

Bachelor of Science HE-ARC in Engineering

Orientation : Développement Logiciel et Multimédia (DLM)

Flipp3r - 202



Réalisé par :

Bruno Costa

Sous la direction de :
Prof. Benoît Le Callennec
Haute Ecole Arc, HES-SO

Expert :

Dr Sylvain Cardin
Chef de groupe, Laboratoire de muséologie expérimentale, EPFL

28 juillet 2022

Remerciements

Je souhaite remercier avant tout mon collègue et partenaire d'école avec qui j'ai travaillé d'arrache-pied durant la durée du projet, Diogo Lopes da Silva.

Je remercie ensuite nos superviseurs, M. Benoît Le Callennec pour son soutien et son aide, M. Stéphane Gobron pour ses bons conseils.

Merci à M. Max Monti pour la supervision générale de ce projet, à Ricardo Volpe pour l'aide fourni sur les parties électroniques, à M. David Grunenwald pour sa confiance et son expertise. Merci au menuisier, Jean-Marie Liengme, d'avoir créé un flipper aussi beau.

Et finalement, merci aux artistes, Marc Ferrario et Christopher Lanza, pour cette expérience hors du commun et pour cette aventure.

Résumé

Ce rapport traite du parcours emprunté pour le développement du jeu Flipp3r mandaté par l'artiste Neuchâtelois Mandril. L'objectif est de créer un flipper à trois joueurs projeté sur une installation physique où du dessin sera réalisé afin de créer un effet de profondeur : du dessin augmenté. Les règles de ce flipper sont simples : il faut abattre le boss en visant ses points faibles disponibles sur le terrain. Néanmoins, comme c'est un flipper à trois joueurs, le flipper est sectionné en trois zones où la gravité y est respectivement différente et où les balles peuvent librement se déplacer.

Le jeu est dans un état stable et fonctionnel et le cahier des charges est respecté. Le projet est également prêt à accueillir de futures améliorations et autres concepts de nouveau gameplay.

Abstract

This report deals with the path taken for the development of the game Flipp3r commissioned by the Neuchâtel artist Mandril. The objective is to create a three player pinball game projected on a physical installation where drawings will be made in order to create a depth effect : augmented drawing. The rules of this pinball game are simple, you have to kill the boss by aiming at his weak points available on the field. However, as it is a three player pinball game, the pinball machine is divided into three zones where the gravity is respectively different and where the balls can freely move.

The game is in a stable and functional state and the specifications are met. The project is also ready for future improvements and new gameplay concepts.

Table des matières

1	Introduction	1
1.1	Mise en contexte	1
1.2	Résumé du projet	2
1.3	Cahier des charges	2
2	Gestion de projet	3
2.1	Communication et documentation	3
2.2	Planification et gestion des sprints	4
2.2.1	Issues, milestones et board	4
2.3	CI / CD	5
2.4	Journal de travail - Toggl Track	6
2.5	Répartition des tâches	6
2.6	Outils et technologies utilisées	6
3	État de l'art	7
3.1	Flipper ou Pinball	7
3.2	Animation 2D de la bille	8
3.3	Détection de collision d'objets rigides	9
3.3.1	Détection de collision discrète	9
3.3.2	Détection de collision continue	9
3.3.3	Solution	9
3.4	Effets spéciaux en temps réel	10
4	Analyse	11
4.1	Contexte du projet	11
4.2	Spécifications et besoins	11
4.2.1	Historique	11
4.2.2	Jeu et Gameplay	12
4.3	Faisabilité	13
4.4	Risques	14
4.5	Tests et validations	15
5	Conception	17
5.1	Concept du jeu	17
5.2	Concepts	17
5.2.1	Collisions	17
5.2.2	Animation	24
5.2.3	Effets spéciaux	25
5.2.4	Audio	26

6	Implémentation	27
6.1	Collisions	27
6.2	Gameplay	28
6.2.1	Flippers	28
6.2.2	Bumpers	28
6.3	Animation	29
6.4	Effets visuels	30
6.4.1	Système de particules	30
6.4.2	VFX Graph	32
6.5	Audio	34
6.6	Intégration	34
7	Tests	36
7.1	Tests sur les collisions	36
7.2	Tests de gameplay	36
7.2.1	Tests utilisateurs	36
7.3	Tests sur les effets spéciaux et les lumières	37
7.4	Tests audio	37
7.5	Tests d'endurance	37
8	Résultats	38
8.1	État du programme	38
8.2	NIFFF	38
8.2.1	Validation utilisateurs	38
9	Limitations et perspectives	40
9.1	Limitations	40
9.1.1	Problèmes encore présents	40
9.1.2	Gameplay	40
9.2	Perspectives	40
9.2.1	Améliorations	40
10	Conclusion	42
Table des figures		44
Bibliographie		46

Chapitre 1

Introduction

Ce rapport contient toutes les informations et toutes les étapes importantes à la réalisation du projet de Bachelor "Flipp3r" : l'analyse, la conception, l'historique de réflexion ou encore la réalisation y sont détaillés.

1.1 Mise en contexte

Chaque année, la HE-Arc collabore avec le NIFFF, Neuchâtel International Fantastic Film Festival, et un artiste régional pour créer une installation qui y sera présentée. Cette fois, c'est avec Marc Ferrario, l'artiste Neuchâtelois du nom de Mandril, que l'école a décidé de travailler. Son idée est de créer un flipper où serait mélangé l'art contemporain, le projection mapping, le dessin augmenté et la technologie. En plus d'être un objet d'art unique, le flipper doit avoir un gameplay dynamique, une interaction intéressante et une certaine particularité : trois joueurs. En 2019, la Haute École Arc a déjà collaboré avec Mandril pour créer l'installation Figura [1]. Grâce à celle-ci, Marc a déjà expérimenté le projection mapping et le dessin augmenté mais uniquement dans le purement contemplatif. C'est là que les étudiants de la HE-Arc, Bruno Costa et Diogo Lopes da Silva, entrent en jeu. Leur objectif est de participer à la création du jeu Flipp3r, non seulement dans son implémentation mais aussi dans sa conception. Cependant, pour créer une installation de cette ampleur, les étudiants ne peuvent pas tout faire. L'installation en bois est créée par des menuisiers recrutés pour l'occasion. Et un ami de Mandril, Christopher Lanza, travaille sur les modèles 3Ds et la création sonore du jeu.

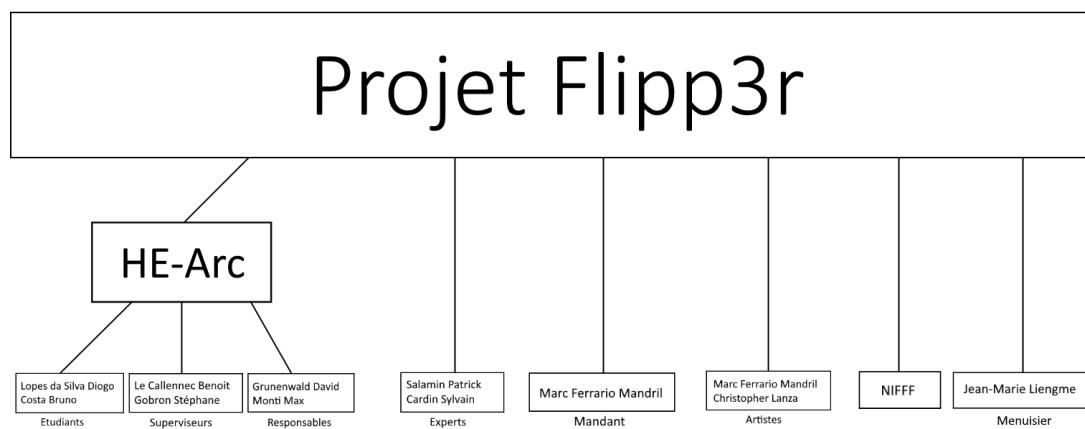


FIGURE 1.1 – Le consortium représentant les membres du groupe.

L'application a été réalisé sur le moteur de rendu temps réel Unity et implémenté en langage C#, de plus, une gestion de l'installation physique du flipper ainsi qu'une administration des projecteurs ont été effectuées.

1.2 Résumé du projet

L'objectif du projet est de réaliser un jeu de flipper à trois joueurs entièrement projeté sur une plateforme en forme de L ressemblant à un flipper.

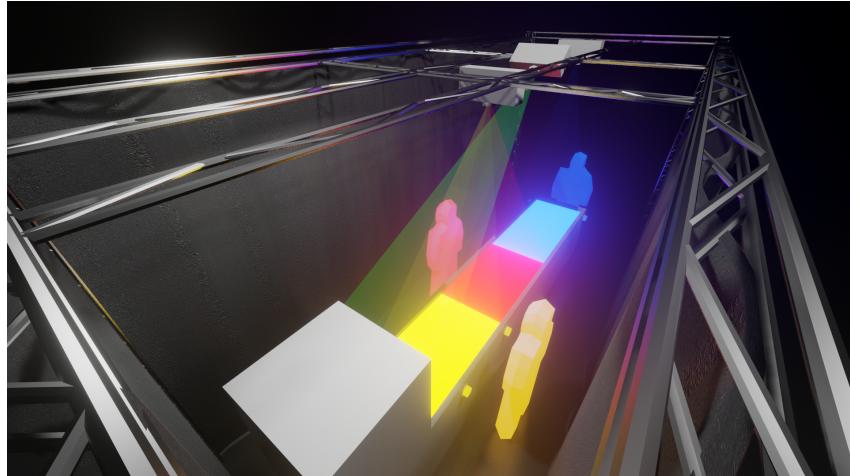


FIGURE 1.2 – Le 1er prototype du Flipp3r.

L'implémentation d'un jeu de flipper unique, d'un système de lumières dynamique, l'ajout de sons et musiques ainsi que la gestion de l'installation physique et de la projection représentent le sujet du projet. De plus, une jouabilité inégalée et un gameplay amusant sont les piliers de l'application : comme le jeu sera utilisé par plusieurs joueurs grand public, il est important qu'il soit apprécié.

1.3 Cahier des charges

Comme le projet est réalisé à deux, nous avons décidé de séparer les différentes tâches et responsabilités selon les affinités des étudiants. Chaque étudiant a rédigé le cahier des charges pour correspondre au projet mandé par Mandril et en prenant en compte les objectifs et tâches séparés plus tôt. Finalement, il a été signé par l'enseignant, l'expert, le mandant et l'étudiant concerné. La version PDF de ce cahier des charges est disponible en annexe de ce rapport.

Chapitre 2

Gestion de projet

Le projet Flipp3r est un travail de bachelor complexe quant à son organisation et sa gestion. En effet, comme démontré dans le consortium illustré plus haut, plusieurs acteurs ont pris part à la réalisation de l'installation. Que ce soit les menuisiers qui ont construit la structure en bois, les artistes qui ont créé le design du jeu ou encore les étudiants qui l'ont implémenté, cette grande quantité de rôles implique une gestion de projet et une communication importantes et délicates.

2.1 Communication et documentation

Pour la communication intergroupe et intra-groupe, plusieurs moyens et technologies ont été utilisés. Un serveur discord regroupant toutes les informations sur le projet a été créé au début du semestre. Les liens, informations et autres démonstrations sont disponibles sur les différents canaux de discussions. De plus, les pages Wiki du GitLab ont constamment été mises à jour pour contenir toutes les dernières informations.

Cependant, les artistes ont préféré communiquer sur WhatsApp par question de simplicité, les étudiants ont alors pris l'habitude de transférer les informations pertinentes reçues par téléphone sur le serveur discord afin de regrouper toutes les informations du projet en un seul lieu.

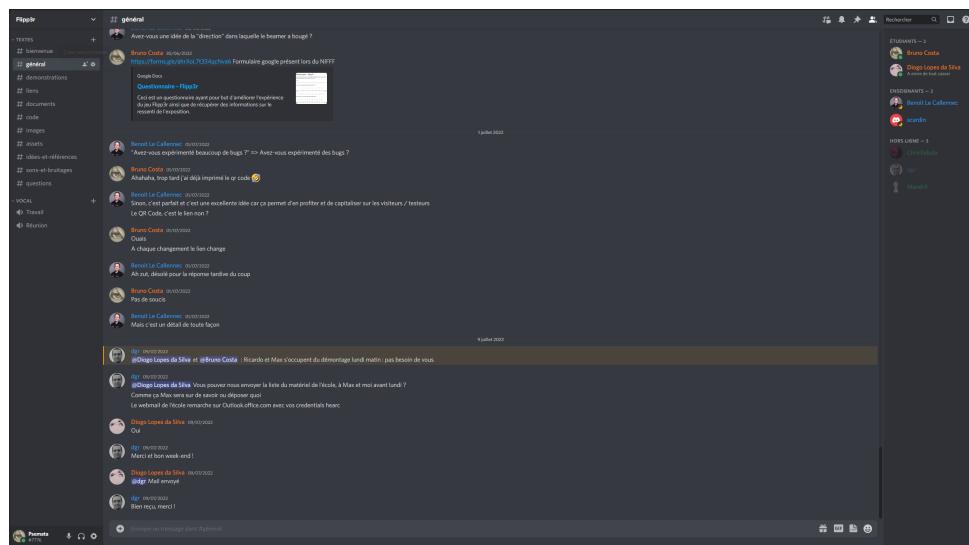


FIGURE 2.1 – Le serveur discord utilisé.

