Ecole Publique d'Ingénieurs en 3 ans

Internship Report 2nd Year Computer Science

DEVELOPMENT OF VIRTUAL REALITY APPLICATIONS USING UNITY 3D

25 of August 2023

Jordy Gelb,
Année Universitaire 2022/2023
Informatique / ISIA

Company Supervisor : **Michael Callaghan**
ENSICAEN Supervisor : **Régis Clouard**

www.ensicaen.fr

TABLE OF CONTENTS

Acknowledgments

I would like to express my heartfelt thanks to my internship supervisor, Michael Callaghan, a professor at the University of Ulster, for welcoming me into his laboratory, for the trust he placed in me, and for his encouragement throughout the project.

I also wish to extend my gratitude to Professor Hubert for his attention to the progress of this project, as well as to my team members: Nicolas Rousseau, Etienne Langlois, and Benjamin Maignan, for the support they provided me throughout this period.

I am also grateful to the entire teaching staff at ENSICAEN for the quality of their teachings, which have managed to captivate my curiosity and interest over the past two years.

Finally, I would like to thank all the people who assisted me in the writing of this report: my internship supervisor, Régis Clouard, and my family.

All these individuals have contributed with their availability and good spirits to make my internship enriching and motivating.

Introduction

As part of my training for the Computer Engineering degree at ENSICAEN, I completed a four-month internship at the University of Ulster in the United Kingdom, specifically at the Intelligent Systems Research Center located on the Magee Campus in Londonderry.

This institution has been working for several years on the use of new technologies in virtual reality and augmented reality for educational purposes. Different groups of students have worked to create, refine, and improve educational software in virtual reality, offering various interactive and playful experiences, such as an AI-powered discussion on the human brain or a learning game. The latest project developed is a mathematics game where players navigate through a cartoony world and solve various operations by interacting with the surrounding numbers.

This game, "Numbers & Letters," was published on Steam on October 6, 2022, and received an award at the "GALA" serious games competition the same year. My supervisor, Mr. Michael Callaghan, wanted to continue the development of this game to bring it to new platforms and add new features.

The main challenge of this internship, apart from the constraints of living in a foreign country, was learning a new tool: the Unity game engine, along with the accompanying C# programming language, and working collaboratively on a large-scale project.

Regarding the structure of this report, I will first present the context in which the internship took place. Then, I will describe the period of learning the technologies related to Unity and virtual reality. Finally, I will explain the translation, optimization, and porting work of "Numbers & Letters" to new platforms that I carried out during the second part of my internship. To conclude, I will provide an overview of the knowledge acquired during this experience.

1. Présentation of the University and Context of the Internship

1.1. Ulster University

The University of Ulster is the largest university in Northern Ireland. Established in 1984 through the merger of the New University of Ulster and Ulster Polytechnic, it now comprises four campuses located in Belfast, Coleraine, Jordanstown, and Derry. Additionally, the university offers distance learning programs in London and Birmingham.

With over 27,000 students and 1,665 academic staff [2], the university offers a wide range of courses in various fields, including computer science, engineering, social sciences, and arts. The institution upholds four core values: integrity, collaboration, inclusion, and unlocking potential.

The university is internationally recognized for the quality of its research. They emphasize this on their website, stating: "International experts have rated Ulster University among the top 25% of UK universities for world-leading research based on research power, and 72% of our research activity is considered world-leading and internationally excellent." [3].

My internship took place at the Magee Campus located in Londonderry. This campus offers a variety of academic programs in the following areas:

- Arts, Humanities, and Social Sciences
- Computer Science, Engineering, and Spatial Planning
- Life and Health Sciences
- Business School

Specifically, my research period was conducted within the "Intelligent Systems Research Centre" (ISRC) in the "School of Computing and Intelligent Systems" department. More than 90 researchers and numerous students work in this building. The center is organized into several research teams, each led by a university professor.

The mission of this center is to gain a deeper understanding of brain functioning and apply this knowledge to create models and technologies capable of solving complex problems.

1.2. Context

Our project leader, Mr. Callaghan Michael, has been working for several years on the effectiveness of learning through gaming and Virtual Reality (VR). In collaboration with Mr. Cecotti Hubert, a professor at the "California State University," and several teams of students, several learning games on various topics have been published on Steam. Among these, we can mention "Electronic Circuit Simulator," a learning game on the subject of electronic circuits, and "Medical Imaging VR," an application that allows the consultation of medical imaging scans in VR.

In 2022, the VR video game "Numbers & Letters" was published on Steam and won the first prize at the GALA serious games competition [4]. It is a mathematics game designed to improve the player's mental calculation skills. To achieve this, the player must solve equations using the numbers floating in the scene. By combining these numbers with the available operators, the user can solve exercises of varying complexity.

Figure 1: Finalist Logo of the GALA 2022 Competition

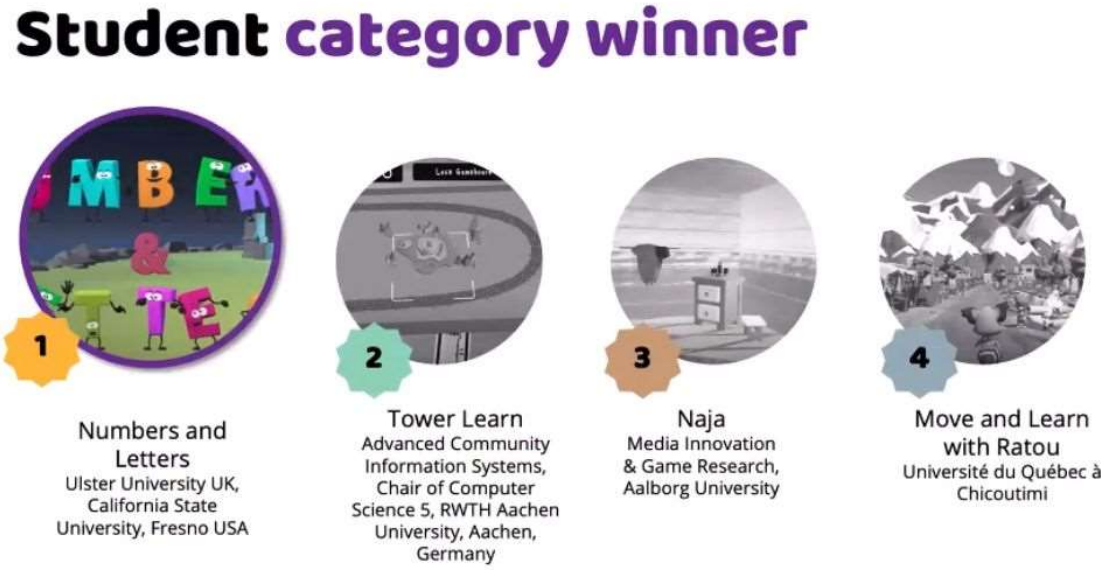


Figure 2: Montage showcasing the winning games of the student category at GALA 2022.

Two new game modes were introduced as a result of frequent game updates. The first one is the "Inverted" mode, where the player is given an equation and must find the result among the numbers present in the scene. The second mode is the "Letter" mode, where the objective is to find the missing letter of a word to earn points.

Mr. Callaghan's wish for this research period was to make the game accessible to as many people as possible and to add new game modes. This work was divided among the various

groups of French students who participated in the development of the application this year. I primarily worked on the first part, which involved making the game cross-platform and translating the interface into several languages.

1.3. Tools and Working Method

To carry out this project, several tools were imposed on me.

Firstly, the development tools:

- Unity game engine with its C# programming language. This was a constraint imposed by our supervisor, as the first part of the game was developed using Unity.
- Visual Studio 2022 integrated development environment (IDE), which is the default IDE for Unity. The most significant advantage I noticed with this IDE is its Unity extension, which greatly facilitates programming through an intelligent auto-completion system and accessible documentation from within the software.
- Plastic SCM version control software. This allowed us to retrieve the project as it was left by our predecessors and continue its development. Unity acquired Plastic SCM in 2020, and it greatly facilitates collaborative work on large-scale projects with its powerful conflict management system and user-friendly interface.
- During the research period, we used the HTC Vive and the Oculus Quest 2 as the VR headsets. The HTC Vive is a wired headset accompanied by two external stations that enable real-time body tracking. This setup allows the user to play more powerful games, as all the calculations are performed by the computer. In contrast, the Quest 2 is a

Figure 4: Photo of the HTC Vive

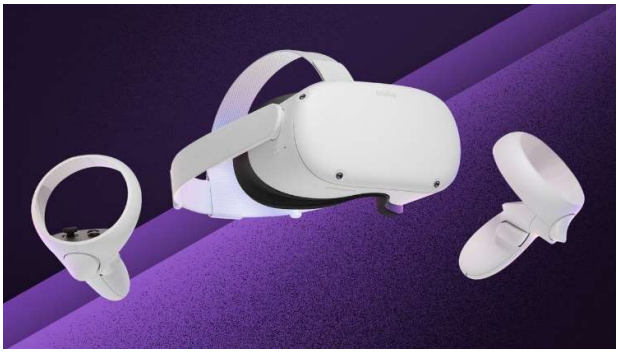


Figure 3: Photo of the Meta Quest 2

standalone headset with an integrated processor, enabling the user to play games without the need for a wired connection to another device. This grants the user greater

freedom of movement but results in a trade-off between display quality and gaming power compared to the HTC Vive.

Next, the communication tools:

- Slack communication platform: This platform allowed us to maintain a thread of discussion among the different project members and to exchange with our supervisor throughout our internship.

- Zoom video conferencing software: We used Zoom for periodic voice calls to discuss the project's progress in real-time.

- Trello project management tool: The interactive online boards provided by Trello were very useful at the beginning of the project. They helped us organize our learning process with the software and stay updated on each person's progress through the tutorials provided by our supervisor.

Figure 5: Screenshot of the Unity board on Trello.

As for the working method employed, it resembled the Agile methodology. The development team participated in two Zoom meetings per week with our supervisor and Mr. Hubert Cecotti. During these meetings, we discussed the project's progress and set the objectives to be achieved before the next meeting. The work was divided into short sprints of two to three days, focusing on implementing small, specific features.

