

Bachelor of Science HE-ARC in Engineering

Orientation : Développement Logiciel et Multimédia (DLM)

Projet n°228
Flipp3r

Réalisé par
Diogo Lopes da Silva

Encadrement : Prof. Dr. Gobron Stéphane
HE-Arc, HES-SO
Expert : Dr. Salamin Patrick
Logitech

Version française

Ce rapport de Bachelor parle du développement du jeu Flipp3r mandaté par l'artiste Neuchâtelois Marc Mandril. L'envie est de créer un flipper jouable à trois le tout étant projeté sur une grande surface de quasiment trois mètres. Un dessin viendra agrémenter la projection pour donner un effet de profondeur et réaliser un dessin augmenté. Pour arriver au bout du jeu, il faut défaire le boss du jeu en attaquant ses points faibles. Comme il y a trois joueurs, chacun aura droit à sa zone particulière avec sa gravité personnalisée. Chaque joueur peut envoyer sa bille dans la zone d'un autre afin de coopérer dans les objectifs.

Le jeu est dans un état stable et le cahier des charges respecté, le projet est également prêt à recevoir des améliorations futures.

Mots clefs : Flipper ; Multijoueur ; Projection ; Dessin augmenté

English version

This Bachelor's report talks about the development of the game Flipp3r commissioned by the Neuchâtel artist Marc Mandril. The idea is to create a pinball game playable by three people, projected on a large surface of almost three meters. A drawing will be added to the projection to give an effect of depth and to realize an increased drawing. To reach the end of the game, you have to defeat the boss of the game by attacking his weak points. As there are three players, each one will have his own zone with his own gravity. Each player can send his marble in the zone of another one in order to cooperate in the objectives.

The game is in a stable state and the specifications met, the project is also ready to receive future improvements.

Keywords : Pinball ; Multiplayer ; Projection ; Augmented drawing

Remerciements

Je tiens à remercier premièrement mon collègue Bruno Costa, sans qui ce projet n'aurait pas pu être réalisé en si peu de temps.

Je remercie nos superviseurs, dans un premier temps M. Gobron Stéphane pour ses précieux conseils. Bien que nous n'ayons pas eu la possibilité d'utiliser son expertise en matière de gameplay, il a tout de même su nous donner de bons conseils pour le bon déroulement ce projet. Dans un second temps, merci M. Le Callennec Benoît pour son aide sur les spécificités de Unity.

Merci à M. Monti Max pour la supervision du projet et la communication avec le NIFFF. Merci à Volpe Ricardo pour l'aide sur le montage du flipper ainsi que les raccordements électriques. Merci à M. David Grunenwald pour la confiance qu'il nous a apportée et sa supervision. Merci aux menuisiers pour avoir construit le flipper et sans qui le projet n'aurait pas été aussi impressionnant.

Finalement, merci aux artistes, Marc Mandril pour nous avoir donné l'occasion de créer ce projet et Christopher Lanza pour l'immense travail qu'il a fourni.

Table des matières

1	Introduction	1
1.1	Contexte	1
1.2	Spécificité du jeu	2
1.3	Cahier des charges	2
1.4	Consortium	2
2	Gestion de projet	5
2.1	Communication	5
2.2	Sprints	6
2.3	Intégration Continue	6
2.4	Suivi de projet	6
2.5	Outils et technologies utilisés	6
3	Etat de l'art	7
3.1	Flipper ou Pinball	7
3.2	Flipper numérique	7
3.3	Unity	7
3.4	Multijoueur	8
3.5	Projection	8
4	Méthodologie	11
4.1	Concepts et modèles	11
4.2	Solutions aux problèmes exposés	13
4.3	Implémentation et déploiement	14
5	Tests et résultats	23
5.1	Résultats de l'approche	23
5.2	Test utilisateur / sondage pour valider le modèle	24
6	Discussion	27
7	Conclusion	29
7.1	Réalisation	29
7.2	Perspective	29

0Définitions

Acronymes

HE-Arc	Haute Ecole Arc, Neuchâtel
HES-SO	Université des Sciences et Arts Appliquée de Suisse Occidentale
OpenCV	<i>Open Computer Vision.</i>

Lexique

OpenCV . Bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel [Wikipédia].

Chapitre 1

Introduction

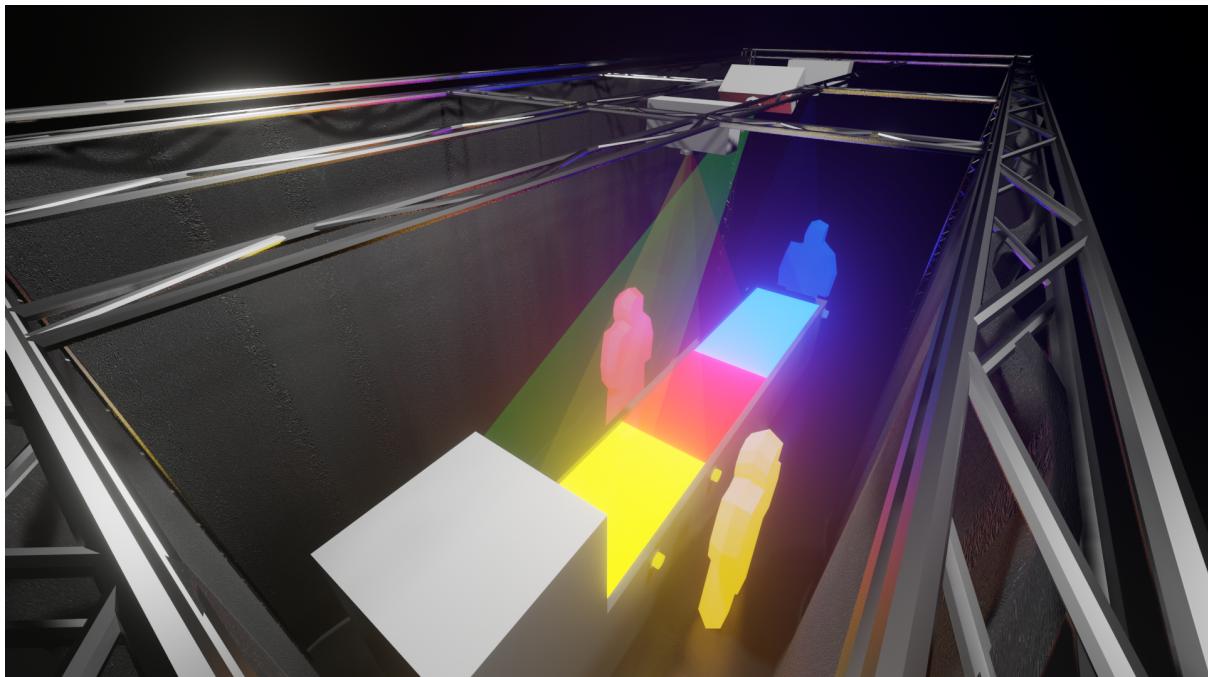


FIGURE 1.1 – Premier prototype du Flipp3r

1.1 Contexte

Le jeu "Flipp3r" a été réalisé dans le cadre du projet de bachelor de fin de 3e année à la Haute École Arc de Neuchâtel. Il est mandaté par l'artiste Marc Mandril et en collaboration avec Christopher Lanza pour les modèles 3D. Le projet a pour but d'être exposé pour toute la durée du «NIFFF», un célèbre festival neuchâtelois, et ainsi être utilisé par plusieurs utilisateurs. Les joueurs vont donc avoir une expérience inédite d'un flipper à trois joueurs, agrémenté par des dessins pour donner un effet de profondeur. Le tout est projeté sur une grande table en bois de 2m80 rappelant la forme L du Tetris. L'application a été réalisée sur le moteur de rendu temps réel Unity et implémentée en langage C#.