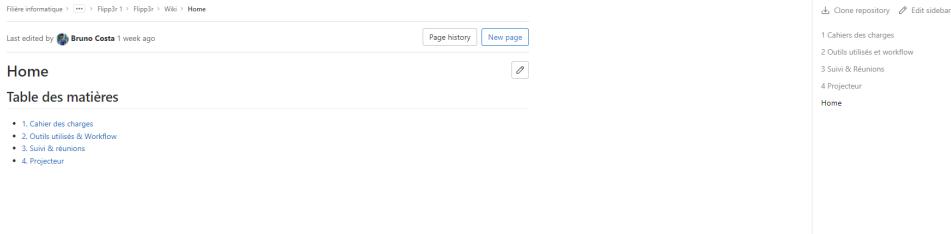


FIGURE 2.2 – Le Wiki GitLab utilisé.

2.2 Planification et gestion des sprints

Afin de rester flexible, les étudiants ont décidé de travailler de manière itérative et ainsi de fonctionner via des sprints. La célèbre technique agile "Scrum" n'a pas été utilisée parce que le groupe est trop petit pour qu'elle soit efficace.

Au début du projet, les artistes et les étudiants se rencontraient toutes les deux semaines afin de montrer l'avancée et de discuter de la suite. Pour chacune de ces rencontres, une vidéo de démonstration et un PV de réunion étaient préparés.

Lors du passage au plein-temps, une grande réunion regroupant tous les acteurs du projet a été dirigée par les étudiants afin d'établir une organisation poussée et détaillée de la fin du projet. Les sprints sont passés de 2 à 1 semaine et les dernières semaines ont été organisées pour pouvoir mieux gérer un potentiel état de crise. De plus, les réunions se sont faites quotidiennement tout au long des derniers sprints. Les PV de réunion quant à eux ont été abandonnées car les réunions étaient trop fréquentes.

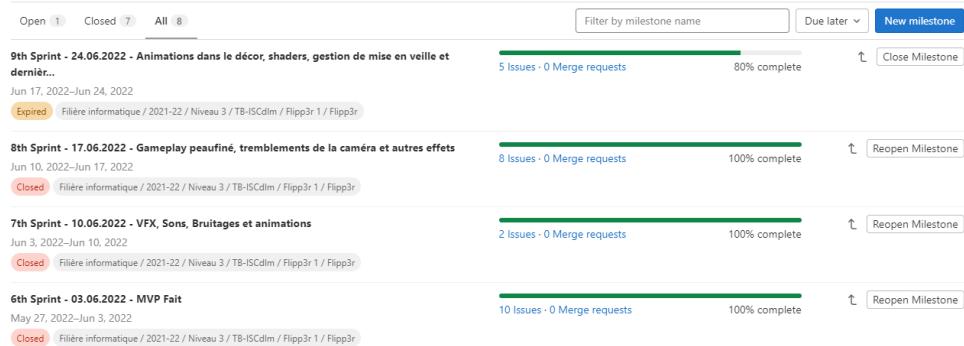


FIGURE 2.3 – La fonctionnalité de milestones de GitLab.

2.2.1 Issues, milestones et board

Un diagramme de Gantt n'est pas un outil adapté pour s'occuper de l'organisation de ce type de projet car le changement y est fréquent. Une tâche créée un jour, peut changer le lendemain. Cependant, il est nécessaire de planifier et organiser les tâches à réaliser. Pour pouvoir faire ceci, les étudiants ont utilisé les fonctionnalités des milestones, des issues et du board de GitLab.

Chaque milestones représentent un sprint et les issues qui y sont liées représentent les tâches. Des tags ont été créés afin de catégoriser et différencier les tâches. Pour finir, le système de kanban "board" de GitLab a été employé pour gérer de manière graphique l'organisation du projet. Les tableaux utilisés sont :

- New - Nouvelles tâches pas encore entamées du sprint

- WIP - Les tâches en cours du sprint
- Review & Testing - Les tâches presque finies qui nécessitent une dernière vérification ou quelques tests
- Done - Les tâches terminées en attente d'être fermées à la fin du sprint

Filière informatique > ... > Flipp3r 1 > Flipp3r > Issues > #44

Open Created 4 weeks ago by  Bruno Costa Owner 7 of 12 tasks completed Close issue ⋮

Bugs existants

La balle se déplace dans le portail (Sa vitesse n'est pas annulée)

Les liens électriques restent un instant lors de la mort des balles et essaient d'accéder à un gameObject détruit

La gestion des portails des rampes ferment les entrées via des colliders, ils ne ferment pas assez vite et la balle retourne en arrière ce qui la fait se bloquer dans la rampe

La balle se destroy dans la méthode update, dépendant du framerate, il se peut qu'elle trigger la mort 2 x de suite ce qui enchaîne plusieurs bugs

Le bug de la mort multiple peut engendrer un bug de spawn multiples qui fait que plusieurs balles apparaissent au lieu d'une seule

La balle entre dans le piston ou est mal renvoyée par le piston

La balle peut encore passer par un mur ; Création d'une zone mort à l'extérieur

La balle reste l'enfant du flipper à l'éjection et ainsi subi les rotations du flipper même à l'extérieur

À cause du tremblement du bumper, les collisions sont répétées plusieurs fois

Comme la balle suit le long du flipper à la collision, si elle arrive par le bout, elle entre dans le flipper

Balle coincée contre un bumper ou contre une autre balle

Problème de rotation de la caméra au gameover

Edited 2 weeks ago by Bruno Costa

FIGURE 2.4 – Une issue de GitLab.

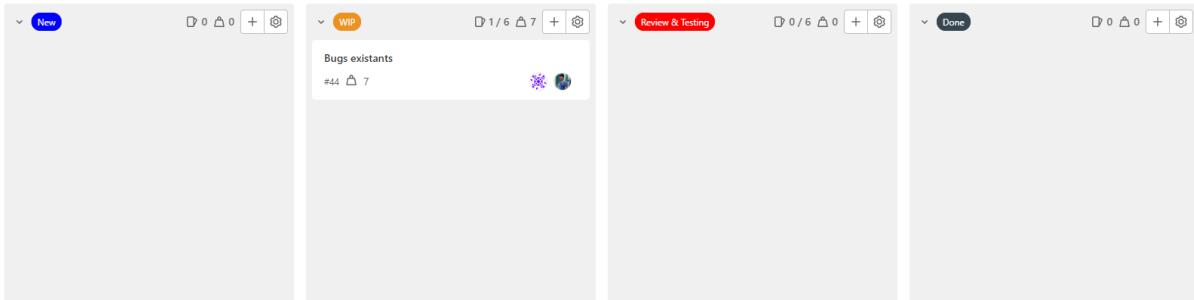


FIGURE 2.5 – La fonctionnalité de board de GitLab.

2.3 CI / CD

Les itérations fréquentes ont demandé des tests fréquents. Un système de CI/CD, Continuous Integration/Continuous Delivery, a donc été mis en place pour pouvoir tester les dernières modifications sur l'installation finale et ainsi assurer un fonctionnement correct et sous conditions réelles. C'est pourquoi le service de runner offert par GitLab a été utilisé et un fichier ".yaml" a été créé pour instaurer une routine de déploiement à chaque push exécuté sur la branche principale du "repository" Git. Différents tutoriels ont été suivis pour mettre en place ce système [2] [3] [4] [5] [6]. De plus, pour assurer une bonne continuité dans le travail, le projet a été organisé de manière à accueillir une intégration simple et fluide. L'utilisation de scènes attribuées et des

prefabs dans Unity d'une part et des mises en commun fréquentes d'une autre ont permis un parcours sans accroches et résultant en une version du programme à jour continuellement.

2.4 Journal de travail - Toggl Track

Le logiciel Toggl Track a été utilisé au lieu de rédiger un simple journal de travail. Grâce à cette application, chaque tâche a pu être chronométrée précisément et un compte-rendu, disponible en annexe, a pu être généré. Ce tableau détaillé permet d'analyser la gestion du temps et l'efficacité tout au long du travail.

2.5 Répartition des tâches

Les tâches ont été réparties par les deux étudiants selon leurs affinités respectives. De cette manière, les deux étudiants ont pu travailler sur les parties qui les intéressaient.

Bruno Costa s'est occupé des parties suivantes :

- Gameplay et environnement
- Gestion des collisions
- Animations
- Effets spéciaux
- Sons

Diogo Lopes da Silva s'est occupée des parties suivantes :

- Projection
- Calibrage
- Shaders
- Gestion du déroulement du jeu
- Lumières
- Gestion du boss

La répartition détaillée est disponible dans les cahiers des charges disponibles en annexe.

2.6 Outils et technologies utilisées

Les outils de développement utilisés sont :

- Unity - 2021.3.2f1
- Visual Studio Code

Unity a été choisi car c'est un logiciel de rendu en temps réel connu des deux étudiants. Quant à sa version, elle a été choisie car, au moment du développement, c'était une des dernières version "LTS" proposée.

Pour les outils de gestion et de communication :

- Git - Versioning du projet
- Discord - Utilisé pour la communication et le transfert de fichier simple
- DropBox - Transfert de fichier important (Principalement entre les artistes et les étudiants)
- Toggl Track - Gestion et mesure du temps de travail
- Zotero - Gestion des sources et de la recherche
- TeamViewer - Logiciel utilisé pour avoir un accès remote à la machine hôte du logiciel

Chapitre 3

État de l'art

À présent, il est facile d'apercevoir la complexité et l'originalité du projet. En effet, un flipper à trois joueurs qui mélange l'aspect "Projection Mapping" et l'interaction réelle des joueurs sur une installation physique est inédit. Afin de mieux cerner les enjeux et ainsi avoir un parcours plus simple, il est important de posséder un bon état de l'art qui indique ce qui a déjà été fait. Comme indiqué dans le précédent chapitre, les tâches auxquelles l'étudiant à contribuer sont la création du flipper, l'animation 2D de la bille, la détection de collision d'objets rigides, les effets spéciaux comme la fumée, les gerbes d'étincelles ou la foudre et finalement les sons. C'est pourquoi chacun de ces thèmes sera exploré dans le but d'édifier un état de l'art correct et utilisable.

3.1 Flipper ou Pinball

Le premier jeu de flipper, ou pinball, a vu le jour en 1871 et a été créé par l'inventeur britannique Montague Redgrave lorsque celui-ci a souhaité améliorer le jeu déjà existant nommé "Bagatelle". Le but du jeu était alors d'utiliser une queue de billard pour envoyer une bille et ainsi générer des points selon l'endroit où elle termine son parcours.



FIGURE 3.1 – Un jeu de bagatelle.

L'intervention de Mr. Redgrave a été d'ajouter un système de ressort et de rétrécir l'objet afin d'obtenir un jeu portable et facilement utilisable. C'est ensuite à partir des années 1930 que les premiers flippers comme nous les connaissons sont apparus, mais sans pieds.

Finalement, c'est en 1932 que les fabricants ont commencé à ajouter des pieds aux machines [7]. Jusqu'à ce jour, le jeu de flipper a bien évolué et de multiples modèles ont été créés. Cependant, bien que quelques expériences aient été faites comme dans le cadre de l'étude d'implémentation



FIGURE 3.2 – L'ancêtre du flipper.

de jeu sur une machine de flipper [8], il n'existe pas de version "multijoueur". De plus, aucune expérience de projection mapping ou de dessin augmenté sur un flipper n'a été trouvée. Il y a bien entendu eu des créations de flippers virtuels comme le célèbre "Space Cadet", mais pas de mélange : jeu virtuel - multijoueur - projection mapping.



FIGURE 3.3 – Le jeu Space Cadet sur Windows XP.

