Ecole Publique d'Ingénieurs en 3 ans

Rapport de stage 2A Informatique

DEVELOPPEMENT D'APPLICATIONS DE REALITE VIRTUELLE SOUS UNITY 3D

25 août 2023

Jordy Gelb,
Année Universitaire 2022/2023
Informatique / ISIA

Tuteur entreprise : **Michael Callaghan**
Tuteur ENSICAEN : **Régis Clouard**

www.ensicaen.fr

TABLE DES MATIERES

Introduction **4**

1. Présentation de l'université et mise en contexte du stage **5**

1.1. L'université d'Ulster 5

1.2. Contexte 6

1.3. Outil et méthode de travail 7

2. Début du stage **9**

2.1. Apprentissage de Unity 9

2.2. Premier jeu VR : VRArchery 11

2.3. Expérience sur l'implémentation d'une interface entre le cerveau et des membres virtuelles 13

3. Portage du jeu sur Quest **14**

3.1. Application multiplateforme 14

3.2. Optimisation du jeu 15

3.2.1. Modification du Render Pipeline 15

3.2.2. Optimisation CPU 16

3.2.1. Optimisation GPU 17

3.2.2. Optimisation de la compilation 19

3.3. Localisation de Numbers & Letters 20

Conclusion **22**

Remerciements

Je tiens à remercier vivement mon tuteur de stage, Michael Callaghan, professeur à l'université d'Ulster, pour m'avoir accueilli au sein de son laboratoire, pour la confiance qu'il m'a accordée et pour ses encouragements tout au long du projet.

Je souhaite également exprimer ma gratitude au professeur Hubert pour l'attention qu'il a portée à la progression de ce projet, ainsi qu'aux membres de mon équipe : Nicolas Rousseau, Etienne Langlois et Benjamin Maignan pour le soutien qu'ils m'ont apporté tout au long de cette période.

Je remercie aussi toute l'équipe pédagogique de l'ENSICAEN pour la qualité de leurs enseignements qui a su captiver ma curiosité et mon intérêt tout au long de ces deux dernières années.

J'adresse enfin mes remerciements à toutes les personnes qui m'ont apporté leur aide lors de la rédaction de ce rapport : mon tuteur de stage Régis Clouard et ma famille.

Toutes ces personnes ont contribué par leur disponibilité et leur bonne humeur à rendre mon stage enrichissant et motivant.

Introduction

Dans le cadre de la formation au Diplôme d'ingénieur informatique à l'ENSICAEN, j'ai réalisé un stage d'une durée de quatre mois au sein de l'Université d'Ulster au Royaume-Uni et plus spécifiquement dans le centre de recherche sur les systèmes intelligents situé à Londonderry sur le Campus de Magee.

Cette institution travaille depuis plusieurs années sur la question de l'utilisation des nouvelles technologies en matière de réalité virtuelle* et augmentée* dans l'enseignement. Différents groupes d'étudiants se sont succédés pour créer, affiner et améliorer des logiciels éducatifs en réalité virtuelle afin de proposer différentes expériences ludiques (IA de discussion sur le cerveau humain, jeu d'apprentissage…). Le dernier projet réalisé est un jeu de mathématiques ou le joueur évolue dans un monde « cartoonesque » et doit résoudre tout type d'opérations en interagissant avec les nombres qui l'entourent.

Ce jeu, « Numbers & Letters », publié sur Steam* le 6 octobre 2022, a gagné la même année un prix à la compétition de jeux sérieux « GALA » [1]. Le souhait de mon tuteur, Monsieur Callaghan Michael, était de continuer le développement de ce jeu pour le porter sur de nouvelles plateformes et ajouter de nouvelles fonctionnalités.

Le principal défi que m'a posé ce stage, en dehors des contraintes liées à la vie dans un pays étranger, a été l'apprentissage d'un nouvel outil : le moteur de jeu Unity* avec le langage de programmation C# qui l'accompagne ainsi que le travail en équipe sur un projet de grande ampleur.

Concernant l'organisation de ce rapport, je vais tout d'abord vous présenter le contexte dans lequel s'est déroulé ce stage. Ensuite, je vous décrirai la période d'apprentissage des technologies liées à Unity et à la réalité virtuelle. Enfin, je vous ferai part du travail de traduction, d'optimisation et de portage de « Numbers & Letters » sur de nouvelles plateformes que j'ai effectué lors de la seconde partie de mon stage. Pour terminer, je vous exposerai le bilan des connaissances acquises.

1. Présentation de l'université et mise en contexte du stage

1.1. L'université d'Ulster

L'université d'Ulster est la plus grande université d'Irlande du Nord. Créée en 1984, suite à la fusion entre la New University of Ulster et l'Ulster Polytechnic, elle comprend aujourd'hui 4 campus situés à Belfast, Corelaine, Jordanstown et Derry. L'université possède également une branche en distanciel à Londres et Birmingham.

Comptant plus de 27 000 étudiants et 1665 personnels académiques [2], l'université propose des formations dans de nombreux domaines tels que l'informatique, l'ingénierie, les sciences sociales, l'art… Les quatre valeurs principales abordées sont l'intégrité, la collaboration, l'inclusion et la valorisation du potentiel.

Cette institution est internationalement reconnue pour la qualité de sa recherche. Ils mettent celle-ci en avant sur leur site web : « *Des experts internationaux ont jugé l'Université d'Ulster comme faisant partie des 25% des meilleures universités britanniques pour la recherche de pointe mondiale basée sur la puissance de la recherche, et 72% de notre activité de recherche est jugée de premier plan et d'excellence internationale.* » [3].

Mon stage s'est déroulé sur le campus de Magee situé à Londonderry. Ce campus propose un grand nombre de programmes universitaires dans les domaines suivants :

- Arts, sciences humaines et sociales
- Informatique, Ingénierie et Aménagement du Territoire
- Sciences de la vie et de la santé
- École de commerce

Plus précisément, ma période de recherche s'est effectué au sein du « Centre de Recherche sur les Systèmes Intelligents » (ISRC) dans le pôle de « l'Ecole d'informatique et de système intelligents ». On compte plus de 90 chercheurs et de nombreux étudiants travaillant dans ce bâtiment. Le centre est organisé en plusieurs équipes de recherche avec à la tête, un professeur de l'université.

La mission de ce centre est d'en apprendre plus sur le fonctionnement du cerveau et d'appliquer ces connaissances à la création de modèles et de technologies capables de résoudre des problèmes complexes.

1.2. Contexte

Notre chef de projet, Monsieur Callaghan Michael, travaille depuis plusieurs années sur la question de l'efficacité de l'apprentissage par le jeu et la Réalité Virtuelle (VR). En collaboration avec M. Hubert Cecotti, professeur au « California State University », et plusieurs équipes d'étudiants, plusieurs jeux d'apprentissage sur des thèmes variés ont été publiés sur Steam. Parmi ceux-ci, on peut citer « Electronic Circuit Simulator », un jeu d'apprentissage sur le thème des circuits électroniques ou encore « Medical Imaging VR », une application permettant la consultation de scanners d'imagerie médicale en VR*.

En 2022, le jeu vidéo en VR « Numbers & Letters » a été publié sur Steam et a gagné le premier prix à la compétition de jeux sérieux GALA [4]. C'est un jeu de mathématiques qui a pour but d'améliorer les capacités du joueur en matière de calcul mental. Pour ce faire, le joueur doit résoudre des équations en utilisant les nombres naviguant dans la scène. En combinant ceux-ci à l'aide des opérateurs disponibles, l'utilisateur peut résoudre des exercices plus ou moins complexes.

Figure 1: Logo de finaliste de la compétition GALA 2022

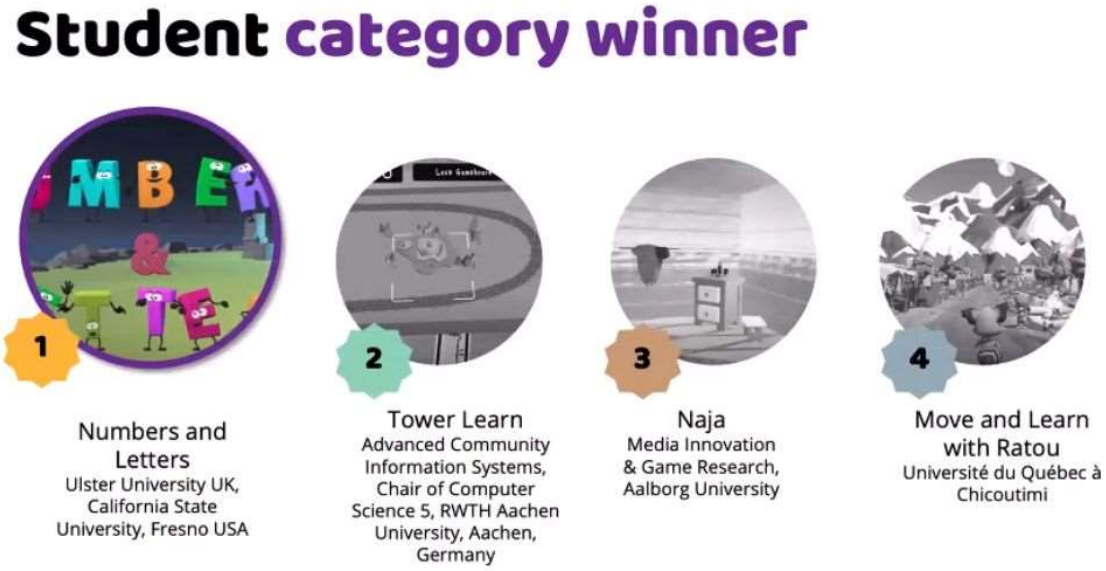


Figure 2: Montage montrant les jeux gagnants de la catégorie étudiant du GALA 2022

Deux nouveaux modes de jeux sont ensuite apparus suite à des mises à jour fréquentes du jeu. Le premier est le mode Inversé où cette fois-ci une équation est donnée au joueur et il doit trouver, parmi les nombres présents dans la scène, le résultat. Le second est le mode « Letter » dont le but est de trouver la lettre manquante d'un mot afin de gagner des points.

Le souhait de M. Callaghan pour cette période de recherche, était de rendre le jeu accessible au plus grand nombre et d'ajouter de nouveaux modes de jeu. Ce travail a été réparti entre les différents groupes d'étudiants français qui ont participé cette année au développement de l'application. J'ai principalement travaillé sur la première partie, à savoir rendre le jeu multi-plateforme et traduire l'interface dans plusieurs langues.

1.3. Outil et méthode de travail

Afin de réaliser ce projet, un certain nombre d'outils m'ont été imposés.

Tout d'abord, les outils de développement :

- Le moteur de jeux Unity avec son langage de programmation C#. Celui-ci est une contrainte imposée par notre tuteur, car la première partie du jeu a été développée dessus.

- L'environnement de développement intégré Visual Studio 2022 qui est celui par défaut de Unity. Le plus gros avantage que j'ai pu constater chez celui-ci est son extension Unity facilitant grandement la programmation au travers d'un système d'autocomplétions intelligent et une documentation accessible depuis le logiciel.

- Le logiciel de gestion de version Plastic SCM. Celui-ci nous a permis de récupérer le projet tel qu'il nous a été laissé par nos prédécesseurs et ainsi continuer son développement. Plastic SCM a été acquis par Unity en 2020 et facilite grandement le travail à plusieurs sur un projet de grande ampleur grâce à son système de gestion de conflit puissant et son interface utilisateur facile d'utilisation.