1.2 Spécificité du jeu

Comme dit précédemment, le «Flipp3r» sera joué par trois utilisateurs en simultané. Pour ce faire, le plateau sera séparé en trois régions distinctes où une "pseudo-gravité" fera effet dans chacune d'elles. Le but du jeu sera de coopérer afin de viser certains éléments du plateau pour infliger des dégâts au boss du jeu. Il existe plusieurs phases de jeu où le boss change de forme selon les points de vie qu'il lui reste.

1.3 Cahier des charges

Étant deux sur ce projet, les tâches ont été séparées afin de convenir le mieux aux deux étudiants.

1.3.1 Objectifs principaux

- Gestion de la projection
 - Position, réglages, calibrage
- Choix du vidéoprojecteur
 - Nombre de vidéoprojecteurs
 - Quel vidéoprojecteur
- Interfaçage des boutons et capteurs
- Gestion de la lumière
 - Illumination directe et indirecte
- Gestion de la caméra
- Tests gameplay et projection
- Affichage panneau vertical
 - Score
 - Animation finale

1.3.2 Objectifs secondaires

- Shader
- Transformation du terrain
- Tweaking des paramètres du flipper
- Vérifier que le storytelling soit bien intégré comme le veulent les artistes
- Mise en veille

1.4 Consortium

Le projet inclu beaucoup d'acteurs dont les métiers sont complètement différent. Il est bon pour ça de représenter l'entièreté des participants au projet en indiquant leur part dans le projet.

Projet Flipp3r

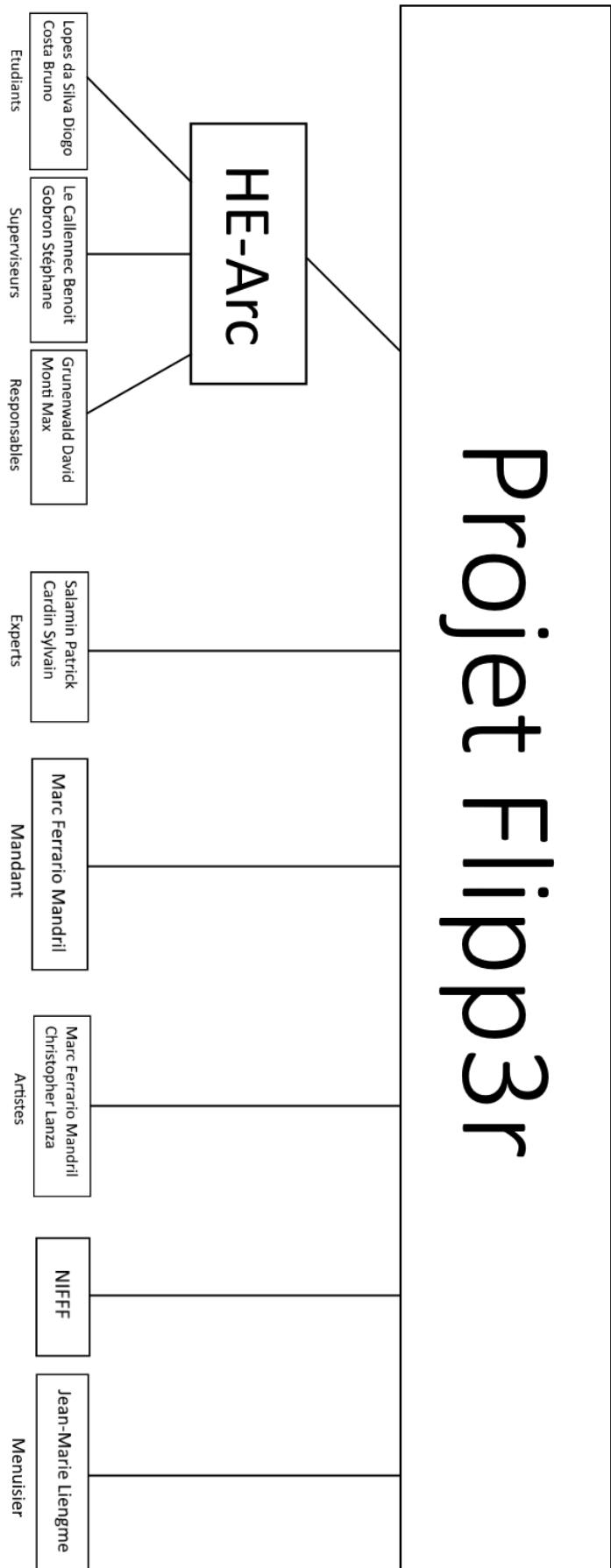


FIGURE 1.2 – Consortium du projet Flipp3r

Chapitre 2

Gestion de projet

Comme précisé précédemment, la communication entre personnes n'étant pas dans le même domaine est difficile. Dans ce chapitre, les différents moyens de communication seront listés et expliqués.

2.1 Communication

Pour pouvoir communiquer entre les différents acteurs de ce projet, trois outils ont principalement été utilisés.

2.1.1 Discord

Discord est un outil de messagerie instantanée similaire à Slack ou encore Telegram. Ce logiciel a été choisi parce que les étudiants, certains enseignants et les artistes avaient déjà l'habitude de l'utiliser. Il nous permet donc de communiquer par écrit ou vocal, partager des fichiers ou encore partager son écran lors de réunions. Cependant, certaines personnes n'avaient pas la possibilité d'utiliser Discord. Deux autres moyens ont donc été utilisés pour passer les messages.

2.1.2 Mail

Le mail est le meilleur moyen d'envoyer une information à tous les acteurs du projet simultanément. De plus, certaines personnes n'étaient joignables que par ce biais. M. Monti, s'occupait de la communication avec le personnel du NIFFF et relayait les informations par mail.

2.1.3 Microsoft Teams

Microsoft Teams est un logiciel appartenant à Microsoft similaire à Discord. La He-Arc Industrie avait mis en place un environnement Teams pour les étudiants et le personnel lors de la pandémie du Covid-19. Cet outil a été utilisé pour les réunions hebdomadaires avec M. Gobron quand il n'était pas possible de le faire en présentiel. Il est aussi possible que les réunions se passent parfois par appel téléphonique lorsqu'une des deux parties n'est pas disponible sur Teams.

2.1.4 WhatsApp

Enfin, un groupe WhatsApp a été créé entre les deux étudiants et artistes qui ont eu besoin de beaucoup discuter. En effet, les concernés étaient généralement joignable plus rapidement via ce moyen, ce qui permet de poser une question et de recevoir une réponse rapidement.

2.1.5 Dropbox

Discord pose certains problèmes, premièrement le partage de beaucoup de fichiers manque d'organisation. Étant donné que l'artiste des modèles 3D partageait plusieurs dizaines de fichiers Blender, il était impossible d'utiliser uniquement Discord. Deuxièmement, Discord limite la taille des fichiers 8Mo gratuitement ce qui n'était pas suffisant. Les étudiants et artistes ont donc décidé d'utiliser Dropbox pour tout ce qui est partage de modèles, vidéos et sons.