Nos contributions :

- Flipper en projection mapping, interaction réelle sur l'installation et utilisation de dessin augmenté

3.2 Animation 2D de la bille

John Lasseter présente en 1987 12 techniques de l'animation traditionnelle appliquées à l'animation par ordinateur [9]. Ces principes sont présents afin d'aider à créer une animation fluide et

attirante. Pour pouvoir déplacer une bille de manière dynamique le long d'un chemin, comme dans une rampe par exemple, l'idée d'utiliser de l'animation traditionnelle plutôt qu'un déplacement physique, une simulation, est très intéressante. En effet, de cette manière, la bille ne sera jamais coincée dans une rampe et un effet "cartoon" sera présent. Le "Keyframing" est alors utilisé : technique inspirée de l'animation traditionnelle. L'animation est posée à des instants précis, et on interpole les images intermédiaires (in-betweens). Cette interpolation est en général faite grâce à des splines, des fonctions mathématiques comme des courbes de Bézier.

Nos contributions :

- Utilisation des courbes de Bézier.
- Le support des courbes de Bézier sur Unity est uniquement disponible à partir de la version 2022, c'est pourquoi une implémentation triviale a été réalisée pour supporter ces courbes dans la version 2021.

3.3 Détection de collision d'objets rigides

La détection de collision d'objets est un milieu complexe où encore aujourd'hui de multiples recherches sont réalisées. Il existe plusieurs types de détection de collision [10] :

- **Collision d'objets déformables** : entre deux éponges ou objets mous comme des balles en mousse.
- **Collision de l'objet avec lui-même** : une liane bousculée qui se touche elle-même.
- **Collision de tissus** : la simulation de rideaux.
- Et plein d'autres.

Comme le flipper utilise des billes métalliques rigides, c'est la détection de collision d'objets rigides en temps réel qui a été étudiée.

3.3.1 Détection de collision discrète

La détection de collision discrète fonctionne ainsi :

- La simulation physique est avancé pas à pas. Dans notre cas, le déplacement de la bille.
- On vérifie si il existe une collision.
- À chaque pas, une liste de tous les corps s'intersectant est créée.

Le problème de cette façon de faire est qu'elle ne trouve pas le réel instant de collision, mais uniquement le moment **après** que la collision ait eu lieu.

3.3.2 Détection de collision continue

La détection de collision continue utilise un algorithme qui est capable de prédire très précisément la trajectoire des éléments physiques. L'instant de la collision est calculée avec grande précision et les corps ne se rentrent pas dedans car la collision est "prédite". En effet, la prédiction est effectuée **avant** la mise à jour de la simulation physique, ce qui permet de prédire les collisions et non de vérifier s'il en existe une ou non. Technique idéale pour une balle de flipper à grande vitesse.

3.3.3 Solution

L'objectif est de pouvoir contrôler et animer la balle librement. Pour pouvoir faire cela, il faut indiquer à la balle qu'elle n'est plus sous l'influence du moteur physique de Unity. Cependant, cela bloque également l'utilisation de la détection de collision continue. Une solution a alors

été imaginée pour assurer une détection de collision stable et fonctionnelle. Voir le chapitre 5 - "Conception" pour plus de détails.

Nos contributions :

- Utilisation des méthodes de détection de collision offertes par les moteurs de jeu actuels, Unity dans le cadre de ce projet
- Modification de certaines de ces méthodes pour créer un système de détection de collision correspondant aux besoins du projet.

3.4 Effets spéciaux en temps réel

La création d'effets spéciaux en temps réel est spécifique, si on la compare à de la création d'effets en pré-rendu, les techniques sont complètement différentes. En général, le moindre effet visuel, comme de la fumée, est difficile à générer et demande beaucoup de travail. Les moteurs de jeu offrent des moyens de créer simplement des effets spéciaux en temps réel. Unity donne accès au "Particle System" qui permet de donner naissance à des effets simples de particules comme des étincelles, de la fumée ou des explosions. Néanmoins, dès que la demande d'un effet plus complexe apparaît, le "Particle System" ne suffit plus. Il faut alors utiliser le "VFX Graph". "VFX Graph" s'inspire des outils de pointe utilisés dans l'industrie du film et permet de créer une large gamme d'effets et peut gérer des millions de particules sur le GPU sans écrire une ligne de code. De plus, il est actuellement possible de fusionner l'utilisation du "Shader Graph" et du "VFX Graph" pour créer de meilleurs effets visuels.

Nos contributions :

- Utilisation du Particle System pour les effets simples
- Utilisation du VFX Graph pour de multiples effets

Chapitre 4

Analyse

Le projet Flipp3r est complexe et ambitieux, il place des contraintes et des objectifs jamais vus jusqu'ici. En effet, l'état de l'art fait précédemment démontre l'originalité et l'inédit du projet et permet de comprendre son étendu. Grâce aux informations recueillies, il est facile d'analyser quels sont ses spécifications, ses besoins et ses points chauds.

4.1 Contexte du projet

Le projet est un mandat proposé par l'artiste neuchâtelois Mandril. son but est de créer un objet artistique jamais vu sous la forme d'un flipper à trois joueurs. Cette installation doit être présentée au NIFFF, Neuchatel International Fantastic Film Festival, du 2 au 9 juillet 2022. Ce qui induit une date de rendu plus proche que pour les travaux de bachelor conventionnels.

Ce travail de bachelor possède de nombreuses facettes, en plus de devoir implémenter un jeu entier, le projet utilise des projecteurs et des éléments "hardware" spécifiques comme des boutons de flippers ou des enceintes sonores. Comme cité plus haut, les tâches ont été séparées entre les deux étudiants : l'un s'occupe des détails du jeu et l'autre de la partie "hardware".

4.2 Spécifications et besoins

Le projet s'est déroulé de manière itérative, l'état et la direction du jeu changeait donc souvent. Par conséquent, ses spécifications et ses besoins aussi s'en sont également retrouvés modifiés.

4.2.1 Historique

Le jeu a été analysé à de multiples reprises et a changé bien des fois jusqu'à arriver à un état correct. En effet, de nombreuses réunions et discussions avec les acteurs du projet ont résulté en le jeu que nous connaissons aujourd'hui. Cependant, il est intéressant de connaître les différentes étapes que celui-ci a parcourues sous la forme d'un historique.

1. La première étape était une simple idée d'un flipper à trois joueurs avec une gravité pour chaque joueur.
2. Ensuite l'idée était que Unity gère les collisions et la physique, de cette manière, le flipper aurait été séparée en trois parties penchées dans leurs sens respectifs.
3. Le problème venait de la perspective, si nous utilisons trois zones orientées de manières différentes, les points de vue des trois joueurs seraient faussés. C'est pourquoi le jeu est devenu orthographique.

4. Les premiers "concepts art" ont plu à tout le monde, un effet non réaliste et cartoon a alors été décidé. Une vidéo est disponible en annexe.
5. Après de multiples tests, le groupe s'est rendu compte que Unity ne permet pas de détections de collisions d'objets rigides à grande vitesse. Ainsi, une méthode manuelle a été choisie.
6. Les "concepts art" suivants ont introduit un robot faisant office de boss.



FIGURE 4.1 – Le premier concept art du boss du jeu.

7. Le gameplay est alors presque décidé. Une coopération entre les trois joueurs pour battre le robot.
8. Une réunion importante a alors pris place. La structure du plateau et les éléments de jeu.



FIGURE 4.2 – Le premier plateau de jeu.

9. Dès lors, le groupe avait une idée finale des éléments de jeu, du placement du terrain et du gameplay.

4.2.2 Jeu et Gameplay

Dans cette section sera analysé les parties du jeu correspondant au gameplay, les collisions, les animations de la balle et les effets spéciaux. Toutes autres parties concernant le jeu, comme la lumière, les shaders, la projection ou la gestion du boss, sont analysés dans le rapport de l'étudiant Diogo Lopes da Silva.

Gameplay et animations

Le jeu Flipp3r comprend plusieurs éléments de gameplay.

- **Les flippers :** les flippers servent à récupérer les billes et les projeter aux endroits voulus. Ils forment la base du gameplay du jeu. Un flipper est activé lorsque le bouton respectif est pressé.

- **Les slingshots** : les slingshots sont des protections qui permettent d'amortir la balle une fois arrivée sur un flipper. Ils ont également des flippers cachés à l'intérieur activables par les autres joueurs dès que leurs deux boutons sont pressés.
- **Les pistons** : les pistons s'activent en même temps que les flippers et permettent de propulser la bille en direction opposée du piston.
- **Les bumpers** : les bumpers sont des éléments de gameplay sur lesquelles les billes vont rebondir et générer du score au moment de la collision. Si la bille accumule suffisamment de score, elle s'enflammera, ira plus vite et générera plus de score sur les prochaines collisions.
- **Le portail** : le portail permet d'accueillir une bille en son sein et la charge pour la rendre électrifiée. Ainsi la balle ira plus vite et générera plus de score.
- **Les murs** : les murs sont de simples éléments bloquants qui font rebondir la bille.

Finalement, le but du jeu a été décidé par l'ensemble du groupe et les règles sont les suivantes :

- Le boss possède une barre de vie cachée et affiche son pourcentage via son état visible.
- Au début du jeu, des bumpers sur le terrain sont sélectionnés en tant que "faiblesses" du boss.
- En les touchant avec la bille, le boss reçoit des dégâts et va perdre de la vie. Ensuite, de nouveaux bumpers sont choisis.
- Le boss possède 5 phases de destruction, pour passer à la suivante, 3 faiblesses doivent être touchées.
- Ainsi, pour finir le jeu, il faut toucher 15 fois une faiblesse.
- Chaque joueur possède également un score pour voir qui a le mieux joué.
- Si une bille change de terrain, la gravité affecte alors sa vitesse.
- Une bille apparaît dans le terrain des joueurs lorsqu'une déjà présente quitte le terrain ou lorsque la dernière bille du joueur meurt.
- Il ne peut y avoir que 15 balles en tout. Au bout de 15 balles perdues, le jeu est alors perdu, un gameover apparaît.
- Chaque joueur ne peut avoir que 3 balles maximum, ainsi 9 balles maximum sur le terrain.

Collisions

Les collisions du jeu Flipp3r ne sont pas réalistes, la balle ne perd pas de vitesse après chaque collision et rebondit de manière illogique. De plus, le rebond change dépendant sur quoi la balle collisionne : comme sur un flipper ou sur un mur.

Ceci est fait pour avoir accès à un sentiment "cartoonesque" et très dynamique.

Effets visuels

Les effets visuels ont été imaginés pour correspondre au style du décor et aux souhaits du mandataire.

4.3 Faisabilité

Les contraintes du projet Flipp3r sont nombreuses :

1. Une date de rendu proche
2. Un travail à deux étudiants
3. Utilisation de matériel spécifique
 - (a) Projecteurs

(b) Utilisation de boutons physiques

(c) Ordinateur puissant

(d) Structure en bois

4. Projet conséquent

Avec ces contraintes viennent de multiples questions quant à la faisabilité des différentes tâches et les risques du projet. Est-ce que les tâches sont séparées correctement entre les deux étudiants ? Est-ce que la quantité de travail est faisable à deux ? Comme le concept est original, beaucoup de tests doivent être effectués avant de lancer le projet pour de bon. De plus, Unity n'est peut-être pas capable d'offrir le nécessaire pour créer ce type de jeu que ce soit dans le domaine de l'animation ou dans le domaine des collisions.

4.4 Risques

Dans cette section, les différents risques du projet seront analysés, détaillés puis ordonner dans un tableau pour mieux saisir leurs gravités.

Problèmes Hardware

- 01 ;
- Un des éléments hardware crash, se casse, n'est simplement plus utilisable ou ne convient pas ;
- Probabilité : 0.6 ;
- Impact : 0.6 ;
- Niveau de criticité : 0.36 ;
- Faire des tests fréquents sur l'installation finale de chaque élément de hardware pour s'assurer de leurs bons fonctionnements.
- Changer le matériel, implémenter une solution permettant de sauver la situation.

Problèmes de projection

- 02 ;
- La projection a un problème de calibrage ou/et de taille ;
- Probabilité : 1 ;
- Impact : 0.8 ;
- Niveau de criticité : 0.8 ;
- Faire des tests fréquents sur l'installation finale, implémentation d'un système de calibrage manuel et/ou automatique.
- Utilisation du système de calibrage, paramétrage du dit projecteur.

Problèmes Unity

- 03 ;
- Unity crée un problème spécifique au moteur de rendu ;
- Probabilité : 0.4 ;
- Impact : 1 ;
- Niveau de criticité : 0.4 ;
- Étudier les limites du moteur de jeu et en discuter avec les superviseurs.
- Implémenter le jeu selon ces limites.