2. Beginning of the internship

The first month of the internship was mainly dedicated to learning the Unity game engine, creating our first VR games, and familiarizing ourselves with the laboratory and the various project teams within it.

2.1. Learning Unity

During the first two weeks of my internship, my primary task was to learn the Unity game engine. Several tutorials were made available to us on the project team's Trello platform, allowing our supervisor to track our progress.

Firstly, I completed the official Unity tutorial called "Create with code"[5] , which lasted approximately 41 hours. This tutorial taught me the basics of the software through several guided mini-projects and fun quizzes. A more detailed description of each of these projects can be found in the appendix.

Next, I followed a 10-hour video by the YouTuber Code Monkey [6] , which was recommended by my supervisor. The video focused on creating a cooking game and allowed me to reinforce the knowledge I had acquired earlier while introducing me to different key aspects of managing a Unity project. During this tutorial, I learned to adhere to strict rules regarding programming style, which made my work more efficient in the later stages. You can find my version of the game on my Itch.io page.

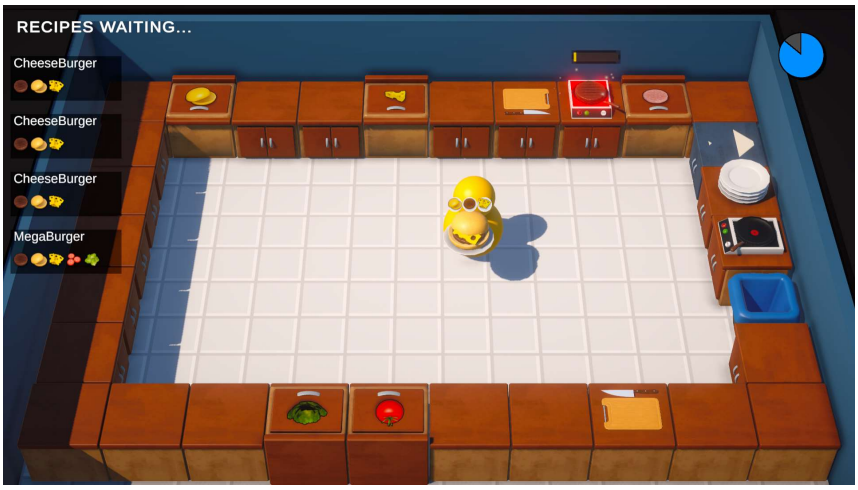


Figure 6: Screenshot of the game GetCoocked

Finally, I followed numerous tutorials describing software architecture concepts applied to the Unity game engine. The principles that I used the most are :

- Using events to invert dependencies between scripts.
- Utilizing "ScriptableObjects" to significantly reduce dependencies between my classes and make my project open to extensions. You will find an explanation supported by a diagram showing an implementation of ScriptableObjects in the appendix , as well as an official Unity video describing the qualities of this component in my references [7].
- Implementing singletons (e.g., game manager, input manager) to facilitate access to an object in the scene and make it unique. (Appendix : Singleton in Unity)
- Segregating the logic and visual parts of my components to keep my code simple and easily readable.

Finally, at the request of my supervisor, I worked in a team with Nicolas Rousseau, Etienne Langlois, and Benjamin Maignan to create the game "Number And Gun" during the third week of the internship. It is a first-person game where the goal is to score as many points as possible by shooting characters moving in the scene. The player has three minutes to beat their record while avoiding attacks from multiple robots.

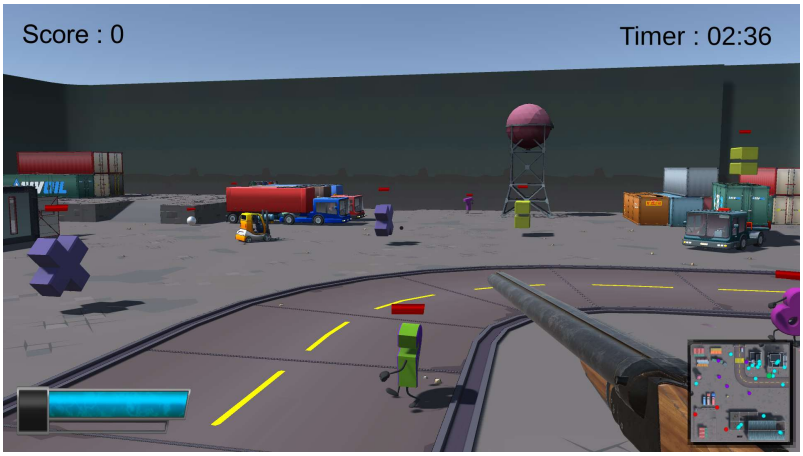


Figure 7: Screenshot of Number & Gun

This first team project allowed us to learn how to use the version control software Plastic SCM and, most importantly, to come to an agreement on essential aspects for the smooth progress of our future projects. We realized there were some issues related to divergent programming practices among certain members of the group. Therefore, we took the opportunity to align our knowledge gained during the first weeks of learning and plan the execution of our future projects.

2.2. First VR Game : VRArchery

Access to the premises was restricted during the first three weeks, so I was able to start working on virtual reality from my fourth week of the internship. The final task assigned by my supervisor to conclude this first part of Unity training was the creation of my first VR game. The only requirement was that it should use the assets present in "Numbers & Letters." Thus, with the same team as during the creation of "Number & Gun," we developed the game "VRArchery."

VRArchery is a VR archery simulator where the user appears in a scene with a bow and arrows nearby. The goal is to shoot at numbers present around the user to earn points. The larger the number, the more difficult it is to hit, and the more points it yields.

For this game, I primarily worked on implementing user interactions with the elements of the scene. This is where I learned about the functionality of Unity's XR Interaction Toolkit.

The XR Interaction Toolkit is an overlay offered by Unity that allows the implementation of basic VR interactions between the player and the scene components using simple components. This package consists of a set of classes of the "Interactor" and "Interactable" types, which interact with each other, along with an interaction manager that supervises all operations between these two types of objects. There are different types of "Interactors" that simulate various behaviors commonly observed in VR applications, such as the "Gaze Interactor" (using gaze to interact with the scene), the "Simple Interactor" (distance-based interaction with arms), and the "Laser Interactor" (a laser pointer extending from the hands for remote interactions).

We implemented a system for our game using the "Simple" and "Laser Interactor" to allow the player to retrieve arrows placed on a table at a distance (laser), manually notch them to the bow (simple), and draw/release the string to launch the projectile.

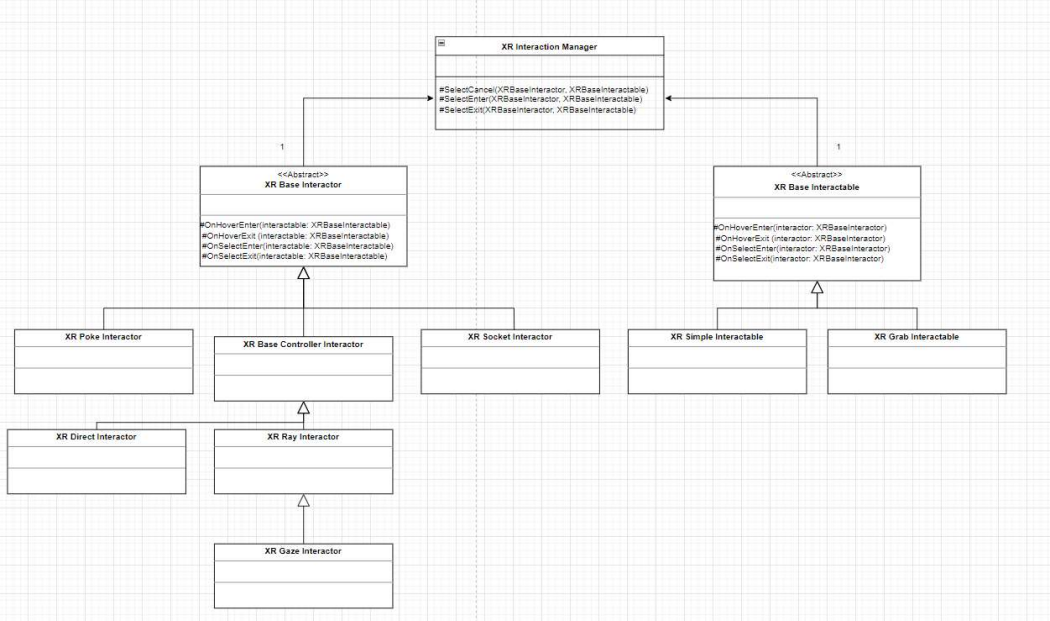


Figure 8: UML diagram of the basic classes of the XR Interaction Toolkit.

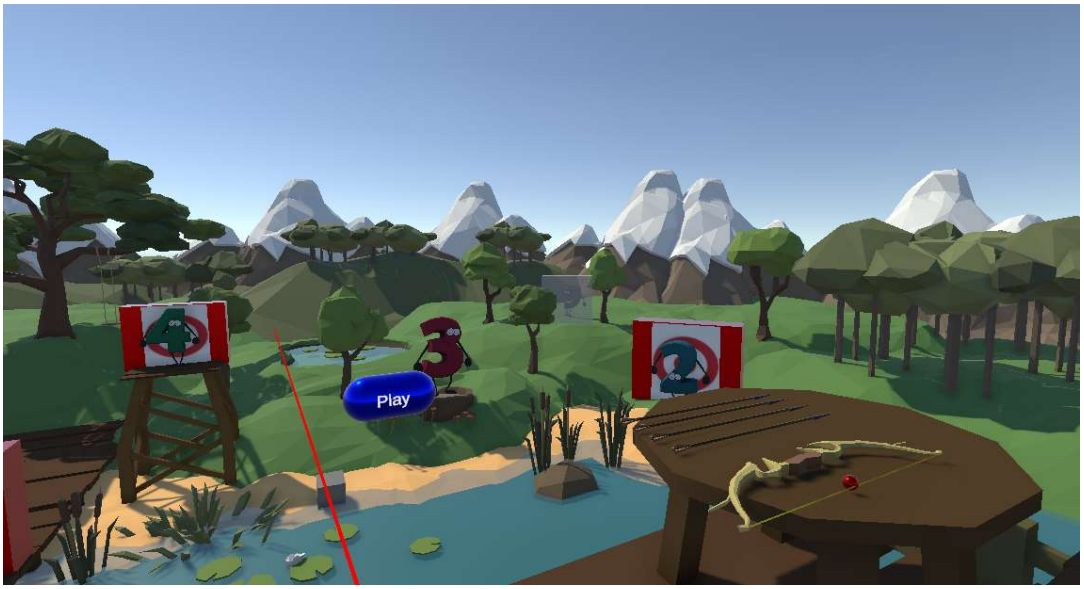


Figure 9: Screenshot of the game VR Archery

2.3. Experience with the Implementation of an Interface Between the Brain and Virtual Members

During this first month of my internship, I also had the opportunity to get in touch with one of the researchers working at the ISRC's Space Innovation and Neurotechnology Lab: Mr. Niall McShane. I participated in a scientific experiment aimed at training a neural network to recognize the signals sent by my brain to perform a movement.

