Academy of Economic Studies
Faculty: Cybernetics, Statistics and Economic Informatics
Specialization: ***Economic Informatics (in English)***

Practical stage project

Project theme: Making a video game

Coordinating teacher: **Student:**

Asist. Univ. Dr. Bogdan Iancu **Maria Condrea, group 1050**

Bucharest

2018

SUMMARY

Figure List

Introduction

This project is the result of my practical stage at SIVECO Romania, where I had the occasion to study thoroughly the C# programming language, work with programs such as Photoshop, Autodesk Maya, and work in the game engine Unity in the eLearning department. For me, this has been the second experience of working in the IT field after my former job as a front-end web developer.

The game I have worked on is called Zombie Toys and it's a third-person endless shooter/arcade game where the player, a young boy or girl, has awoken to find out that their toys came to life as zombies who threaten them. The aim of the game is to survive as long as possible, as the player avoids and kills the zombified toys. Points are earned by shooting the monsters using a special remote control toy, having four attacks, Lightning Attack, Frost Attack, Stink Bomb and Slime Attack. By gaining points the player might spawn an ally, which will attract the enemies giving the player a chance to take them down.

This game was based not only on coding the behaviors of the players, the enemies or the attacks, also game design was an important part of it, as I've become accustomed with UI, creating the players, learning about the power of prefabs when making a game, how important lighting is to a game and how it can change the overall effect of the game, adding physics to them etc.

It is possible to play the game on PC and Android phones.

1. SIVECO Romania SA Company Presentation

1.1General Presentation

SIVECO Romania S.A. is a company specialized in the development of IT projects, from complex to large-scale programs for education, health, agriculture, customs organizations, also European institutions, private companies and the public sector. For more than 26 years, SIVECO is ranked on the first place in the top of the local IT services market. The result of the International Data Corporation (a point reference for analyzing the local business environment) has confirmed that SIVECO has been synchronized in real time with the market requirements, proposing mature and viable solutions based on modern technology. In the year 2016, SIVECO has been the leader on the Romanian IT market, with a market share of 1.97%.

With more than 1.700 clients in 27 countries, SIVECO has managed to complete more than 3.500 projects in various domains of activity on four continents with the aid of their international offices, the company offers an optimized package and perfectly integrated software for businesses.

SIVECO Romania S.A. is a Romanian company founded in 1992, with its headquarters in Bucharest, in the Northern area, and local offices in Constanta and Brasov, with a vast domain of completed projects and it aims at creating better solutions in order to generate positive change, create prosperity and competitiveness for their customers and rivals. In the year 2016, the company had roughly 400 employees, with a turnover of more than 33 mil. Euro, with direct income from selling software products of about 46.7 mil. Euro.

Even though the company faced some backlash a few years ago, after its founder, Irina Socol, was sentenced to prison for tax evasion, SIVECO has managed to keep itself steady and continue to innovate the Romanian IT business.

Having won more than 200 prizes over the years, in the year 2005, Intel Capital and Polish Enterprise Fund V become shareholders of the company, holding more than 32%, with an investment of 12 mil. Dollars, making it the biggest international investment in Romania. Back in 2013, the

company managed to buy back the shares. Currently, the two major shareholders are the SIVECO Management Team, with 36.29% and SIVECO Netherlands B.V. with 63.71%.

With a vast experience in European projects, large-scale projects at national level for the education, agriculture and health systems, the company organized and implemented complex programs, offered training and an extensive and efficient use of resources.

Over the years, SIVECO Romania managed to secure partnerships with global leaders from around the globe and secure important certifications, like Oracle, Microsoft, Intel, system integrators like HP and Bull.

SIVECO's strategy in based on strengthening and enlarging the actual pool of partners, from global ones, multinationals that can help SIVECO expand on international markets, and local partners, which are companies with a very strong local presence that can support commercial efforts, and are SIVECO's complementary.

At SIVECO, employees are an important resource that matter a lot. Without the people behind desks and computers, the company would not be where it is today.

This is why the career path chosen is deeply respected and trained. SIVECO has a special program for career development, grouped into five components, all deeply connected within each other, starting with the Skill Matrix Application, which manages the competences of each employee structured into three sections: basic information, forming and professional experience.

The next step is formed by the individual career development plan, which is revised every two years, based on the analysis made of Skill Matrix, which contains KPI, Key Performance Indicators; The training plan, which is elaborated each year, based on the results from the app, from the individual career plan and the results of the evaluation of professional competencies.

The evaluation take place each year and its goal is to improve the employees' performance, aligned with the company's performance, it is mandatory for all employees to take part in this step.

Finally, there is team building, which consists of periodic programs aiming at improving team and competition skills, trust and ability of communication and improve the overall morale of the employees.