Système d'animation de Unity

- 04 ;
- Le système d'animation de Unity ne permettant pas de faire ce qui est souhaité (rampes, collisions spécifiques, effet cartoon) ;
- Probabilité : 0.3 ;
- Impact : 0.6 ;
- Niveau de criticité : 0.18 ;
- Étudier les limites du moteur de jeu.
- Implémenter un cache-misère, une solution bricolage.

Retard

- 05 ;
- Les étudiants prennent du retard sur l'implémentation du programme ;
- Probabilité : 0.5 ;
- Impact : 1 ;
- Niveau de criticité : 0.5 ;
- Faire suivre la planification et l'avancée aux superviseurs pour éviter de dériver ou de prendre trop de retard.
- Si le retard est trop grand, couper dans les objectifs pour pouvoir rendre le projet à temps.

Changement d'avis du mandant

- 06 ;
- Le mandant change d'avis soudainement et le projet subit un changement de direction ;
- Probabilité : 0.3 ;
- Impact : 0.6 ;
- Niveau de criticité : 0.18 ;
- Il n'existe pas de méthode de prévention.
- Proposer un entre-deux pour pouvoir créer un compromis entre ce qui existe déjà et ce que souhaite l'artiste afin de ne perdre que peu de temps.

N°	Nom	Probabilité	Impact
02	Problèmes de projection	1	0.8
05	Retard	0.5	1
03	Problèmes Unity	0.4	1
01	Problèmes Hardware	0.6	0.6
04	Système d'animation de Unity	0.3	0.6
06	Changement d'avis du mandant	0.3	0.6

TABLE 4.1 – Le tableau des risques du projet.

4.5 Tests et validations

Plusieurs tests ont été pensés afin de valider l'état du projet :

- Tests d'endurance
- Un test qui laisse tourner le jeu longtemps pour voir si un crash apparaît.
- Tests de gameplay
- Faire jouer plusieurs personnes pour avoir un avis sur certaines fonctionnalités de gameplay.

- Tests physique
 - Faire tourner le jeu avec plein de collisions pour voir si un problème apparaît.
- Tests utilisateur
 - Faire jouer le jeu à des personnes tiers pour avoir un avis autres.
 - La création d'un formulaire via l'outil Google Forms pour avoir un bon aperçu de la perception du travail par le grand public lors du festival.

Chapitre 5

Conception

Après une analyse complète, le concept du jeu prend une direction consentie par tous les acteurs du groupe. Le jeu Flipp3r est un jeu de flipper à trois joueurs où l'objectif est d'anéantir un maléfique robot en atteignant ses points faibles disséminés sur le terrain.

5.1 Concept du jeu

Trois joueurs et ainsi trois zones, chacune d'entre-elles est un flipper pour un joueur avec sa propre gravité. Ces parties comprennent les éléments de base du flipper ainsi que d'autres éléments originaux de gameplay. La particularité est que les billes peuvent passer d'une zone à l'autre pour influencer le cours de la partie. Grâce aux premiers Concept Art des artistes et aux discussions du groupe, il a été décidé de créer un style basé sur l'Art Déco et sur des mouvements de vieux jeux cartoons et dynamiques. Des décors ressemblant au légendaire jeu "BioShock" ou au célèbre film "Metropolis" était le souhait du mandant. Finalement, le jeu Flipp3r possède un style bien défini où l'art déco semi-futuriste se mêle à des animations rapides et non physiquement réaliste.

5.2 Concepts

Ce chapitre regroupe la conception de tous les points chauds des tâches réalisées par l'étudiant Bruno Costa.

5.2.1 Collisions

Le jeu de flipper implémenté dans ce projet utilise un système de collision à haute vitesse où la balle ne se déplace pas par le moteur physique de Unity, mais par un déplacement manuel. Ainsi, elle n'est plus affectée par la physique de Unity et à cause de ça, il est impossible d'utiliser une technique de détection de collision continue. En effet, la balle est déplacée via script et sans gravité pour ainsi avoir un contrôle total sur elle et pouvoir lui donner des mouvements spécifiques à volonté comme des tremblements ou des déplacements dans des rampes. Si la balle possède une grande vitesse, suffisamment grande pour outrepasser le temps entre chaque appel de la méthode "FixedUpdate" et par la même occasion la vérification des collisions, la balle passe à travers les "Triggers". Ce problème est lié au fait que la collision est détectée trop tard par rapport au déplacement, et de facto la balle ne trouve aucun élément sur lequel rebondir.

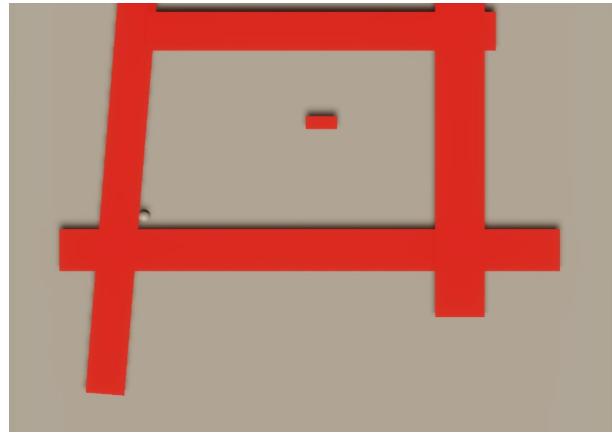


FIGURE 5.1 – La bille avant la collision.

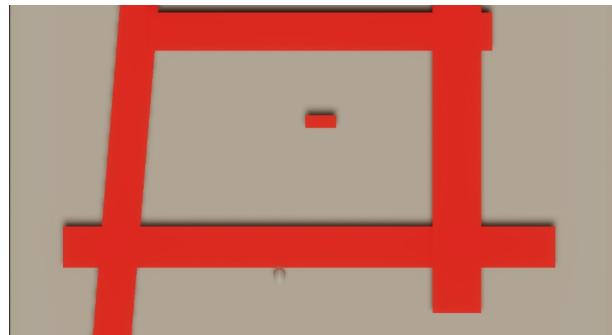


FIGURE 5.2 – La bille après la collision.

Unity ne permet pas de gérer des collisions d'objet rigide "IsKinematic" à haute vitesse, une recherche a donc dû être faite.

Afin de mieux comprendre la situation et trouver une solution, plusieurs tests ont été effectués :

1. Changement de la taille et de la vitesse de la balle.
2. Augmentation de la taille des murs.
3. Changement du type de détection de collision.
4. Augmentation de la quantité d'appel de FixedUpdate.

Grâce à ces tests, le problème a mieux été cerné et certaines de ces modifications se trouvent être des solutions mais uniquement temporaires. Si la balle augmente encore en vitesse ou change de taille, cela ne fonctionne à nouveau plus.

Un poste sur le forum de Unity a donc été rédigé pour trouver la moindre solution possible [11]. L'idée proposée ici par l'utilisateur Edy est d'utiliser la fonction "Rigidbody.MovePosition" à partir de la fonction "FixedUpdate" et d'activer la détection de collision continue spéculative ainsi que l'interpolation pour adoucir les déplacements de la bille.

La méthode MovePosition avait été testée mais dans la méthode Update, ce qui n'avait montré aucun résultat intéressant, l'idée de le mettre dans la méthode FixedUpdate a été une révélation. Cependant, l'activation de la détection de collision continue spéculative et l'interpolation altérait négativement le déplacement de la bille pour une raison inconnue. En effet, la bille possédait un déplacement saccadé et la détection de collision s'en retrouvait affectée, ces paramètres sont alors restés désactivés.

À présent, la balle utilise ce système pour se déplacer, les collisions sont bien mieux détectées et une haute vitesse et utilisable. Bien que, si la vitesse soit trop grande, le problème réapparaît, une grande marge est désormais disponible.

Rebonds et coins

La gestion des rebonds et des coins est passée par de nombreuses étapes avant d'être fonctionnelle. En effet, même si la détection de collision fonctionne, il faut pouvoir gérer le rebond de la balle et surtout le rebond de celle-ci dans un coin.

Les rebonds de la balle ont d'abord été gérés d'une manière très simple. Au moment de la collision, un rayon était envoyé pour détecter la collision, le point de collision et la normale de collision. Grâce à ces informations, le nouveau vecteur résultant de la collision est calculé et la balle part dans l'autre sens. Le problème est que ce rayon était alors envoyé dans le sens de la collision, ce qui est faux. L'envoi doit se faire dans la direction de la balle, pour voir la collision d'où la balle se dirige.

Première amélioration La première amélioration imaginée était de simplement envoyer un unique rayon depuis le centre et dans le sens de la vitesse de la balle. Le problème est que si la collision se fait sur le côté de la balle, le calcul se fera avec le mur en face de la balle et du coup la balle rebondira dans le vide.

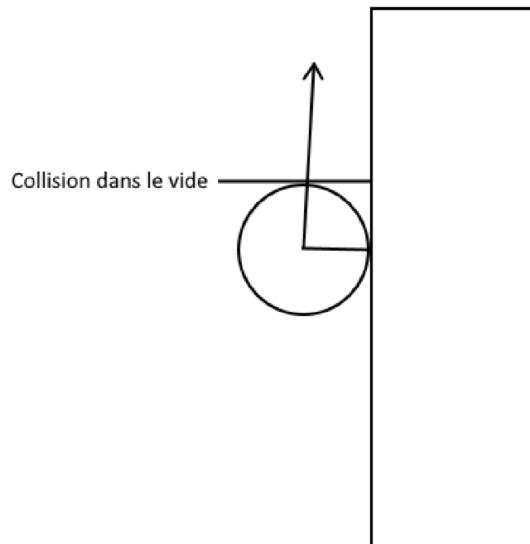


FIGURE 5.3 – La bille collisionne dans le vide.

Deuxième amélioration La deuxième amélioration imaginée était d'envoyer deux voire trois rayons depuis la balle, de cette manière :

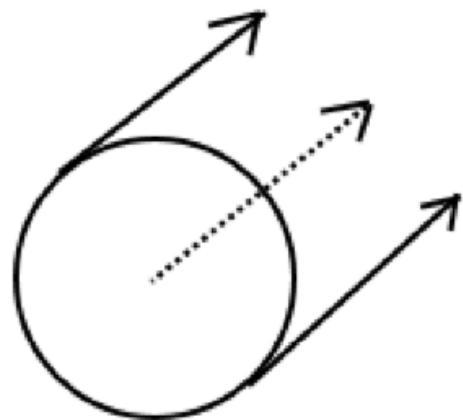


FIGURE 5.4 – Trois rayons émis depuis la bille.

Le problème de cette amélioration est son manque de précision conséquent. Il se peut que le rayon passe juste à côté d'un obstacle et ainsi permettrait à la balle de passer à travers le dit obstacle.

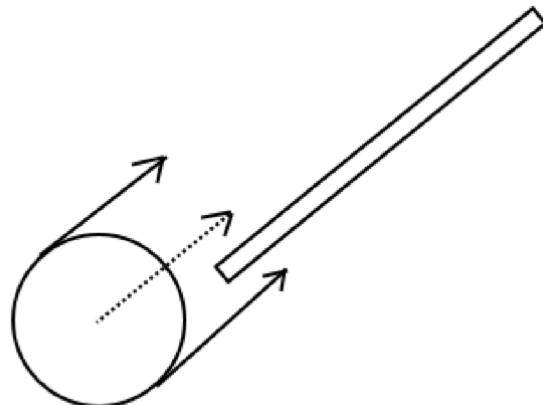


FIGURE 5.5 – Un objet passe dans les mailles du filet de collision.

De plus, la gestion du rebond sur un coin, externe ou interne d'un élément ne fonctionnerait pas bien. Quel rayon étudier pour avoir la bonne normale ? Est-ce que le coin est bien détecté comme tel ?

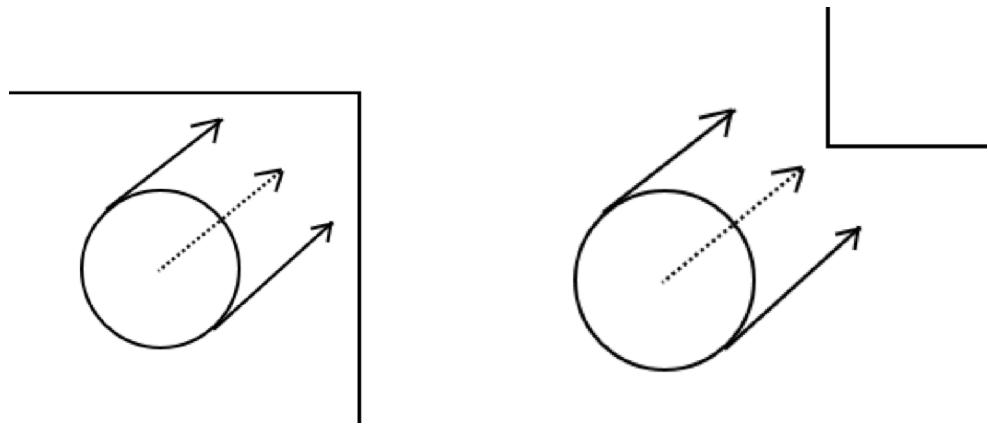


FIGURE 5.6 – La bille va avoir une collision avec un coin.

