsFlapping", false);
                myTransform.position += Vector3.down * 100 * Time.deltaTime;
            }
    }
```

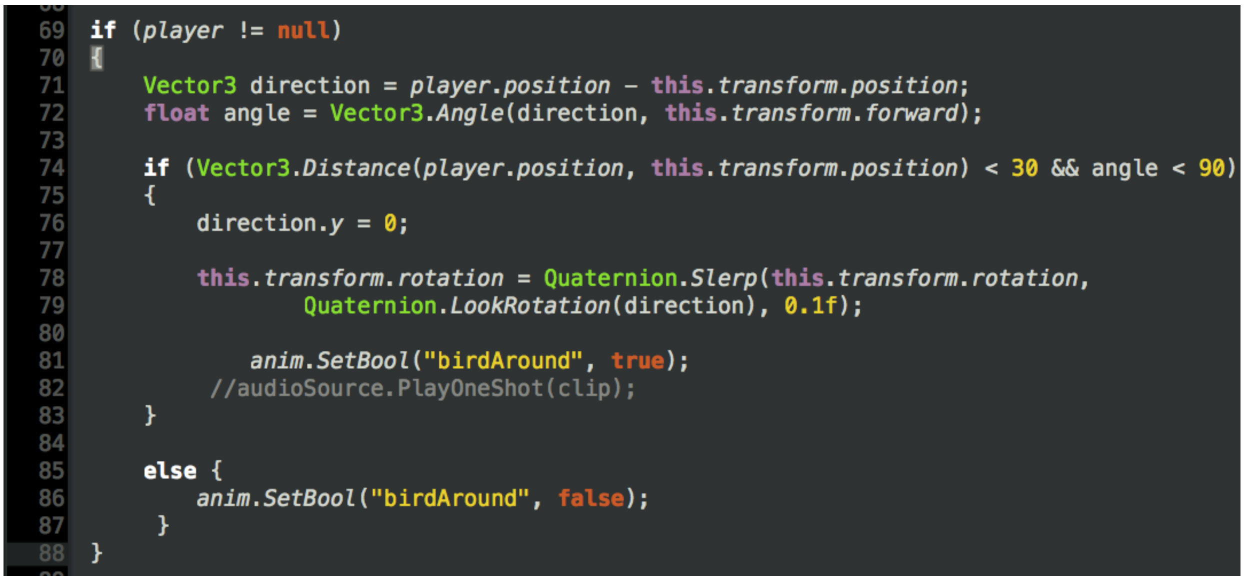
Figure 4. Physics.Raycast Function

4.2 Enemy Implementation

Enemy Implementation is a main component in developing an interesting game. It is important to keep a balanced and fair challenge throughout the game to provide players with a content which is not too difficult and not too simple and dull at the same time. This project does not include a level design and the difficulty increases from the starting point of the game through the different paths in the environment.

The enemy characters applied in this project are a bird known as great spotted woodpecker and a brown bear. These animals are inhabitants of the forests of Finland, which is a contributing factor for this project's aim of raising awareness about the forest. These enemies being non player characters (NPCs), require a different way of implementation than player characters, which are controlled by players. AI designs, recorded animations and controller scripts were used to control NPCs in this project.