On the following page a few of the most important projects developed by SIVECO's eLearning department are presented, aiming at showing and proving that good skills, communication and team play can help achieve fantastic products, that make society much better:

SEI, IT-based Education System

One of the most known products of the eLearning department is SEI, which is one of the most important European programs trying to introduce IT services into the pre-university educational system, was launched back in 2001. The supplier of the SEI program is a group of companies, where SIVECO Romania provides software applications, multimedia educational content and services such as software installation and configuration, training and technical support. There result of such an application are significant, and the number speak for themselves: over 7 million people have been involved in this project either directly or indirectly, from the software developers, to the teachers, instructors and pupils that used the program. The management of the SEI educational portal, the most important educational source with more than 232.478 registered users, which reached in July 2014 more than 5.6 million unique visits.

Also, other several IT systems have been developed within the project, aiming at supporting the organization of national exams for middle school and high school students or teachers , like ADLIC program, between 2001-2014 and the Baccalaureate version, with over 2.8 million students over the past 10 years, and teacher's nomination upon vacant position, with more than half a million teachers using the program.

Interactive Encyclopedia Romania 1918: how to teach History to the digital generation

This application dedicated to the Centenary, was presented in the opening of eLSE, one of the most important eLearning conferences in Romania.

It is a unique collection of hundreds of rare images, films, 3D views of monuments commemorating the National Union or od World War I weapons, testimonies from the front and

behind the front, the application was presented at the opening of the 14th edition of the eLearning and Educational Software International Conference, held in Bucharest. On such an occasion, participants from across Europe gathered in the aula of the Central Library od the Polytechnic University to present the latest trends in educational software.

The Encyclopedia is being offered to all Romanians from all around the globe as a reminder of the most important national project, according to its authors. The digital Encyclopedia, "Romania 1918. People, moments and images" is private project, created by SIVECO Romania, in partnership with prestigious cultural institutions and personalities of the Romanian society.

There are over 370 rare images from private collections or museums (some of which have been animated), 8 vintage films, 12 3D objects, 24 interactive digital maps and graphical information, and even 2 games that simulate historical battles.

The project has been created by the volunteer work of a team of specialists from many fields, from historians, researchers, collectors, university professors, museographers, writers, bloggers, developers, graphic designers, testers, web designers, marketing people, weapons specialists and others.

1.2 Identifying a problem, an activity that can be improved

During my stay at SIVECO, I have noticed that there was not a lot of interaction between the people programming in C# and those programming in Java, even though their cubicles were close. Only when there was a problem or it was necessary for them to interact they would leave their desk. I believe that it is important to maintain a healthy relationship with the people around us and we should try to be friendlier. In addition, when our technical leader was gone, most of the team would lag and not do much, sometimes they would him and ask what to do next.

I think that it would be better for the leader to exactly plan in advance task and present them to his subordinates, making sure this way, everyone has something to accomplish in the given time, obviously taking care that the project moves accordingly to the schedule. And when it comes to socializing, I think that the creation of a common area other than the kitchen would help employees interact and get to know each other.

2. Unity 3D Presentation

Created and developed by Unity Technologies, Unity 3D is a cross-platform game creation system that comes with an integrated development environment for engineering 2D and 3D games with consistent graphics, great layouts, intuitive design and engaging game play. It is widely used for developing video games for desktops, consoles, mobile devices and websites.

The Unity engine targets the following graphics APIs: Direct3D on Windows and Xbox One; OpenGL on Linux, Mac and Windows; OpenGL ES on Android and IOS and WebGL on the web, also proprietary APIs on video game consoles.

Within 2D games, Unity allows importing sprites and an advanced 2D world rendered. For 3D games, Unity allows specification of texture compression, mipmaps and resolution settings for each platform that the game engine supports, also, it provides support for bump mapping reflection, mapping, screen space ambient occlusion, dynamic shadows using shadow maps, render-to-texture and full-screen post-processing effects. Unity also offers services to developers such as: Unity Ads, Unity Analytics, Unity Certification, Unity Cloud Build, Unity Multiplayer, Unity Performance Reporting.

Unity supports the creating on custom vertex, pixel, compute shaders and Unity's own surface shaders using Cg, a modified version of Microsoft's High-Level Shading Language.

According to data on mobile video games, 34% of top 1000 free mobile games are Made with Unity. With the integration of different social media platform SDKs, Unity supports all the relevant features of the websites with consistency.

There are plenty of reasons why Unity is a fantastic game engine, from the integration of different social media platform SDKs that support relevant features, the graphical editor makes it easy to design and layout scenes, insert objects, scripts, lights effects, overall reducing the development time significantly.

The engine has a low learning curve and it can be easily adopted by the developers seamlessly due to the great Unity documentation. The scripting is uncluttered and straightforward that brings fast iteration and execution of the programming environment. C# is the most commonly used language, also JavaScript was used, but now it's considered deprecated. It is very easy to create, debug and design simple c-routines.

It is very easy to publish a game on any platform. You simply have to select a platform; press the "Build" button, save the game and Unity builds it. Using the platform, it becomes easier to track and deal with bugs, because they are shown in the console.