- L' HTC Vive et l'Oculus Quest 2 ont été les casques VR que nous avons utilisés au cours de la période de recherche. Le premier est un casque filaire qui est accompagné de deux stations externes permettant le suivi en temps réel du corps. Ce mode de

Figure 4: Photo de l'HTC Vive

Figure 3: Photo du Meta Quest 2

fonctionnement permet à l'utilisateur de jouer à des jeux plus puissants, car tous les calculs sont réalisés par l'ordinateur. Contrairement au HTC Vive, le Quest 2 est un casque avec un processeur intégré permettant donc de jouer à des jeux sans avoir besoin d'une connexion filaire avec un autre appareil. L'utilisateur jouit ainsi d'une plus grande liberté dans ses mouvements, mais perd, au final, en qualité d'affichage et de puissance pour les jeux.

Ensuite, les outils de communication :

- La plateforme de communication Slack. Cette plateforme nous a permis de garder un fils de discussion entre les différents membres du projet et d'échanger avec notre tuteur tout au long de notre stage.

- Le logiciel de visiophonie Zoom pour échanger périodiquement de vive voix sur la progression du projet.

- L'outil de gestion de projet Trello. Ses tableaux interactifs en ligne nous ont été très utiles au début du projet pour nous organiser dans l'apprentissage du logiciel et rester à jour sur la progression de chacun dans les tutoriels mis à disposition par notre tuteur.

Figure 5: Capture d'écran du tableau Unity sur Trello

Quant à la méthode de travail employé, celle-ci s'apparentait davantage à la méthode agile. En effet, l'équipe de développement participait à deux réunions Zoom par semaine avec notre tuteur et M. Hubert Cecotti. On discutait alors de la progression du projet et mettait en place les objectifs à réaliser avant la prochaine réunion. Le travail était donc découpé sous forme de courts sprints de deux à trois jours sur de courtes fonctionnalités.

2. Début du stage

Le premier mois de stage était principalement dédié à l'apprentissage du moteur de jeu Unity, la création de nos premiers jeux VR, et la découverte du laboratoire et des différentes équipes de projet en son sein.

2.1. Apprentissage de Unity

Mon travail au cours des deux premières semaines de stage a été de me former sur le moteur de jeu Unity. De nombreux tutoriels ont été mis à notre disposition sur la plateforme Trello de l'équipe de projet permettant notamment à notre tuteur de suivre notre progression.

J'ai, dans un premier temps, complété le tutoriel officiel d'Unity « Create with code » [5] d'une durée estimé de 41h. Celui-ci m'a appris les bases du logiciel au travers de plusieurs minis projets guidés et d'examens ludiques. Une description plus détaillée de chacun de ces projets est consultable en annexe.

J'ai ensuite suivi la vidéo de 10h du youtuber Code Monkey [6] , recommandée par mon tuteur, sur la création d'un jeu de cuisine. Celle-ci m'a permis de renforcer mes connaissances acquises au préalable tout en m'introduisant à différents points clés de la gestion d'un projet Unity. J'ai notamment appris au cours de ce tutoriel à suivre des règles strictes en termes de style de programmation qui m'ont facilité la tâche par la suite. Vous trouverez ma version du jeu sur ma page Itch.io.

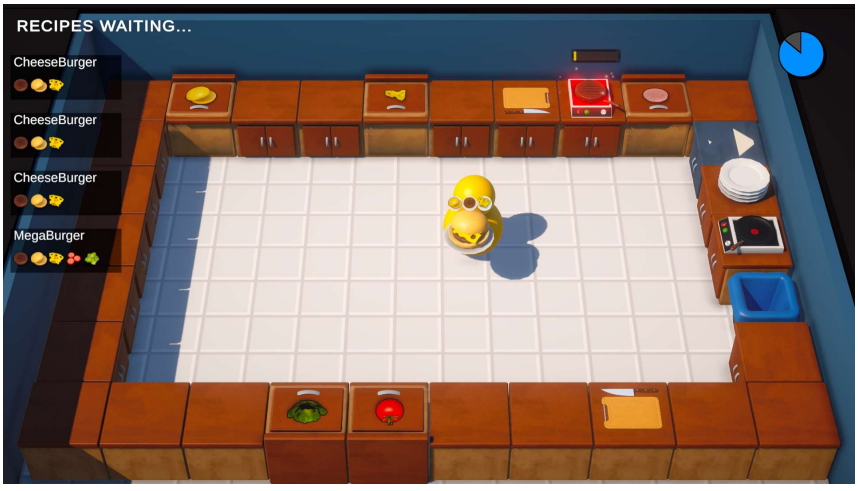


Figure 6: Capture d'écran du jeu GetCoocked

Enfin, j'ai suivi de nombreux tutoriels décrivant des concepts d'architecture logiciel appliqué au moteur de jeu Unity. Les principes, que j'ai par la suite le plus utilisés sont :

- L'utilisation d'événements pour inverser les dépendances entre les scripts.
- L'utilisation de « ScriptableObjects » afin de diminuer grandement les dépendances entre mes classes et de rendre mon projet ouvert aux extensions. Vous trouverez notamment une explication appuyée d'un schéma montrant une implémentation des ScriptableObjects en annexe ainsi qu'une vidéo officielle de Unity décrivant les qualités de ce composant dans mes références [7].
- L'implémentation de singletons (Manageur de jeu, d'inputs…) afin de faciliter l'accès à un objet dans la scène et de rendre celui-ci unique. (Annexe : Singleton sous Unity)
- La ségrégation entre la partie logique et la partie visuelle de mes composants afin de garder mon code simple et facilement lisible.

Finalement, à la demande de mon tuteur, j'ai créé en équipe avec Nicolas Rousseau, Etienne Langlois et Benjamin Maignan le jeu « Number And Gun » au cours de la troisième semaine de stage. C'est un jeu à la première personne où le but est de gagner le plus de points possibles en tirant sur des personnages se déplaçant dans la scène. Le joueur à trois minutes pour battre son record et doit en même temps éviter les attaques de plusieurs robots.

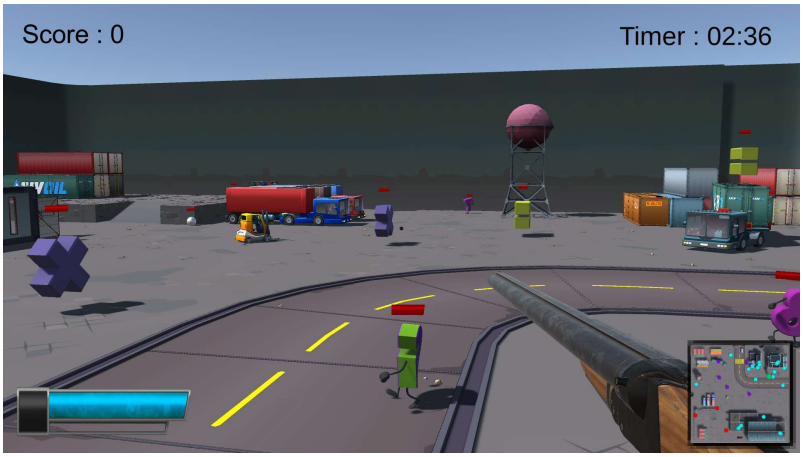


Figure 7: Capture d'écran de Number & Gun

Ce premier projet en équipe nous a permis d'apprendre à utiliser le logiciel de gestion de version Plastic SCM et surtout de nous accorder sur les points essentiels au bon déroulement de nos futurs projets. En effet, nous nous sommes rendu compte d'un certain nombre de problèmes concernant les pratiques de programmations divergentes entre certains membres du groupe. Nous avons donc profité de l'occasion pour aligner nos connaissances obtenues lors des premières semaines d'apprentissage et planifier le déroulement de nos futurs projets.

2.2. Premier jeu VR : VRArchery

L'accès aux locaux étant restreint pendant les trois premières semaines, j'ai pu commencer à travailler sur de la réalité virtuelle à partir de ma quatrième semaine de stage.

L'ultime tâche assignée par mon tuteur pour conclure cette première partie de formation à Unity a été la création de mon premier jeu en VR. La seule consigne concernant celui-ci était qu'il devait utiliser les assets***** présents dans « Numbers & Letters ». Ainsi, avec la même équipe que pendant la création de Number & Gun, nous avons créé le jeu VRArchery.

Ce jeu est un simulateur VR de tir à l'arc. L'utilisateur apparaît dans une scène avec à proximité un arc et des flèches. Son but est de tirer sur des nombres présents autour de lui pour gagner des points. Plus un nombre est grand, plus il est difficile à toucher, et plus il rapporte de points.

Sur ce jeu, j'ai principalement travaillé à l'implémentation des interactions de l'utilisateur avec les éléments de la scène. J'ai ainsi découvert le fonctionnement du XR interaction Toolkit de Unity.

Le XR Interaction toolkit est une surcouche proposée par Unity permettant, à l'aide de composants simples, la mise en place des interactions de base en VR entre le joueur et les composants de la scène. Ce package est composé d'un ensemble de classes de type « Interactor » et « Interactable » interagissant l'un avec l'autre, ainsi qu'un gestionnaire d'interaction supervisant toutes les opérations entre ces deux types d'objets. Il existe différents « Interactor » permettant de simuler un certain nombre de comportements habituellement observés dans les applications VR tel que le « Gaze Interactor » (Utilisation du regard pour

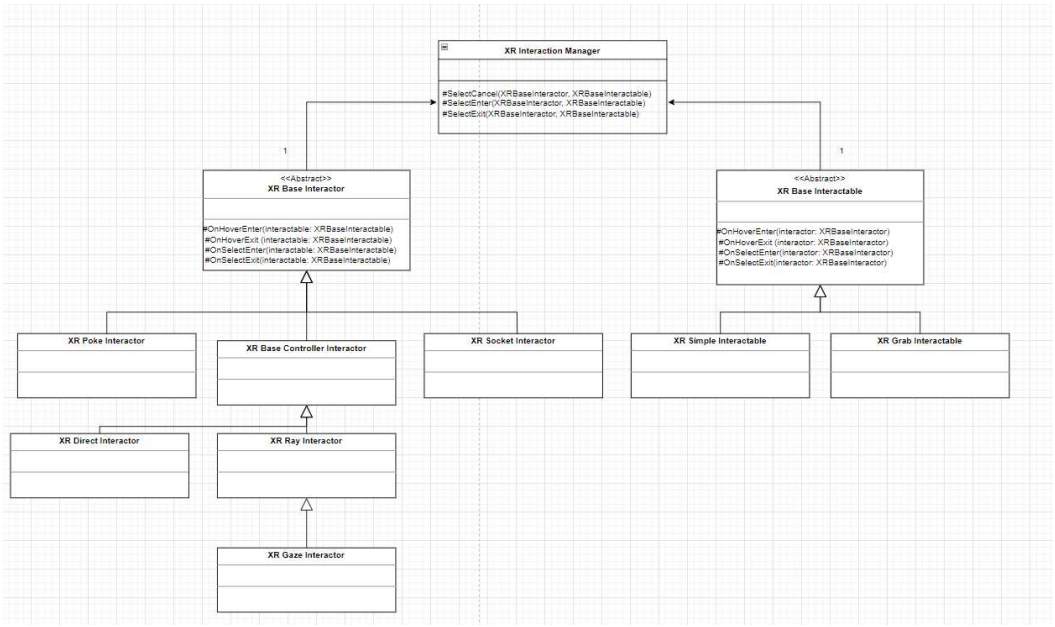


Figure 8: Shéma UML des classes de base du XR Interaction Toolkit

interagir avec la scene), le « Simple Interactor » (Interaction à distance de bras), ou encore le « Laser Interactor » (Pointeur laser partant des mains et permettant des intéractions à distance).