2.2 Sprints

Afin d'assurer une bonne avancée du projet, les étudiants ont décidé de faire des sprints toutes les semaines. Ainsi, chaque début de semaine, les deux étudiants s'organisaient les tâches à réaliser pour la semaine. En début de semaine, une réunion était organisée principalement avec les artistes afin de voir où en est le projet. Les milestones de Gitlab ont été utilisées pour que tout le monde puisse voir le sprint en cours avec les différentes tâches.

2.3 Intégration Continue

À chaque nouvel ajout, il le faut tester sur la PC de jeu, mais sans perdre trop de temps à installer la nouvelle version. Pour ça, à chaque ajout, l'application s'installait automatiquement du le PC via le runner de Gitlab.

2.4 Suivi de projet

Afin de garder un bon suivi de projet et des ressources utilisées, deux applications sont sélectionnées. Premièrement Toggl Track qui est l'équivalent d'un journal de travail, mais précis à la seconde. L'utilisateur peut indiquer quand commence une tâche, et l'arrêter quand il le souhaite. L'application va ainsi calculer automatiquement le temps pris sur chaque tâche.

Zotero est une application pour sauvegarder les ressources utilisées sur Internet. L'avantage est qu'il peut automatiquement détecter si c'est un forum, un blog, un document académique, un livre et d'autres catégories. C'est l'équivalent d'un Bookmark de n'importe quel navigateur mais permettant une meilleure organisation. Il est également possible de créer des groupes d'utilisateurs, ainsi les sources se regroupent et sont disponibles à tout un chacun.

2.5 Outils et technologies utilisés

Comme précisé dans l'introduction, le jeu a été développé sur Unity avec langage C# pour les scripts avec l'IDE Visual Studio Code. Pour contrôler l'ordinateur de jeu à distance, notamment lors du NIFFF lorsque les étudiants ne sont pas sur place, TeamViewer est utilisé.

Chapitre 3

Etat de l'art

Maintenant qu'il est possible de comprendre la grandeur et l'ambition du projet, il est possible de s'attaquer à son originalité. En effet, si un flipper à trois joueurs projeté existait déjà, ce projet n'aurait pas grand intérêt.

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. Son souhait était d'améliorer le jeu déjà existant de la "Bagatelle". Le but du jeu était d'utiliser une queue de billard pour envoyer une bille et générer des points selon l'endroit où elle termine. 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.

La révolution que nous voulons apporter est l'application d'un mode multijoueurs coopératif. Là où l'on pouvait imaginer deux flippers qui communiquent entre eux pour s'aider, nous, nous voulons que l'expérience soit commune et dans un seul flipper.

3.2 Flipper numérique

Étant en informatique, il est évident que le développement de ce flipper se fera en partie numériquement. Pour ça, il faut s'inspirer de ce qui a été fait en matière de flipper. On pense bien évidemment au pinball de Windows, mais des dizaines d'autres exemples existent. La thèse [1] explique le développement d'un flipper en Java et C++, elle permet de voir les réflexions déjà prises pour développer ce genre de jeu.

Une autre source [2] explique également d'une autre façon comment développer son flipper mais avec le framework Qt. Il intègre également une notion de fluide, qui permet de faire des interactions intéressantes avec les balles.

3.3 Unity

Unity était un des prérequis dans ce Bachelor, de plus, les étudiants avaient déjà de l'expérience avec ce moteur de jeu en temps réel.

Le document [3] explique de manière complète les étapes de développement d'un jeu 3D sur Unity. Il va aussi bien dans des notions de base comme des notions avancées comme les UV Layout utilisés avec les lumières.

En parlant de lumière, l'ambiance d'un jeu est fortement affectée par sa lumière et ses ombres. Nous pouvons notamment voir dans le document [4] une explication des différentes lumières dans Unity et de comment les utiliser. Elles permettent de faire beaucoup d'effet et de changer drastiquement l'impression d'un joueur.

Un autre point qui affecte énormément le visuel est les shaders, car ils permettent de plus ou moins tout faire visuellement. Le document [5] explique bien les notions de shader dans l'infographie et comment s'en servir dans des moteurs de rendus comme Unity avec le ShaderGraph.

Maintenant, pourquoi devons-nous obligatoirement utiliser le moteur Unity ? Ne pourrait-on pas utiliser d'autres moteurs tels qu'Unreal Engine ? Sur quels points est-ce que nous pourrions les comparer ?

Ce bachelor [6] permet exactement la comparaison entre ces deux moteurs. Ce qu'essaie d'expliquer l'auteur, est que dans certaines instances, Unity est meilleur et d'autres Unreal est meilleur. Mais, quelle que soit l'issue, Unreal permet de dépasser des limites que Unity pose, mais sans bonne raison il n'y a pas besoin de changer pour Unreal.

Ce document [7] quant à lui compare les performances entre les différentes techniques d'éclairage globale. Il compare d'un même moteur de rendu et entre différents moteurs de rendu tant en termes de performance qu'en termes de qualité visuelle. Il permet d'avoir une bonne idée des limites d'un moteur et à quel moment il faut changer.

Pour le choix d'un moteur de rendu en temps réel il n'y a pas de vérité absolue. Il faut se fier à plusieurs paramètres changeant selon la situation ainsi qu'au confort et développeur sur ces moteurs.

3.4 Multijoueur

Composante dominante et plus originale de ce projet, il est très important de vérifier s'il existe déjà quelque chose de ce style. La ressource [8] est le développement d'une application contrôlant un réel flipper. Ce qui est intéressant avec ce document c'est qu'il y a une implémentation d'une dimension multijoueur dans le jeu. Chaque joueur lance à son tour une balle et doit faire le plus de scores possible.

Même si ça peut valoir comme du multijoueur, ça ne reste pas ressemblant à ce que nous voulons faire. Une autre ressource [9] indique le développement d'une IA contrôlant un flipper. L'implémentation multijoueur de ce flipper est faite pour faire affronter l'IA face à de véritables joueurs. Mais chaque joueur/IA contient son propre flipper distinct.

3.5 Projection

Autre grande composante de ce projet, la projection du flipper. Idée intéressante et innovante, aucune ressource n'a été trouvée pour un flipper projeté. Cependant, qui dit projection dit décalage et qui dit décalage dit calibrage. Heureusement, le calibrage est un sujet très recherché et

documenté et il est facile de trouver des ressources à ce sujet. Par exemple, la ressource [10] est un document expliquant une calibration de caméra via un pattern donné. Ce document contient des éléments mathématiques expliquant comment fonctionne la calibration d'une caméra avec des matrices.

Comme dit avant, il existe d'autres patterns, [11] est un document qui montre la calibration d'une caméra utilisant un damier qui est une solution utilisable pour calibrer le projecteur. Il est même possible d'aller jusqu'à l'analyse de la distorsion d'une lentille comme le montre la ressource [12] et comment le calibrer.

Finalement, une autre technique est d'utiliser un encodage qui s'appelle un Arucos. L'aruco est simplement un identifiant sous la forme d'une image de la même façon que les codes QR avec les chaînes de texte. Ils appartiennent à un dictionnaire qui permet de les reconnaître. La ressource [13] explique la calibration d'une caméra en utilisant cette fois-ci les arucos. La source est en portugais, mais l'étudiant rédigeant ce rapport est bilingue et comprend complètement ce qu'il se dit.