The experiment is divided into 10 training sessions spaced at least one day apart. During each session, the goal is to reach targets, first physically using my arm, and then mentally, by imagining the movement. The brainwaves produced by my brain are collected using sensors present on the "g.Nautilus RESEARCH 32" headset. Each session is preceded by a hardware setup phase during which the subject responds to anonymous questionnaires. The headset is then placed on the subject's head, and gel is applied through the pores of the headset to facilitate the transfer of signals from the head to the device.

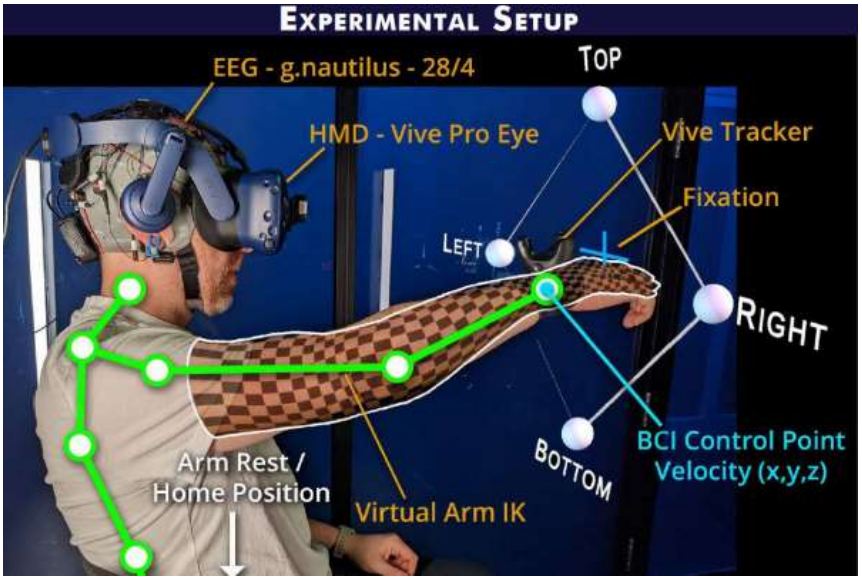


Figure 10: Explanatory diagram of the experimental setup

Half of the sessions were conducted using a VR headset, while the other half used a two-dimensional screen.

During this period, I had the opportunity to interact with Mr. Niall McShane. This experience allowed me to explore the field of research related to brainwave analysis. I also observed the scientific protocols required to conduct tests and extract results.

3. Porting the Game to Quest

The main objective of this second part of my internship is to make the application "Numbers & Letters" accessible to a wider audience, especially by deploying it on the Oculus Quest Store. To achieve this, I had to modify and adapt the existing game code to make it multi-platform, perform well on both Windows and Android, and support multiple languages.

3.1. Multi-plateform Application

To be able to launch the game on the Oculus Quest 2 running on an Android OS, I had to first change the API used to communicate with the VR hardware. The SteamVR plugin, which was used for handling interactions with the environment, had some bugs when used on the Quest 2.

Therefore, I migrated the game controls to Unity's XR Interaction Toolkit. This toolkit has the significant advantage of supporting the vast majority of VR headsets currently on the market. It also facilitates controlling different types of headsets based on the platform on which the game is compiled. As a result, I allowed all available headsets from the API for builds on the Windows platform, and I only allowed Oculus headsets for Android builds. This significant code change allowed me to maintain a single project for both targeted platforms, making it easier to maintain and add new features.

Next, I worked on the parts of the code directly related to the platform on which the game is executed. The three main concepts I focused on are the achievement system, the leaderboard system, and file access.

Since the code was not written with the possibility of new platforms being implemented, I had to modify many classes to reduce their dependencies on the Steam platform. I ensured that the parts of the code related to using a platform were isolated and encapsulated in generalized functions. This way, it is easy to add new platforms to the game without modifying the code of those already in place. I also took the opportunity to enclose platform-specific parts of the code in compilation directives. This modification helps avoid having code compiled for a platform other than the one intended.

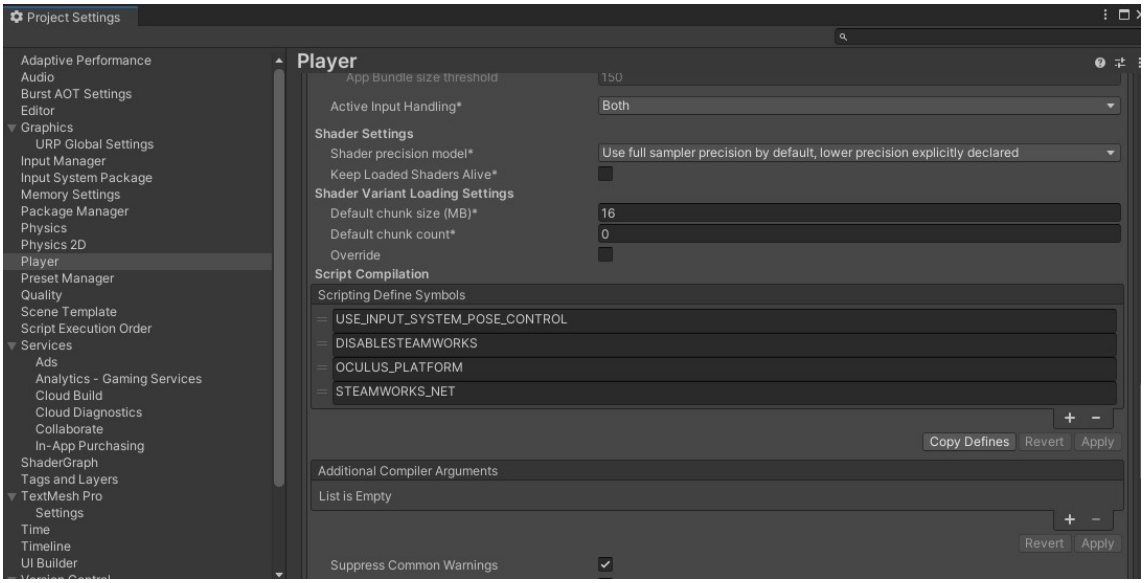


Figure 11: Screenshot of the compilation directives in the Android project

3.2. Game optimisation

One of the biggest challenges I faced during the porting of the game from SteamVR to the Oculus Quest 2 was the optimization issues of the application. Indeed, the game was not originally developed with the performance constraints of the device in mind. The Oculus Quest 2 uses the "Android" operating system and is capable of running games locally without the need for a connection to a computer. During the initial tests, the game suffered from significant FPS drops, which I addressed through several steps of optimization.

3.2.1. Modification of the Render Pipeline

The first action I took to improve the game's performance was to change the Render Pipeline from the existing "Built-in Render Pipeline" to the "Universal Render Pipeline" (URP). This new version, regularly updated, is more efficient and utilizes modern shader-based rendering techniques to achieve high performance while preserving visual quality. It also offers greater flexibility in customizing the rendering pipeline, allowing me to disable visually costly effects for the GPU. One of my initial tasks in this regard was to replace the water plane on the main scene, which was moving a large number of pixels on the screen to simulate a wave effect, with an optimized shader using a normal map*. The final effect was very similar, and I observed a significant improvement in the overall stability of FPS.

Figure 12 Screenshot of the visual change in the water plane (on the left, the old version, on the right, the new version)

3.2.2. Optimisation CPU

The second action I undertook was to study Unity's documentation to better understand the workings of the game engine and the reasons that could cause these performance slowdowns. I came across an excellent guide [8] that explains the various reasons that can lead to such significant performance loss. In the appendix, you will find a diagram describing the process I had to go through to identify the origin of these issues.

I was able to identify, with the help of Unity's analysis tools, a significant CPU overload (Render Thread Bound) caused by a large number of consecutive calls to the GPU, also known as "Draw calls." A Draw call is a call to Unity's graphics API made to display shapes on the screen. Preparing a draw call creates a considerable load on the CPU. According to Meta's documentation, it is recommended not to exceed 600 draw calls to maintain a stable FPS count. Some scenes of the game had over 1000 draw calls, mainly due to the diversity of the environment and the simultaneous appearance of many elements on the camera.

To address this optimization issue, I decided to combine the meshes* of my objects that make up the environment using a tool available on the Unity Assets Store. This action reduced the number of draw calls by two-thirds for the most costly scenes and by an average of one-third for the others. Additionally, I configured the occlusion culling system for the scene components. When objects in the environment go out of the user's field of view, they disappear and are no longer counted among the objects transmitted to the GPU for scene rendering.

The two mentioned techniques conflicted with each other since the now merged objects into a single mesh could not be separated to work with occlusion culling. To resolve this, I grouped the objects in the environment based on their position relative to the camera. This way, I could make parts of the scene disappear based on the player's orientation while maintaining an acceptable number of draw calls.

This change had a significant impact on the game's performance, making it stable on the majority of its scenes. The only negative effect observed was an increase in the size of the Unity project due to the creation of new Meshes, some of which could be large. However, this modification does not affect the size of the final build, as Unity removes all unused Meshes from the final application.