Nous avons implémenté un système pour notre jeu utilisant le « Simple » et le « Laser Interactor » afin de pouvoir récupérer les flèches disposées sur une table à distance (laser), encocher manuellement celles-ci à l'arc (simple), et bander/relâcher la corde pour faire partir le projectile.

Figure 9: Capture d'écran du jeu VR Archery

2.3. Expérience sur l'implémentation d'une interface entre le cerveau et des membres virtuelles

Au cours de ce premier mois de stage, j'ai également pu entrer en contact avec un des chercheurs travaillant au Pôle d'innovation en informatique spatiale et en neurotechnologie de l'ISRC : Mr Niall McShane. J'ai ainsi participé à une expérience scientifique visant à entraîner un réseau de neurones à reconnaître les signaux envoyés par mon cerveau pour réaliser un mouvement.

L'expérience se divise en 10 sessions d'entraînement espacées d'au moins une journée. Au cours de chacune d'entre elles, le but est d'atteindre des cibles, dans un premier temps, physiquement, avec mon bras, puis dans un second temps, mentalement, en s'imaginant le mouvement. Les ondes cérébrales alors produites par mon cerveau sont récupérées à l'aide des capteurs présents sur le casque « g.Nautilus RESEARCH 32 ». Chacune de ces sessions est précédée par une étape d'installation du matériel pendant lequel le sujet répond à des questionnaires anonymes. Le casque est alors fixé sur la tête du cobaye et du gel est inséré par les pores du casque afin de faciliter le transfert des signaux de la tête vers le casque.

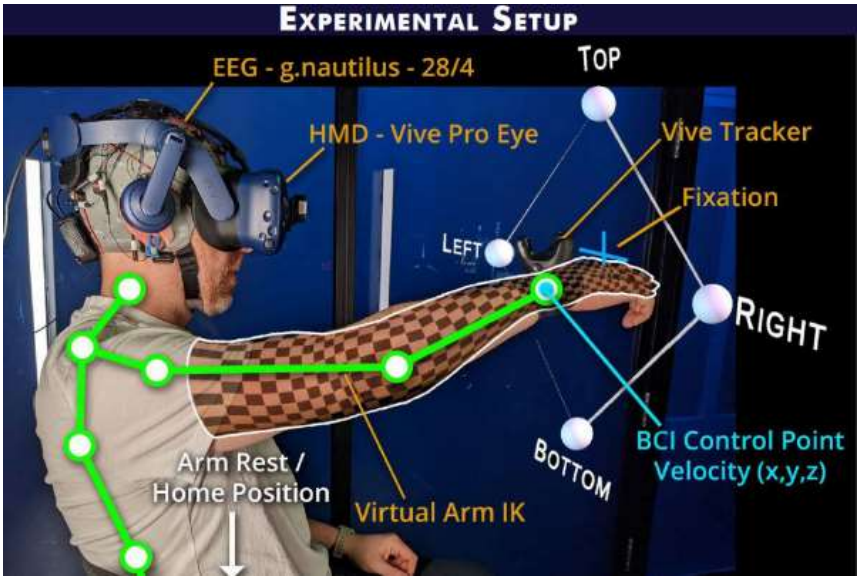


Figure 10: Schéma explicatif du montage de l'expérience

La moitié des sessions ont été réalisées à l'aide d'un casque VR, l'autre moitié à l'aide d'un écran en deux dimensions.

J'ai pu profiter de cette période pour échanger avec Mr Niall McShane. Cette expérience m'a permis de découvrir le domaine de la recherche lié à l'analyse des ondes cérébrales. J'ai également pu observer les protocoles scientifiques à mettre en place afin de réaliser des tests et d'en extraire des résultats.

3. Portage du jeu sur Quest

L'objectif principal de cette deuxième partie de mon stage est de rendre l'application « Numbers & Letters » accessible au plus grand nombre et notamment de déployer celle-ci sur l'Oculus Quest Store. Pour ce faire, j'ai dû modifier et adapter le code du jeu déjà existant pour le rendre multi-plateforme, performant sur Windows et Android, et multilingue.

3.1. Application multiplateforme

Afin de pouvoir lancer le jeu sur l'Oculus Quest 2 tournant sur un OS Android, j'ai dû, dans un premier temps, changer l'API utilisé pour communiquer avec le matériel VR. En effet, le plugin SteamVR, alors utilisé pour s'occuper des interactions avec l'environnement, présentait certains bugs lorsqu'utilisé sur le Quest 2.

J'ai donc migré les contrôles du jeu vers le XR interaction toolkit de Unity. Celui-ci présente comme gros avantage de supporter la vaste majorité des casques présents actuellement sur le marché. Il facilite également le contrôle des différents types de casques pouvant être utilisé selon la plateforme sur lequel le jeu a été compilé. Ainsi, j'ai autorisé tous les casques disponibles de l'API pour les « build » réalisé sur la plateforme Windows et je n'ai autorisé que les casques Oculus pour les « build » Android. Ce changement important du code m'a notamment permis de conserver un seul projet pour les deux plateformes visées, facilitant ainsi sa maintenance et l'ajout de nouvelles fonctionnalités.

Dans un second temps, je me suis occupé des parties du code directement liées à la plateforme sur lequel le jeu est exécuté. Les trois principaux concepts sur lesquels je me suis penché sont : le système de succès, le système de classement, et l'accès aux fichiers.

Le code n'ayant pas été écrit en prenant en compte la possibilité que de nouvelles plateformes puissent être implémentées, j'ai dû modifier un grand nombre de classes afin de réduire les dépendances de celles-ci vis-à-vis de la plateforme Steam. J'ai fait en sorte d'isoler le plus possible les parties du code liées à l'utilisation d'une plateforme que j'ai encapsulées dans des fonctions généralisées. Ainsi, il est facile d'ajouter de nouvelles plateformes au jeu sans avoir à modifier le code de celles déjà en place. J'en ai également profité pour englober les parties du code liées à une plateforme en particulier de directives de compilations. Cette modification permet notamment d'éviter d'avoir du code compilé pour une autre plateforme que celle visée.

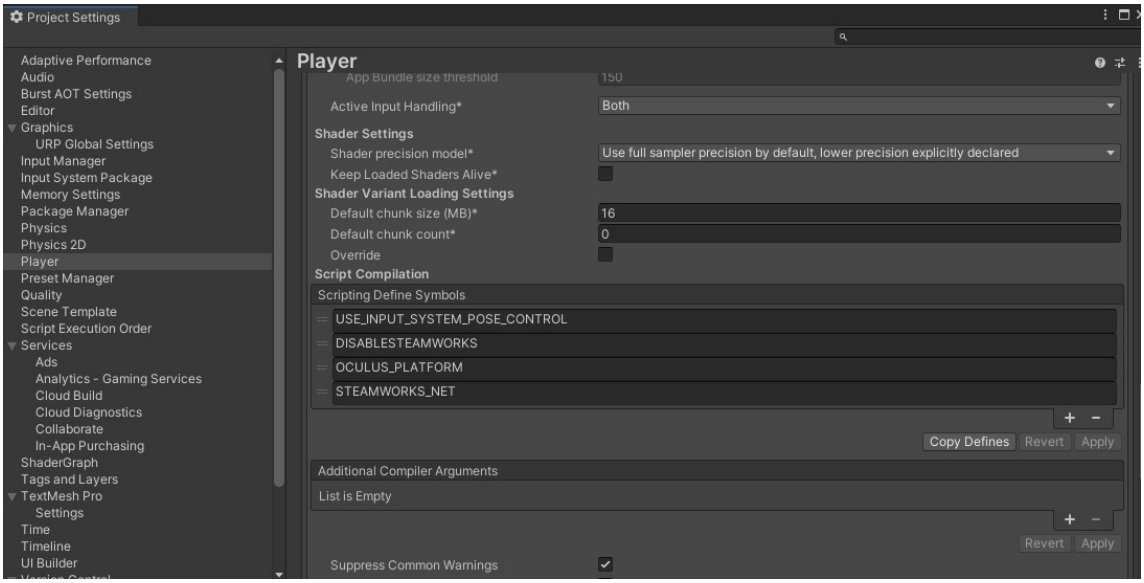


Figure 11: Capture d'écran des directives de compilation du projet sous Android

3.2. Optimisation du jeu

Un des plus grands challenges que m'a posé le portage du jeu de SteamVR vers L'Oculus Quest 2 a été les problèmes d'optimisation de l'application. En effet, celle-ci n'a pas été développée en prenant en compte les contraintes de performance de l'appareil. L'Oculus Quest 2 utilise comme système d'exploitation « Android » et est capable de faire tourner des jeux en local sans avoir besoin d'une connexion à un ordinateur. Lors des premiers tests, le jeu souffrait de lourds problèmes de chute de FPS* que j'ai corrigés au travers de plusieurs étapes.

3.2.1. Modification du Render Pipeline

La première action que j'ai réalisée afin d'améliorer les performances du jeu a été de changer le « Render Pipeline » alors en place (Built in Render pipeline) pour « l'Universal Render Pipeline ». Cette nouvelle version, régulièrement mise à jour, est plus performante. Elle utilise des techniques de rendu basé sur les shaders modernes pour atteindre des performances élevées tout en préservant la qualité visuelle. Elle permet également d'offrir une plus grande flexibilité au niveau de la personnalisation du pipeline de rendu me permettant notamment de désactiver les effets visuels trop coûteux pour le GPU. L'un de mes premiers travaux à ce sujet a été de changer le plan d'eau sur la scène principale qui déplaçait une grande quantité de pixels à l'écran pour simuler un effet de vague, par un shader optimisé avec une normal map*. L'effet, au final, était très ressemblant et j'ai pu constater une amélioration généralisée de la stabilité des FPS.

Figure 12: Capture d'écran du changement visuel du plan d'eau (à gauche l'ancienne version, à droite la nouvelle)

3.2.2. Optimisation CPU

La seconde action que j'ai entreprise a été d'étudier la documentation de Unity afin de mieux comprendre le fonctionnement du moteur de jeu et les raisons qui pourraient pousser à ces ralentissements. J'ai découvert un excellent guide [8] qui explique les différentes raisons qui peuvent mener à une perte de performance aussi importante. Vous trouverez en annexe un schéma décrivant le processus par lequel je devais passer afin de déterminer l'origine de ces problèmes.

J'ai su identifier à l'aide de l'analyser de Unity une surcharge importante du CPU (Render Thread Bound) dû à un trop grand nombre d'appels consécutifs au GPU aussi nommés « Draw calls ». Un Draw call est un appel à l'API graphique de Unity réalisé pour afficher des formes à l'écran. La préparation d'un draw call créé une charge non-négligeable pour le CPU. En effet, d'après la documentation de Meta, il est recommandé de ne pas dépasser les 600 draw calls afin de garder son nombre de FPS stable. Certaines scènes du jeu présentaient plus de 1000 draw calls notamment dû à la diversité de l'environnement et l'apparition de beaucoup d'éléments à la caméra en même temps.