Game development can be easily accelerated with the asset store. It is easy to use a vast exhibit of ready-made artworks, scripts, visual programming solutions, editor extensions and much more.

C# Programming Language

C# is a programming language stemming from C and C++, it is a Microsoft programming language developed to compete with Sun's Java language. C# is an object-oriented programming language used with XML-based Web services on the .NET platform and designed for improving productivity in the development of Web applications. C# blusters type-safety, garbage collection, simplified type declarations, versioning and scalability support, and other features that makes developing solutions easier and faster.

Created in January 1999, Dutch software engineer Anders Hejlsberg formed a team to develop C# as a complement to Microsoft's .NET framework. Initially, C# was developed as C-like Object Oriented Language (Cool). The actual name was changed to avert potential trademark issues. In January 2000, the language was released.

C# improved and updated many C and C++ features, like one of the following: it has a strict Boolean data variable type, such as bool, while the C++ bool variable types may the returned as integers or pointers to avoid common errors. The garbage collector is useful because it eliminated the concerns related to memory leaks.

C# type is safer than C++ and has safe defaults conversions only, which are implemented during compile or runtime. No implicit conversions between Booleans, enumeration members and integers other than zero may be converted into an enum type. User defined conversions must be specified as explicit or implicit, versus the C++ default of implicit conversion operators and copy constructors.

The programming language has many distinguishing features, most notably being the portability that most directly reflects the underlying Common Language Infrastructure (CLI), the easiness of writing methods and functions. In C#, programmers must use the keyword virtual to allow methods to be overridden by subclasses.

Unlike C++, C# doesn't support multiple inheritance, although a class can implement any number of interfaces. This was a design choice in order to avoid complications and simplify architectural requirements throughout CLI. When implementing multiple interfaces that contain a method with the same signature, C# allows implementing each method depending on which interface that method is called through, or, similarly with Java, allows for implementation of the method once, and have that being the only invocation on a call through any of the class's interfaces.

However, unlike Java, C# supports operator overloading and only the most commonly overloaded operators in C++ may be overloaded in C#.

Worth noting is also that in the C# programming language, global methods and variables are not supported. Methods and variables must be contained within a class or struct.

3. Making a video game

In order to create and complete the game, the workload was spread throughout 20 lessons, each encompassing various, but important steps in the inception, making and deployment of a video game. From understanding the video game industry, to writing scripts and making UIs, the challenging part was to grasp as much information as possible.

The first lesson was based on the Unity engine, on how to differentiate between the services offered by Unity, the video game industry practices, what is and how game development is an important step in assuring that a game is made smoothly.

Next, I was introduced and familiarized with the production phases of a game, in depth explanations of the Asset store, growing skills in building a game, a few basics of the game industry, its terminology and principles.

During the next few lessons, I was more familiarized with the Unity user interface, the idea of GameObject and assets, how useful they are. This way, I've learnt about distinguishing between the Hierarchy window, the editor, the views and windows, the Project view and how to reorganize the interface in order to make it easier to use.

Understanding the role of the Hierarchy, Inspector and Project window has been an important milestone in the long path of understanding Unity.

The Hierarchy window contains the list of every GameObject in the current scene. Some of them are direct instances of Asset files while others are instances of Prefabs, which are custom objects that usually make up most of a game. As more objects are added into a scene or removed from it, they will appear and disappear from the Hierarchy. It is easy to parent GameObjects into others, by making "parent"-"children" relationships.

The Inspector Window displays detailed information about the currently selected GameObject, including all attached components and their properties, and it allows you to modify the functionality if GameObjects in your scene. One can use the Inspector to view and edit properties and

settings of almost everything in the Editor, from physical game items, to Assets and Material, as well as in-Editor settings and preferences.

In this window, it is possible to manage the assets that belong to the project. The folders used in the project are all stored there, ready to be accessed at any time. One important feature is the Favorites section, where you can pin any folder you deem important in the project, making is easier to access it from any state. There is also a create menu that make is even more easy to create inside the window a new folder, Asset, Material, add a new script, or even an Animator.

It is important to keep Assets in different folder, each with its own meaning in order to maintain a sense or order in the project. It is easier this way to search for a missing item, or to organize when exporting or importing new Assets.