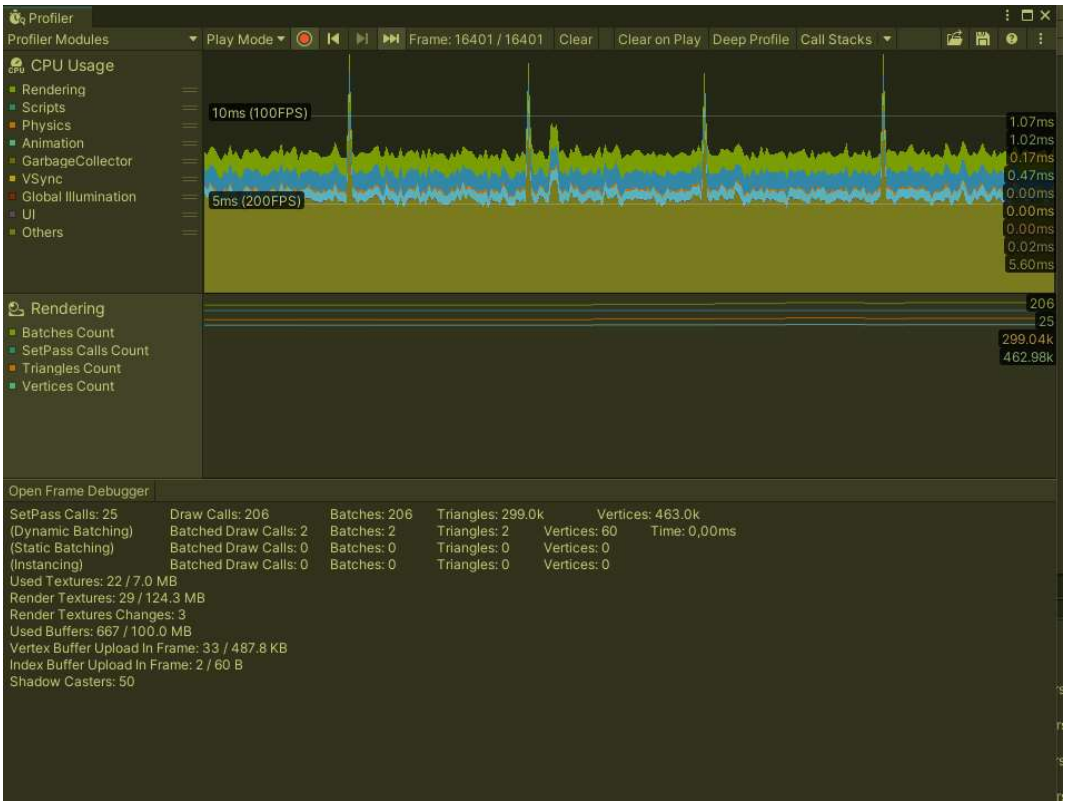


Figure 13 : Screenshot of Unity Profiler in Editor mode on the "Beach 180" scene of "Numbers & Letters"

3.2.1. Optimisation GPU

The third action taken to improve the game's performance was reducing the number of polygons displayed on the screen. Some camera angles that showed too many objects at once could cause significant drops in performance.

This problem arose from GPU overload (GPU bound) caused by the excessive complexity of certain elements in the environment for the Quest 2. Meta's documentation recommends not displaying more than 1 million triangles per viewport. To address this issue, I used the « UnityMeshSimplifier » plugin by Whinarn, which allowed me to simplify a mesh. This operation inevitably leads to a degradation of the mesh quality, but since the modified elements are often at a long distance from the player and in large quantities in a scene, the difference is not noticeable. This method helped me reduce the number of polygons displayed in the scene and ultimately solved my remaining performance issues..

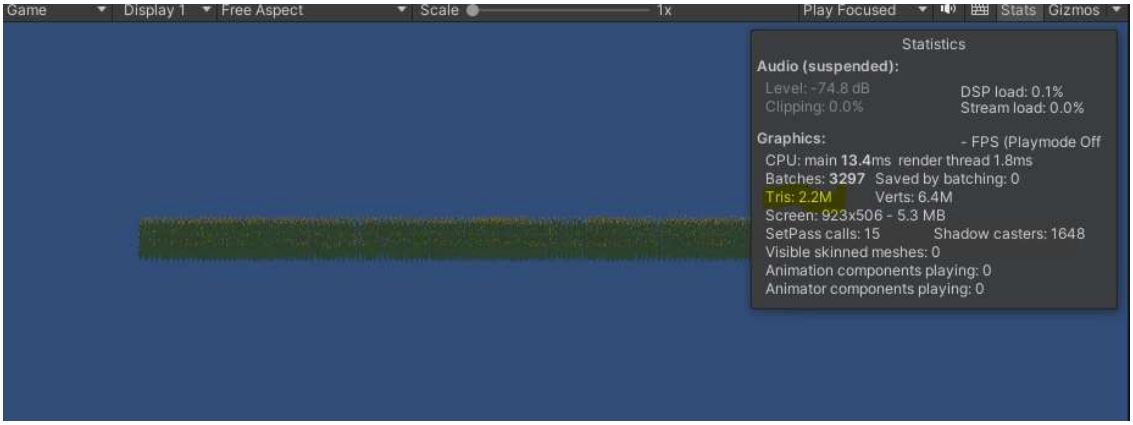


Figure 14: Screenshot of the cornfield before optimization.

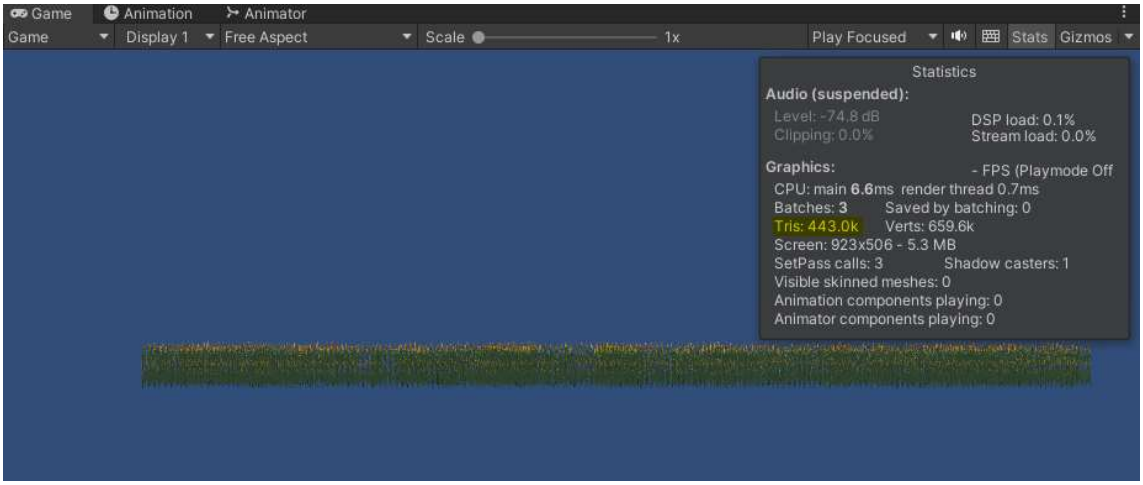


Figure 15: Screenshot of the cornfield after optimization.

3.2.2. Compilation optimization

The game now running at a stable 60 FPS, I was able to address another aspect of optimization, which is the compilation of the project. The game was facing heavy compilation issues under Android due to excessive usage of computer RAM (> 20 GB).

Upon diagnosing the problem, I discovered a C# file containing a list of over ten thousand words used to generate the words to be found in the "Letter" mode. During compilation, this array was loaded into memory, causing a significant increase in RAM usage and leading to crashes during compilation, or in extreme cases, crashing the entire computer. I resolved this issue by moving this word list to a ".txt" file, reducing the compilation time and allowing users to customize the word list for the "Letter" mode.

Another aspect of compilation that I addressed was the shaders. Initially, shader compilation could take from 40 minutes up to 1 hour and 30 minutes. However, subsequent compilations took only around ten minutes. After studying Unity's documentation, I decided to enable "Stripping" of shaders in the project, allowing the compiler to ignore those that are not used in our game. Along with some modifications made possible by the "Universal Render Pipeline," I managed to reduce the compilation time to around ten minutes for the first build and less than 3 minutes for subsequent builds, saving valuable time for developers testing the game on the Quest..

All these optimization changes allowed me to maintain a single project that can be developed simultaneously on both the Steam and Oculus Quest platforms without major changes to the environment or visual effects.

3.3. Localisation of Numbers & Letters

During the final stage of my internship, in addition to handling the integration and optimization of other parallel projects, I was tasked with translating the game "Numbers & Letters" into five languages: French, Italian, Spanish, German, and Portuguese. To achieve this, I developed a suite of user-friendly and easily extensible components for the developer.

The initial idea was to make the application translatable by anyone, so the translations needed to be stored in a JSON file that could be easily modified by the user. I first created the "localizationData.json" file, which contains a list of supported languages at the top and an array associating a key to all available translations for a set of words.

Next, I created a static class that serves as an interface between other components and the JSON file. This class also has a C# event that notifies subscribed classes of any changes in the interface language.

Finally, the last script is a Unity component that is attached to an object containing a text field. Upon application startup, the script automatically subscribes to the language change event and requests the translation for its key from the translation interface. Several "SetText" functions were created in this class to facilitate word changes during gameplay. These functions accept different types of parameters, such as integers, floating-point numbers, or strings, and retrieve the translation data to replace the changing elements inside the game (such as a timer, score display, etc.). These changing elements are identified by a 0 in the translation fields of the JSON file.