Chapitre 4

Méthodologie

Après avoir compris à quel point le projet est original, nous pouvons passer aux premiers concepts puis définir les problèmes qui le composent.

4.1 Concepts et modèles

4.1.1 Concept du jeu

Trois joueurs et ainsi trois zones, chacune d'entre elle est un flipper avec sa propre gravité. Ces parties comprennent les éléments de bases d'un flipper ainsi que d'autres originaux. 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, il a été décidé de créer un style basé sur l'Art Déco. Des décors ressemblants au jeu "BioShock" ou encore au film "Métropolis" étaient le souhait du mandant. Finalement, le jeu Flipp3r possède un style bien défini où l'art déco semi-futuriste se mélange à des animations rapides et non réalistes.

4.1.2 Plateau de jeu

L'une des premières choses qui ont été intensivement recherchées avec les artistes est le plateau de jeu. Il était difficile pour les deux parties d'avancer sans avoir une image d'à quoi le plateau ressemblerait. Le focus principal était d'avoir trois zones que les joueurs pourraient exploiter d'eux-mêmes mais qu'ils pourraient tout aussi bien communiquer entre eux.

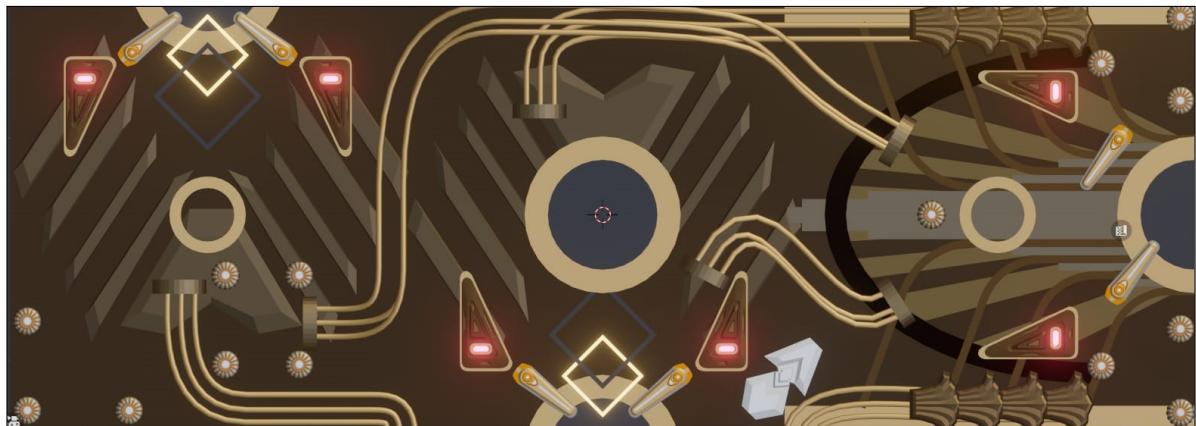


FIGURE 4.1 – Première version du plateau décidée après plusieurs heures de réunions

On peut voir dans la figure 4.1 la volonté d'avoir des rampes qui permettent une meilleure circulation, le plateau étant très grand.

4.1.3 Boss du jeu

Une partie intégrante des flippers est l'histoire, elle permet d'agrémenter l'envie d'avancer dans le jeu et donne des objectifs. Ici l'histoire est de défaire le robot maléfique. Le design a totalement été laissé à Christopher Lanza qui a modélisé tous les assets 3D du jeu.

L'idée est que le boss est constitué d'une base mécanique recouverte par un visage destructible, le tout attaché à une colonne vertébrale articulée.



FIGURE 4.2 – Première version du boss dans un état de standby



FIGURE 4.3 – Première version du boss dans un état de taunt



FIGURE 4.4 – Première version du boss dans un état de destruction avancé

On peut donc voir dans chacune de ces images la volonté d'avoir un sentiment de progression avec une destruction du boss. Pour avoir une fin impactante, l'idée est que derrière toute cette armature se cache le cerveau du boss qui serait en réalité une bille. Une fois toutes les parties détruites, le cerveau tomberait sur le plateau à très grande vitesse en détruisant tout sur son passage.

4.1.4 Projection

Un document dans l'annexe nommé "Calibrage - Rapport - Traitement d'image.docx" offre déjà un historique de l'évolution du calibrage. Cependant, des évolutions ont été apporté notamment l'utilisation des arucos. Ces arucos sont en réalité simplement des identifiants codés en image comme peuvent l'être les codes QR avec les chaînes de caractères. Ainsi il y a 4 identifiants utilisés, de 0 à 3, et qui sont imprimés. L'idée et d'avoir deux photos avec dans une les arucos imprimés et dans l'autre les mêmes arucos, placés aux mêmes endroits, mais possiblement déformés. Avec ces deux images, il est possible de calculer une matrice pour passer d'une image à l'autre.

4.2 Solutions aux problèmes exposés

4.2.1 Projection

Un des plus impactant se passe notamment au niveau hardware. En effet, comme nous utilisons des projecteurs fixés, il est possible qu'un décalage se crée. La cause peut être simplement la fixation qui n'est pas suffisamment solide ou bien les utilisateurs qui seraient plutôt physiques. Pour pallier à ce problème, il faut trouver un moyen de replacer la projection, et ce par du calibrage. Dans Unity, pour afficher quelque chose en temps réel à l'écran il faut passer par des caméras. Il est possible de jouer avec les matrices de projection afin de déformer, ou bien dans notre cas reformer, une image.

Maintenant la question qui se pose est, comment avoir la matrice de projection permettant de reformer l'image ? Comme montré dans l'état de l'art, des damiers ou des arucos nous permettent

de calculer ensuite cette matrice. Tout ce qu'il reste à faire c'est d'utiliser ces valeurs dans Unity et le tour est joué.

Malheureusement, cette situation, bien qu'elle paraisse idéale, n'est pas satisfaisante étant donné que ce sont des bénévoles qui gèrent l'installation. Devoir poser une caméra, poser les arucos, prendre une photo, changer de scène, reprendre une photo, lancer le programme, intégrer la matrice puis finalement lancer le jeu nécessite beaucoup d'étapes. En réalité, il est plus simple de sélectionner les valeurs intéressantes dans la matrice et créer des raccourcis pour les changer à la main.

4.2.2 Lumières

À l'origine, le jeu ne devait pas être totalement sombre, mais uniquement deux zones dans la partie rouge qui devait cacher des bumpers. Cependant, Unity n'offre pas la possibilité de créer des zones sans "toit" non affectées par des lumières. Le seul moyen de n'avoir aucune lumière et de cacher des zones est de retirer toutes les lumières ambiantes mais le jeu dans ce cas est totalement sombre. Pour essayer d'avoir le meilleur des deux mondes, il a été décidé que chaque objet et balle allaient émettre un peu de lumière. Ainsi, chaque objet est visible de lui-même et la balle éclaire la majorité du plateau donnant ainsi un sentiment de découverte au joueur.