Pour corriger ce problème d'optimisaiton, j'ai décidé de combiner les mesh* de mes objets composant mon environnement entre eux à l'aide d'un outil présent sur l'Unity Assets Store réduisant ainsi de deux tiers le nombre de draw call pour les scènes les plus couteuses, et un tier en moyenne pour les autres. J'ai également pris le temps de configurer le système d'occlusion oculling* pour les composants de la scène afin que lorsque les objets appartenant à l'environnement sortent du champ de vision de l'utilisateur, ils disparaissent et ne sont donc plus comptés parmi les objets transmis au GPU pour l'affichage de la scène. Les deux

techniques énoncées ci-dessus rentrent en conflit, car les objets désormais fusionnés ensemble, en un seul mesh, ne peuvent être séparés et donc fonctionner avec l'occlusion oculling. J'ai donc décidé de regrouper les objets composant l'environnement en fonction de leur position par rapport à la caméra afin de pouvoir faire disparaître une partie de la scène selon l'orientation du joueur tout en gardant un nombre de draw calls satisfaisant.

Ce changement a eu un impact important sur les performances du jeu le rendant ainsi stable sur la majorité de ses scènes. Le seul effet négatif observé est une augmentation de la taille du projet Unity dû à la création de nouveaux Mesh pouvant prendre pour certains une grande taille. Cette modification n'impacte pas la taille du build, car Unity se charge de retirer tous les Mesh non utilisés de l'application final.

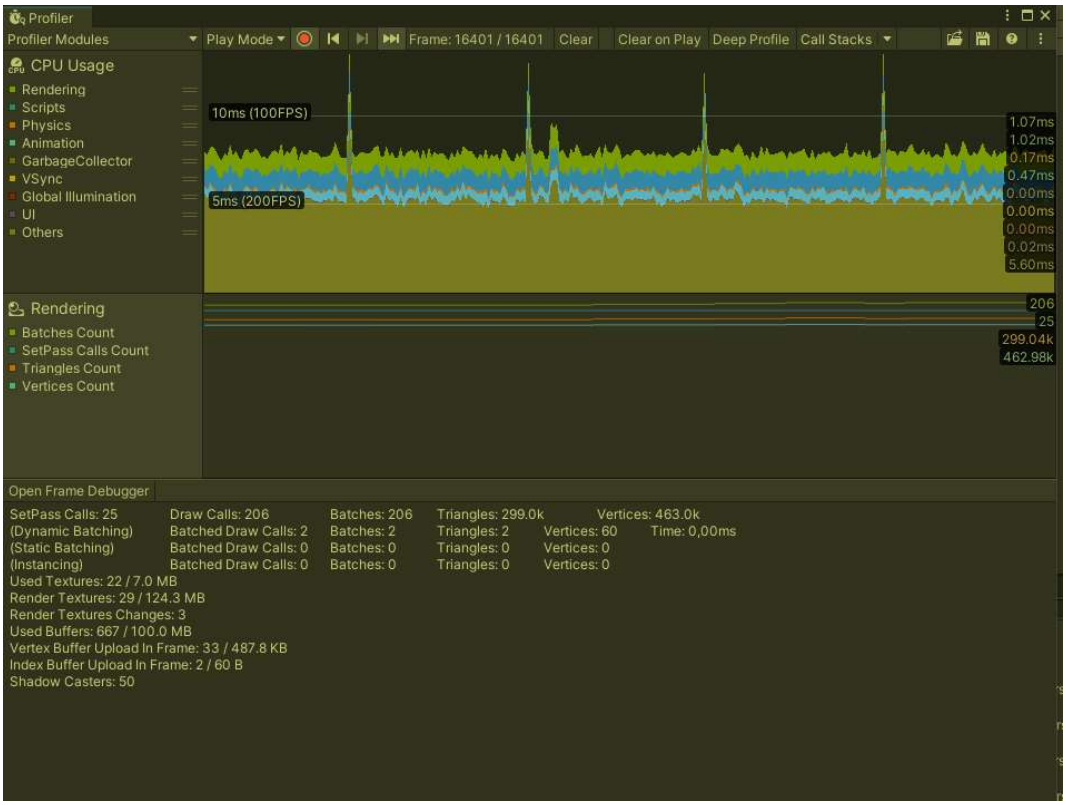


Figure 13 : Capture d'écran de l'analyser de Unity en mode éditeur sur la scène Plage 180 de "Numbers & Letters"

3.2.1. Optimisation GPU

La troisième action entreprise afin d'améliorer les performances du jeu a été la réduction du nombre de polygones affiché à l'écran. En effet, certains angles de caméra affichant trop d'objets en même temps pouvaient provoquer de fortes chutes de performance.

Ce problème venait d'une surcharge du GPU (GPU bound) causé par la complexité trop importante de certains éléments du décor pour le Quest 2. La documentation de Meta à ce

sujet recommande de n'afficher à l'écran pas plus de 1 million de triangles par fenêtre d'affichage. Afin de résoudre ce problème, j'ai utilisé le plugin « UnityMeshSimplifier » de Whinarn me permettant de simplifier un mesh. Cette opération induit forcément une dégradation de la qualité de celui-ci, mais les éléments modifiés étant le plus souvent à longue distance du joueur et en très grande quantité sur une scène, la différence n'est pas visible. Cette méthode m'a permis de réduire mon nombre de polygones affichés dans ma scène et a donc corrigé mes derniers problèmes de performance.

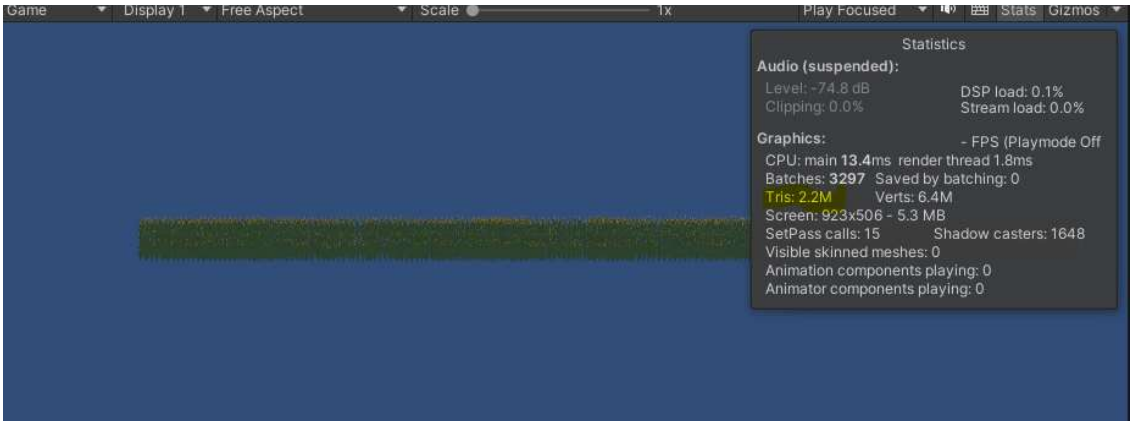


Figure 14: Capture d'écran du champ de maïs avant optimisation

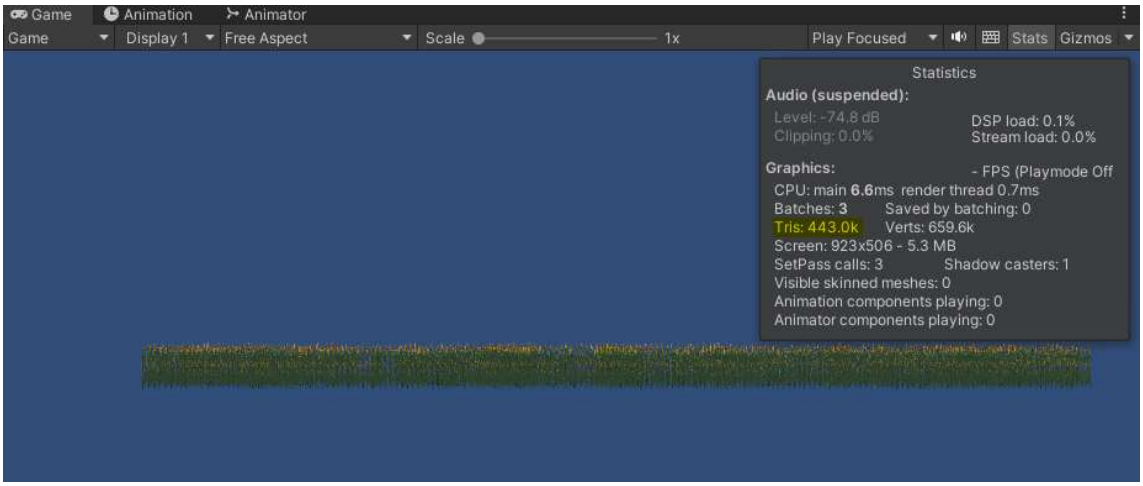


Figure 15: Capture d'écran du champ de maïs après optimisation

3.2.2. Optimisation de la compilation

Le jeu tournant désormais à 60 FPS de manière stable, j'ai pu aborder un autre aspect de l'optimisation qui est celle de la compilation du projet. En effet, le jeu souffrait alors de lourds soucis de compilation sous Android à cause d'une utilisation trop importante de la mémoire RAM de l'ordinateur (> 20 Go).

Après diagnostic du problème, j'ai découvert l'existence d'un fichier C# contenant une liste de mots de plus de dix mille éléments utilisée pour générer les mots à trouver du mode Letter. Lors de la compilation, ce tableau est chargé en mémoire et engendre une hausse importante de l'utilisation de la mémoire vive qui provoque un crash de la compilation dans le meilleur des cas, un crash de l'ordinateur dans le pire. J'ai corrigé ce problème en déplaçant cette liste de mots dans un fichier au format « .txt » réduisant ainsi le temps de compilation du projet et permettant, au passage, à l'utilisateur de customiser la liste de mots avec laquelle il souhaite jouer au mode Letter.

Un autre point de la compilation sur lequel je me suis penché est celui des shaders. Lors de la première compilation, ceux-ci pouvaient prendre de 40 minutes jusqu'à 1h30 à être générés. Ce temps était réduit à une dizaine de minutes pour les compilations suivantes. Après étude de la documentation de Unity, j'ai décidé d'activer le « Stripping » des shaders du projet permettant au compilateur d'ignorer ceux qui ne sont pas utilisés dans notre jeu. Ajouté à cela quelques modifications permises par l' « Universal Render Pipeline », j'ai réussi à réduire le temps de compilation à une dizaine de minutes lors du premier build, et à moins de 3 minutes lors des build suivant sauvant ainsi un temps précieux aux développeurs du projet souhaitant tester leur jeu sur le Quest.

Tous ces changements opérés au niveau de l'optimisation du projet m'ont permis de conserver un seul et unique projet sur lequel il est possible de développer en parallèle sur la plateforme Steam et sur la plateforme d'Oculus Quest, et ce, sans avoir à faire de changement important au niveau de l'environnement ou des effets visuels.

3.3. Localisation de Numbers & Letters

La dernière étape de mon stage, en plus de m'occuper de l'aspect intégration et optimisation des autres projets en parallèle, a été de traduire le jeu « Numbers & Letters » dans les cinq langues suivantes : français, italien, espagnol, allemand et portugais. Pour ce faire, j'ai développé une suite de composants simples d'utilisation pour le développeur, et facilement extensibles par l'utilisateur.

L'idée initiale étant de rendre l'application traduisible par tout le monde, il fallait stocker les traductions dans un fichier JSON facilement modifiable par l'utilisateur. J'ai donc, dans un premier temps, créé le fichier « localizationData.json » contenant à son sommet une liste des langues supportées, puis un tableau associant une clé à toutes les traductions disponibles d'une suite de mots.

J'ai ensuite créé une classe statique servant d'interface entre les autres composants et le fichier JSON. Celle-ci possède également un événement C# notifiant aux classes abonnées du changement de langue de l'interface.