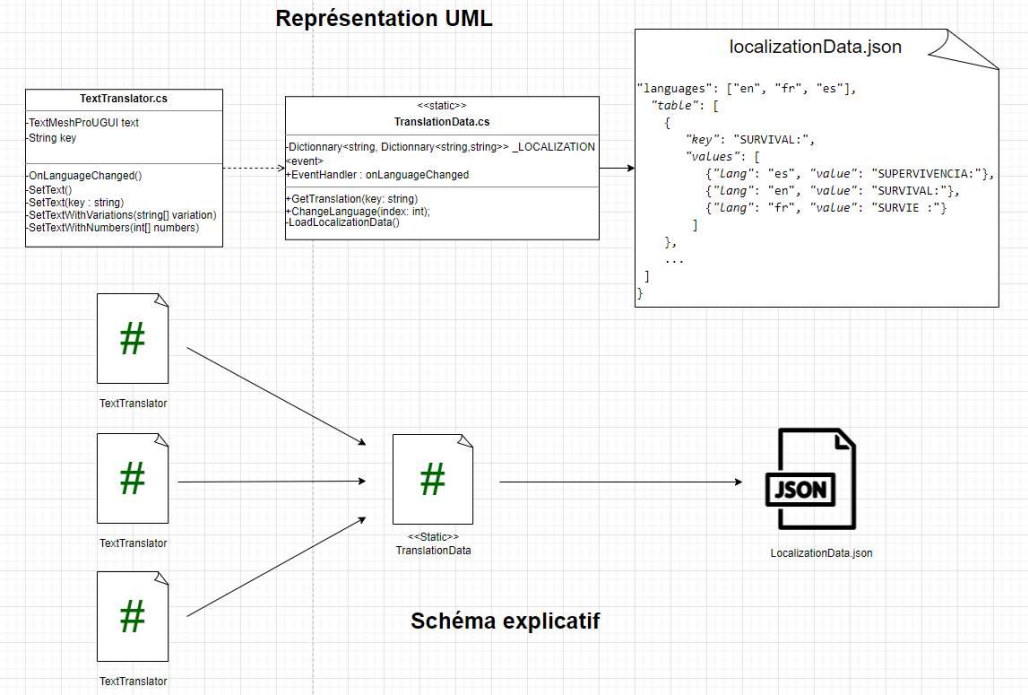


Figure 16: UML diagram and explanation of the implementation of localization in "Numbers & Letter."

If the developer wants to make a new field translatable, they just need to add the "TextTranslator.cs" component to the object that has the text field, with the translation key inside. If a user wants to add a new language to the game, they just need to add it to the list of supported languages in the JSON file, and then add their translations to the existing ones.

Figure 17: Screenshot of the "Number" game menu translated into Spanish

Conclusion

These 16 weeks at ISRC in Derry allowed me to discover the world of VR and video game development. This internship was highly rewarding in terms of practical experience, learning, applying theoretical knowledge, and also on a personal level.

My mission to make the game "Numbers & Letters" accessible to a wider audience was a real challenge. Having no prior knowledge of multi-platform development or the Unity game engine, I invested a lot of time and effort to understand the workings of these technologies and deliver quality work.

Learning new optimization practices, software design, and project management opened up new horizons for me. I plan to deepen these skills to gain a better overview of my future projects and anticipate potential difficulties.

Additionally, continuously practicing English during this internship allowed me to enrich my vocabulary and gain confidence in oral communication.

I appreciated the opportunity to work directly in the laboratory with state-of-the-art equipment. My interest in VR technologies for gaming has grown, and during my double degree program in Canada, I plan to invest in VR equipment to continue developing applications while closely following advancements in this field.

Regarding this, my supervisor, Mr. Michael Callaghan, recommended various references to deepen my knowledge. I am already looking forward to studying this information during the remainder of the summer.

In conclusion, this internship has been highly formative and motivating for my future studies. It confirmed my career plan in the video game industry, which I will pursue in Chicoutimi starting in September.

GLOSSARY

AR : Augmented Reality (AR) is a technology that superimposes virtual elements such as images, videos, holograms, etc., onto the real world, creating an interactive and immersive experience.

VR : Virtual Reality (VR) is a technology that creates an immersive virtual environment, replacing the user's perception with the simulated one provided by devices such as headsets and controllers.

Steam : An online platform developed by Valve for the distribution of video games and digital content.

Unity : It is a video game development platform that allows creators to design, build, and deploy interactive experiences on various platforms. It is a powerful and popular tool in the gaming industry.

Asset : In Unity, an asset refers to any content element used in game development, such as 3D models, textures, sounds, scripts, and more.

FPS : The FPS or "Frames Per Second" is an indicator that refers to the number of images displayed on the screen per second. This value is used to evaluate the graphic performance and smoothness of an application. The higher the FPS value, the smoother the movement on the screen.

Mesh : A mesh is a structure composed of points connected by edges to form a geometric representation of a 3D object or surface.

Normal map : A normal map is a special texture used to create the illusion of bumps and roughness on 3D objects without having to modify its mesh.

Occlusion oculling : Occlusion culling is a rendering technique that aims to not draw objects that are hidden or occluded by other elements, thereby optimizing performance.

APPENDICES

TABLE DES FIGURES

APPENDICES

Appendice 1 : Projects completed through the "Create with code" learning program.

Each of the following projects consists of 3 phases. Firstly, the learning phase on a project provided by the tutorial. Here, we have to follow the presentation videos to achieve the expected result. Then comes the programming challenge phase, where we have to correct errors in another project related to what we have learned previously. Finally, the tutorial's bonus phase offers several features to implement, ranging from easy to expert difficulty.

1.1. Project Unit 1 : Player Control

This first tutorial taught me how to use the basics of the input system in Unity. During the learning phase, I created a small car game where the user drives on a highway and collides with crates on the road.

The programming challenge associated with this tutorial is a flight simulator that reuses, with some differences, the input system used for the car game.

As for the bonus features offered by the tutorial, I implemented the following in order of difficulty:

- Adding more obstacles in different shapes (pyramids, columns, etc.)
- Adding an oncoming vehicle (to create a high-speed collision with the player)
- Adding a mode to change the camera position (rear view, front view)
- Local multiplayer mode with split-screen.

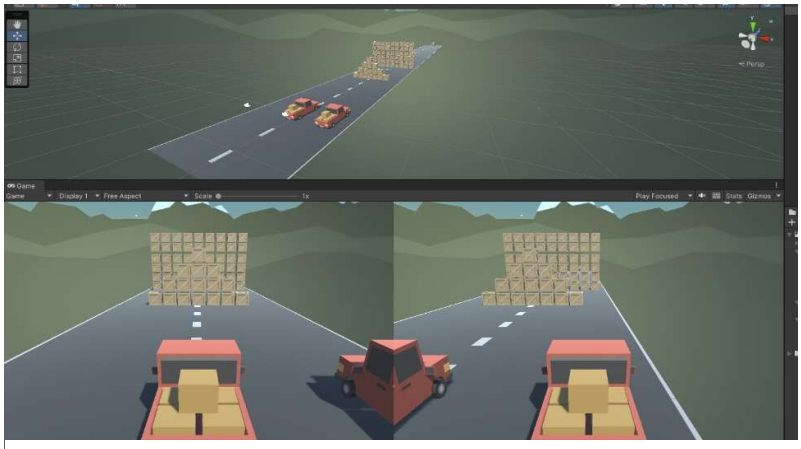


Figure 18: Screenshot of the main project in Unit 1

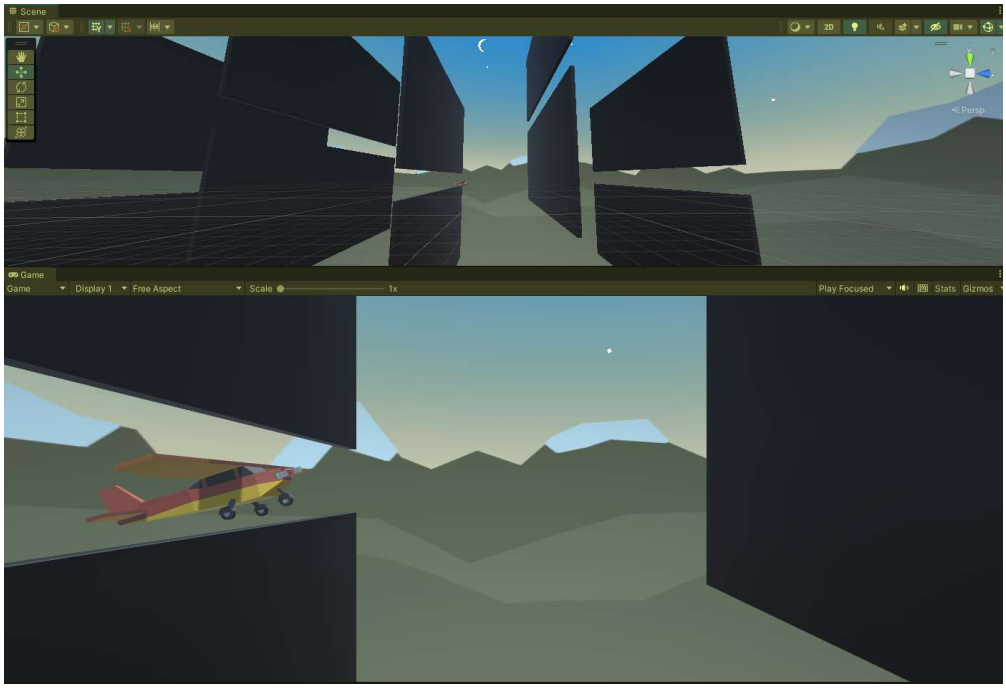


Figure 19: Screenshot of the challenge in the Unit 1

1.2. Project Unit 2 : Basic Gameplay

This second tutorial taught me how to implement my first game mechanics in Unity. During the learning phase, I created a small top-down shooting game where the objective is to feed dogs coming from all directions with bone-shaped projectiles while avoiding getting hit by the animals.

The programming challenge associated with this tutorial is a precision game where you have to time the launch of a dog to catch a ball falling from the sky.

As for the bonus features proposed by the tutorial, I implemented the following ones in order of difficulty:

- Diversification of player movements (up, down, left, right)
- Animals appearing from all sides and causing the game to end if they touch the player
- Adding a field with the player's number of lives and a message in case of game over
- Adding a health bar above the animals.