4.2.3 Changement de vidéos

Pour le boss, il a été décidé de ne pas créer les animations sur Unity pour gagner du temps et des ressources. Les animations ont été remplacées par des vidéos qui sont simplement joués lorsque les conditions sont réunies (e.g. attaque du boss). Cependant, le système de vidéo de Unity n'est pas très avancé. Dans Unity, pour jouer une vidéo il faut un "Player" et un player contient une "Source" qui peut être un fichier mp4, wav, etc. Ainsi il est techniquement possible d'avoir un Player et de modifier sa source à chaque fois. Malheureusement, quand il y a un changement de source dans un Player, un flash blanc apparaît qui est dû au buffering de la vidéo. Sans aucun système de prébuffering, une solution est d'avoir un Player pour chaque vidéo et de simplement mettre en premier plan la vidéo correspondante.

4.3 Implémentation et déploiement

Après avoir compris le concept, les problématiques et leurs solutions, il est temps de s'attarder sur comment elles ont été réalisées.

4.3.1 Projection

Comme expliqué dans les problématiques, il est plus simple de modifier à la main la projection petit à petit avec des raccourcis. Cependant, le code pour pouvoir récupérer la matrice de projection avait déjà été réalisé. Le code est un script python qui utilise la librairie OpenCV, très connue pour réaliser des manipulations sur des images. Cette librairie a entre autres, une implémentation des arucos et permet une détection de ces derniers dans une image. La fonction OpenCV demande en paramètre l'image, le dictionnaire ainsi que des paramètres tels que des thresholds. Il essaie de détecter les arucos en "aiguisant" l'image, la rendant noir et blanc via le threshold puis les détecter. Il fonctionne mieux quand les arucos ont des zones blanches autour d'eux, car le passage noir et blanc est très contrasté. Il retourne les coins qui ont été détectés comme faisant partie d'un aruco, les identifiants correspondant ainsi que les coins qui ont été

rejetés lors de la détection.

Donc l'implémentation consiste en récupérant deux images, les passer chacune dans cette fonction, récupérer les coins et les identifiants puis récupérer la matrice d'homographie. Bien heureusement, OpenCV offre également une solution pour ça, une fonction nommée "findHomography" qui demande deux tableaux de points. Ce qu'il va faire c'est trouver et retourner la matrice qui permet de passer du premier tableau de points au deuxième. De cette façon, la matrice de projection peut être sauvegardée et utilisée plus tard dans Unity.

Même si cette façon fonctionne, comme dit précédemment, ce n'est en réalité pas très pratique ou faisable pour des bénévoles. Il a donc été préférable d'utiliser un changement manuel, certes moins précis, mais beaucoup plus rapide.

La matrice de projection des caméras Unity est une matrice 4 par 4 (3 dimensions avec coordonnées homogènes) qu'on peut très facilement modifier. Elle est accessible sous le nom "projectionMatrix" et toutes ses 16 valeurs sont publiques et modifiables.

$$P = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{-2}{far-near} & \frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

FIGURE 4.5 – Matrice de projection utilisée dans Unity

Les valeurs de quatrième colonne deuxième ligne et celle de la deuxième colonne et quatrième ligne sont celles qui sont ressorties comme les plus intéressantes. Elles correspondent à un décalage soit à gauche ou à droite soit en haut ou en bas. Il a été remarqué que si décalage il y avait, il était uniquement dans ces axes-là. Étrangement, ce décalage n'arrive pas tout le temps et est parfois résolu en relançant l'application. Il a donc été conclu que le décalage ne vient pas de l'installation, mais de Unity ou Windows 10. Selon le Dr. Sylvain Cardin, expert de M. Costa, qui travaille énormément avec Unity, ce problème viendrait du viewport qui serait mal calculé au lancement. Pour ne pas utiliser tout "l'écran" du projecteur mais seulement la partie qui couvre la table, nous passons par le viewport de Unity. Également, quand le décalage arrive, la partie du haut du flipper, le boss, elle n'est pas décalée et cette partie n'utilise pas de viewport.

C'est donc une potentielle piste pour rechercher comment résoudre ce problème dans une future version de ce projet.

4.3.2 Lumière

Pour les lumières, il a été vu précédemment dans les concepts qu'il fallait désactiver la lumière ambiante. Pour se faire, il faut savoir que dans Unity, il est possible d'utiliser du post-processing et que cette lumière vient de là. Dans le High Rendering Definition Pipeline, qui est celui de plus haute qualité dans Unity, un post-processing est déjà mis en place sous le nom de "Volume". Il faut donc simplement désactiver cet effet de post-processing ou bien simplement mettre son intensité à 0 et il n'y a plus de lumière.

Ainsi, chaque objet va de lui-même créer de la lumière via une "Point Light" qui peut s'apparenter à une ampoule. Il est possible de régler la couleur de cette ampoule, son intensité, la distance à laquelle elle émet et la génération d'ombre qu'elle peut produire.



FIGURE 4.6 – Image rapprochée d'un flipper vert et de l'intéraction de la lumière avec un élément du décors

C'est donc un travail où la difficulté ne se trouve pas dans sa technicité, mais dans le temps qu'il faut y mettre pour avoir un résultat satisfaisant. Même si dans Unity il est possible d'utiliser les prefabs pour avoir plusieurs objets identiques dans une scène ce n'est pas suffisant. Les trois zones différentes qui ont chacune des couleurs différentes font qu'il y a un besoin de changer les prefabs et d'en créer des nouveaux. Le seul cas dans lequel les prefabs sont réellement utiles est pour les bumpers qui eux se répètent souvent dans une même zone. De plus, il a fallu beaucoup de concertation avec les artistes pour savoir si l'aspect général du jeu leur plaisait.

