Purest Atom Exam

Rapport de projet

OGAML (P1A) :

Guillaume CARRIÈRE

Marwan DAHOU

Louis GASNAULT

Amayun HOUERY

Sous la direction de M. Rémi VERNAY

Table des matières

1 Remerciements

Nous souhaiterions remercier toutes les personnes qui nous ont aidés lors de la réalisation de ce projet.

En premier lieu, nous remercions Monsieur Rémi Vernay, directeur d'EPITA Toulouse, pour avoir supervisé le projet et nous avoir aidés à améliorer nos présentations orales.

Nous remercions également Monsieur Salah Kabous, intervenant en Techniques d'Expression à EPITA, pour ses conseils prodigués tout au long de l'année pour que nous améliorions nos exposés.

Nous remercions aussi Asbjørn Thirslund, le vidéaste de la chaîne Brackeys dont les tutoriels très bien réalisés nous ont beaucoup aidés pour appréhender bon nombre des fonctionnalités d'Unity.

Nous finirons en remerciant nos familles respectives pour nous avoir supportés et soutenus pendant toute la durée de ce projet.

2 Introduction

2.1 Présentation générale

Purest Atom Exam est un jeu de combat et d'action en multijoueur local. Le jeu se déroule en manches d'environ 3 à 5 minutes, dans lesquelles plusieurs joueurs s'affrontent entre eux dans une arène en deux dimensions possédant plateformes et/ou obstacles, le tout dans des graphismes pixel art pour coller à une ambiance arcade et nerveuse tout en restant lisible.
La particularité de ce jeu est le système de combat dans lequel le seul moyen d'infliger des dégâts est de projeter des objets sur ses ennemis à l'aide d'un simili force de gravité permettant de faire orbiter les objets autour de son combattant et de les accélérer pour enfin les relâcher au bon moment, à l'image d'une fronde antique.
Cependant, tous les objets n'auront pas le même comportement, le même poids ou la même taille. Certains objets seront bien plus puissants que d'autres et attireront les convoitises.
Enfin, il est possible de jouer sur quatre cartes différentes possédant des caractéristiques et une topologie uniques, forçant les joueurs à adapter leur style de jeu pour remporter la victoire.

2.2 Histoire

En 2248, l'humanité s'est développée à une vitesse exponentielle jusqu'à coloniser les autres planètes du système solaire en partie grâce à la multispatiale Spazon. Cette entreprise avait commencé par développer un moteur de recherche, il y a de ça plusieurs décennies. En diversifiant ses activités, elle a progressivement dominé toutes ses concurrentes pour finalement devenir une des grandes puissances du monde humain. Au fil des années, Spazon a permis de nombreuses avancées scientifiques, allant même jusqu'à créer un nouveau type de soldat d'élite : les THEUS. Ces clones génétiquement modifiés, capables de maîtriser la télékinésie, ont rapidement servi à asseoir le pouvoir de Spazon sur toutes les colonies humaines.
Mais un groupe de rebelles marginaux refusant la domination de Spazon grandit de

jour en jour sur les astéroïdes troyens de Jupiter. Cette colonie très pauvre, regroupant les rebus de la société « ordonnée » de Spazon, est prête à tout pour conserver sa liberté face aux tentatives répétées de Spazon pour la « normaliser ».
Afin de rivaliser avec les THEUS, les meilleurs miliciens troyens se sont alors dotés d'implants leur octroyant les mêmes capacités. Au cours des affrontements qui suivirent, certains THEUS tombèrent entre les mains des Troyens et furent faits prisonniers.
D'abord esclaves de Spazon et maintenant retenus par un peuple qui les hait, les THEUS n'ont plus qu'un rêve : obtenir leur liberté. Mais ils ne sont pas encore au bout de leur peine. Certains Troyens peu scrupuleux commencent à organiser des combats entre les THEUS pour en tirer profit, leur promettant la liberté s'ils sortent vainqueurs. Les membres de cette presque famille vont-ils supporter longtemps de s'entretuer ainsi ? La réponse se trouve dans l'arène.