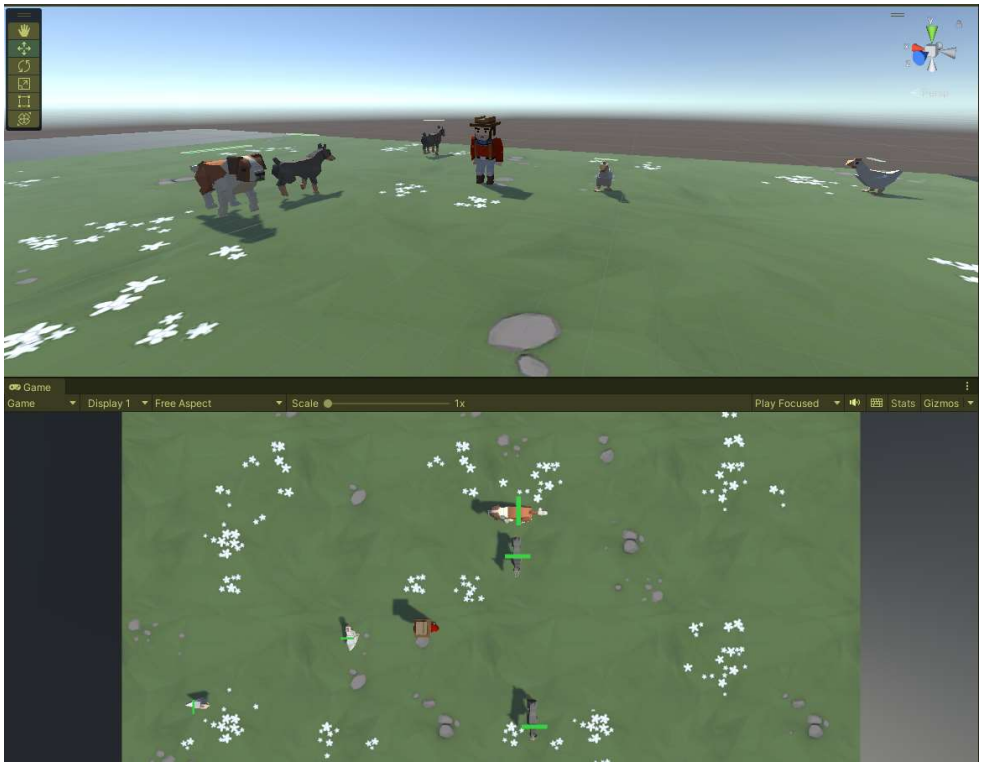


Figure 20: Screenshot of the main project in Unit 2

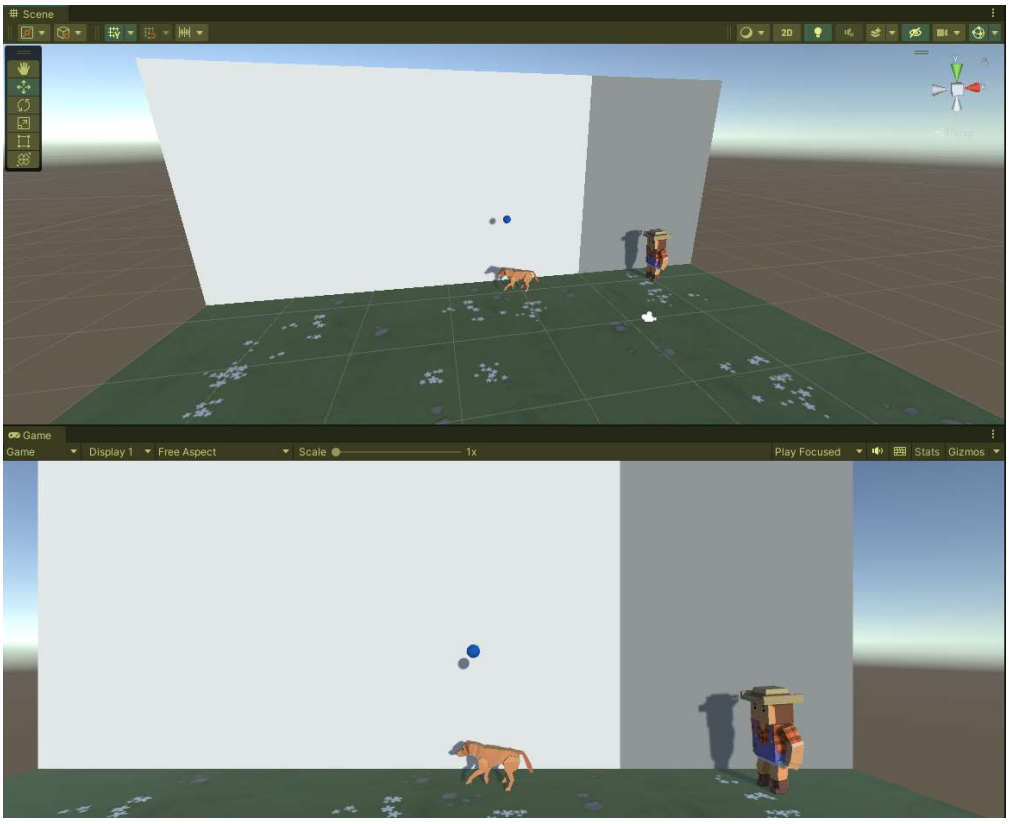


Figure 21: Screenshot of the challenge in the Unit 2

1.3. Project Unit 3 : Sound and Effects

This third tutorial taught me how sounds, animations, and particle effects work in Unity. During the learning phase, I created a 2D front-view racing game where the goal is to avoid obstacles appearing on the road. The game has its own music, a scrolling background that repeats in a loop, and sound and visual effects triggered during key actions (running, jumping, hitting an obstacle).

The programming challenge associated with this tutorial is a similar game to the previous one, but this time controlling a balloon to which you can apply force, and it must collect as many dollars as possible without touching a bomb.

For the bonus features proposed by the tutorial, I implemented the following in order of difficulty:

- Randomization of obstacles (positions, types of obstacles...)
- Adding a double jump
- Adding a speed boost and a score system
- Adding a game start animation (player's entry on the scene)

Figure 22 Screenshot of the main project in Unit 3

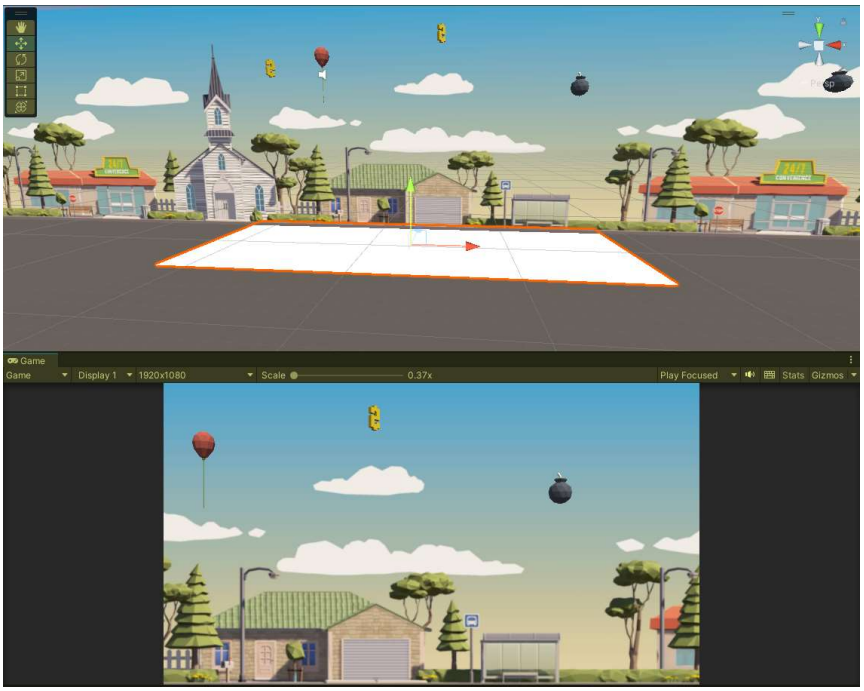


Figure 23: Screenshot of the challenge in the Unit 3

1.4. Project Unit 4 : Gameplay Mechanics

This fourth tutorial taught me how to create game management systems and consumables in Unity. During the learning phase, I created a top-down 3D sumo game where the goal is to eject other participants from the game platform. The player must survive successive waves of opponents and can use randomly appearing power-ups on the field. The combat system is based on Unity's physics and collision system.

The programming challenge associated with this tutorial is a game that uses the same combat mechanics as the previous one. This time, the player is on a soccer field. The opponent's balls aim to score in the player's goals, while the player must push them into the opposing goals.

For the bonus features proposed by the tutorial, I implemented the following in order of difficulty:

- Adding a new type of more powerful enemy
- Adding a homing missile
- Adding a superpower that crashes the player to the ground, repelling nearby enemies
- Adding a boss that uses a finite state machine to control its states (summoning new enemies, pursuing the player, repelling attacks, etc.)