Troisième amélioration La troisième amélioration a été de remplacer les rayons par des sphères pour détecter l'entièreté des possibilités de collisions.

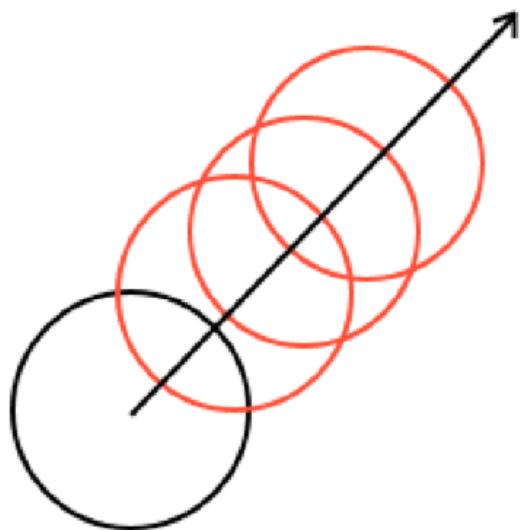


FIGURE 5.7 – Des sphères sont envoyées depuis la bille.

De cette manière, la majorité des cas spécifiques sont pris en compte : peu importe la forme de la surface, des données de collisions seront détectées et utilisables pour calculer le rebond. Néanmoins, un autre problème est alors apparu. Comme ces sphères étaient envoyées au moment de la collision, elles se trouvaient d'ores et déjà dans l'élément avec lequel la bille a collisionné. De ce fait, les sphères envoyées agissaient comme si aucune collision ne s'est produite.

Quatrième amélioration Ici, l'idée appliquée a été de constamment envoyer des sphères pour pouvoir "prédir" les informations de collisions. Ainsi, à chaque instant, des sphères étaient envoyées dans la direction de la bille pour scruter les informations de collisions futures. Avec cette amélioration, les billes rebondissent de manière optimale à chaque collision trouvée avec un élément du décor. Le seul problème restant est la gestion des coins, à ce stade, les billes

collisionnent avec un coin comme un mur plat et ainsi, passe à travers le mur car la seconde collision est trop proche ou déjà trop tard.

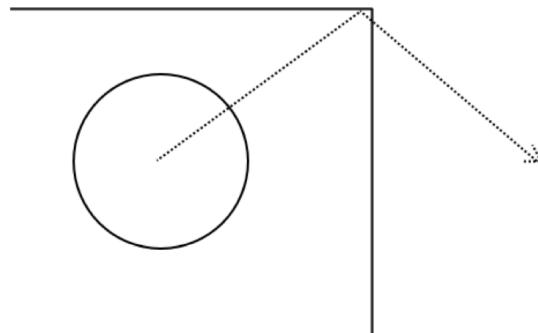


FIGURE 5.8 – La collision d'une bille dans un coin.

Cinquième amélioration La détection d'un coin est finalement basique : au lieu d'envoyer une seule sphère, nous en envoyons deux différentes. Si la première détecte une future collision, alors elle désactive temporairement cette surface pour la rendre inexistante. La seconde est alors envoyée, et si une autre collision est détectée, alors on évalue la distance entre les deux. Si elles sont suffisamment proches, on peut donc établir l'existence d'un coin.

Le calcul du vecteur de rebond est ensuite trivial, l'addition des normales des deux collisions correspond à un retour en arrière de la balle.

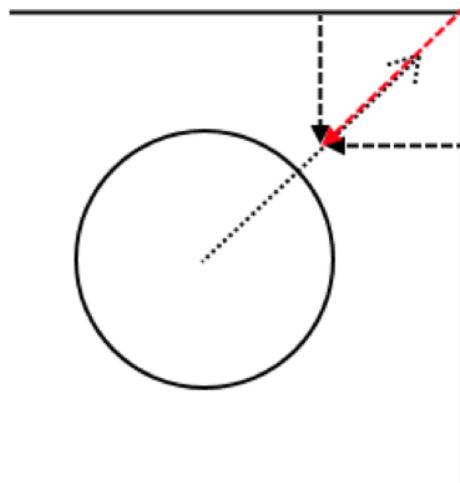


FIGURE 5.9 – Calcul du vecteur de retour après collision dans un coin.

Proche du but, mais il existe encore deux problèmes :

1. Lorsque le coin est serré, la collision se fait sur les deux parties. Avec la seconde faussée car la vitesse de la balle a déjà été changée par la première collision.

2. Lorsque le coin est trop large, l'information de la seconde collision à le temps de changer et sauvegarde ainsi une collision faussée.

Amélioration finale Afin de pallier ces derniers problèmes la dernière amélioration a été de :

- Verrouiller la modification des informations de collisions lorsqu'un coin a été détecté et ce jusqu'à ce que la collision soit effectuée.
- Vérifier au moment de la collision le sens du vecteur de vitesse de la balle par rapport au vecteur normal de collision. S'ils sont dans le même sens, ce qui normalement ne devrait pas arriver, le sens de la vitesse est corrigé.

Finalement, les billes ont la capacité de rebondir sur les surfaces rencontrées et même sur les coins de celles-ci.

Algorithm 1: GetCollisionNormal

Result: Sauvegarde la/les normale(s) de collision des éléments sur le chemin de la balle

```

if 1ère collision détectée par le premier envoi de sphère then
    if La dite collision n'a jamais été rencontrée OU La dite collision a été rencontrée
        mais n'est pas verrouillée then
            | Sauvegarde de sa normale dans le conteneur en tant que non-verrouillée;
        end
    Désactivation du collisionneur du premier objet détecté;
    if 2ème collision détectée par le second envoi de sphère then
        if Les deux collisions sont suffisamment proches then
            | Calcul du vecteur résultant des deux normales;
            | Sauvegarde du nouveau vecteur en tant que verrouillée;
        end
    end
    Réactivation du collisionneur du premier objet détecté;
end
```

Gameplay - Flipper

Les flippers ont un fonctionnement spécifique quant aux collisions avec la balle. Ils possèdent plusieurs états qui changent la manière dont la balle rebondit à leurs contacts.

Si le flipper est "inactif", c'est-à-dire soit levé et maintenu, soit baissé, alors la balle va glisser sur le flipper. Lorsque la balle est en train de glisser le flipper et qu'il est activé, alors la bille va être éjectée. Sa nouvelle vitesse sera égale à la moyenne pondérée de la normale de collision et du vecteur formé par le point de rotation du flipper et de la position de la balle.

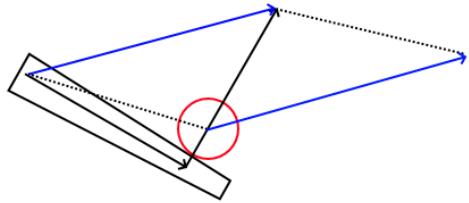


FIGURE 5.10 – Rebond de la balle lorsqu'elle glisse sur le flipper.

Si la balle tape le flipper au moment de son activation, c'est-à-dire lorsqu'elle subit une reprise de volée, alors la balle va être éjectée mais de manière plus forte. Sa nouvelle vitesse sera simplement un calcul de collision simple, une trajectoire calculée par rapport à la normale de collision. Cependant, cette trajectoire sera multipliée par la distance "bras de levier", c'est-à-dire, la distance entre le point de rotation du flipper et la position de la balle.

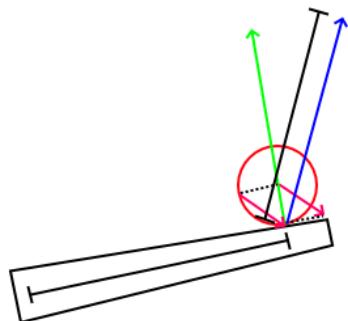


FIGURE 5.11 – Rebond de la balle lorsqu'elle est renvoyée à la collision, une reprise de volée.

5.2.2 Animation

Comme expliqué précédemment, afin d'animer une balle se déplaçant dans une rampe, des splines sont utilisées. Néanmoins, Unity ne possède pas le support de création de spline, du moins dans la version utilisée pour ce projet. C'est pourquoi un système de création de spline via des courbes de Bézier a été entièrement implémentée. Ces splines sont définies par plusieurs courbes et chacune d'entre-elles sont définies par 4 points. Un point de départ, un point de fin et deux points d'interpolation.

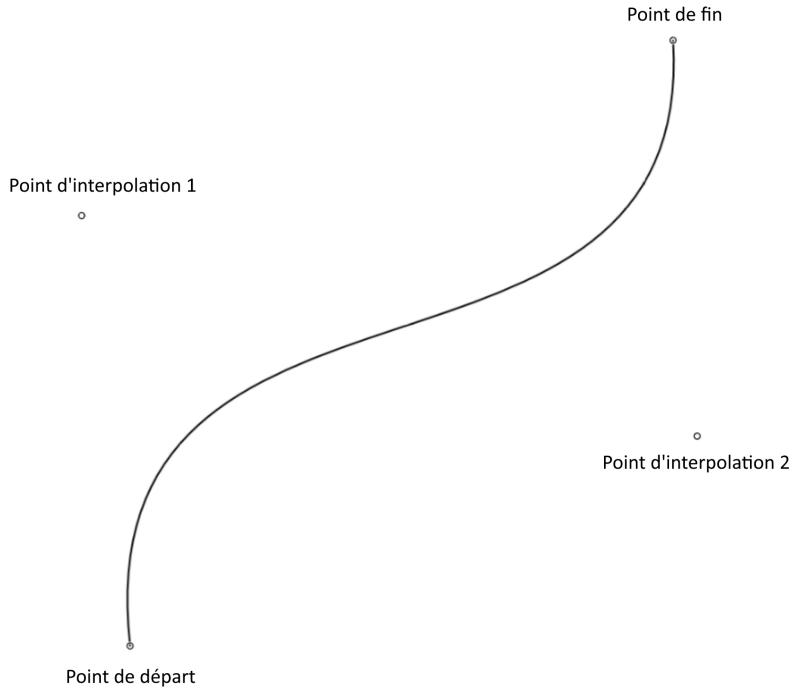


FIGURE 5.12 – Exemple de courbe de Bézier

Ces splines sont créables manuellement en déplaçant chaque point individuellement. Il est possible de les parcourir avec un indice, allant de 0 à 1, indiquant la progression de cette spline. Ainsi, une animation de rampe est réalisable pour autant qu'une spline correspondante est créée.

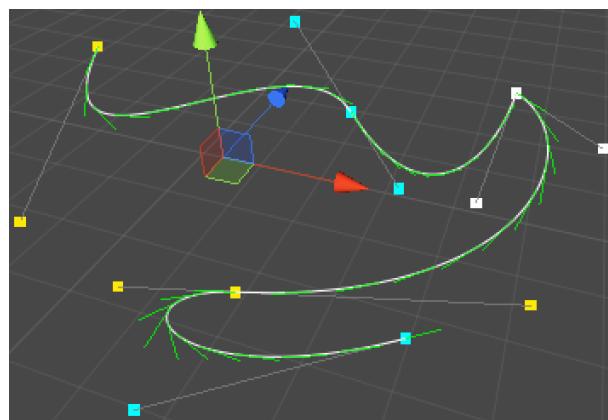


FIGURE 5.13 – Exemple d'une spline de rampe

5.2.3 Effets spéciaux

Pour ajouter de la vie et du dynamisme au jeu, il est nécessaire d'ajouter des effets visuels. Sans effets, le jeu semble vide et peut facilement ennuyer le public. Le projet est un mandat, c'est pourquoi les effets créés ont été voulus et commandés par le groupe artistique. Avec le thème Art déco, les effets spéciaux visuels appropriés sont des jets d'étincelles, de la foudre ou tout ce qui

paraît robotique. Dès qu'un effet est créé, il est montré aux artistes afin de subir une évaluation et de potentiels changements pour atteindre le meilleur état possible. Parmi ceux créés, voici un effet de flammes autour de la balle indiquant son accumulation de points.