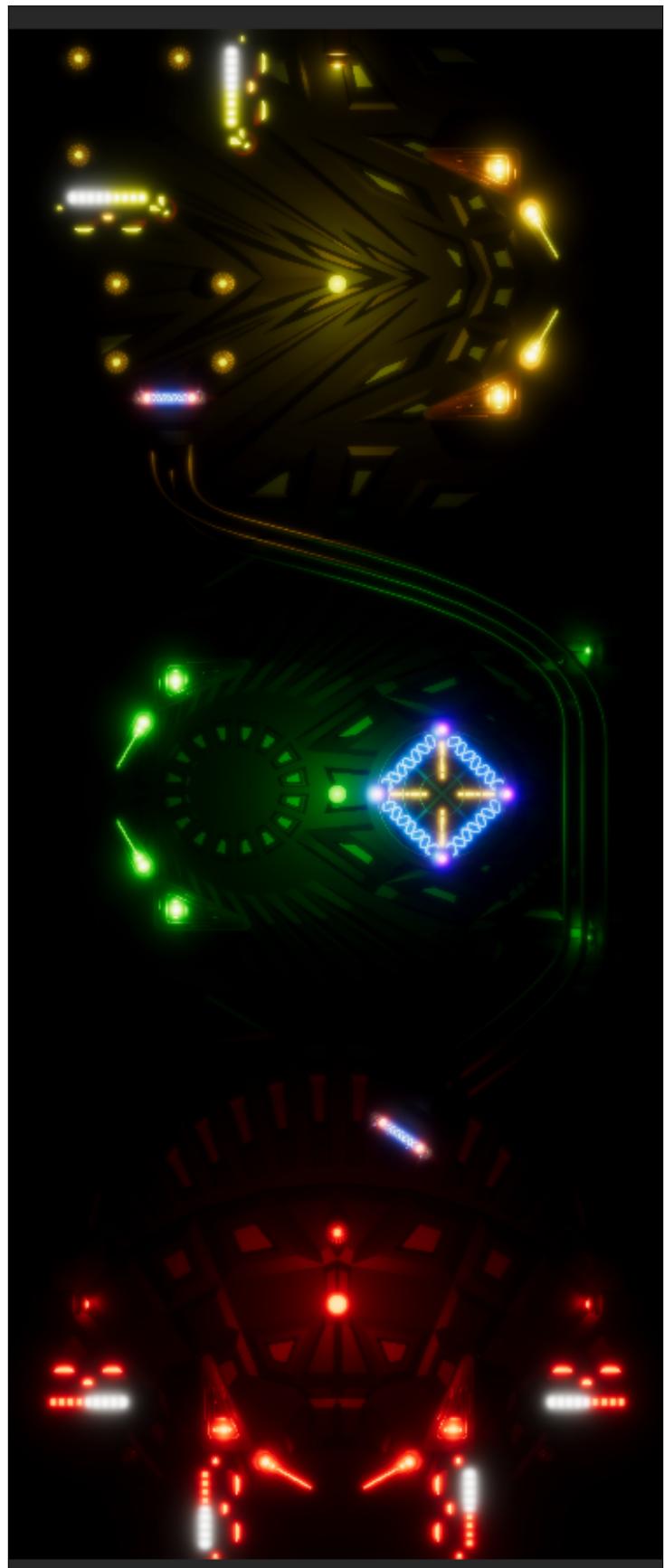


FIGURE 4.7 – Image du plateau de jeu dans le noir avec uniquement les lumières des objets

4.3.3 Gameplay

Une fois le jeu lancé on se retrouve dans une phase silencieuse, le boss est éteint, aucune balle et seulement certains bruits de fond. Une fois qu'un joueur appuie sur un des boutons, le boss s'enclenche et une fois allumées les balles apparaissent.



FIGURE 4.8 – Boss du jeu en attente d'une activité d'un joueur

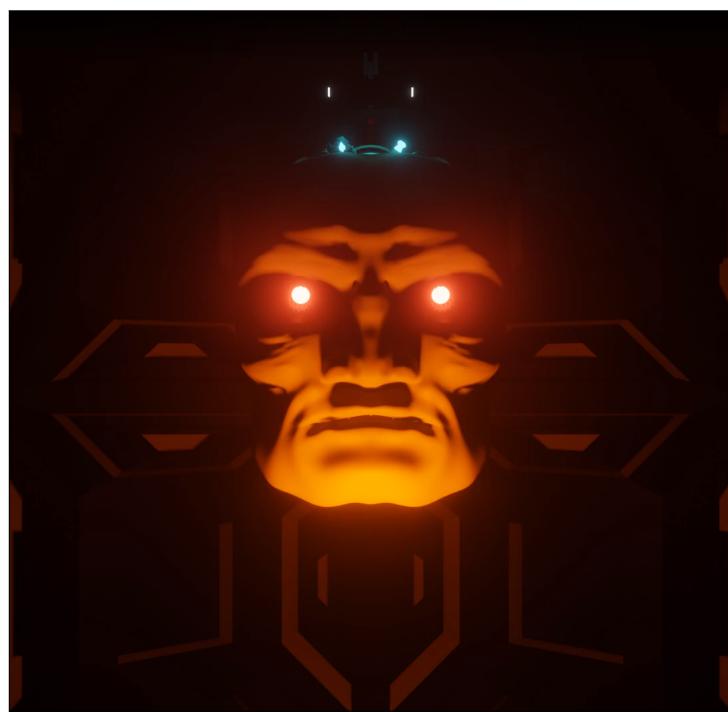


FIGURE 4.9 – Boss du jeu dans sa première phase

À partir de ce là, le déroulement du gameplay est assez linéaire. Dans chaque zone un bumper peut devenir une faiblesse, il devient bleu et une cible le pointe. Quand un joueur réussit à taper dans ce bumper, il envoie des dégâts au boss. À ce moment il y a deux choix, soit le boss a suffisamment de vie et se prend juste des dégâts, soit il subit une destruction. Dans le cas d'une attaque simple, s'il n'a pas provoqué dans cette phase de jeu, il provoque puis il repasse en mode "idle". Si le boss a perdu suffisamment de vie, il se détruit en partie et passe à la prochaine phase.



FIGURE 4.10 – Boss du jeu dans sa deuxième phase

À tout moment, excepté quand la dernière séquence de destruction se lance, les joueurs peuvent perdre. Les conditions sont de perdre 15 balles, si plus aucune balle ne peut réapparaître la séquence de gameover se lance.

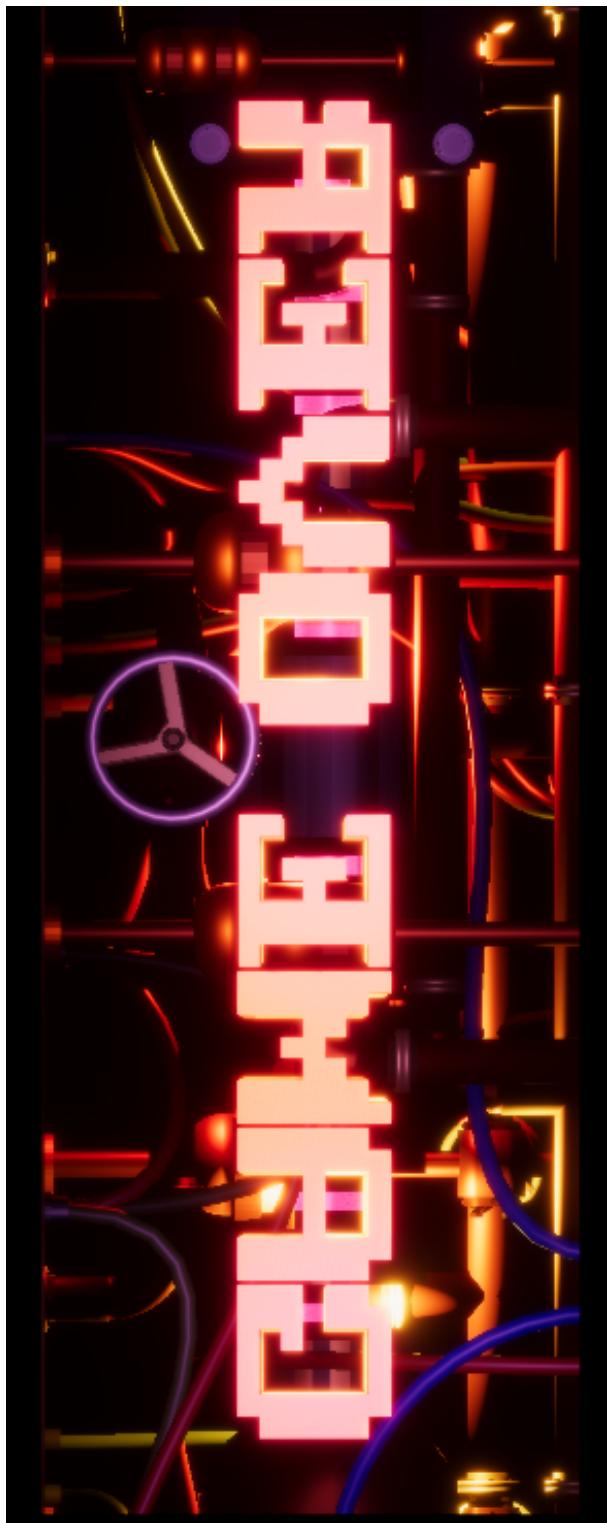


FIGURE 4.11 – GameOver du jeu quand il n'y a plus de balles qui peut apparaitre

Quand la séquence de destruction est finie, il faut s'assurer que ce n'est pas la phase finale, et on passe aux vidéos des prochaines phases. Ce système de répète pendant 5 phases et lors de la destruction finale, le cerveau du boss tombe dans le terrain. La balle commence par aller à très grande vitesse en ignorant les collisions, excepté les murs, et produisant un bruit oppressant. Après un certain temps, des explosions apparaissent avec un fondu au blanc progressif. Quand

l'entièreté du plateau est blanche, une musique douce et calme se joue pour signifier la fin du jeu.