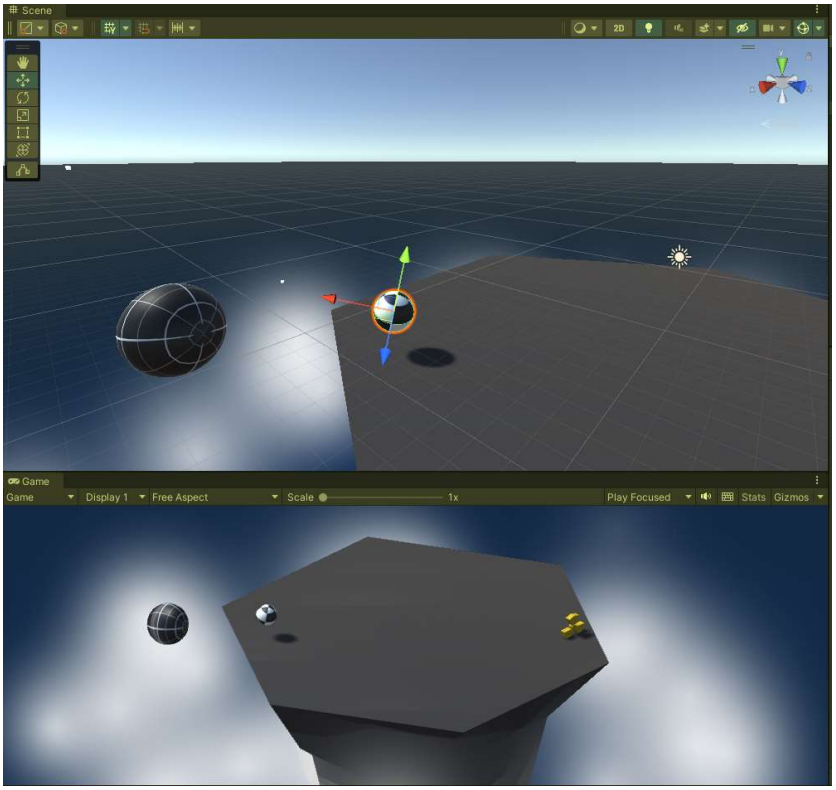


Figure 24: Screenshot of the main project in Unit 4

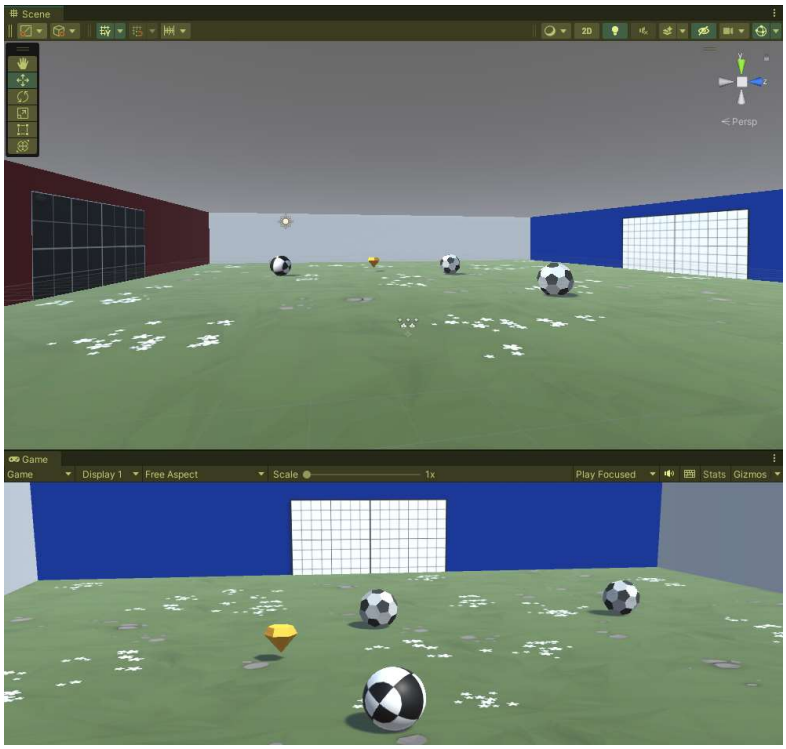


Figure 25: Screenshot of the challenge in the Unit 4

1.5. Project Unit 5 : Gameplay Mechanics

This fifth tutorial taught me how to create a user interface in Unity with a title screen offering different game difficulties, a display area for the player's score, and an end game screen that allows replay. The application I created during the learning phase is a reflex game where the player must click on randomly launched boxes before they go out of the screen.

The programming challenge associated with this tutorial is a 2D game where the goal is to click on food appearing on a grid within a time limit and avoiding clicking on a skull head.

For the bonus features proposed by the tutorial, I implemented the following in order of difficulty:

- Adding a text field displaying the player's number of lives
- Adding a slider to change the game's volume
- Adding a pause menu to halt the game's progress
- Adding a system to slice boxes by dragging the mouse over them (click and swipe)

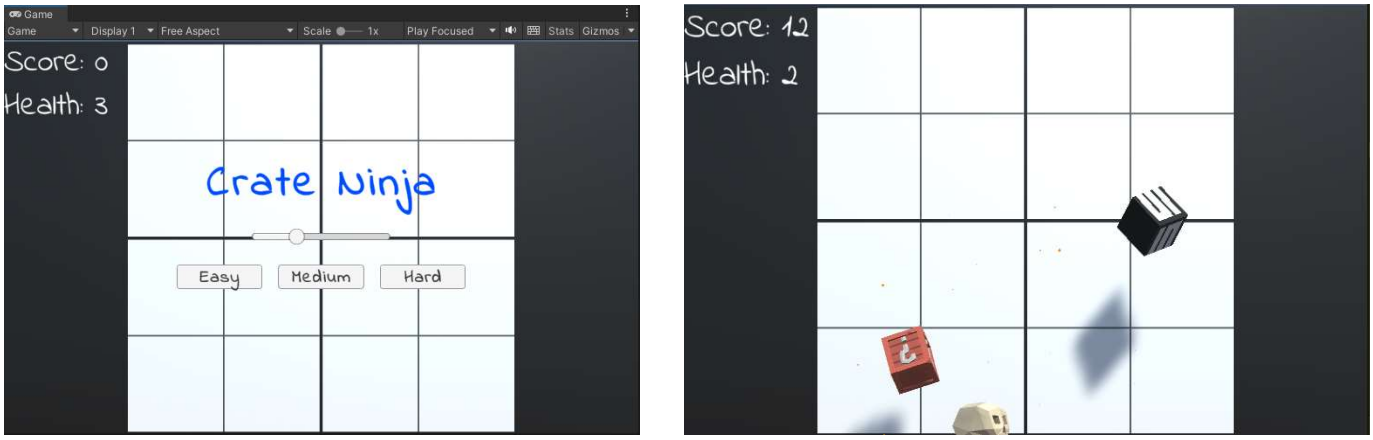


Figure 26: Screenshot of the main menu and the gameplay screen of the main project in Unit 5

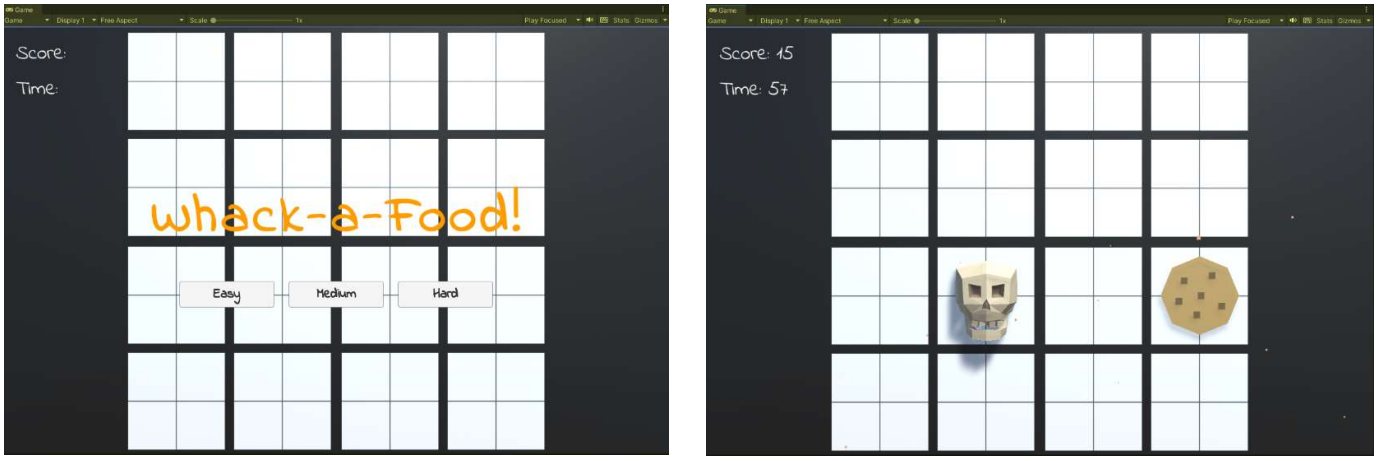
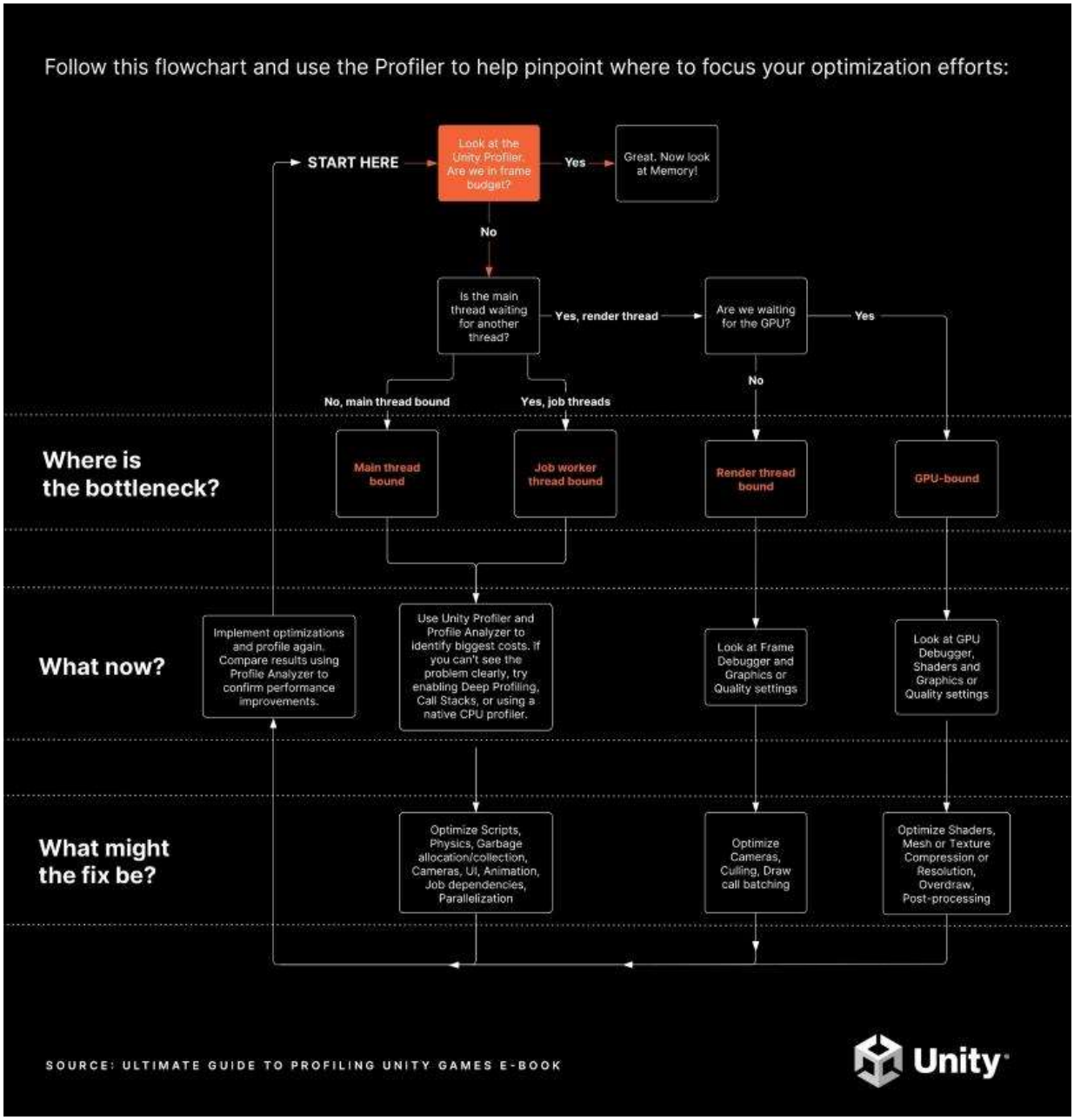


Figure 27: Screenshot of the main menu and the gameplay screen of the challenge in Unit 5

This project concludes the Unity "Create with Code" tutorial.