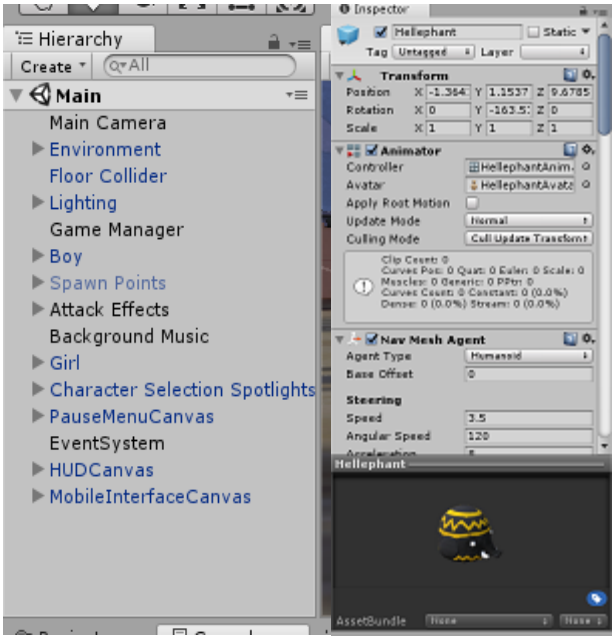

Figure 4.1 The Hierarchy Window in the finished game

Figure 4.2 The Inspector Window in the finished game

Figure 4.3 The Project Window with all the Assets of the finished game

The following lessons got more in depth with GameObjects and Assets, learning more about Prefabs and why they are more preferred to GameObjects, differentiating between the various types of GameObjects and their components, what makes them be, like Materials, their properties. Understanding GameObjects and knowing how to utilize and manage Prefabs is an important step into understating what actually builds a game.

Next, Models have been covered thoroughly, how they are stored and managed in the Project View. Creating Materials, textures was then introduced ,learning how to evaluate them based on their properties, knowing the differences between materials and effects, also, how to export or import them into the Unity Editor. Managing and optimizing textures was also touched, refining materials properties. A great deal is about shaders, mostly about the texture ones, that can pretty much change drastically the appearance of an object.

Materials, Shaders and Texture all have a close relationship between them:

Materials define how a surface should be rendered by including references to the Texture it uses, tilling information, Color tints and more. The available options for a Material depend on which Shader the Material is using.

Shaders are small scripts that contain the mathematical calculations and algorithms for calculatinf the Color of each pixel rendered, based on the lighting input and the Material configuration.

Textures are bitmap images. A Material can contain references to textures, so that the Material's Shader can use the textures while calculating the surface color of a GameObject. In addition

to basic Color (Albedo) of a GameObject's surface, Textures can also represent many other aspects of a Material's surface such as its reflectivity or roughness.

The next thing I had to do for the game was to assemble the Game Level. Zombie Toys is a continuous game that is played on a single Scene, the one of a child's bedroom. As I assembled the scene and added elements like the floor, the walls, the toys on the ground or the dressing, I have learned about the usefulness of physics in a 3D games, by adding RigidBodies to all my objects in the room. Also, I have added the Colliders that would trigger the RigidBody component.

RigidBodues enable the GameObjects to act under the control of physics. The RigidBody can receive forces and torque to make the objects move in a realistic way. Any GameObject must contain a RigidBody to be influenced by gravity, act under added forces via scripting, or interact with other objects.

Collider components define the shape of an object for the purposes of physical collisions. A Collider, which is invisible, does not need to be the exact shape as the object's mesh and in fact, it is a rough approximation of it, often more efficient and indistinguishable in gameplay. The simplest Colliders used in 3D are the Box Collider, Sphere Collider and Capsule Collider.

Following to this was introduction to Lighting in Unity. I have learned more about lighting used in gaming, how it can manipulate the gameplay and change it, how to distinguish between all the types of lighting used in games. Light tools and processes were presented, as the Sprite Editor. Next, Light Baking was presented in detail, from the light types and how to bake the lighting in my scene.

Unity makes it easy to extract elements from a composite image by providing a Sprite Editor for the purpose.

Global illumination, or 'GI', is a term used to describe a range of techniques and mathematical models which attempt to simulate the complex behaviour of light as it bounces and interacts with the world. Simulating global illumination accurately is challenging and can be computationally expensive. Because of this, games use a range of approaches to handle these calculations beforehand, rather than during gameplay.

It is easy to use Baked mode for Lights used for local ambience, rather than fully featured Lights. Unity pre-calculates the illumination from these Lights before run time, and does not include

them in any run-time lighting calculations. This means that there is no run-time overhead for baked Lights.

Unity bakes direct and indirect lighting from baked Lights into light maps (to illuminate static GameObjects) and Light Probes (to illuminate dynamic Light GameObjects). Baked Lights cannot emit specular lighting, even on dynamic GameObjects. Baked Lights do not change in response to actions taken by the player, or events, which take place in the Scene. They are mainly useful for increasing brightness in dark areas without needing to adjust all of the lighting within a Scene.

Baked Lights are also the only Light type for which dynamic GameObjects cannot cast shadows on other dynamic GameObjects.

The next chapter explained in detail Animations and their role in the game. Animator Controllers, Animation types, states, making transitions using the Animator Window.

An Animator Controller allows us to arrange and maintain a set of Animation Clips and associated Animation Transitions for a character or object. In most cases it is normal to have multiple animations and switch between them when certain game conditions occur. For example, one could switch from a walk Animation Clip to a jump Animation Clip whenever the spacebar is pressed. However even if we only have a single Animation Clip we still need to place it into an Animator Controller to use it on a GameObject.

Unity automatically creates an Animator Controller when you begin animating a GameObject using the Animation Window, or when you attach an Animation Clip to a GameObject.