3 Présentations des membres

3.1 Guillaume Carrière

Je ne me suis intéressé à l'informatique réellement qu'à partir de la classe de première, au lycée, mais j'y ai trouvé une passion puis un réel plaisir à achever des projets primitifs sur mon temps libre. C'est donc naturellement que je me suis orienté vers la spécialité ISN une fois arrivé en Terminale, ce qui m'a apporté un peu plus de connaissances sur la programmation. Cela m'a permis de rendre mes créations un peu plus complexes et intéressantes, renforçant ainsi mon goût pour l'informatique.
C'est pour ces raisons que je suis désormais à l'EPITA dans l'espoir de pouvoir achever des projets plus ambitieux sur lesquels je pourrai exploiter le reste de mes passions telles que la musique ou les jeux vidéo, que je pratique depuis bien plus longtemps que la programmation.
Partageant cet espoir avec les membres du groupe OGaml, j'avais donc la certitude de pouvoir apprendre beaucoup de ce projet, qui aura été le plus concret sur lequel j'ai pu travailler jusqu'à maintenant.

Classe : DPS

3.2 Marwan Dahou

J'ai fait une terminale S-SVT et, bien qu'étant déjà intéressé par l'ingénierie, l'informatique et la conception vidéoludique, cette spécialité s'en éloignant, elle m'a apporté plus de rigueur et a affiné mes attentes. Suite à cela, bien que retenu dans plusieurs bonnes écoles d'art en spécialité "game design", une voie plus scientifique m'a bien plus intéressé afin de comprendre le monde avant d'en créer un. L'aspect relationnel, managérial et "travail d'équipe" d'une formation d'ingénieur sont aussi au cœur de mes attentes. Cela combiné avec le vécu d'un ami présent depuis un an à Epitech m'a donc orienté vers EPITA.

Un tel projet est alors pour moi un moyen parfait de me tourner vers ce qui me motive le plus dans la voie que j'ai choisie et d'acquérir enfin des bases et une expérience concrète du domaine qui m'est le plus attrayant.
C'est donc une expérience importante qui, je l'espérais, allais contenter ou renforcer ma vision du travail en équipe et la gestion d'un projet ou alors les remettre en question visant à ensuite mieux faire.

Classe : Marabout

3.3 Louis Gasnault

J'ai commencé la programmation en 3ème sur Python, je suivais alors un cours sur le site OpenClassrooms, qui m'a permis de comprendre certaines bases. J'ai rapidement essayé d'appliquer ce que j'avais appris en créant de petits programmes. Bien qu'ils ne soient pas très complexes, ces premiers "projets" m'ont familiarisé avec le débuggage d'un programme et son amélioration, le tout, seul ou avec mon père. Arrivé au lycée j'ai rencontré une personne qui partageait ce qui était devenu ma passion : la programmation. Nous nous sommes donc associés pour créer des projets d'une plus grande ampleur, en nous organisant comme un vrai studio de développement (chef de projet, game designer, développeurs, graphistes), mais avec les cours en parallèle, nous n'avons jamais réussi à finir entièrement un projet.
J'ai donc choisi une terminale S spécialité ISN pour pouvoir passer plus de temps sur mes projets. Pendant les vacances d'été 2018, j'ai également participé à une "game jam" (compétition de création de jeu) de 48h.
Comme je me suis déjà investi dans la réalisation de jeux dans plusieurs domaines : programmation, web, graphisme et game design, j'espérais pouvoir mettre en œuvre ce que j'ai appris ces dernières années à chaque étape de ce projet et dans chaque tâche.

Classe : Healer

3.4 Amayun Houery

Amateur de jeux vidéo depuis tout jeune, l'informatique m'intéresse depuis maintenant la 3ème, résultant d'une curiosité d'en apprendre plus sur la création de ces œuvres. Similairement aux autres membres du groupe, j'ai suivi des tutoriels sur internet, j'ai appris en bidouillant chez moi et en faisant des recherches. J'ai ainsi appris les bases de quelques langages comme le C++ et le Javascript qui m'ont entre autres permis de comprendre bien assez tôt la notion de POO. Jusqu'ici, la programmation était un loisir fait principalement de découvertes et de pratiques dans lesquelles je me suis bien amusé. J'ai aussi fait des recherches sur l'art du game design, toujours dans la curiosité d'en apprendre sur le jeu vidéo et sa création.
En terminale je me suis aussi tourné vers la spécialité ISN qui m'a permis de m'essayer à l'exercice ardu du projet en groupe. La tâche est compliquée mais apporte beaucoup, c'est pourquoi j'ai compté sur ce projet pour aller plus loin dans la conception d'un jeu vidéo et vivre des expériences à partir desquelles je tirerai des leçons utiles pour le domaine professionnel.