All completed exercises can be found on my Github page at the following address: https://github.com/JojoGelb/UnityCreateWithCode

Appendice 2 : Official Unity diagram on game performance analysis.



Appendice 3 : Software architecture using Unity's "ScriptableObjects"

"ScriptableObjects" is a Unity-derived class used when creating an object that does not need to be attached to a game scene element. It is particularly useful for storing data that can be easily modified through the software's inspector window. It also has the unique feature of preserving values between the editor mode and play mode, thus facilitating real-time adjustments, among other things.

One common issue observed in Unity projects is the difficulty some components have in communicating with each other within a scene. The simplest solution often used is implementing a singleton, which, while making access to the class easier, has the negative effect of creating strong dependencies between objects, preventing them from existing in a scene independently. This often leads to importing multiple singletons and related classes to use a functionality that may not necessarily require so many components. Implementing a Model-View-Controller (MVC) architectural pattern using ScriptableObjects can be a solution to this problem. It allows for better control of dependency injection and keeps track of all the components in the scene.

One example of using ScriptableObjects that I would like to detail is its use as an event bus/interface. In this case, ScriptableObjects serve as a platform to relay an event from one class to any other class that wants to listen to it.

Let's take the example of a player class with an integer variable representing its life. When this value changes, the class triggers an event notifying all subscribed classes, such as a health bar, a sound manager, or enemies reacting differently based on the player's health points. Using a ScriptableObject in this case avoids strong coupling between these classes. Instead of subscribing to the player, the classes listening to the event will subscribe to the ScriptableObject handling the life change event. This allows the player to trigger the event without worrying about the existence of scripts listening to it, while the scripts listening to the event do not need to directly know the player's reference. Since the ScriptableObject is not attached to a game object, it persists between scenes, guaranteeing its existence.

This architecture allows for easily adding or removing functionalities that require knowledge of the player's life changes without impacting the class's functionality. It also allows for individual testing of class responses to the event without having to create a player.

This clever implementation adheres to the SOLID principles with:

- Single Responsibility: The class is solely responsible for communicating the event and delegates the event management to the player class.
- Open/Closed Principle: New functionalities can easily be added without the need to know the classes behind them.
- Interface Segregation: The only dependency between classes here is the event.

- Dependency Inversion: The class does not require a trigger or receiver to function and does not depend on the scene context thanks to the ScriptableObject implementation.

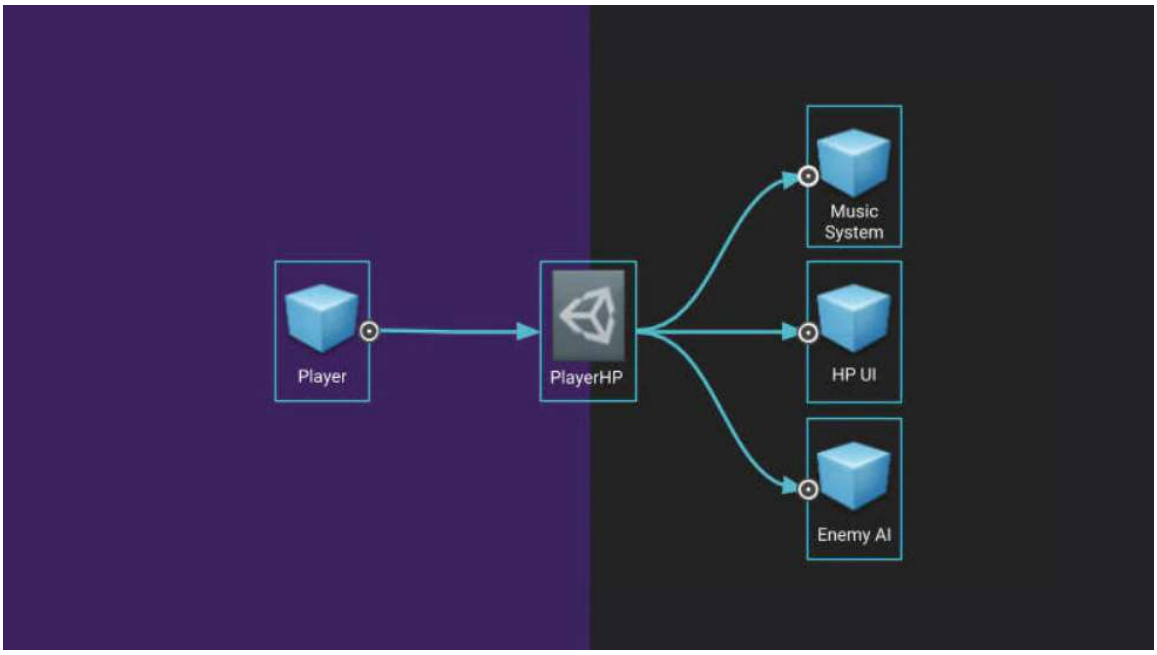


Figure 28: Diagram of how a ScriptableObject functions as an event distribution platform.

Appendice 4 : Code of a Singleton easy to use in Unity

```
/*
 A static instance is like a singleton but instead of destroying any new
instances,
it overrides the current instance. Usefull for resetting the state
 */
public abstract class StaticInstance<T> : MonoBehaviour where T :
MonoBehaviour
{
    public static T Instance { get; private set; }
    protected virtual void Awake() => Instance = this as T;

    protected virtual void OnApplicationQuit()
    {
        Instance = null;
        Destroy(gameObject);
    }
}

/*
 Basic singleton: Destroy any new version created.
 */
public abstract class Singleton<T> : StaticInstance<T> where T: MonoBehaviour
{
    protected override void Awake()
    {
        if (Instance != null) Destroy(gameObject);
        base.Awake();
    }
}


/*
 Will survive scene load
 Perfect for system classes, persistent data, audio source between scenes...

 */
public abstract class PersistentSingleton<T> : Singleton<T> where T :
MonoBehaviour
{
    protected override void Awake()
    {
        base.Awake();
        DontDestroyOnLoad(gameObject);
    }
}
```

REFERENCES

[1] Serious Game Society `Gala‘, 22 Juillet 2023: https://conf.seriousgamessociety.org/game-competition/

[2] Wikipedia ‘Ulster University’, 22 Juillet 2023: https://en.wikipedia.org/w/index.php?title=Ulster_University&oldid=1098814430

[3] Ulster [Online] ‘Research Excellence’, 14 Juillet 2023: https://www.ulster.ac.uk/research/our-research/research-excellence

[4] Serious Game Society `Gagnant des années précédentes du Gala‘, 22 Juillet 2023: https://seriousgamessociety.org/2022/08/22/4393/

[5] Unity, « Learn with Code », 22 Juillet 2023 : https://learn.unity.com/course/create-with-code?uv=2021.3

[6] Code Monkey ‘Learn Unity beginner/Intermediate 2023 ’ 22 Juillet 2023 : https://www.youtube.com/watch?v=AmGSEH7QcDg

[7] Unite Austin 2017, ‘Game Architecture with Scriptable Objects’, 22 Juillet 2023 : https://www.youtube.com/watch?v=raQ3iHhE_Kk

[8] Unity ‘Performance profiling tips for game developers’, 22 Juillet 2023 : https://unity.com/how-to/best-practices-for-profiling-game-performance

[9] Unity livre électronique : ‘Ultimate guide to profiling unity games ’ https://resources.unity.com/games/ultimate-guide-to-profiling-unity-games

Résumé

Ce document vise à présenter le stage que j'ai effectué du 24 avril 2023 au 11 août 2023 au sein du « Centre de Recherche sur les Systèmes Intelligents » de l'université d'Ulster à Londonderry.

Dans le cadre de la recherche de l'Université d'Ulster sur l'utilisation de la réalité virtuelle et des jeux vidéo pour l'apprentissage, mon tuteur, Mr Callaghan Michael, m'a proposé de poursuivre le développement du jeu éducatif « Numbers & Letters ». L'objectif de ce stage était d'ajouter des modes de jeu supplémentaires et d'augmenter l'accessibilité du jeu. J'ai principalement travaillé sur la seconde partie en portant le jeu sur une nouvelle plateforme : « L'oculus Quest Store » et en le traduisant dans plusieurs langages.

La réussite de ce stage a résidé dans les nouvelles connaissances et compétences que j'ai acquises tant en matière de programmation (découverte de Unity, maîtrise de C#, optimisation logicielle, etc.) que d'organisation (Travail en équipe, gestion des dates butoir du projet, vie dans un pays anglophone, etc.).

Mots Clés : Unity, Réalité virtuelle, Jeu vidéo, Optimisation, Université d'Ulster

Summury

This report summarizes the period of research I did at the Ulster University in Londonderry between the 24th of April 2023 and the 11th of August 2023.

My tutor, M. Callaghan Michael, works on the question of the effectiveness of VR and video games as a teaching tool.

The project I was given was to further develop the serious game "Numbers & Letters". My main goal was to port the game to a new platform: "The Oculus Quest Store" and translate it in several languages to makes the game more accessible. The biggest challenge I met during this period was the optimization of the game so that it could run smoothly on a performance constrained device.

This experience was a success for me. I learned a lot about video game development (Unity, C#, Software Architecture …) and project organization (Team works, deadlines management, agile method …).

It was also an excellent opportunity to live abroad and increase my masteries of English.

Keywords : Unity, Virtual reality, Video game, Optimization, Ulster University

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Ecole Publique d'Ingénieurs en 3 ans

6 boulevard Maréchal Juin, CS 45053
14050 CAEN cedex 04