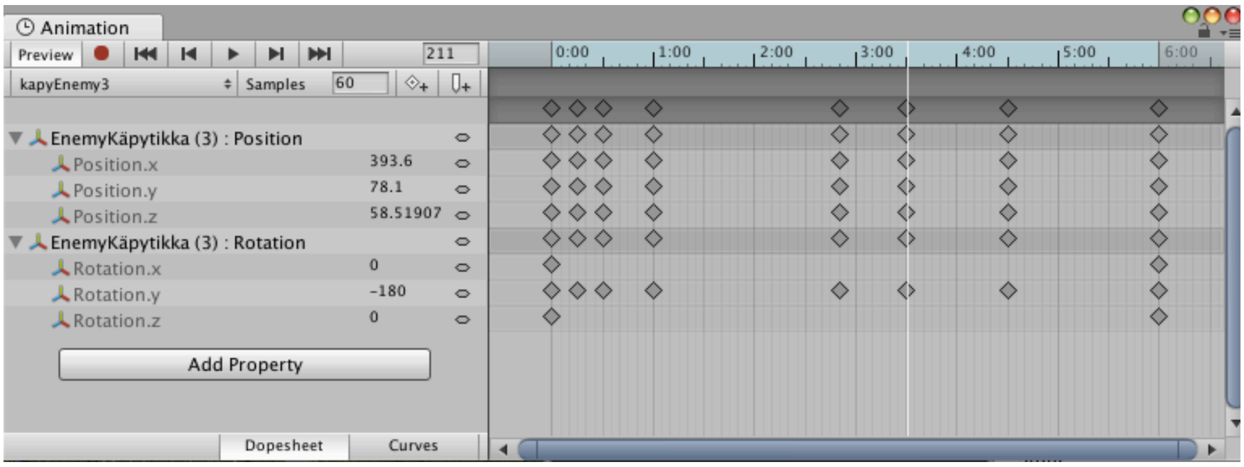
AI in games develop NPCs to behave in a realistic and believable manner inside the environment by causing them to respond appropriately to events happening in the game (Kyaw, Aung Sithu, et al. 2013). In this project, an AI was designed for the bear NPC using physics attributes in a controller script. As can be seen in Figure 5 below, first, the player's direction, and the angle between the direction the NPC is facing and the player's position is calculated. Next, using these calculations a range of distance and an angle is defined inside the if-statement. Afterwards, the NPC is able to detect if the player is within the defined range of values and performs a rotation to face the player. Afterwards, the Animator.SetBool () method is used to send Boolean values to trigger transitions of animation states inside the animator controller of the NPC. As a result, the bear plays an attack animation when the player is in range and it plays an idle animation when the player goes out of range. Additionally an audio clip is played up on the NPC attacking the player.



```
if (player != null)
{
    Vector3 direction = player.position - this.transform.position;
    float angle = Vector3.Angle(direction, this.transform.forward);

    if (Vector3.Distance(player.position, this.transform.position) < 30 && angle < 90)
    {
        direction.y = 0;

        this.transform.rotation = Quaternion.Slerp(this.transform.rotation,
                Quaternion.LookRotation(direction), 0.1f);

            anim.SetBool("birdAround", true);
         //audioSource.PlayOneShot(clip);
    }

    else {
        anim.SetBool("birdAround", false);
     }
}
```

Figure 5. Bear Controller Script

In case of the enemy bird NPCs, a controlling script was written and an animation was recorded using the Unity animation view and Timeline window. The timeline editor was used to connect a set of the enemy bird's animation and create a sequence to be played simultaneously with the NPC's movement inside the game world. The timeline consists of useful features. For instance, there is an option to set a loop mode on for the created playable. It's also possible to either set an initial time to start the playable, or play it on awake. In addition, the animation view was used to create and modify animations directly inside unity. As can be seen from Figure 6 below, the position and rotation properties were used to record values for the NPC's activities in the different key frames. Due to the nature of the game, this short animation is running in loop for the specific enemy on its defined position.



211
kapyEnemy3
Samples 60

EnemyKäpytikka (3) : Position	
Position.x	393.6
Position.y	78.1
Position.z	58.51907
EnemyKäpytikka (3) : Rotation	
Rotation.x	0
Rotation.y	-180
Rotation.z	0

Figure 6. Enemy NPC's Recorded Animation

The enemy birds are designed to throw obstacles on the player character while the player is roaming in the environment to collect food. These enemies are running when the game loads but are set in the corners of the scene in an unpredicted manner. Besides, the food game objects are spawned randomly with a short life span, leading the player to move and collect them more randomly. This increases the probability of the player taking a hit from the obstacles being continually thrown.

In addition to these, there are two triggers positioned to throw another obstacle on the player. Whenever player reaches these triggering position, sets of logs will start falling from the trees on to the bypassing player bird caused unexpectedly by the windy weather. The more logs hit the bird, the more health is damaged. Particle systems were used to create the wind effect and design the falling of logs and leaves from the trees.

Generally, the player always starts the game in a full health and the health is threatened by those enemy NPCs and the obstacles described above. A counter is placed in the GUI to record the duration of the game play and observe for how long the player survived in the windy weather with these enemies and obstacles in the game world, and how much food it collected before reaching a zero health level.