During Play Mode, the Animator pans the view so that the current state being played is always visible. The Animator Controller respects the independent zoom factors of the Base Layer and Sub-State Machine, and the window pans automatically to ensure visibility of the active state or states.

The following lesson was about Scripting in Unity and it was mostly about C#. The lesson introduced simple things such as variables, modifiers, classes, several programming terms, exhaustive explanations of all loops, introduction to co-routines and Raycasts.

Raycast casts a ray, from point origin, in direction direction, of length maxDistance, against all colliders in the scene. It is sometimes needed to add a LayerMask, to filter out any Colliders that

are not important because no collisions will begenerated.

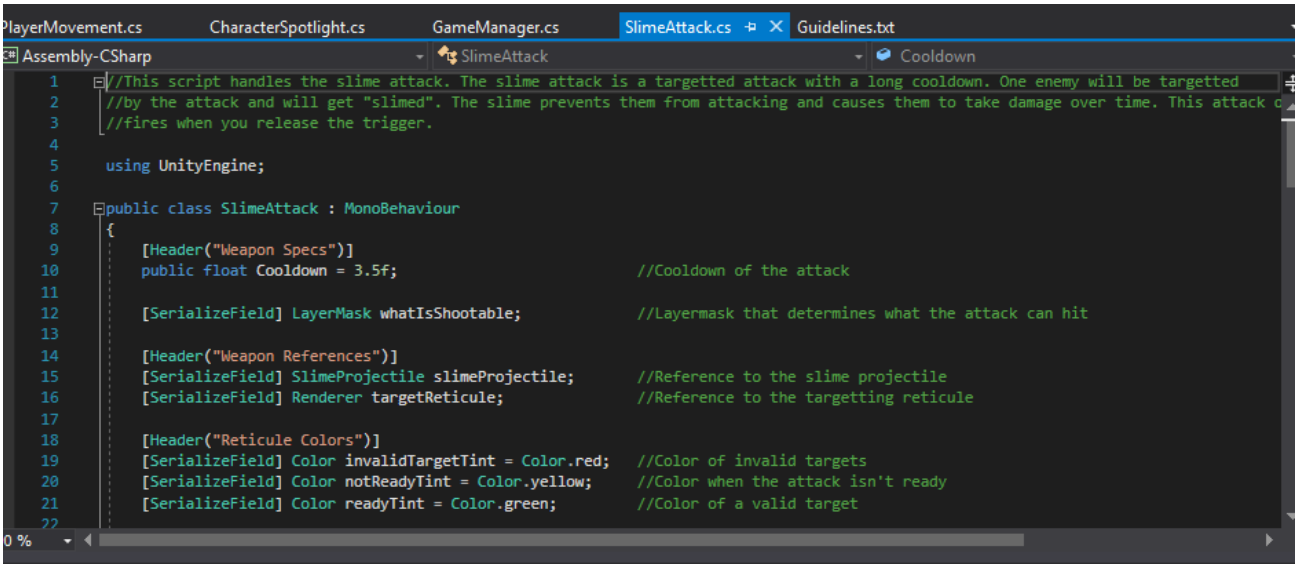

Figure 4.4 Excerpt from one of the scripts used in the game

The next chapter was dedicated to navigation and pathfinding. This way, I have learnt about NavMeshes, how to bake them, the existence of Max Slope and how obstacle avoidance works.

The process of creating a NavMesh from the level geometry is called NavMesh Baking. The process collects the Render Meshes and Terrains of all Game Objects which are marked as Navigation Static, and then processes them to create a navigation mesh that approximates the walkable surfaces of the level.

There are four steps for creating a NavMesh: select scene geometry that should affect the navigation, walkable surfaces and obstacles. Next, it is necessary to check Navigation Static on to include selected objects in the NavMesh baking process. Adjusting the bake settings to match the agent size is done, by modifying the following: Agent Radius, Agent Height, Max Slope and Step Height. Then, the scene is ready for NavMesh baking.

I then created the Allies and the Player of the game. In order to better manage the player, the scripts and effects associated with it, I have created a Game manager to add the scripts and behaviors to it, the music and effects of the Player attacks. For this game, I chose to use a Sheep as an ally, its scope is to make the Enemies follow it, in order to give some time for the Player to reorganize or kill easier the Enemies.