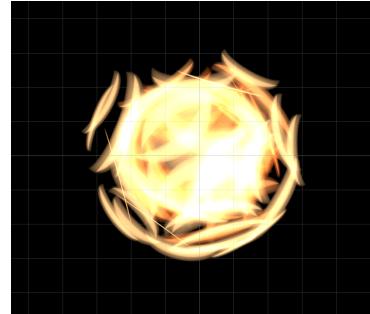


FIGURE 5.14 – Effet visuel de flammes.

Ou encore un effet d'électricité indiquant le voyage d'une bille dans une rampe.

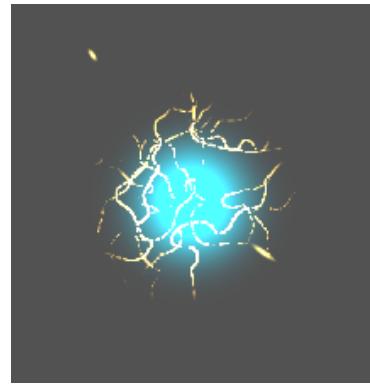


FIGURE 5.15 – Effet visuel d'électricité.

Afin de créer un effet visuel, il faut normalement passer par bien des outils et cela représente un travail fastidieux. Ces deux effets ont été créé de manière différente via des moyens offerts par Unity. Le premier a été fait via le "Particle System" de Unity, un outil qui permet de générer une quantité définie d'un élément, d'une particule, donné et ainsi avoir un effet de masse. Le second est un peu plus complexe et a été fait via le "VFX Graph" de Unity, cet outil peut créer des effets de toutes sortes et d'ampleur plus grand. Il peut surtout utiliser le préexistant "Shader graph".

5.2.4 Audio

La gestion de l'audio s'est basée sur les sons créés par l'artiste Christopher Lanza, sur les enceintes utilisées avec l'installation ainsi que sur la salle où se trouvera l'installation. Comme il n'y a que deux enceintes, que le son dans la salle résonne un peu et que les sons ont été reçus un peu tard, aucune spatialisation du son n'a été mise en place. Tous les sons, bruitages ou musiques proviennent de la caméra au centre du jeu.

Chapitre 6

Implémentation

Les concepts importants du projet ont à présent été détaillés et c'est dans ce chapitre qu'est expliqué comment ils ont été appliqués.

6.1 Collisions

Dans le jeu Flipp3r, les balles se gèrent elles-mêmes. Chacune d'entre-elles possède un script qui va gérer les déplacements de la balle ainsi que les collisions et autres. À chaque instant la balle va récupérer la normale de collision des objets vers lesquelles elle se dirige et les sauvegarder dans un conteneur de la manière évoquée dans le chapitre précédent.

L'envoi de sphère décrit plus haut est réalisé par la fonction Physics.SphereCast().

```
if (Physics.SphereCast(this.transform.position, this.radius, this.speed, out  
RaycastHit hitFirst, 5f, layerMask, QueryTriggerInteraction.Collide)) {  
    // Une collision a été détectée  
}
```

Cette fonction nécessite plusieurs paramètres :

- La position source de l'envoi
- Le rayon de la sphère envoyée
- La direction dans laquelle elle va être envoyée
- L'objet qui va sauvegarder les informations de collision
- La distance maximum de l'envoi
- Le masque de collision (Ce qui doit ou ne doit pas être compris comme étant collisionnable)
- Et finalement la manière dont il faut considérer la potentielle collision : ici, une collision normale.

Ensuite, lorsque la balle entre en collision avec une surface, un comportement différent va être effectué selon le type de l'objet collisionné. Si c'est contre un mur ou un élément décoratif, la balle va simplement rebondir via une trajectoire calculée par rapport à la normale de celui-ci via la fonction Vector3.Reflect().

```
Vector3 newDirection = Vector3.Reflect(this.speed.normalized, collisionNormal);
```

Une valeur aléatoire est ajoutée à cette trajectoire afin d'éviter que la balle se coince dans une boucle infinie de rebonds.

Par contre, si la balle tape sur un flipper, un bumper ou un autre élément de gameplay, un rebond spécifique sera effectué. Le bumper fera trembler la balle avant de l'éjecter, le flipper éjectera la balle dans une trajectoire bien définie, etc. Une fois la collision effectuée, les verrouillages mis en place pour éviter les problèmes des coins sont retirés.

6.2 Gameplay

Cette section va détaillée comment les éléments de gameplay importants ont été implémentés.

6.2.1 Flippers

Les flippers sont des éléments fondamentaux du jeu et fonctionnent via des "Hinge Joints" : des joints qui ont un fonctionnement identique à des ressorts réglables.



FIGURE 6.1 – Un flipper du jeu.

Un GameObject avec un "Hinge Joint" possède deux positions, une de repos et une active. Ces positions sont des angles, au repos, le GameObject flipper aura une orientation "basse", mais lorsque le ressort est activé le flipper subi une rotation et son orientation sera "haute". Au relâchement du ressort, le flipper regagne son orientation "basse".

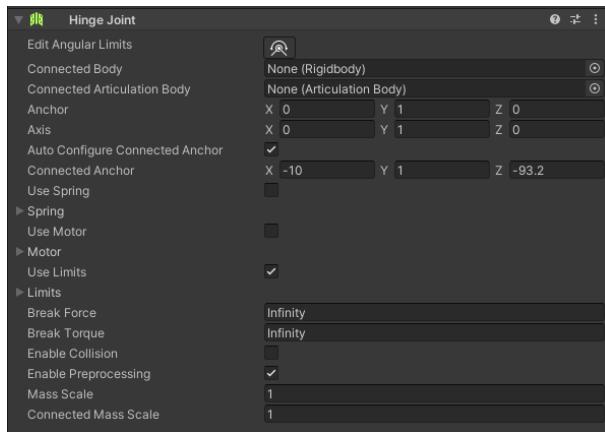


FIGURE 6.2 – Composant Hinge Joint d'un flipper.

6.2.2 Bumpers

Les bumpers sont les éléments circulaires sur le terrain qui font rebondir les balles dans tous les sens.

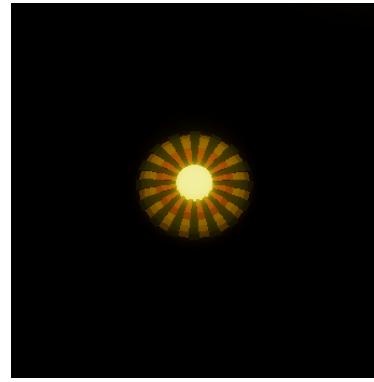


FIGURE 6.3 – Un bumper du jeu.

Le souhait de l'artiste était de donner un style "Cartoon" à la collision. Pour cela, une animation de tremblement a été imposée au bumper et à la balle. Lorsque la balle frappe un bumper, celui-ci entre en mode "animation" et par la même occasion va trembler. Ce tremblement est assez simple, à chaque instant t la position x du bumper est remplacée par une position x aléatoire située dans un petit cercle autour du bumper. Une fois l'animation finie, le bumper retrouve sa position initiale.

6.3 Animation

Les splines faites à partir de courbes de Bézier ont un indice qui permet de les parcourir. Lorsque cet indice est à 0, le point de départ est atteignable et utilisable tandis que lorsqu'il est à 1, c'est le point de fin que l'on peut employer.



FIGURE 6.4 – Un exemple d'une spline utilisée pour une rampe.

Bien entendu, un indice à 0.5 indiquera la moitié de la spline. Ainsi, lors de l'entrée d'une

rampe, la bille parcours toutes les positions de la spline avec un indice changé par le temps. Pour influencer la vitesse de déplacement de la bille dans une rampe, une durée de parcours a été définie. Par cette durée, l'indice vaut alors à l'instant t : $t / \text{durée}$. De cette manière, l'indice vaudra toujours entre 0 et 1 selon l'avancée de la spline.

6.4 Effets visuels

De nombreux effets spéciaux ont été implémentés pour le jeu Flipp3r, uniquement les plus importants et les plus complexes seront expliqués et détaillés ici.

6.4.1 Système de particules

Cet outil permet de générer et d'émettre plusieurs particules et fonctionne via de multiples réglages :

- Le type d'émission
- La forme de l'émission
- La couleur des particules sur sa durée
- La taille des particules sur la durée
- ...

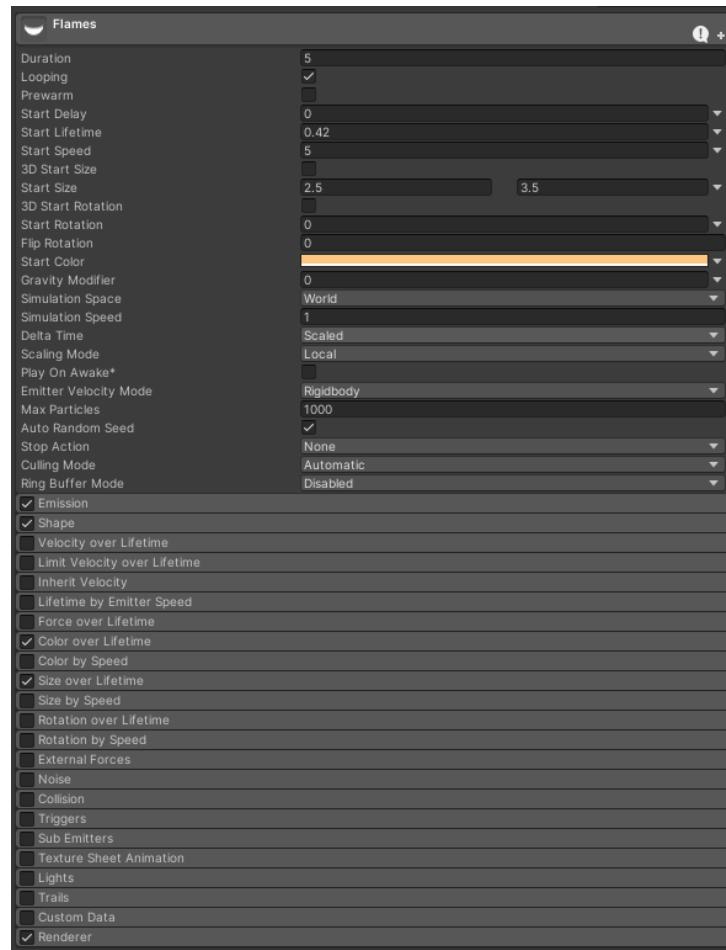


FIGURE 6.5 – Réglages du système de particules de Unity.

Flammes

Les flammes ont été réalisés via le système de particule de Unity et sont fondamentalement des sprites générés très rapidement. En effet, les particules peuvent être de simples sprites gérés de manière additive : un dessin de noir et de blanc qui utilise un de l'alpha pour créer un effet de "glow". Les couleurs de ce sprites sont alors changées pour afficher l'effet voulu.

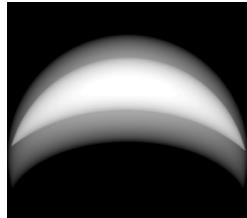


FIGURE 6.6 – Particule de l'effet de flammes.

Afin d'avoir un effet "plat" et en direction de l'émission, les particules sont définies en tant que "billboard" et sont émises via une sphère.

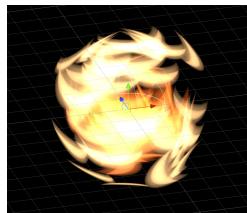


FIGURE 6.7 – Émission des flammes.

Cependant, comme le jeu n'a pas de perspective, la vue par le haut donne un effet tout autre.

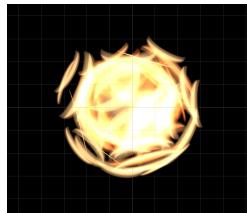


FIGURE 6.8 – Émission des flammes vue de dessus.

Finalement, c'est avec une émission de 200 particules par seconde que l'on a un effet dynamique de feu tout autour d'une sphère, et donc de la bille.

Explosion

L'effet visuel d'explosion est un peu différent de celui des flammes. Il emploie plusieurs effets en un. Chacun d'entre eux utilise des meshes et non des sprites. Les particules sont des sphères 3Ds et des trails qui ne sont pas émis sur la durée mais en une seule et unique fois. Cette émission est appelée "bursts". Ainsi, 200 particules sont envoyées une seule fois tout autour d'une sphère.

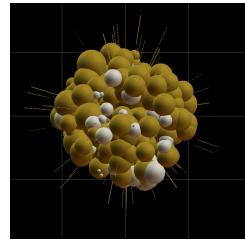


FIGURE 6.9 – Effet visuel de l’explosion sans lumière.