Enfin, le dernier script est un composant Unity ne s'attachant qu'à un objet possédant un champ de texte. Automatiquement, au démarrage de l'application, le script s'abonne à l'événement de changement de langue et demande la traduction de sa clé à l'interface de traduction. Plusieurs fonctions « SetText » ont été créées dans cette classe afin de faciliter le changement du mot au cours de l'exécution du jeu. Ces fonctions, acceptant différents types de paramètres tel que des entiers, des nombres à virgule ou encore des chaînes de caractères, récupèrent les données de traduction et remplacent les particules changeantes à l'intérieur (tel qu'un timer, un affichage de score…). Ces particules changeantes sont désignées par un 0 dans les champs de traduction du fichier JSON.

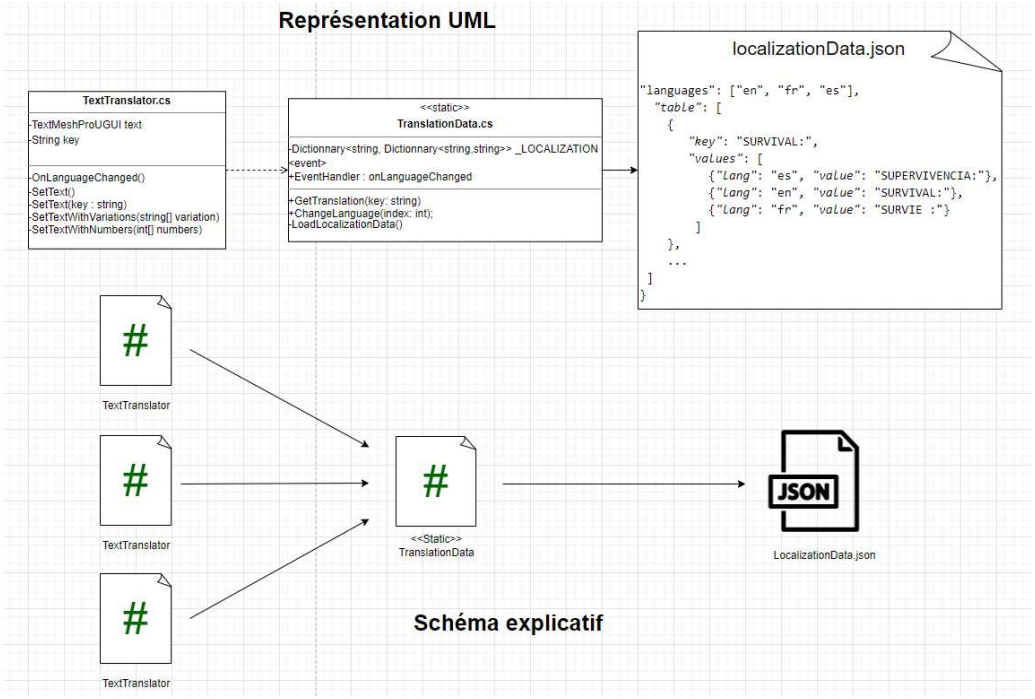


Figure 16: Schéma UML et explicatif de l'implémentation de la localisation dans "Numbers & Letters"

Si le développeur souhaite rendre un nouveau champ traduisible, il lui suffit d'ajouter le composant « TextTranslator.cs » à l'objet possédant le champ de texte avec à l'intérieur la clé de traduction. Si un utilisateur souhaite ajouter une langue au jeu, il lui suffit juste d'ajouter celle-ci aux champs des langues du fichier JSON, puis d'ajouter ses traductions à la suite de celles déjà existantes.

Figure 17: Capture d'écran du menu de jeu "Number" traduit en Espagnol

Conclusion

Ces 16 semaines passées au sein de l'IRSC à Derry m'ont permis de découvrir le monde de la VR et du développement de jeux vidéo. Ce stage fut très enrichissant en terme de pratique, d'apprentissage, d'utilisation des acquis théoriques, mais aussi humainement.

Ma mission consistant à rendre le jeu « Numbers & Letters » accessible au plus grand nombre a été un réel défi à relever. Ne connaissant ni le domaine du développement multi-plateforme, ni le moteur de jeu Unity, j'ai investi beaucoup de temps et d'énergie afin de bien comprendre le fonctionnement de ces technologies et de fournir un travail de qualité

Ensuite, l'apprentissage de nouvelles pratiques en matière d'optimisation, de conception logicielle, et de gestion de projet m'a ouvert de nouveaux horizons. Je compte approfondir ces compétences dans l'objectif d'avoir une meilleure vue d'ensemble sur mes futurs projets et donc anticiper d'éventuelles difficultés.

Enfin, la pratique continue de l'anglais au cours de ce stage m'a permis d'enrichir mon vocabulaire et d'acquérir une aisance à l'oral.

J'ai apprécié le fait de pouvoir travailler directement au sein du laboratoire avec à disposition des appareils de pointe. Mon intérêt pour les technologies VR en matière de jeu vidéo s'est accru. Je compte investir lors de mon double diplôme au Canada dans de l'équipement VR afin de pouvoir continuer à développer des applications tout en m'intéressant étroitement à tout ce qui touche à ce domaine.

A ce sujet, mon tuteur, M. Callaghan Michael, m'a recommandé de nombreuses références afin d'approfondir mes acquis. Je me réjouis déjà à la perspective d'étudier ces informations pendant la fin de cet été.

En conclusion, ce stage a été particulièrement formateur et motivant pour la suite de mes études. Il a confirmé mon projet d'orientation dans le domaine du jeu vidéo que je vais pouvoir concrétiser à Chicoutimi à partir du mois de septembre.

GLOSSAIRE

AR : Réalité Augmentée. Une technologie superposant des éléments virtuels (images, vidéo, hologramme…) sur le monde réel, créant ainsi une expérience interactive et immersive.

VR : Réalité Virtuelle. Une technologie créant un environnement virtuel immersif remplaçant la perception de l'utilisateur par celle simulée par les dispositifs (casque, manette …)

Steam : Une plateforme en ligne de distribution de jeux vidéo et de contenu digital développée par Valve.

Unity : C'est une plateforme de développement de jeux vidéo qui permet aux créateurs de concevoir, construire et déployer des expériences interactives sur différentes plateformes. C'est un outil puissant et populaire dans l'industrie du jeu.

Asset : Dans Unity, un asset désigne tout élément de contenu utilisé dans le développement de jeux, tel que les modèles 3D, les textures, les sons, les scripts …

FPS : Le FPS ou « Frame Per Seconds » est un indicateur désignant le nombre d'images affichées à l'écran par seconde. Cette valeur est utilisée pour évaluer les performances graphiques et la fluidité d'une application. Plus cette valeur est grande, plus le mouvement à l'écran est fluide.

Mesh : Un mesh est une structure composée de points connectés par des arêtes pour former une représentation géométrique d'un objet ou d'une surface en 3 dimensions.

Normal map : Une normal map est une texture spéciale utilisée pour créer l'illusion de reliefs et de rugosités sur des objets 3D sans avoir à modifier son mesh.

Occlusion oculling : L'occlusion oculling est une technique de rendu visant à ne pas dessiner les objets qui sont cachés ou occultés par d'autres éléments et permettant ainsi d'optimiser les performances.

ANNEXES

Annexe 1 : Projets réalisés au travers du programme d'apprentissage « Create with code » 25
Annexe 2 : Schéma officiel de Unity sur l'analyse des performances d'un jeu 33
Annexe 3 : Architecture logicielle utilisant les « ScriptableObjects » de Unity 34
Annexe 4 : Code d'un modèle de Singleton facilement utilisable 35

TABLE DES FIGURES

Figure 1: Logo de finaliste de la compétition GALA 2022 6
Figure 2: Montage montrant les jeux gagnants de la catégorie étudiant du GALA 2022 6
Figure 3: Photo du Meta Quest 2 7
Figure 4: Photo de l'HTC Vive 7
Figure 5: Capture d'écran du tableau Unity sur Trello 8
Figure 6: Capture d'écran du jeu GetCoocked 9
Figure 7: Capture d'écran de Number & Gun 10
Figure 8: Shéma UML des classes de base du XR Interaction Toolkit 12
Figure 9: Capture d'écran du jeu VR Archery 12
Figure 10: Schéma explicatif du montage de l'expérience 13
Figure 11: Capture d'écran des directives de compilation du projet sous Android 15
Figure 12: Capture d'écran du changement visuel du plan d'eau (à gauche l'ancienne version, à droite la nouvelle) 16
Figure 13 : Capture d'écran de l'analyser de Unity en mode éditeur sur la scène Plage 180 de "Numbers & Letters" 17
Figure 14: Capture d'écran du champ de maïs avant optimisation 18
Figure 15: Capture d'écran du champ de maïs après optimisation 18
Figure 16: Schéma UML et explicatif de l'implémentation de la localisation dans "Numbers & Letters" 21
Figure 17: Capture d'écran du menu de jeu "Number" traduit en Espagnol 21
Figure 18: Capture d'écran du projet principal de l'Unit 1 25
Figure 19: Capture d'écran du challenge de programmation de l'Unit 1 26
Figure 20: Capture d'écran du projet principal de l'Unit 2 27
Figure 21: Capture d'écran du challenge de programmation de l'Unit 2 27
Figure 22 : Capture d'écran du projet principal de l'Unit 3 28
Figure 23: Capture d'écran du challenge de programmation de l'Unit 3 29
Figure 24: Capture d'écran du projet principal de l'Unit 4 30
Figure 25: Capture d'écran du challenge de programmation de l'Unit 4 30
Figure 26: Capture d'écran du menu de démarrage et de l'écran de jeu du projet principal de l'unit 5 31
Figure 27: Capture d'écran du menu de démarrage et de l'écran de jeu du challenge de programmation de l'unit 5 32
Figure 28: Schéma du fonctionnement d'un ScriptableObject en tant que plateforme de distribution d'événements 35

ANNEXES

Annexe 1 : Projets réalisés au travers du programme d'apprentissage « Create with code »

Chacun des projets suivants est constitué de 3 phases. Tout d'abord, la phase d'apprentissage sur un projet fourni par le tutoriel. On doit ici suivre les vidéos de présentation afin d'arriver au résultat attendu. Ensuite la phase du défi de programmation où l'on doit par nous-même corriger des erreurs présentes dans un autre projet en rapport avec ce que l'on a appris précédemment. Enfin, la phase bonus du tutoriel ou plusieurs fonctionnalités à implémenter sont proposées allant de difficulté facile à expert.

1.1. Projet Unit 1 : Player Control

Ce premier tutoriel m'a appris à utiliser les bases du système d'input sur Unity. J'ai ainsi créé pendant la phase d'apprentissage un petit jeu de voiture où l'utilisateur conduit sur une autoroute et percute des caisses sur sa route.

Le défi de programmation associé à ce tutoriel est un simulateur de vol réutilisant avec quelques différences le système d'input utilisé pour le jeu de voiture.

Pour ce qui est des fonctionnalités bonus proposées par le tutoriel, j'ai implémenté les suivantes par ordre de difficulté :

- L'ajout de plus d'obstacles sous différentes formes (pyramides, colonnes …)
- L'ajout d'un véhicule venant d'en face (pour créer une collision à toute vitesse avec le joueur)
- L'ajout d'un mode permettant de changer la position de la caméra (vue de derrière, vue de devant)
- Un mode multi-joueurs local avec écran séparé.