4.3 Profiling Performance

In software development profiling is examining the performance of a software product in order to get visibility of the execution of the code. There are different kinds of profiling tools used to optimize the performance of a software based on the problem types. When it comes to profiling games, Unity engine offers a built in profiling tool that can be accessed from the unity editor. Profiler provides a comprehensive information about the functionalities of the different parts of a game. It is important to find the hotspots causing poor performances before trying to fix the problems. Profiler helps to locate those problematic areas by using the information gathered after profiling a running game. For instance, Memory usage of the game, CPU time usage, game logics and physics calculations frequency and duration, the GUI elements being rendered are among the data that can be gathered using the profiler. Hence this tool helps fix the most common performance problems which are related to memory shortage and low frame rates for instance. Furthermore, it is advised to run the profiling tool whenever making changes, to ensure the changes made are not affecting the performance negatively. Therefore it’s important to keep note of previous measurements and compare with new profiling results before proceeding to further modifications. (Dickinson, C. 2015; Unity technologies 2017d.)

Besides unity profiler, built in developer tools from web browsers can be used for profiling Unity WebGL games. In case of web browser games, performance can be affected by the web browser the game is running on, since unity WebGL does not maintain equal support to all of the web browsers. Additionally, the probability of getting performance issues is high because there is not a specific target device and the range of the end users’ devices is vast. However, most web browsers offer the JavaScript console which is an essential tool for developers to inspect web contents and help to identify performance problems.

4.4 Performance Optimization

Optimization is a methodology used to develop and improve a program to increase its effectiveness and performance as well as utilize assets and resources more efficiently. It can also be used to expand the game’s functionalities by allowing more features to the game following optimizations of resources, GPU and CPU usages. There are a wide range of approaches to do optimization but the fundamental principles hardly differ from one another. (Safdarzadeh 2010) states that “the steps are benchmark, detect, solve, check and repeat” as shown on Figure 7 below. (Safdarzadeh 2010.)

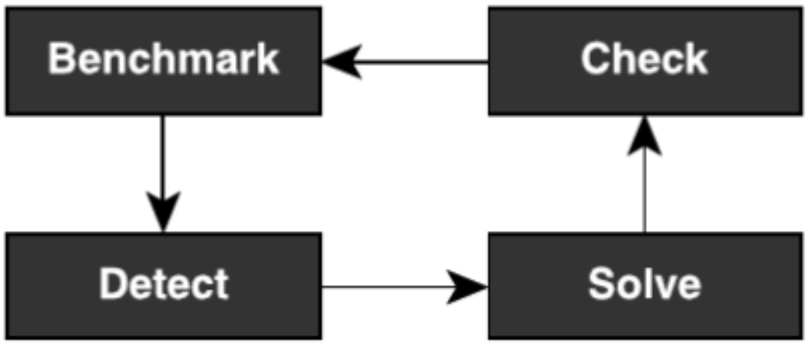


Figure 7. Optimization Lifecycle (Safdarzadeh 2010)

A benchmark is a relevant and consistent game-situation or problem chosen from the game as a standard to be used for performance comparisons in the optimization process. After choosing a benchmark, the biggest performance bottlenecks need to be detected among other insignificant problems in the game. This can lead to a good performance win in the end. Here, profiling tools can be used to help get a more precise information on which area to optimize. Afterwards, changes can be made to solve or improve the shortcomings that were detected earlier. This step might cause a lot of attempts before suggesting a final solution. Finally it is checked if the solution has improved the performance by using the benchmark defined. This process is repeated until the desired performance is met. (Safdarzadeh 2010.)

While optimizing Unity WebGL games, Memory usage analysis takes a big part because memory related errors occur most commonly. Figure 8 below shows the different categories in a web browser's memory for Unity WebGL content. All of this memory sections can affect the browser's performance but distinctively, poor Unity Heap size allocation can be a possible bottleneck for memory management. Unity Heap is a presumed memory that is allocated for all the run-time contents in unity. This memory needs to be assigned in the Unity WebGL player settings before game build. Its default value is 256 MB but it is important to give a specific memory size for optimum performance. (Kongregate Developer 2010.)