4.3.4 Vidéos

Comme dit précédemment, il n'est pas possible d'utiliser un seul Player mais un par vidéo. Nous pouvons interpréter ça comme avoir un tableau de Player pour chaque situation (e.g. idle, destruction, hit, etc.). Qui dit tableau dit index, il y a un index par tableau (idleIndex, destructionIndex, hitIndex, etc). Ces index sont incrémentés à chaque destruction qui signifie un changement de phase, exceptés pour les vidéos de "hit" car il y en a deux par phase. Il faut donc un index local à la phase qui est simplement 0 ou 1 et hit est incrémenté de 2 à chaque destruction.

La première chose qui viendrait à l'esprit pour mettre une vidéo en évidence est simplement de toutes les désactiver et d'activer la bonne. Cependant, cette situation est soumise aux performances du PC et sur un ordinateur portable il arrivait qu'un flash ait lieu. Pour éviter que ça arrive sur l'ordinateur de jeu, toutes les vidéos sont activées, mais arrêtées. La façon de les mettre en avant est d'augmenter de 0.1f son axe Y ce qui la met au premier plan. De cette façon, il n'y a pas de flash, quel que soit le support et les transitions restent fluides.

Chapitre 5

Tests et 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.

5.1 Résultats de l'approche

L'état du programme est stable et fonctionnel, le jeu tient les tests d'endurance et ne crash pas. Bien que certains bugs sont encore présents, ils n'empêchent pas le déroulement du jeu et ne bloquent pas les joueurs.

5.1.1 Plateau de jeu et lumières

On peut voir l'aperçu final du jeu, c'est la première chose que voit les joueurs quand ils arrivent devant le flipper.

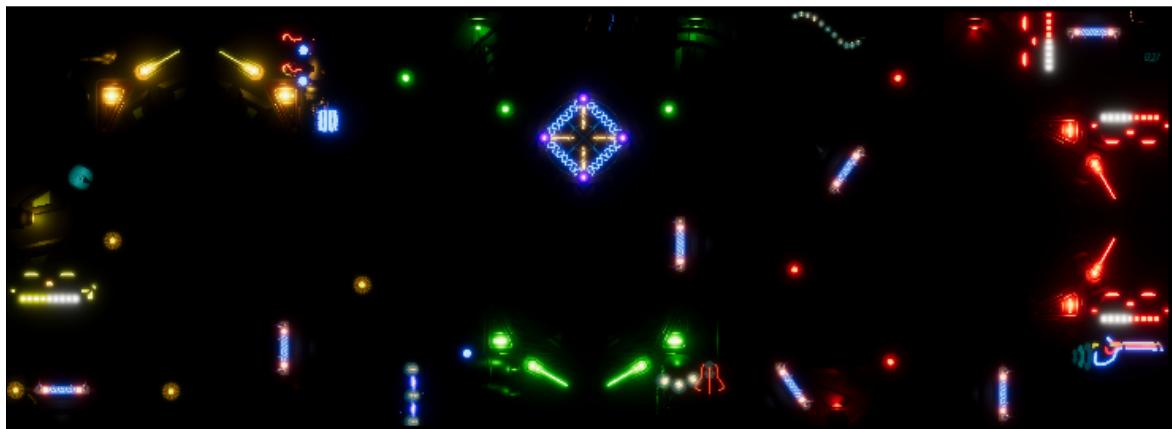


FIGURE 5.1 – Apperçu du plateau de jeu final

Il est intéressant de voir aussi beaucoup d'éléments du décors ne sont pas visibles. Ils ne vont l'être, qu'uniquement au moment où une balle passer proche d'eux.



FIGURE 5.2 – Apperçu du plateau de jeu final avec lumières pour voir les détails

5.1.2 NIFFF

L'installation était à disposition du public pendant une semaine pour l'activité "NIFFF Invasion". Des bénévoles étaient présents pour s'assurer du bon déroulement du jeu.

5.2 Test utilisateur / sondage pour valider le modèle

5.2.1 Tests de gameplay

Les tests de gameplay ont commencé uniquement à partir de la dernière semaine de travail. Il était nécessaire d'avoir l'installation finale pour pouvoir tester correctement. Ce sont d'abord les membres du projet Flipp3r qui ont testé le jeu et donné leurs avis.

- Trop de bumpers difficiles d'accès et certains bloquaient des chemins
- La gravité des balles était trop faible et les balles lentes
- Le portail dans le terrain vert était trop grand et bloquait trop de trajectoires
- Le jeu était trop difficile à terminer

Nous avons donc réajusté ces points, les bumpers ont été mieux repositionnés et seulement les plus faciles d'accès peuvent devenir des faiblesses. La gravité des balles a été graduellement augmentée pour convenir aux souhaits des utilisateurs. Certains éléments, tels que le portail dans la zone verte, ont été réduits pour laisser plus d'espace aux balles.

5.2.2 Tests utilisateurs

Pendant cette semaine, deux étudiants de la He-Arc sont venus tester le jeu. Leur ressenti était globalement pareil, mais en plus ils avaient du mal à saisir la notion de faiblesse. Pour mieux faire comprendre que ce sont ces bumpers qu'il faut prioriser, une cible apparaît en grand et se rétrécit sur les bumpers correspondants.

5.2.3 Tests d'endurance

Une partie de jeu dure entre cinq à dix minutes, il a fallu donc s'assurer que le jeu puisse au moins tourner pendant cette période de temps. Le jeu a tourné pendant toute une nuit pour s'assurer qu'il ne crash pas et le lendemain il tournait encore sans soucis.

5.2.4 Retour des utilisateurs

Un sondage a été réalisé sous la forme d'un Google Forms lors du NIFFF pour que les utilisateurs puissent donner leurs retours. Une douzaine de personnes ont donné leurs avis sur l'installation. Il est possible de retrouver dans les annexes les réponses complètes sous forme d'histogramme, mais voici un résumé des ressentis globaux :

- 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 n'est pas trop haute pour la majorité bien que certains le trouve trop dur ou pas assez
- Le public trouve le jeu assez maniable
- Quelques bugs ont été subis par certains utilisateurs, d'autres n'en n'ont eu aucun

Chapitre 6

Discussion

Il est un peu difficile de montrer des captures d'écran du jeu, car ils ne représentent pas le réel ressenti qu'il donne. Il est préférable de le tester, et pour avoir l'expérience complète d'avoir deux écrans sinon le boss n'apparaît pas. Ceci dit on peut constater avec les retours des utilisateurs que le ressenti global du jeu est plutôt bon même si individuellement il y a un ou deux points en dessous.