Étincelles

Les étincelles sont également des sprites de type additif comme pour les flammes. La particularité de cet effet est que chacune de ses particules gère les collisions.

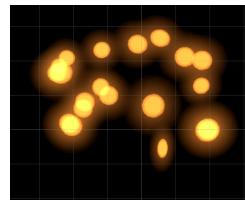


FIGURE 6.10 – Effet visuel des étincelles.

Lorsque l’effet est activé, des étincelles jaillissent et meurent au contact de n’importe quel objet alentour.

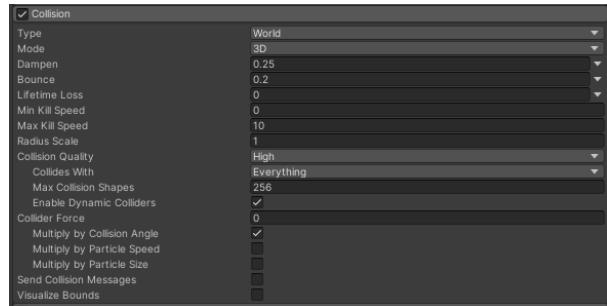


FIGURE 6.11 – Réglage des collisions du système de particules.

6.4.2 VFX Graph

L’outil VFX Graph de Unity fonctionne un peu de la même manière que le Shader Graph. Des nodes sont créés sur un plateau, elles sont reliées à d’autres nodes et via celles-ci des effets sont générés.

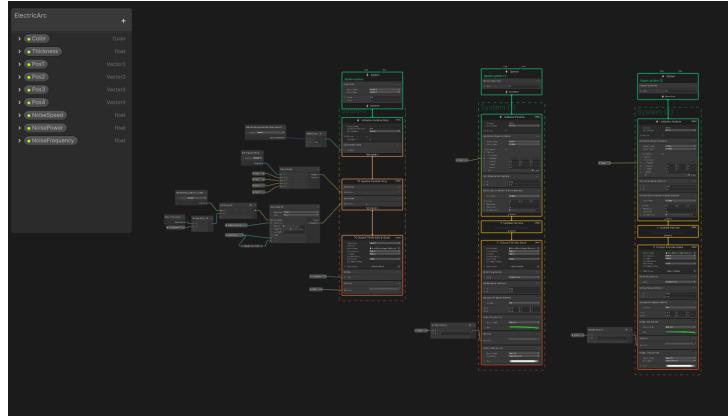


FIGURE 6.12 – Exemple d'un effet visuel créé via VFX Graph.

Arc électrique

La création de l'arc électrique a été faite en suivant le tutoriel de Gabriel Aguilar [12].

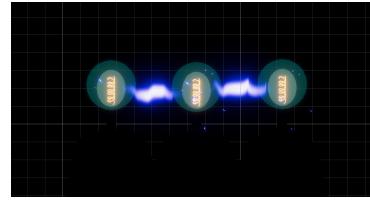


FIGURE 6.13 – Exemple d'arc électrique.

L'effet visuel est composé de particules le long d'une courbe de Bézier. Comme pour les splines d'animations, quatre points sont utilisés ici, des particules y sont alors placées et un bruit les fait se déplacer pour donner un effet de foudre.

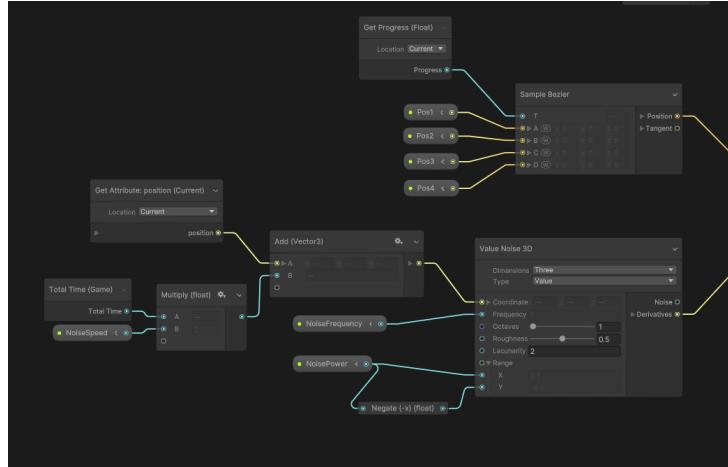


FIGURE 6.14 – Node pour l'effet d'arc électrique.

Électricité magnétique

La création de l'électricité magnétique a été faite en suivant le tutoriel de Gabriel Aguilar [13]. L'effet visuel magnétique est un peu plus compliqué car il emploie un shader. Pour générer un

semblant d'électricité, il faut créer un bruit désordonné dont on ne récupère que les bords. Le shader "Electricity" a été fait pour ça.

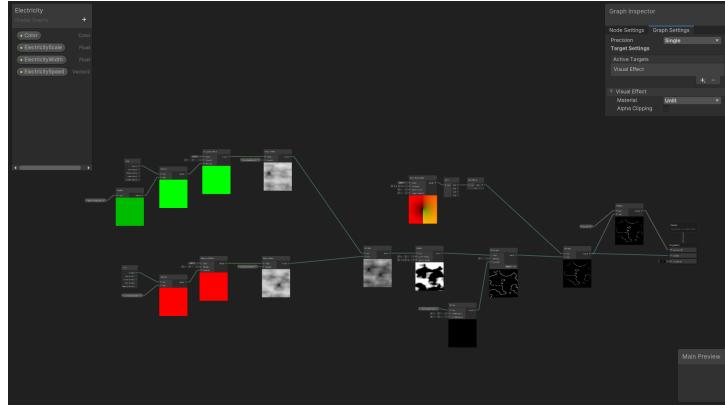


FIGURE 6.15 – Shader graph pour l'effet d'électricité.

Ce shader génère deux bruits multipliés entre eux, ensuite ils sont modifiés pour n'afficher que le noir et le blanc. Finalement, les bords de ce bruit sont extraits et un effet de fondu est appliqué. C'est enfin dans le VFX Graph qu'est utilisée l'électricité, elle est teinte en jaune et elle est couplée à un effet de particules d'étincelles et à un effet de particules générant une lumière palpitante bleue.

6.5 Audio

Pour la partie sonore du projet, un script nommé "AudioManager" a été créé en suivant le tutoriel de Brackeys [14]. Ce contrôleur de son emploie un pattern Singleton qui emploie un tableau de sons qu'il va pouvoir gérer : les jouer, les lancer en boucle ou les stopper. Ces sons sont représentés par une classe personnalisée du nom de "Sound" contenant les données importantes du dit son et qui est "Serializable". Dans Unity, cela implique qu'une instance de cette classe est modifiable depuis l'inspecteur. Ainsi, l'AudioManager est appliquée à la source du son du jeu, c'est-à-dire la caméra, et va émettre et gérer tous les sons du jeu.

6.6 Intégration

L'utilisation de CI / CD via le service de runner de GitLab demande un script ".yaml". Ce script instaure une routine de déploiement où le projet présent dans le répertoire git est utilisé pour créer un build du jeu à chaque "push" effectué sur la branche principale.

```

stages:
  - build

unity-build:
  stage: build
  script:
    - ls
    - rm -r C:\Flipp3r\
    - mkdir C:\Flipp3r\
    - xcopy . C:\Flipp3r\ /E
    - C:\PROGRA~1\Unity\Hub\Editor\2021.3.2f1\Editor\Unity.exe --quit --
batchmode --projectPath C:\Flipp3r\Flipp3r --executeMethod BuildScript.Build

```

```
only:  
  - main  
tags:  
  - flipp3r
```

Son rôle est d'appeler la fonction Build de la classe BuildScript pour générer le build du jeu. Pour utiliser une fonction d'un script sans que le jeu soit lancer, il faut passer par le "batchmode" de Unity. Voici cette fonction :

```
public static void Build()  
{  
    BuildPlayerOptions buildPlayerOptions = new BuildPlayerOptions();  
    buildPlayerOptions.scenes = new[] { "Assets/Scenes/MasterScene.unity" };  
    buildPlayerOptions.locationPathName = "C:/Flipp3rBuilds/Flipp3r.exe";  
    buildPlayerOptions.target = BuildTarget.StandaloneWindows64;  
    buildPlayerOptions.options = BuildOptions.None;  
  
    BuildReport report = BuildPipeline.BuildPlayer(buildPlayerOptions);  
    BuildSummary summary = report.summary;  
  
    if (summary.result == BuildResult.Succeeded)  
    {  
        Debug.Log("Build succeeded: " + summary.totalSize + " bytes");  
    }  
  
    if (summary.result == BuildResultFailed)  
    {  
        Debug.Log("Build failed");  
    }  
}
```

Elle sélectionne la scène voulue, et la build selon les paramètres de base d'un exécutable Windows.

Chapitre 7

Tests

Le projet est bien avancé et la majorité des fonctionnalités sont implémentées. Il faut à présent en tester les moindres recoins pour s'assurer un bon fonctionnement lors du festival.

7.1 Tests sur les collisions

Le fonctionnement des collisions a été testé de trois manières différentes :

1. Une vingtaine de billes ont été enfermées dans une scène vide ayant la même forme que le terrain du flipper et ont été "lachées" à une certaine vitesse. Ce segment de jeu a tourné dans le fond pendant une heure et aucune balle n'est sorti du terrain.
2. Ensuite, plusieurs objets ont été ajoutés à cette scène : des bumpers, des flippers, des murs et autres éléments de gameplay. Cette fois-ci, le jeu a tourné environ 15 minutes, le temps d'une partie, et aucune balle n'est sorti du terrain.
3. Finalement, les derniers tests de collisions ont été faits en situation réelle : c'est-à-dire dans le terrain de flipper final. Il arrive quelques fois qu'une balle sorte du terrain pour une raison inexplicable.

Comme ce problème n'arrive que rarement et comme le temps manquait, il a été décidé de ne pas chercher à le réparer.

7.2 Tests de gameplay

Les tests sur le gameplay ont commencé uniquement à partir de la dernière semaine de travail. L'installation finale était nécessaire pour pouvoir correctement jouer à trois joueurs. En premier lieu, ce sont les membres du groupe qui ont testé le jeu et noté tous les problèmes existants :

- Il y avait trop de bumpers sur le terrain et certains étaient difficiles d'accès. Ce qui rendait le jeu trop difficile.
- La gravité était trop faible et les balles trop lentes.
- Le portail centrifuge au centre du terrain était trop grand et bloquait certaines balles.
- Le jeu était trop difficile à terminer
- ...

Ces problèmes ont été réglés rapidement pour avoir la meilleure expérience utilisateur possible.

7.2.1 Tests utilisateurs

Ensuite sont venus les tests utilisateurs. Ici, les étudiants ont demandé à deux collègues de l'école de venir tester le jeu et de donner un avis. Grâce à ces témoignages, il a été déterminé que le public

ne comprenait pas bien l'idée des "faiblesses" du boss. Un effet indiquant une cible par-dessus les faiblesses à leurs apparitions a été ajouté.

7.3 Tests sur les effets spéciaux et les lumières

Sur l'installation finale avec la projection, certains effets et lumières étaient moins visibles que sur l'écran d'ordinateur. La lumière de la balle jaune était trop forte et empêchait de la discerner du reste des lumières. Certains arcs électriques n'étaient pas très visibles car trop petits. Ces problèmes ont été réglés rapidement avant le début du festival.

7.4 Tests audio

À l'arrivée des enceintes dans la salle de l'installation finale, elles ont immédiatement été branchées pour tester la qualité du son et surtout son écho. En premier lieu, un bon placement des enceintes était important : différentes dispositions ont été testées jusqu'à avoir la bonne position dessous l'installation. Puis, tous les sons du jeu ont été isolés et calibrés pour avoir la meilleure condition sonore possible.

7.5 Tests d'endurance

Une partie de jeu dure environ une quinzaine de minutes, le programme doit alors tenir sans causer un crash au minimum 30 minutes. Le jeu a donc été laissé allumer toute une nuit pour tester son endurance : au lendemain, le jeu tournait encore normalement. De plus, lorsque la partie se termine le jeu se redémarre et ainsi peu de problèmes liés à l'endurance sont présents.

Chapitre 8

Résultats

Le jeu a été présenté durant la semaine du NIFFF du 2 au 9 juillet 2022. Une vidéo résumant cette expérience est disponible en annexe.

8.1 État du programme

Le programme est dans un état stable et fonctionnel. Le jeu tient les tests d'endurance effectués et ne crash pas, il a été utilisé par de nombreux utilisateurs, a été apprécié et une partie entière du jeu est terminable.