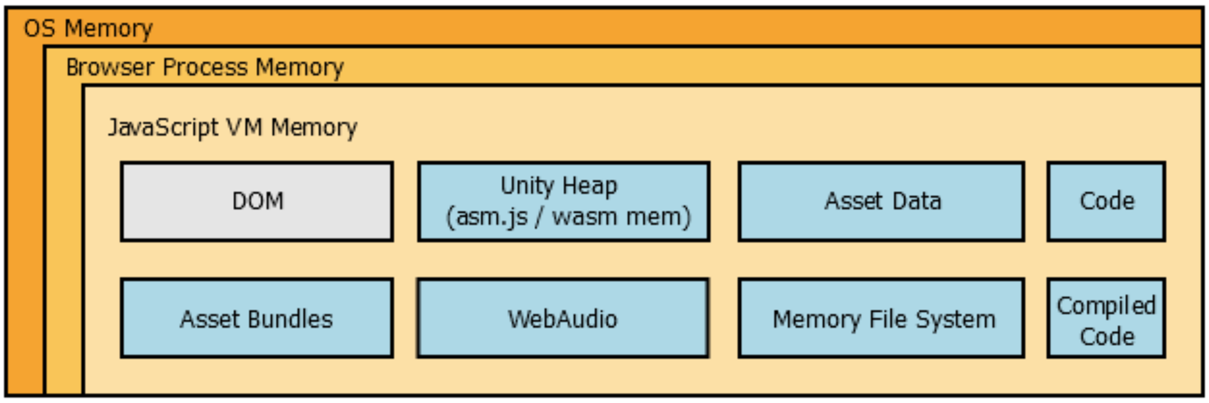


Figure 8. Web Browser's Memory Allocations (Marco 2016)

If the unity heap size is set to be too low, it can cause an out of memory error on the browser in case the unity content exceeds the allocated memory, and if it is set too high, the browser can crash trying to allocate that amount of memory in the browser. Using the unity profiler tool is a good approach to estimate a precise heap size by doing a development build with auto-connect profiler on. Afterwards, by running the game on a browser, the maximum value of memory that had been allocated for the game can be read from the memory allocations section in the profiler, labelled as Reserved Total. It is advised to add some more megabytes 15 to 20 MB on top of this value to assign the unity heap size. This is because there are allocations that are not tracked by the memory manager. (Smith 2016; Fantasy Arts 2016.)

4.5 Platform Comparison

This project was built on two platforms, WebGL and PC. Some of the advantages and limitations observed from these two platforms are listed below.

PC game advantages

- The development targets a specific device in terms of hardware and software.
- The log files are available for debugging after build.

PC game disadvantages

- The game needs to be physically installed on a device.
- The customers are device specific.

WebGL game advantages

- It is easy to share and distribute.
- The game does not need to be installed physically.
- The customers are not device specific.
- It reaches a great number of audience.

WebGL game disadvantages

- The development targets a wide range of devices in terms of hardware and software.
- The performance depends on user’s device and connection.
- It requires optimization for 32-bit browsers and lower spec devices.
- debugging is difficult.
- The log files are not available because WebGL cannot access a device’s file system.

As it is listed above, the WebGL platform has limitations on debugging. It is also necessary to perform a good optimization in order to sustain a good performance for all the users with different levels of devices. Whereas PC platforms comprise a good support for development, and it is also easier to perform debugging for PC builds compared to WebGL builds. On the other hand, it is easier to distribute a WebGL game to a large number of users compared to PC or Mac standalone games, since WebGL is less device specific and encompasses both PC and Mac users. However, the choice of platform is project specific and it depends on the project’s goal and requirements.

5 CONCLUSION

The main objective of this thesis was to develop an interesting educational game for children and the young community. In the development process it was aimed to enhance the developer's skill on unity game development, and gain a better understanding about game development process in general. This project definitely helped the developer to master skills on game development, specifically using unity game engine, C# programming skill from the coding perspective, prototyping, managing and working on a team project using GitLab, and project planning. The online support from unity forums and different tutorials had a great deal of help throughout this game development.

It was realized that implementing the original game idea, fully into the game was difficult in this project, and there were modifications and changes made continuously. For this reason, the spiral model was found to be effective for this project. The game's playable demo version was completed. However, further developments are required before releasing the game onto the Äkräs mini-game website. Level designs can be implemented since the demo version is a one level game and many other features could be added to the core game idea.