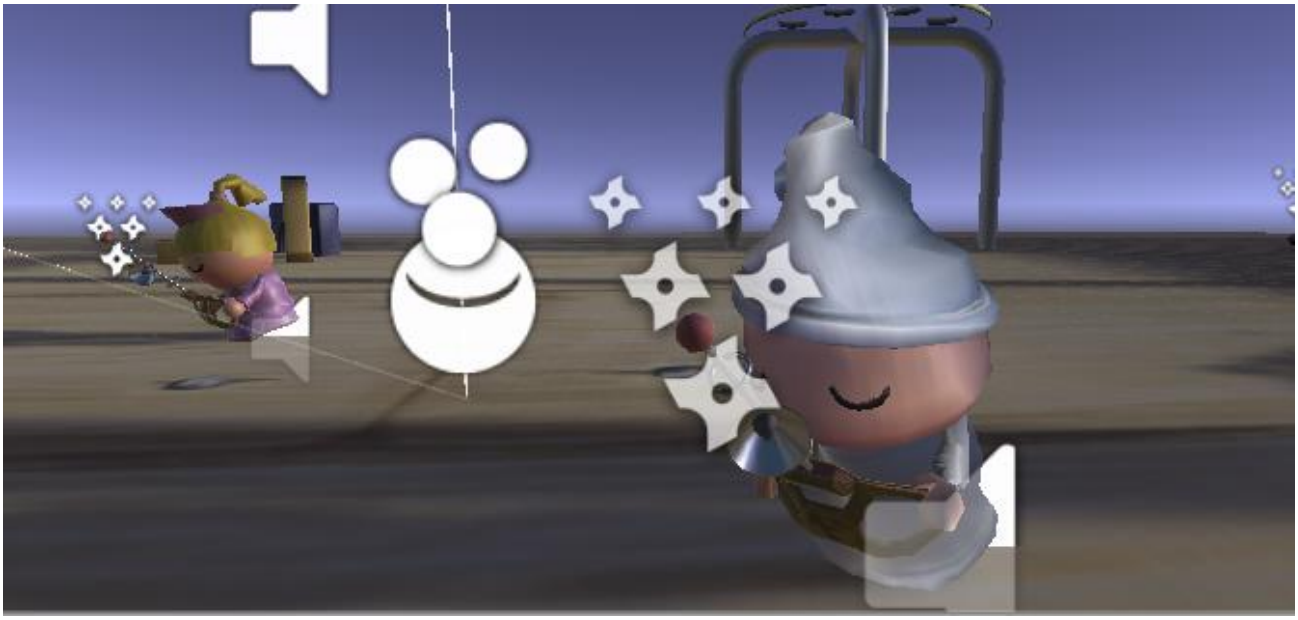

Figure 4.5 The two Players

The Enemies for this game are ZombieDuck, Zombear,Zombunny and Hellephant. They have standard attacks that can bring damage to the Player and ultimately kill him/her. In order to better manage them, I have made a parent Spawn points GameObject where the children elements where the various spawn points. I added the spawning script to the Game Manager to make sure it works!

The following lesson was about Particle Systems. In a 3D game, most characters, props and scenery elements are represented as meshes, while a 2D game uses sprites for these purposes.

Meshes and sprites are the ideal way to depict "solid" objects with a well-defined shape. There are other entities in games, however, that are fluid and intangible in nature and consequently difficult to portray using meshes or sprites. For effects like moving liquids, smoke, clouds, flames and magic spells, a different approach to graphics known as particle systems can be used to capture the inherent fluidity and energy. This section explains Unity's particle systems and what they can be used for.

Particle Systems were introduced in order to create the four attacks of the Player. It is work mentioning that these attacks are called from the Player's wand. The four attacks are Lightning Attack, Frost Attack, Stink Bomb and Slime Attack. It was a tedious work creating all the attacks and each one of them was formed by more components.

This way, the Lighting Attack was built by creating first the attack hit, then the attack emitter, the lightning bolt. For the Frost Attack, at first, I had to build a particle resembling a frost debuff, a frost attack emitter, and a frost cone effect.

For the stink bomb, I have to create a special attack reticle, followed by a reticle and a stink projectile. Finally, for the Slime Attack, I have to make a slime hit effect, its corresponding debuff, attack reticle and then integrate it all on the Player's wand. In order to control easier the attacks, I added an ally manager for the Player.

Particles systems have been a very complicated lesson from my point of view, but it was interesting seeing how much one can accomplish in such little time.

The next lesson introduced Audio, how to control audio properties, enable and examine them, how to identify audio clips and effects, list and manage them and, the most importantly, understand audio properties.

The Audio Listener acts as a microphone-like device. It receives input from any given Audio Source in the scene and plays sounds through the computer speakers. For most applications, it makes the most sense to attach the listener to the Main Camera. If an audio listener is within the boundaries of a Reverb Zone reverberation is applied to all audible sounds in the scene. Furthermore, Audio Effects can be applied to the listener and it will be applied to all audible sounds in the scene.

The Audio Source plays back an Audio Clip in the scene. The clip can be played to an audio listener or through an audio mixer. The audio source can play any type of Audio Clip and can be configured to play these as 2D, 3D, or as a mixture. The audio can be spread out between speakers and morphed between 3D and 2D.

This can be controlled over distance with falloff curves. Also, if the listener is within one or multiple Reverb Zones, reverberation is applied to the source. Individual filters can be applied to each audio source for an even richer audio experience.