8.2 NIFFF

L'installation Flipp3r faisait partie de l'activité "NIFFF Invasion" et était mise à disposition à la villa de la Petite Rochette à Neuchâtel. L'installation était ouverte au public de 13h à 19h tous les jours du festival. Des bénévoles se sont relayés afin d'assurer le bon fonctionnement du jeu. Le grand public a donc pu jouer au jeu durant toute une semaine.

8.2.1 Validation utilisateurs

Une fois le festival terminé, le questionnaire créé sur l'outil Google Forms a été récupéré et analysé. Une douzaine de personnes ont pris le temps d'y répondre. Voici les questions qui ont été posées aux utilisateurs :

- Avez-vous apprécié l'expérience Flipp3r ?
- Comment avez-vous trouvé l'ambiance présente ?
- Est-ce que le jeu est intuitif ?
- Avez-vous apprécié les dessins sur l'installation ?
- Comment jugez-vous la difficulté du jeu ?
- Comment considérez-vous la maniabilité et le gameplay du jeu ?
- Avez-vous expérimenté beaucoup de bugs ?
- Si oui, est-ce que vous pouvez les décrire ?
- Autres remarques ?

Les réponses sont disponibles en annexe. En résumant les retours brièvement :

- La majorité a apprécié l'expérience et l'ambiance présente.
- Le public est mitigé entre trouver le jeu intuitif et moyennement intuitif.
- La majorité apprécie les dessins faits sur l'installation.

- La difficulté est acceptable pour la plupart, bien que certaines personnes trouvent le jeu trop difficile ou pas assez.
- Le grand public trouve le jeu assez maniable.
- Les utilisateurs ont soit expérimenté quelques bugs soit presque zéro bugs.

Via les remarques supplémentaires des utilisateurs, le groupe a pu tirer conclusion que le défaut principal du jeu était son gameplay. Quelques problèmes de compréhension, de maniabilité ainsi que de fluidité de jeu sont présents.

Chapitre 9

Limitations et perspectives

9.1 Limitations

9.1.1 Problèmes encore présents

Bien que l'application soit dans un état stable et fonctionnel, il existe encore quatre bugs connus qui n'ont pas pu être réglés à temps.

- Il arrive encore que la balle sorte du terrain comme expliqué plus haut.
- La gestion de la liste des balles existantes génère une erreur lorsqu'une balle sort du terrain et cette erreur génère un bug dans le spawn des balles. Les balles spawn alors à l'infini.
- La balle peut se coincer au contact d'un bumper et ne plus jamais bougé. La raison est inconnue, mais probablement un problème de thread car la gestion de l'animation qui stoppe la balle et la fait trembler est une coroutine.
- Il se peut que la boucle finale permettant de générer le fond blanc soit interrompue et ainsi le jeu ne se met jamais en "pause" et impossible de le redémarrer. Ce problème est étrange car il fonctionne bien en général, ça n'arrive que rarement.

9.1.2 Gameplay

Grâce aux avis récoltés via le questionnaire Google Form, le groupe a compris que le jeu nécessitait une refonte du gameplay afin d'atteindre un état "parfait". En effet, les remarques ont démontré que le jeu est peu intuitif et le gameplay est incompris par la majorité du grand public. L'idée de coopération entre en conflit avec l'aspect score des joueurs et un voyage des billes non équitables et cela frustre certains utilisateurs. De plus, le gameplay est considéré trop simple par certains et manque un peu de dynamisme.

9.2 Perspectives

L'objectif est de continuer de travailler sur le projet Flipp3r pour en créer d'autres. La présentation de l'installation dans des musées ou autres expositions est envisagée et l'idée serait de créer des flippers à thèmes. Changer les dispositions, corriger les bugs, créer de nouvelles dynamiques de jeu et changer la thématique sont les futurs changements du maintenant célèbre jeu Flipp3r.

9.2.1 Améliorations

Il est vrai, le jeu de Flipp3r est ouvert à des améliorations et à des corrections de bugs.

Bugs

- Pour la sortie de terrain, le domaine de la détection de collision est un domaine vaste et des recherches sont encore en cours. Pour résoudre ce problème, il faut analyser les derniers cas où la balle sort et isoler le problème.
- La gestion de la liste des balles est un problème trivial, l'erreur doit se trouver dans sa mise à jour.
- Comme pour la sortie de terrain, la solution se trouve dans l'analyse du cas. Il faut avant tout reproduire le problème pour le comprendre.
- C'est également le cas pour ce problème. Comme le bug n'arrive pas tout le temps, il est difficile de comprendre quel en est la source.

Pistes d'amélioration

- Créer un level design plus élaboré pour permettre aux balles de mieux voyager entre les zones
- Donner de meilleures interactions au boss, créer des pouvoirs pour le boss
- Retirer le système de score
- Améliorer la compréhension des règles du jeu
- Créer plus d'opportunité de coopération

Chapitre 10

Conclusion

Ce travail de Bachelor respecte le cahier des charges et ses objectifs, autant principaux que secondaires.

Objectifs principaux	Respecté
Gestion de l'installation physique	✓
Gestion de la machine et de Unity	✓
Conception du gameplay	✓
Design et architecture du projet dans Unity	✓
Adaptation des modèles 3D et gestion des imports et des configurations des assets	✓
Gestion du flipp3r, de la scène et du terrain	✓
Simulation physique des éléments (billes, bumpers, etc)	✓
Intégration continue des différentes parties	✓
Gestion du son et des bruitages	✓
Objectifs secondaires	Respecté
VFX	✓
Mécanique d'interaction entre les joueurs - malus et bonus	✓
Gestion des animations passives (décor)	✓

Bien que des bugs mineurs sont encore présents, l'application est dans un état stable et fonctionnel. De plus, l'installation a été appréciée du grand public lors du festival du NIFFF et le groupe a eu de très bons retours.

Le projet Flipp3r a permis de grandement améliorer mes compétences dans le domaine de Unity et du jeu vidéo. La collaboration avec les artistes et mandants, Mandrill et Christopher Lanza, s'est bien déroulée tout au long du projet. Pouvoir vivre une situation réelle de création de jeux avec des artistes passionnés est une des meilleures expériences vécues à la HE-Arc. Je suis fier du travail effectué et n'hésiterais pas à me proposer pour y participer à nouveau si l'occasion se présentait.

Bruno Costa, 28 juillet 2022

Annexes

- Cahier des charges - Bruno Costa : 22ISC-TB202_CC_Flipp3r1_BrunoCosta.pdf
- Cahier des charges - Diogo Lopes da Silva : 22ISC-TB228_CC_Flipp3r2_DiogoLopesDaSilva.pdf
- Flipp3r - Rapport - Diogo Lopes da Sila : BSc_2022_ID228_LopesDaSilva_Diogo_Flipp3r.pdf
- Compte rendu Toggl Track : Toggl Track - Flipp3r - Bruno Costa
- Aide aux bénévoles : Aide pour les responsables du flipp3r.docx
- Autres éléments (lien du dropbox) : Autres éléments.txt
- Rapport du projet de calibrage : Calibrage - Rapport - Traitement d'image.docx
- Documentation Utilisateur, explication du gameplay du jeu : Documentation Utilisateur - Gameplay.docx
- Vidéo de démonstration : FLIP3R_V3.mp4
- Vidéo concept art : GameplayConceptArt.mp4
- Questionnaire google forms : Questionnaire - Flipp3r - Google Forms.pdf
- QR Code vers le questionnaire google forms : Questionnaire - Flipp3r.docx
- Modèle pour les PV de réunion : Réunion - Modèle.dotx
- Présentation du passage au 100% : Réunion Flipp3r - 100% - Organisation.pptx

Table des figures

1.1	Le consortium représentant les membres du groupe.	1
1.2	Le 1er prototype du Flipp3r.	2
2.1	Le serveur discord utilisé.	3
2.2	Le Wiki GitLab utilisé.	4
2.3	La fonctionnalité de milestones de GitLab.	4
2.4	Une issue de GitLab.	5
2.5	La fonctionnalité de board de GitLab.	5
3.1	Un jeu de bagatelle.	7
3.2	L'ancêtre du flipper.	8
3.3	Le jeu Space Cadet sur Windows XP.	8
4.1	Le premier concept art du boss du jeu.	12
4.2	Le premier plateau de jeu.	12
5.1	La bille avant la collision.	18
5.2	La bille après la collision.	18
5.3	La bille collisionne dans le vide.	19
5.4	Trois rayons émis depuis la bille.	20
5.5	Un objet passe dans les mailles du filet de collision.	20
5.6	La bille va avoir une collision avec un coin.	21
5.7	Des sphères sont envoyées depuis la bille.	21
5.8	La collision d'une bille dans un coin.	22
5.9	Calcul du vecteur de retour après collision dans un coin.	22
5.10	Rebond de la balle lorsqu'elle glisse sur le flipper.	24
5.11	Rebond de la balle lorsqu'elle est renvoyée à la collision, une reprise de volée.	24
5.12	Exemple de courbe de bézier.	25
5.13	Exemple d'une spline de rampe.	25
5.14	Effet visuel de flammes.	26
5.15	Effet visuel d'électricité.	26
6.1	Un flipper du jeu.	28
6.2	Composant Hinge Joint d'un flipper.	28
6.3	Un bumper du jeu.	29
6.4	Un exemple d'une spline utilisée pour une rampe.	29
6.5	Réglages du système de particules de Unity.	30
6.6	Particule de l'effet de flammes.	31
6.7	Émission des flammes.	31

6.8	Émission des flammes vue de dessus.	31
6.9	Effet visuel de l'explosion sans lumière.	32
6.10	Effet visuel des étincelles.	32
6.11	Réglage des collisions du système de particules.	32
6.12	Exemple d'un effet visuel créé via VFX Graph.	33
6.13	Exemple d'arc électrique.	33
6.14	Node pour l'effet d'arc électrique.	33
6.15	Shader graph pour l'effet d'électricité.	34

Bibliographie

- [1] Mandril, "figura," 2019. [Online]. Available : http://www.mandrill.ch/#videos?video_id=14
- [2] harish, "gitlab-runner : prepare environment failed to start process pwsh in windows," 2021. [Online]. Available : <https://stackoverflow.com/q/68050125>
- [3] ——, "Answer to "gitlab-runner : prepare environment failed to start process pwsh in windows"," 2021. [Online]. Available : <https://stackoverflow.com/a/68050126>
- [4] gitlab, "Install gitlab runner on windows | gitlab." [Online]. Available : <https://docs.gitlab.com/runner/install/windows.html>
- [5] Y. Loetzer, "How to setup ci pipeline · wiki · yan loetzer / dragon ci," 2021. [Online]. Available : <https://gitlab.com/yanniboi/dragon-ci/-/wikis/How-to-setup-CI-Pipeline>
- [6] M. Gonzalez, "Continuous integration in unity with gitlab ci/cd : Part 1," 2019. [Online]. Available : <https://medium.com/etermax-technology/continuous-integration-in-unity-with-gitlab-ci-cd-part-1-e902c94c0847>
- [7] M. Bellis, "The entire history of pinball and pinball machines," 2019.
- [8] Daniel Wong, Darren Earl, Fred Zyda, Ryan Zink, Sven Koenig, Allen Pan, Selby Shlosberg, Jaspreet Singh, Nathan Sturtevant, *Implementing games on pinball machines*, A. Press, Ed. DOI.org Crossref, 2010. [Online]. Available : <http://portal.acm.org/citation.cfm?doid=1822348.1822380>
- [9] J. Lasseter, "Principles of traditional animation applied to 3d computer animation," *SIGGRAPH '87*, 1987. [Online]. Available : <https://doi.org/10.1145/37401.37407>
- [10] Sunil Hadap, Dave Eberle, "Collision detection and proximity queries siggraph 2004 course," p. 287, 2004.
- [11] Psemata, "Collision detection for fast moving objects," Apr. 2022. [Online]. Available : <https://forum.unity.com/threads/collision-detection-for-fast-moving-objects.1268072/>
- [12] G. A. Prod., "Unity vfx graph - electric arc tutorial - youtube," 2022. [Online]. Available : https://www.youtube.com/watch?v=8NWqTKYEIIU&ab_channel=GabrielAguiarProd.
- [13] ——, "Unity vfx graph - electricity tutorial (procedural shader)," 2021. [Online]. Available : https://www.youtube.com/watch?v=Afh5zY6zxLs&ab_channel=GabrielAguiarProd.
- [14] Brackeys, "Introduction to audio in unity - youtube," 2017. [Online]. Available : https://www.youtube.com/watch?v=6OT43pvUyfY&ab_channel=Brackeys