BIBLIOGRAPHY

Arora, S. 2014, WebGL Game Development, Packt Publishing, Olton Birmingham. E-book. Accessed 27 November 2017 http://ez.lapinamk.fi:2770/lib/ramklibrary-ebooks/detail.action?docID=1674846, Ebrary.

Dickinson, C. 2015. Unity 5 Game Optimization. Birmingham: Packt Publishing Ltd.

Fantasy Arts 2016. Unity WebGL TOTAL_MEMORY allocation. Accessed 25 November 2017 http://fa-games.de/2016/03/16/unity-webgl-total_memory-allocation/.

Fine, R. 2017. UnityScript's long ride off into the sunset. Unity Blogs. Accessed 15 October 2017 https://blogs.unity3d.com/2017/08/11/unityscripts-long-ride-off-into-the-sunset/?_ga=2.171855904.1476417188.1505382569-387934402.1488978546.

Lapland UAS 2017. PLAB-Software Engineering Laboratory. Accessed 10 May 2017 http://www.lapinamk.fi/en/Employers/Development-enviroments/pLab.

Make Human 2016. Open Source tool for making 3D characters. Accessed 28 September 2017 http://www.makehuman.org/.

MDN web docs mozilla 2017. The WebGL API: 2D and 3D graphics for the web. Accessed 15 October 2017. https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API.

Kongregate Developer 2010. Unity WebGL Memory and Performance Optimization. Accessed 10 January 2018 http://blog.kongregate.com/unity-webgl-memory-and-performance-optimization/.

Kyaw, Aung Sithu, et al. Unity 4.x Game AI Programming, Packt Publishing, 2013. E-book. Accessed 27 November 2017 https://ez.lapinamk.fi:2856/lib/ramklibrary-ebooks/detail.action?docID=1220961, Ebrary.

Marco, T. 2016. Understanding Memory in Unity WebGL. Accessed 10 January 2018 https://blogs.unity3d.com/2016/09/20/understanding-memory-in-unity-webgl/.

Pluralsight 2015. Unity, Source 2, Unreal Engine 4, or CryENGINE - Which Game Engine Should I Choose? Accessed 28 September 2017 https://www.pluralsight.com/blog/film-games/unity-udk-cryengine-game-engine-choose.

Safdarzadeh, A. 2010. Video Game Optimization, Course Technology / Cengage Learning, Boston. E-book. Accessed 10 January 2018 https://ez.lapinamk.fi:2856/lib/ramklibrary-ebooks/reader.action?docID=3136286&ppg=24, Ebrary.

Smith, J. 2016. Practical WebGL Advice from the Field. Accessed 28 November 2017 https://www.youtube.com/watch?v=6-S-P7ekdBc&t=145s.

Schell, J. 2015. The Art of Game Design: A Book of Lenses, Second Edition. vol. Second edition. Boca Raton: CRC Press, Taylor & Francis Group.

Tekes views 2016. We got game. Accessed 12 May 2017 https://www.businessfinland.fi/globalassets/julkaisut/views_2016_korjattu.pdf.

Unity Technologies 2017a. Asset Store. Accessed 17 September 2017 https://www.assetstore.unity3d.com/en/.

Unity Technologies 2017b. Creating and Using Scripts. Unity User Manual (2017.2) Accessed 5 September 2017 https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html.

Unity Technologies 2017c. Platforms. Accessed 17 September 2017 https://unity3d.com/unity.

Unity Technologies 2017d. The profiler window. Accessed 20 December 2017 https://unity3d.com/learn/tutorials/temas/performance-optimization/profiler-window.

Unreal Engine 2017. Deep support for the platforms that matter. Accessed 5 December 2017 https://www.unrealengine.com/en-US/what-is-unreal-engine-4.

ValueCoders 2017. Unreal Engine vs Unity 3D Games Development: What to Choose? Accessed 10 December 2017 https://www.valuecoders.com/blog/technology-and-apps/unreal-engine-vs-unity-3d-games-development/.

Yusuf, B. 2017. 26 Best 3D Design/3D Modeling Software Tools (12 are Free). All3DP. Accessed 1 October 2017 https://all3dp.com/1/best-free-3d-modeling-software-3d-cad-3d-design-software/.