There are three Rolloff modes: Logarithmic, Linear and Custom Rolloff. The Custom Rolloff can be modified by modifying the volume distance curve as described below. If you try to

modify the volume distance function when it is set to Logarithmic or Linear, the type will automatically change to Custom Rolloff.

To simulate the effects of position, Unity requires sounds to originate from Audio Sources attached to objects. An Audio Listener attached to another object, most often the main camera, then picks up the sounds emitted. Unity can then simulate the effects of a source's distance and position from the listener object and play them to the user accordingly. The relative speed of the source and listener objects can also be used to simulate the Doppler Effect for added realism.

The next step the in game creation was setting up the Camera, then making the Player selectable. Camera configuration was made in order to smoothly translate from the Player selection window to the gameplay. The Camera choices made were according to the behavior explained in the script. The player selection on PC is simple, when the game starts, you simply have to choose between the boy and the girl and click on it. It was necessary then to refine the character's behaviors for PC, adding specialized scripts.

The game could not be ready without an appropriate UI. In order to set it up, I had to learn about pivots and anchors, the RectTrasform, how to set up text, images, know button properties and be able to distinguish between all the UI components. The game's UI is a simple one, it has plastered some text advising the person playing the game to select a Player, the pause button, and in case of thecomplete mobile game, a special button for changing the attacks. In the PC version, by pressing tab the player can commute between attacks.

Figure 4.6 Game UI

The final step in my game creation was to actually build the game and run it. On PC or Mobile, the steps are the same and very easy to use.

For the mobile implementation, I had to change a few settings for the Player and create a new UI. The touch script was necessary to use in order to move the Player with my finger on the touch screen. For the Mobile interface, the Player selection screen was modified, each character being lighted using a directional light until chosen.