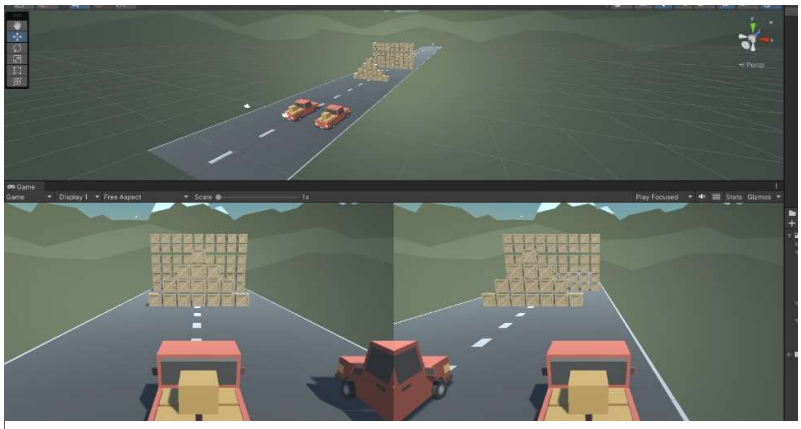


Figure 18: Capture d'écran du projet principal de l'Unit 1

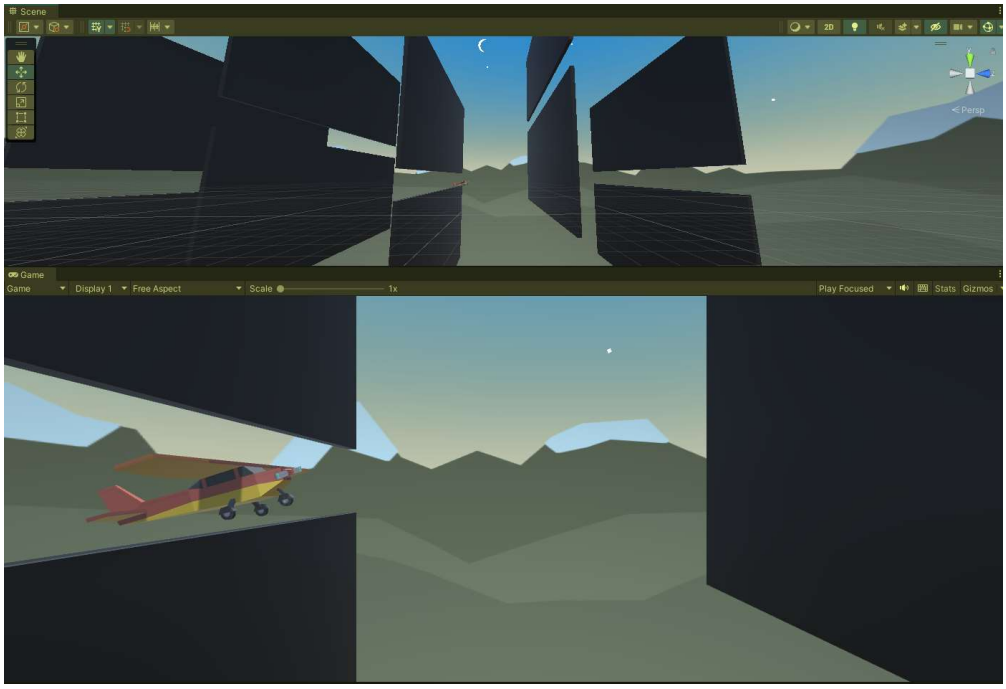


Figure 19: Capture d'écran du challenge de programmation de l'Unit 1

1.2. Projet Unit 2 : Basic Gameplay

Ce second tutoriel m'a appris à mettre en place mes premières mécaniques de jeu sur Unity. J'ai ainsi créé pendant la phase d'apprentissage un petit jeu de tir en vue du dessus où le but est de nourrir des chiens provenant de toutes les directions avec des projectiles sous forme d'os, et ceci, en faisant attention à ne pas se faire toucher par l'un des animaux.

Le défi de programmation associé à ce tutoriel est un jeu de précision où il faut faire s'élancer un chien au bon moment pour qu'il attrape une balle tombant du ciel.

Pour ce qui est des fonctionnalités bonus proposées par le tutoriel, j'ai implémenté les suivantes par ordre de difficulté :

- Diversification des déplacements du joueur (haut, bas, gauche, droite)
- Apparition d'animaux de tous les côtés et provoquant la fin du jeu en cas de contact avec le joueur
- Ajout d'un champ avec le nombre de vie du joueur et un message en cas de fin du jeu
- Ajout d'une barre de point de vie au-dessus des animaux.

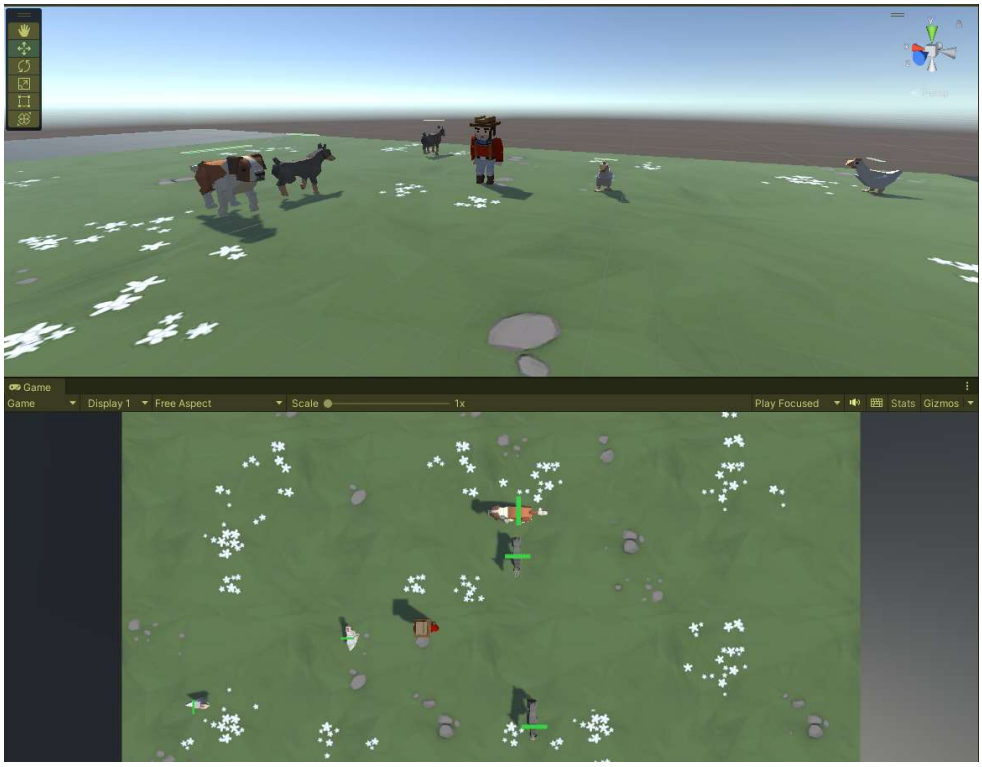


Figure 20: Capture d'écran du projet principal de l'Unit 2

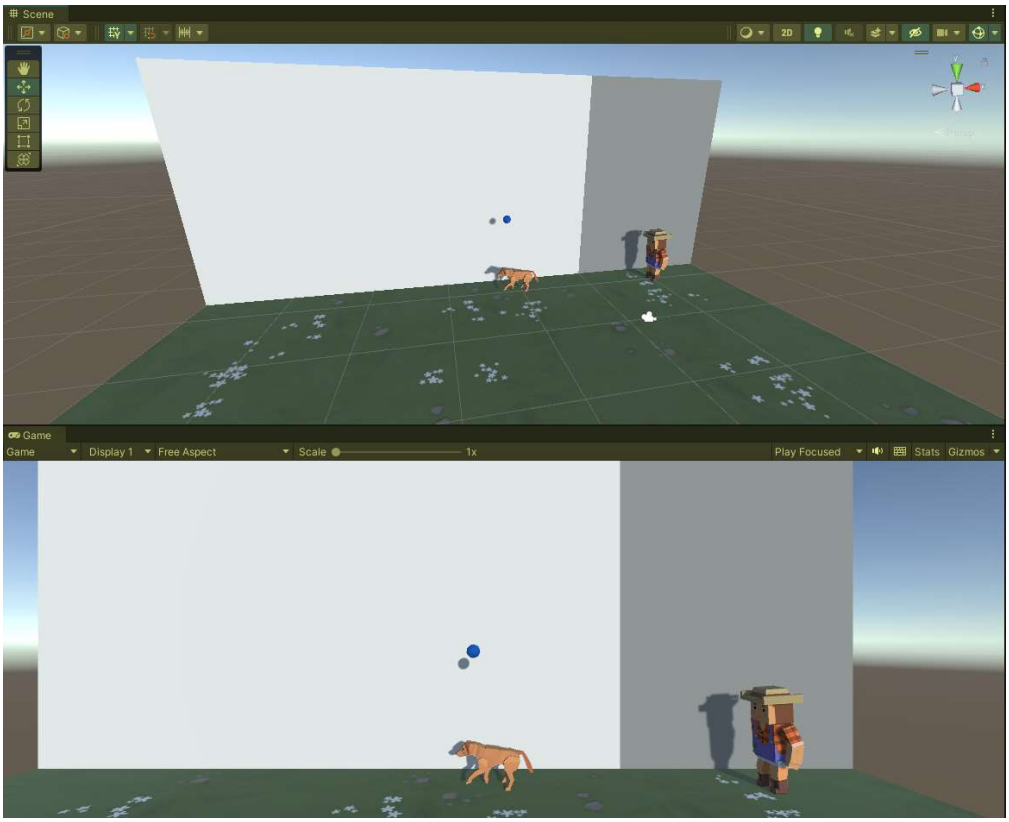


Figure 21: Capture d'écran du challenge de programmation de l'Unit 2

1.3. Projet Unit 3 : Sound and Effects

Ce troisième tutoriel m'a appris le fonctionnement des sons, des animations, et des effets de particules sur Unity. J'ai ainsi créé pendant la phase d'apprentissage un jeu de course en 2D vue de face où le but est d'éviter les obstacles apparaissant sur la route. Le jeu possède sa propre musique, un décor défilant en arrière-plan et se répétant en boucle, et des effets sonores et visuels se déclenchant lors des actions principales. (en courant, en sautant, en touchant un obstacle)

Le défi de programmation associé à ce tutoriel est un jeu similaire au précédent où l'on contrôle un ballon de baudruche auquel on peut donner une force et qui doit collecter le plus possible de dollars sans se faire toucher par une bombe.