Le focus principal pour le NIFFF aux niveaux des bugs était d'éviter ceux qui bloquent la progression du jeu et qu'il faille relancer l'application. Heureusement, aucun bug de ce style n'apparaît et le pire qu'il peut arriver est un spawn de balle infinie.

Finalement, il serait préférable d'avoir plus de temps pour peaufiner le gameplay. Malheureusement, des conditions ont fait que l'installation finale n'était pas disponible assez tôt.

Chapitre 7

Conclusion

Pour conclure, voici les éléments qui ont été remplis dans le cahier des charges et les perspectives.

7.1 Réalisation

7.1.1 Objectifs principaux

- Gestion de la projection ✓
- Position, réglages, calibrage ✓
- Choix du vidéoprojecteur ✓
- Nombre de vidéoprojecteurs ✓
- Quel vidéoprojecteur ✓
- Interfaçage des boutons et capteurs ✓
- Gestion de la lumière ✓
- Illumination directe et indirecte ✓
- Gestion de la caméra ✓
- Tests gameplay et projection ✓
- Affichage panneau vertical ✓
- Score ✓
- Animation finale ✓

7.1.2 Objectifs secondaires

- Shader ✓
- Transformation du terrain
- Tweaking des paramètres du flipper ✓
- Vérifier que le storytelling soit bien intégré comme le veulent les artistes ✓
- Mise en veille ✓

7.2 Perspective

L'objectif serait de continuer de travailler sur le projet à l'avenir. Une exposition dans des musées ou d'autres endroits est envisagée et l'idée serait de créer des flippers avec des thèmes spécifiques. Quant à l'amélioration du projet, on pourrait changer la disposition des éléments du plateau, corriger des bugs, optimiser le code et créer de nouveaux comportements de gameplay.

Diogo Lopes da Silva, 28.07.2022



Annexes

- Affiche : 22INF-TB228_Flipp3r2_DiogoLopesDaSilva-Affiche.pptx
- Cahier des charges - Diogo Lopes da Silva : 22ISC-TB228_CC_Flipp3r2_DiogoLopesDaSilva.pdf
- Cahier des charges - Bruno Costa : 22ISC-TB202_CC_Flipp3r1_BrunoCosta.pdf
- Aides pour les bénévoles : Aide pour les responsables du Flipp3r.docx
- Autres éléments : 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
- Réponses au questionnaire google forms : Questionnaire - Flipp3r_reponses.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	Premier prototype du Flipp3r	1
1.2	Consortium du projet Flipp3r	3
4.1	Première version du plateau décidée après plusieurs heures de réunions	11
4.2	Première version du boss dans un état de standby	12
4.3	Première version du boss dans un état de taunt	12
4.4	Première version du boss dans un état de destruction avancé	13
4.5	Matrice de projection utilisée dans Unity	15
4.6	Image rapproché d'un flipper vert et de l'interaction de la lumière avec un élément du décors	16
4.7	Image du plateau de jeu dans le noir avec uniquement les lumières des objets . .	17
4.8	Boss du jeu en attente d'une activité d'un joueur	18
4.9	Boss du jeu dans sa première phase	18
4.10	Boss du jeu dans sa deuxième phase	19
4.11	GameOver du jeu quand il n'y a plus de balles qui peut apparaître	20
5.1	Apperçu du plateau de jeu final	23
5.2	Apperçu du plateau de jeu final avec lumières pour voir les détails	24

Bibliographie

- [1] K. Kratochvíl, "Hra pinball s využitím knihovny box2d." [Online]. Available : https://is.muni.cz/th/422696/fi_b/?zoomy_is=1
- [2] B. Hohlmann, M. Möller, and L. Scholz, "Developing a pinball game results of a practical course at the chair for computer graphics and multimedia RWTH aachen university, germany)," p. 2. [Online]. Available : https://www.graphics.rwth-aachen.de/media/projects/swp-ss12-c_report.pdf
- [3] M. Labschutz and K. Krosl, "Content creation for a 3d game with maya and unity 3d," p. 8. [Online]. Available : <https://old.cescg.org/CESCG-2011/papers/VUT-Labschuetz-Matthias.pdf>
- [4] A. Tăbușcă, C. Coculescu, and M. Pirnau, "General considerations regarding the development of games using unity technology," vol. 15, no. 2, pp. 267–283, num Pages : 267-283 Place : Bucharest, Romania Publisher : Romanian-American University, Scientific Research Department. [Online]. Available : <https://www.proquest.com/docview/2628335961/abstract/FAA924211C3545D8PQ/1>
- [5] J. Hasu, "Fundamentals of shaders with modern game engines," p. 100. [Online]. Available : <https://lutpub.lut.fi/bitstream/handle/10024/158721/FundamentalsOfShadersWithModernGameEnginesJoonasHasu.pdf?sequence=1&isAllowed=y>
- [6] A. Šmíd, "Comparison of unity and unreal engine," p. 77. [Online]. Available : <https://core.ac.uk/download/pdf/84832291.pdf>
- [7] J. Sundkvist, "AN EVALUATION OF REAL-TIME GLOBAL ILLUMINATION TECHNIQUES," p. 22. [Online]. Available : <https://www.diva-portal.org/smash/get/diva2:1351894/FULLTEXT01.pdf>
- [8] D. Wong, D. Earl, F. Zyda, R. Zink, S. Koenig, A. Pan, S. Shlosberg, J. Singh, and N. Sturtevant, "Implementing games on pinball machines," in *Proceedings of the Fifth International Conference on the Foundations of Digital Games - FDG '10*. ACM Press, pp. 240–247. [Online]. Available : <http://portal.acm.org/citation.cfm?doid=1822348.1822380>
- [9] A. Metcalf, "Pinball : High-speed real-time tracking and playing," p. 64. [Online]. Available : https://era.library.ualberta.ca/items/f871a51f-bf93-4d34-8aa5-817049cd97b5/view/604013d9-51ad-4a52-91bc-2760d7e92e7a/Metcalf_Adam_Fall-202011.pdf
- [10] Z. Zhang, "A flexible new technique for camera calibration," vol. 22, no. 11, pp. 1330–1334, conference Name : IEEE Transactions on Pattern Analysis and Machine Intelligence. [Online]. Available : <https://ieeexplore.ieee.org/document/888718?arnumber=888718>
- [11] A. De la Escalera and J. M. Armingol, "Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration," vol. 10, no. 3, pp. 2027–2044, number : 3 Publisher : Molecular Diversity Preservation International. [Online]. Available : <https://www.mdpi.com/1424-8220/10/3/2027>

- [12] R. Fabio and F. Clive, “Digital camera calibration methods : Considerations and comparisons,” medium : application/pdf Publisher : ETH Zurich. [Online]. Available : <http://hdl.handle.net/20.500.11850/158067>
- [13] S. L. A. d. Silva, A. M. G. Tommaselli, and A. O. Artero, “UTILIZAÇÃO DE ALVOS CODIFICADOS DO TIPO ARUCO NA AUTOMAÇÃO DO PROCESSO DE CALIBRAÇÃO DE CÂMARAS,” vol. 20, pp. 626–646, publisher : Universidade Federal do Paraná. [Online]. Available : <http://www.scielo.br/j/bcg/a/px7XCRGrHLKYwWvkR8ZTXz/?lang=pt>