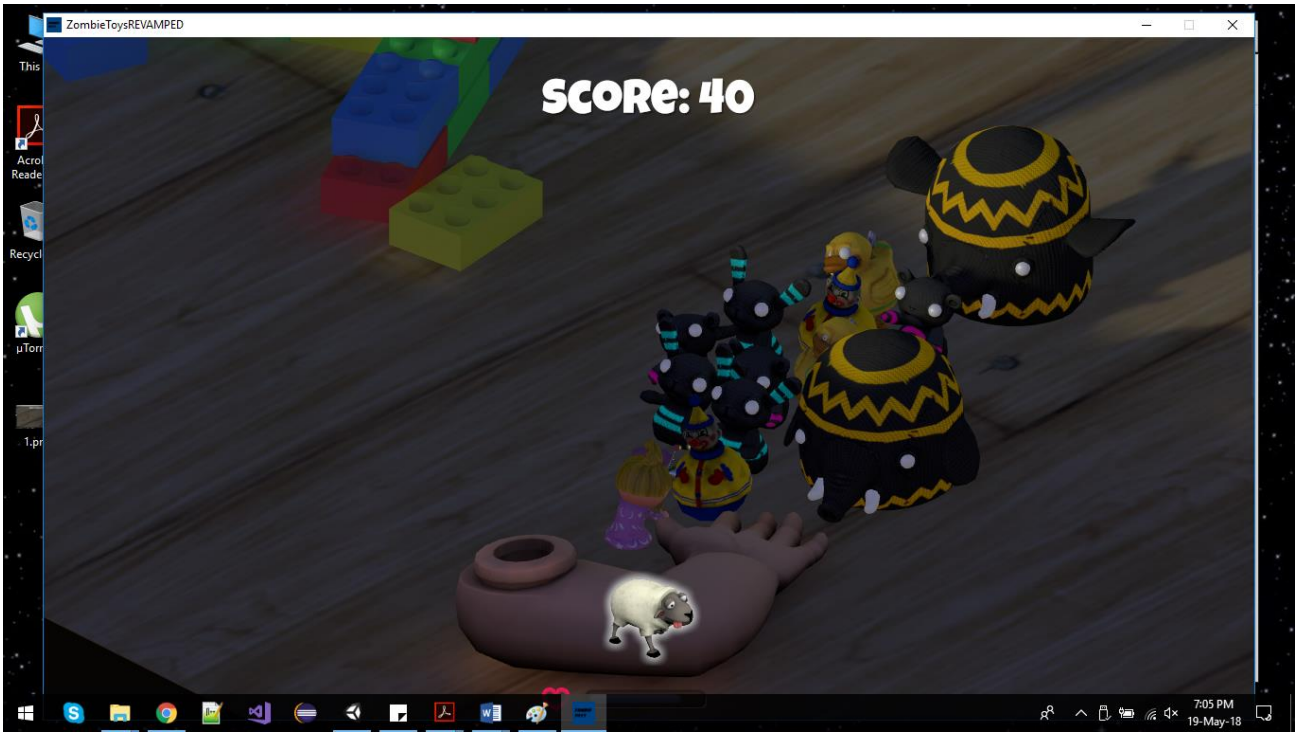

Figure 4.7 Screenshot of the game on PC

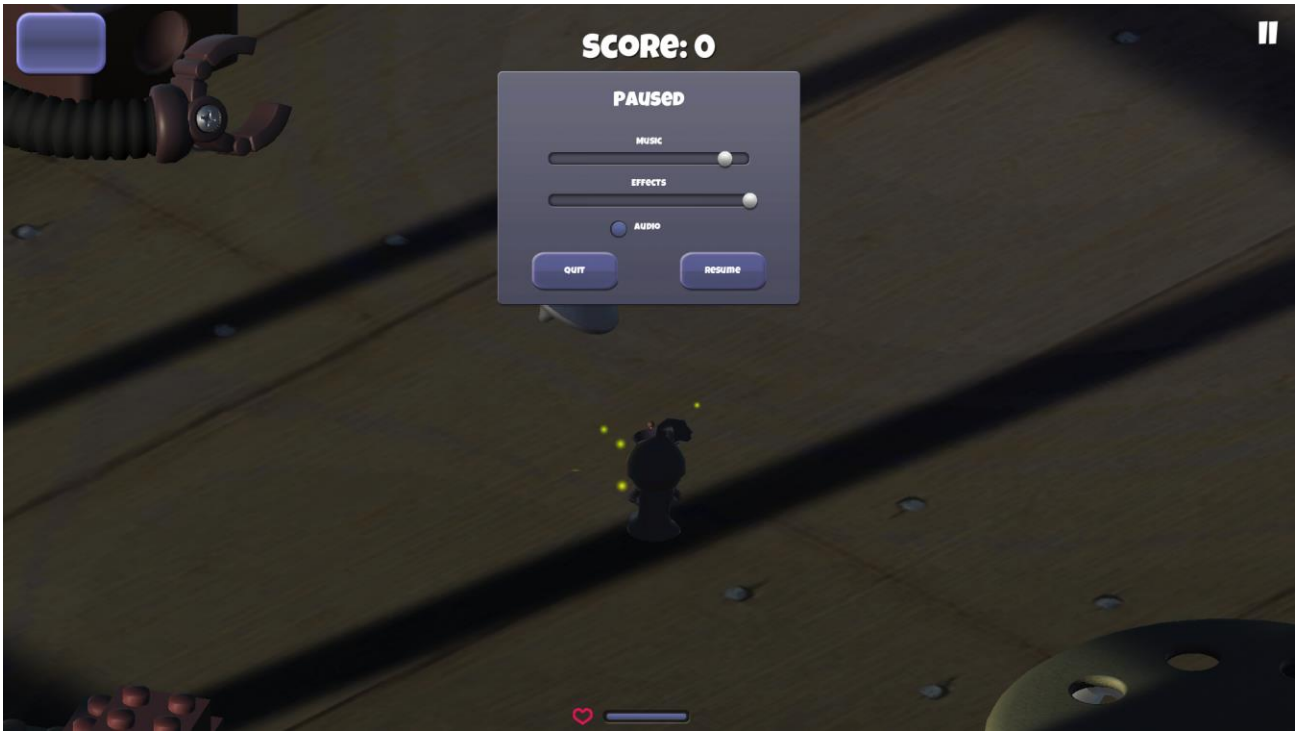

Figure 4.8 Screenshot of the game on Mobile

Conclusions

Working in Unity is accessible to anyone interested in creating a video game. At first, it may seem overwhelming for a beginner, but you can easily manage to navigate between the various windows, create GameObjects, add or modify scripts inside Unity. I had some previous experience with Unity, but I was surprised to see that there is more functionality than what one sees when opening the program. Using Unity might seem dodgy at first, but with experience, it gets better and easier to navigate between all the intricacies of the game engine.

Scripting in C# in Unity seemed easier than I had expected. Even though this semester I started studying C#, I was not sure what to expect when it comes to building a video game. There are plenty of resources available on the official Unity website, to the Internet, all explaining in detail how to code behaviors and add more functionality to a game.

Even though I had to build my game for myself, many times I found myself asking the people around me or searching on Google why do I have to do a certain thing in a certain way. Making a video game was a challenge worth completing, because I just kept asking myself a bunch of things, trying to assess myself and the knowledge gained.

One thing I really enjoyed was discussing the game with my fellow interns at SIVECO. This way, it was easier for all of us to understand the game and see where we might be wrong or where we might be hurrying. Even though the game was individual to make, collaborating with other people proved healthy for everyone.

I consider that finishing this game has been a fantastic milestone to reach. I never thought I could be able to create something like this, and I am very proud of myself and my peers who managed to build their games, as well. For the future, I would to expand the game, maybe add more functionality to it, and rebrand it even, if I can.

I received a lot of help and support from my technical leader and I am content with the result. It was a nice adventure, creating a video game from scratch and I hope to create many more in the near future

Bibliography

- https://unity3d.com/
- http://www.siveco.ro
- https://www.credencys.com/blog/14-reasons-why-developers-love-unity-3d-as-cross-platform/
- https://unity3d.com/learn/tutorials/s/scripting