Pour ce qui est des fonctionnalités bonus proposé par le tutoriel, j'ai implémenté les suivantes par ordre de difficulté :

- Hasardisation des obstacles (positions, type d'obstacle …)
- Ajout d'un double saut
- Ajout d'un déplacement rapide ainsi qu'un score
- Ajout d'une animation de début de jeu (entrée en scène du joueur)

Figure 22 : Capture d'écran du projet principal de l'Unit 3

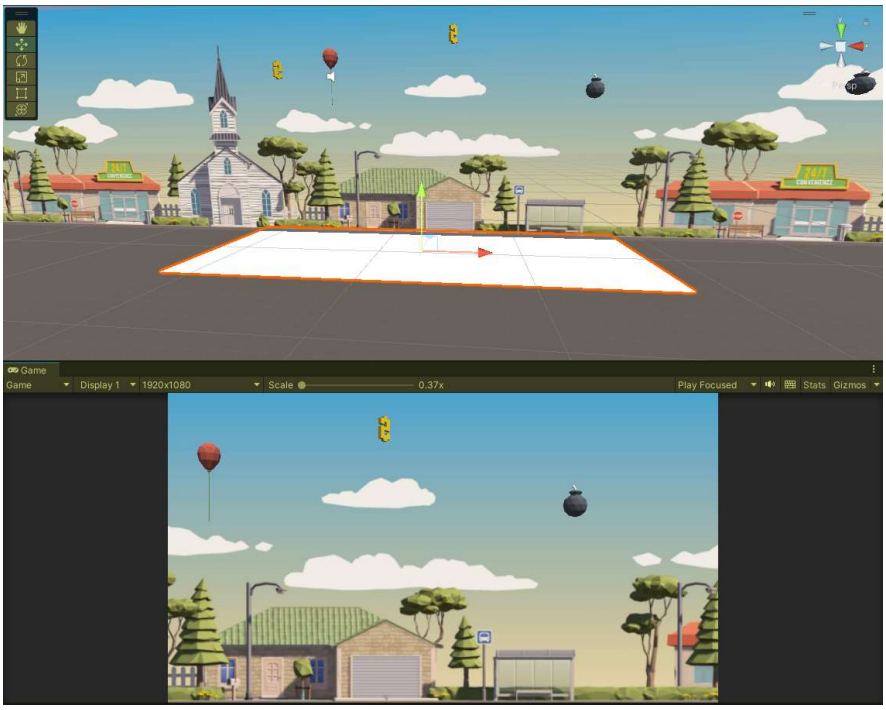


Figure 23: Capture d'écran du challenge de programmation de l'Unit 3

1.4. Projet Unit 4 : Gameplay Mechanics

Ce quatrième tutoriel m'a appris à créer des systèmes de management de jeu et des consommables sur Unity. J'ai ainsi créé pendant la phase d'apprentissage un jeu de sumo en 3D vue du dessus où le but est d'éjecter les autres participants en dehors de la plateforme de jeu. Le joueur doit survivre à des vagues d'adversaires successives et peut se servir de super pouvoir apparaissant aléatoirement sur le terrain. Le système de combat se base sur la physique et le système de collision de Unity.

Le défi de programmation associé à ce tutoriel est un jeu utilisant les mêmes mécaniques de combat que le jeu précédent. Le joueur cette fois-ci est sur un terrain de foot. Les balles adversaires ont pour but de franchir les buts du côté du joueur tandis que le joueur doit les repousser dans les cages en face.

Pour ce qui est des fonctionnalités bonus proposées par le tutoriel, j'ai implémenté les suivantes par ordre de difficulté :

- Ajout d'un nouveau type d'ennemis plus puissants
- Ajout d'un missile téléguidé pour le joueur
- Ajout d'un super pouvoir faisant s'écraser le joueur au sol repoussant les ennemis aux alentours

- Ajout d'un boss utilisant un automate fini pour contrôler ses états. (Invoque de nouveaux ennemis, poursuit le joueur, attaque repoussante…)

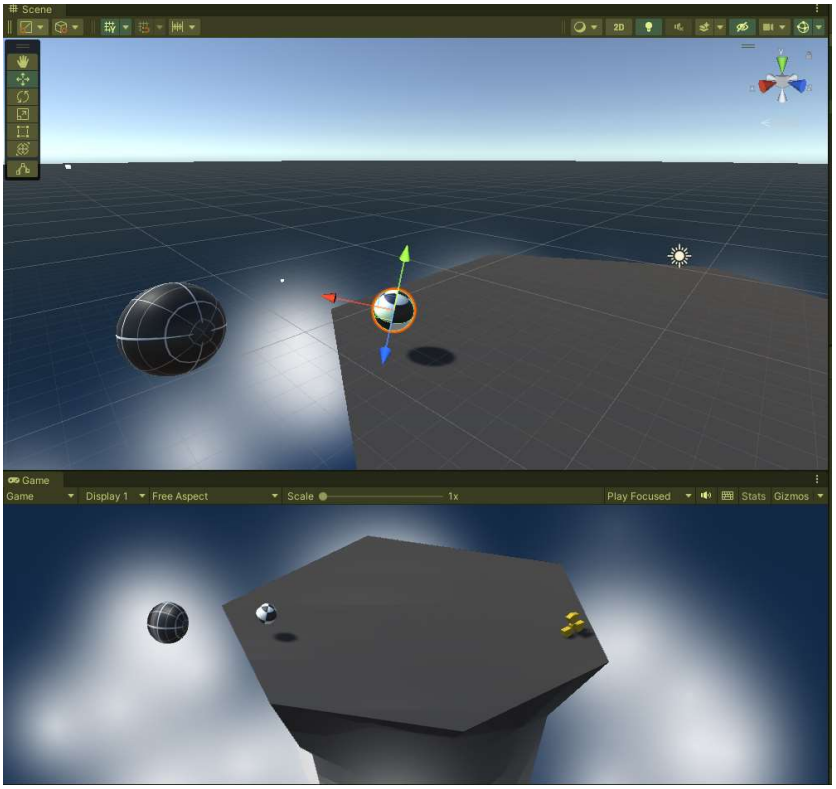


Figure 24: Capture d'écran du projet principal de l'Unit 4

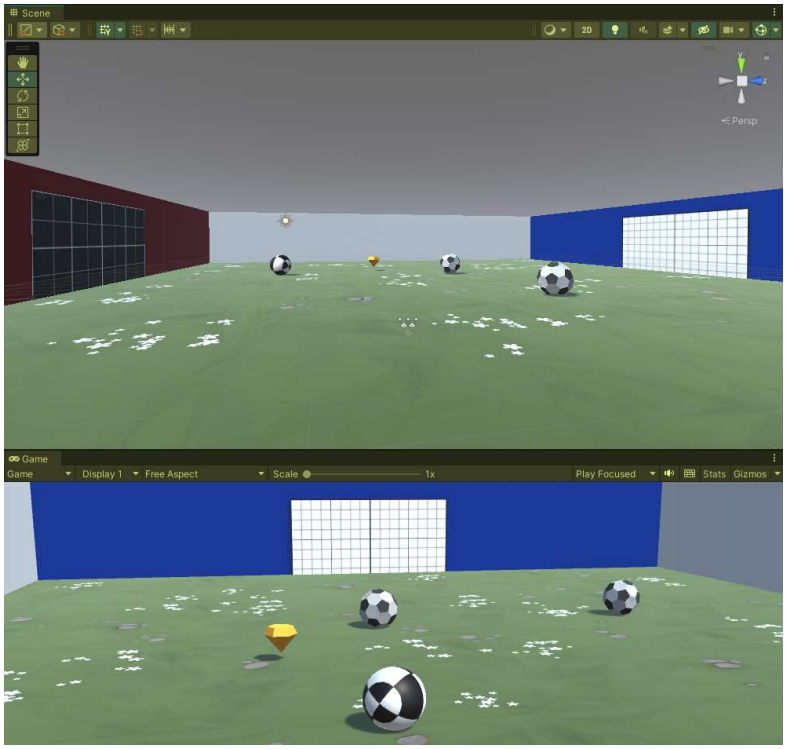


Figure 25: Capture d'écran du challenge de programmation de l'Unit 4

1.5. Projet Unit 5 : Gameplay Mechanics

Ce cinquième tutoriel m'a appris à créer une interface utilisateur sur Unity avec un écran titre proposant différentes difficultés au jeu, une zone d'affichage du score du joueur, et un écran de fin de partie permettant notamment de rejouer. L'application que j'ai créée pendant la phase d'apprentissage est un jeu de réflexe où le joueur doit cliquer sur des boites lancées aléatoirement dans les airs avant que celles-ci ne sortent de l'écran.

Le défi de programmation associé à ce tutoriel est un jeu en 2D où le but est de cliquer sur de la nourriture apparaissant sur une grille dans le temps imparti et en faisant attention à ne pas cliquer sur une tête de squelette.

Pour ce qui est des fonctionnalités bonus proposées par le tutoriel, j'ai implémenté les suivantes par ordre de difficulté :

- Ajout d'un champ de texte affichant le nombre de vie du joueur
- Ajout d'un slider permettant de changer le volume du jeu
- Ajout d'un menu pause arrêtant la progression du jeu
- Ajout d'un système permettant de couper les caisses en faisant glisser sa souris dessus (click and swipe)

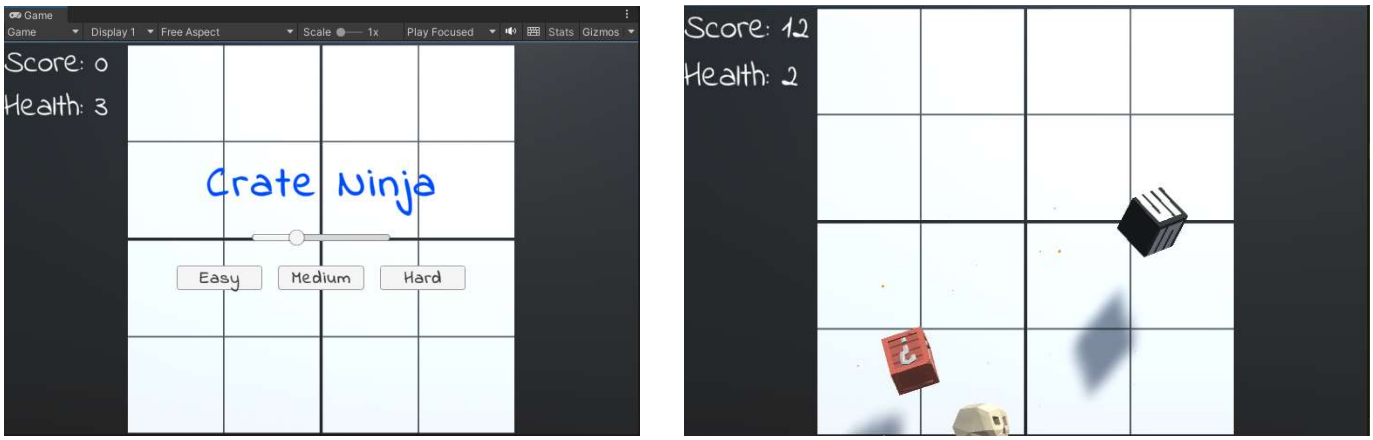


Figure 26: Capture d'écran du menu de démarrage et de l'écran de jeu du projet principal de l'unit 5