Classe : Tank

4 Rappel de nos objectifs

Dans cette partie seront présentés nos objectifs initiaux pour chaque partie au début du projet.

4.1 Graphismes

Nous voulions opter pour le pixel art. Premièrement car c'est un style beaucoup utilisé dans nos inspirations, mais aussi car il peut être rapide à réaliser tout en ayant un résultat intéressant, et que nous en avons tous déjà fait dans le groupe. L'ambiance du jeu devait être très inspirée du genre cyberpunk, qui met en scène un futur proche, souvent dystopique, avec une société technologiquement avancée. De plus, le thème de l'humain augmenté est souvent traité dans ce genre, ce qui correspond à l'univers du jeu.

4.2 Musique, bruitages et menus

Puisque l'ambiance du jeu est du style *cyberpunk*, les musiques créées devaient être du style *synthwave* et, pour renforcer le côté arcade, les bruitages auraient eu un cachet "synthétiseur/rétro".
Un menu simple devait être intégré afin de choisir entre les modes de jeu, les cartes, les personnages et les options.

4.3 Mécaniques du jeu

Purest Atom Exam devait avant tout être un jeu de combat. Il devait se jouer en multijoueur par parties de 5 minutes environ, divisées en manches dans lesquelles les différents personnages du jeu se combattent dans des arènes en deux dimensions dans le style de Towerfall Ascension. L'objectif devait être de faire tomber les points de vie de l'ennemi à 0 et le seul moyen de causer des dégâts à ses adversaires devait être de les faire rentrer en collision avec les différents objets du jeu mis à disposition. Pour les manipuler, les joueurs devaient utiliser la faculté de télékinésie des T.H.E.U.S. Cette capacité devait fonctionner de cette manière : une fois activée une onde se

propage autour du personnage dans un diamètre limité puis disparaît. Tous les objets présents dans cette "aura" commencent à être attirés vers le personnage avec une vitesse modulable par le joueur. L'objectif aurait donc été ensuite de jouer avec cette force d'attraction appliquée par notre personnage pour faire rentrer les objets en orbite autour de ce dernier afin de continuer à les faire s'accélérer et gagner en vitesse sans qu'ils le heurtent, puis ensuite de désactiver la force d'attraction au bon moment pour projeter le(s) objet(s) vers notre cible. Toutes les mécaniques de jeu devaient tourner autour de ce système sur lequel se grefferaient quelques capacités spéciales supplémentaires et différents types d'objets et arènes aux comportements variés.

4.4 Multijoueur

Le jeu avait pour objectif d'être en multijoueur local sur la même machine, avec un objectif d'ambiances "couch gaming". Les contrôles devaient donc être optimisés pour manettes (avec la possibilité d'utiliser si possible quasiment tous types de manettes, Xbox 360, ONE, Dualshock) mais il semblait aussi essentiel d'implémenter des contrôles au clavier, tout d'abord car l'accès à une manette compatible avec l'ordinateur n'est jamais garanti, mais aussi pour permettre plus facilement les sessions de jeu multijoueur local (une seule manette était alors suffisante pour jouer au moins à deux). Le multijoueur en ligne n'était cependant pas envisagé, puisqu'il ne correspondait pas à notre type de jeu.
Parallèlement à cela, nous voulions créer une courbe de progression du joueur du type "easy to learn, hard to master", c'est-à-dire des mécaniques de jeu faciles à prendre en main au début mais difficiles à maîtriser complètement, pour potentiellement donner lieu à des matchs techniques, car c'est un modèle très fréquemment utilisé dans les jeux multijoueurs de combat du style "couch gaming" (en témoignent les licences Super Smash Bros ou Street Fighter et leur scène eSport).