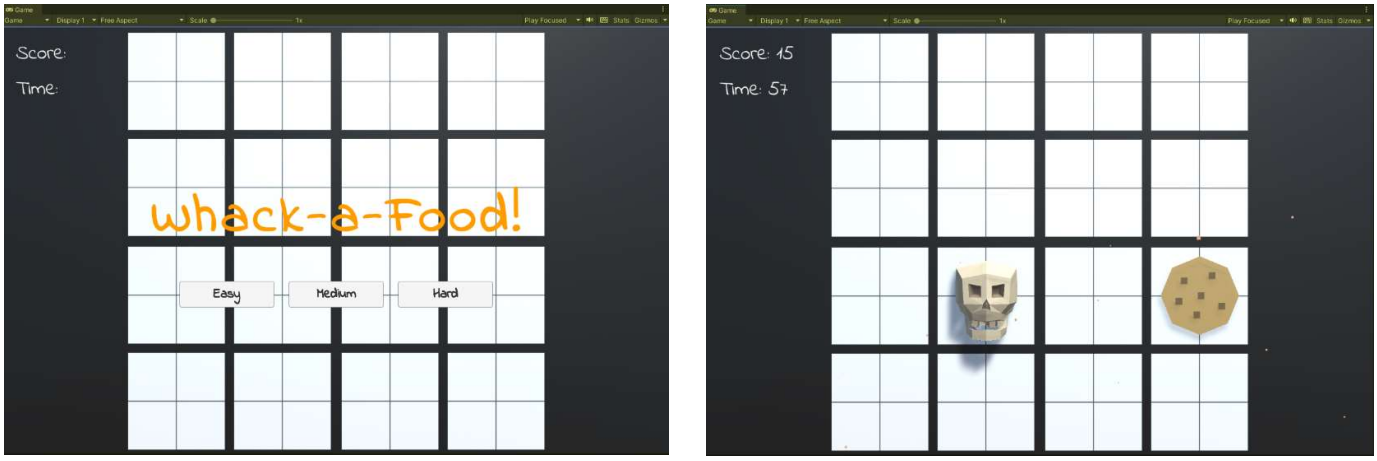
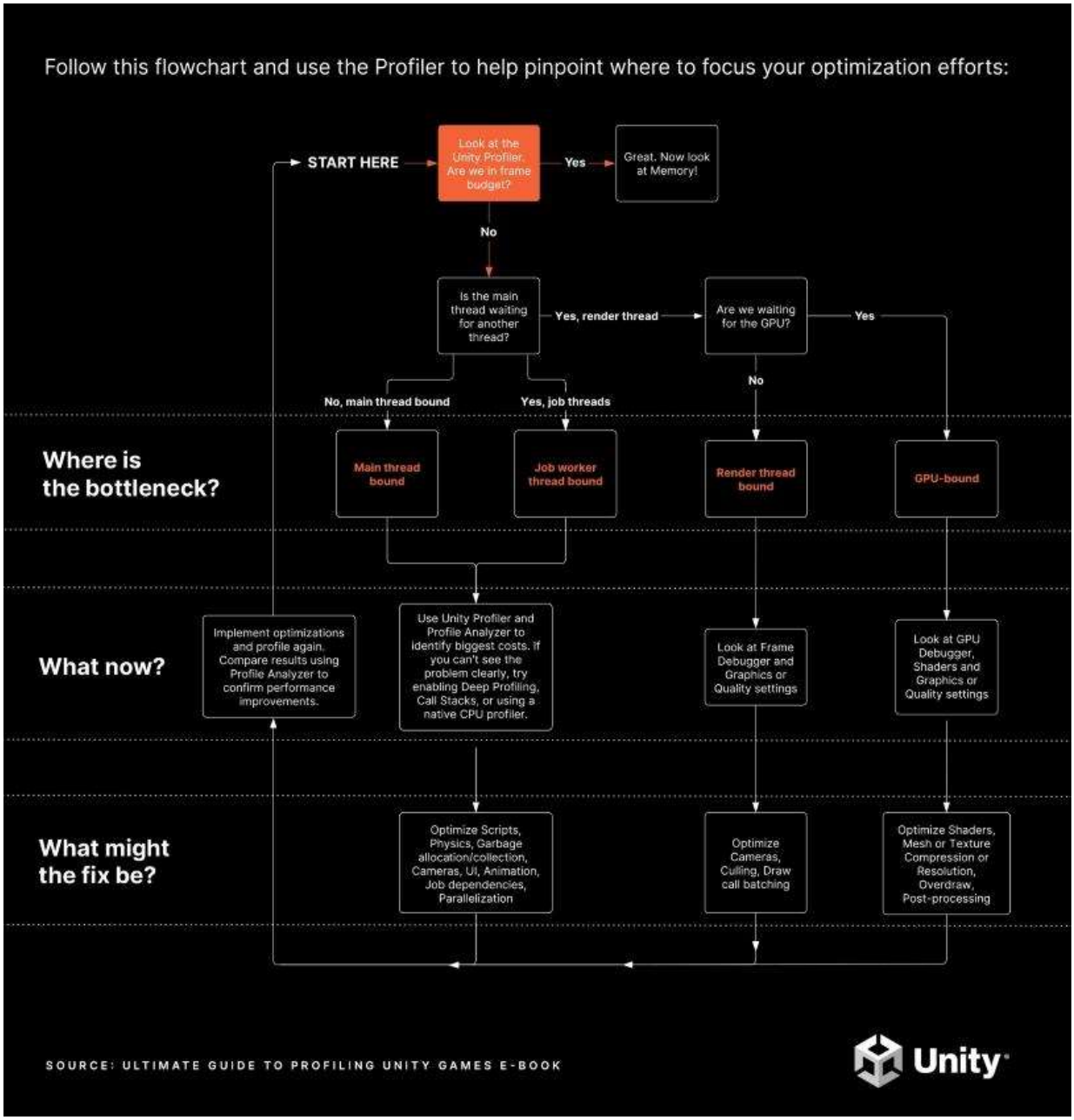


Figure 27: Capture d'écran du menu de démarrage et de l'écran de jeu du challenge de programmation de l'unit 5

Ce projet conclut le tutoriel de Unity « Create with Code ». Tous les exercices complétés peuvent être consultés sur ma page Github à l'adresse suivante : https://github.com/JojoGelb/UnityCreateWithCode

Annexe 2 : Schéma officiel de Unity sur l'analyse des performances d'un jeu



Annexe 3 : Architecture logicielle utilisant les « ScriptableObjects » de Unity

« ScriptableObjects » est une classe dérivable de Unity utilisable lorsque l'on souhaite créer un objet qui n'a pas besoin d'être attaché à un élément de la scène de jeu.

Elle est particulièrement pratique pour stocker des données qui sont ensuite facilement modifiables au travers de la fenêtre d'inspecteur du logiciel. Elle possède également comme particularité de conserver les valeurs entre le mode éditeur et en partie facilitant ainsi, entre autres, les ajustements de valeurs uniquement possibles en direct.

Un problème qui est souvent observé dans les projets Unity est la difficulté qu'ont certains composants à communiquer entre eux au sein d'une scène. La solution la plus simple souvent utilisée est l'implémentation d'un singleton qui, bien que facilitant l'accès à la classe en question, a pour effet négatif de créer de solides dépendances entre des objets empêchant ceux-ci d'exister dans une scène l'un sans l'autre. C'est souvent avec ce type d'implémentation qu'on se retrouve à devoir importer une dizaine de singletons et de classes liés entre elles afin d'utiliser une fonctionnalité qui n'a pas forcément besoin d'autant de composants. L'implémentation d'un patron d'architecture MVC en utilisant les ScriptableObjects comme modèle peut être une solution à ce problème. On peut ici plus facilement contrôler l'injection des dépendances et garder une trace de tous nos composants de notre scène.

Un exemple d'utilisation des ScriptableObjects que je souhaiterais détailler est celui lié à son utilisation en tant qu'interface/bus d'évènements. Dans ce cas, le ScriptableObjects sert de plateforme relayant un évènement d'une classe vers toute autre classe souhaitant écouter celui-ci.

Prenons l'exemple d'une classe joueur possédant une variable entière représentant sa vie. Lorsque celle-ci est modifiée, la classe déclenche un événement notifiant toutes les classes abonnées de ce changement tel qu'une barre de vie, un manager de son ou encore des ennemies réagissant différemment selon le nombre de points de vie du joueur. L'utilisation d'un ScriptableObject dans ce cas permet d'éviter de forts couplages entre ces classes. En effet, au lieu de s'abonner au joueur, les classes écoutant l'événement vont s'abonner au ScriptableObject gérant l'événement de changement de vie. Le joueur peut ainsi déclencher cet événement sans avoir à se soucier de l'existence de script écoutant celui-ci tandis que les scripts écoutant cet événement n'ont pas besoin de connaître la référence du joueur directement. Le ScriptableObject n'étant pas rattaché à un objet en jeu, il persiste entre les scènes garantissant ainsi son existence. Cette architecture permet notamment de facilement ajouter ou retirer des fonctionnalités nécessitant de connaître les changements de points de vie du joueur, et ce, sans impacter le fonctionnement de la classe. Elle permet également de tester individuellement la réponse des classes à cet événement sans avoir à créer de joueur.

Cette implémentation astucieuse respecte les critères SOLID avec :

- Responsabilité unique : la classe ne sert qu'à communiquer l'événement et retire la gestion de celui-ci à la classe joueur
- Ouverture aux extensions et fermeture aux modifications : On peut facilement ajouter de nouvelles fonctionnalités sans avoir besoin de connaître les classes qui se trouvent derrière.
- Ségrégation d'interface : La seule dépendance entre les classes ici est l'événement.
- Inversion des dépendances : La classe ne nécessite ni de déclencheur ni de receveur pour être opérationnel et ne dépend nullement du contexte de la scène grâce à l'implémentation du ScriptableObject.

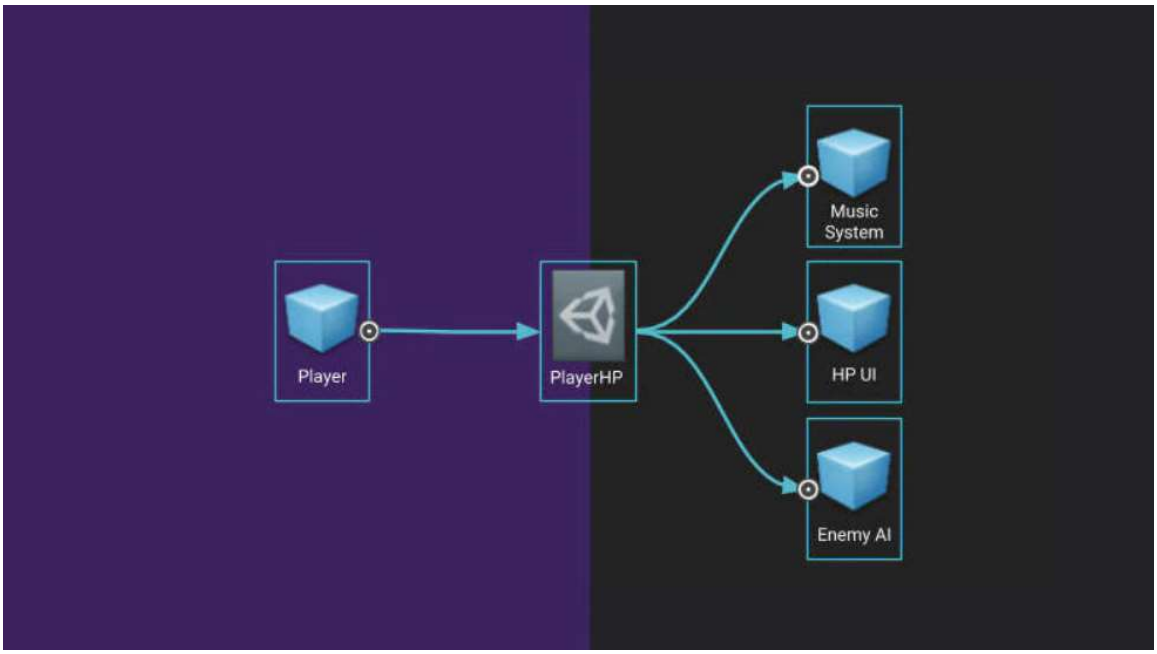


Figure 28: Schéma du fonctionnement d'un ScriptableObject en tant que plateforme de distribution d'événements

Annexe 4 : Code d'un modèle de Singleton facilement utilisable

```
/*
 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