4.5 Autres éléments du gameplay

Cartes : Les cartes devaient être des arènes fermées possédant des plates-formes et obstacles multiples. Plusieurs arènes devaient ainsi être jouables, certaines avec de la gravité, d'autres sans et d'autres encore avec une gravité changeante (à certains moments seulement, et dans différentes directions). Dans chaque arène les objets devaient être lâchés des murs par de nombreux distributeurs de façon aléatoire. Il était aussi envisageable de faire des arènes possédant des issues aux extrêmes du terrain donnant sur l'autre extrême (exemple : on prend une issue en bas qui nous fait arriver à une autre issue en haut).

Objets : Voici quelques exemples d'objets que nous voulions implémenter :

— Caisse : se brise à l'impact, dégâts moyens.

— Mine : explose au bout d'un certain temps après être rentrée en interaction avec un joueur, dégâts liés à l'impact léger, mais dégâts d'explosion importants.

— Boulet : Lourd boulet, difficile à accélérer, fait d'importants dégâts à l'impact, se brise au bout de 3 impacts.

— Débris : léger, se brise au bout de deux coups, facile à accélérer, faibles dégâts à l'impact.

— Flèche : se brise en un coup (ou se plante si elle touche sa cible), dégâts importants si la tête de la flèche touche, minimes sinon, facile à accélérer.

Capacités : Les personnages devaient aussi pouvoir, outre leur force de gravité, utiliser d'autres capacités. En voici quelques exemples que nous voulions implémenter :

— Esquive/sprint : Une capacité pouvant être utilisée fréquemment et permettant d'augmenter très temporairement la vitesse de son personnage.

— Rappel : Une capacité pouvant être utilisée rarement permettant de rappeler des objets projetés par son personnage mais n'ayant pas encore touché quelque chose vers ce dernier avec une grande force, à l'image d'un boomerang.

Nous avons donc synthétisé nos objectifs en différents domaines précis, puis nous nous sommes réparti les responsabilités selon le tableau suivant :

	Marwan	**Guillaume**	**Amayun**	**Louis**
Physique	⊕	+		
Contrôles / Multijoueur		⊕	+	
Graphismes	+			⊕
Site Web	+			⊕
Interfaces / Menus			⊕	+
Game Design		+	⊕	
Level Design	⊕			+
Audio FX		⊕	+	

Légende :

— ⊕ : Responsable

— + : Suppléant

De plus, nos objectifs d'avancement en fonction de chaque domaine pour chaque soutenance correspondaient à cela :

	Soutenances		
	1	**2**	**3**
Physique	30%	60%	100%
Contrôles / Multijoueur	60%	90%	100%
Graphismes	5%	50%	100%
Site web	75%	90%	100%
Interfaces / Menus	20%	50%	100%
Game Design	15%	50%	100%
Level Design	25%	60%	100%
Audio / FX	30%	50%	100%

5 État des lieux

5.1 État des lieux général

Dans sa globalité, le projet comporte un écran-titre et un menu principal avec trois sous-menus ainsi que plusieurs personnages utilisables en jeu sur quatre cartes différentes sur lesquelles pourront être manipulés 7 types d'objets à l'aide des systèmes physiques implémentés. Le jeu possède également un système de manches avec un écran de victoire et un menu de pause. Enfin il est jouable de 2 à 4 personnes, avec des manettes et/ou un clavier. Les pages suivantes vont servir à présenter un état des lieux du projet plus précis, domaine par domaine.

5.2 Physique

La physique dans ce projet comprend toutes les interactions entre les personnages, les projectiles et l'environnement dans lequel ils évoluent. Ces interactions reposent sur un pouvoir d'attraction activable et désactivable, détenu par les différents joueurs. Ce système d'attraction a pour but de permettre aux joueurs d'approximer des orbites avec les projectiles proches et de leur faire prendre de la vitesse. La physique gère donc les trajectoires des objets, leur vitesse et les collisions en fonction des différents modes d'attraction utilisés et les obstacles présents sur le terrain de jeu. Il y a donc une gestion de la physique globale tenant compte des collisions, une gestion de chaque joueur et leurs pouvoirs et une gestion de chaque projectile et les caractéristiques qui lui sont propres. Plus concrètement un joueur possède trois modes d'attraction. Tout d'abord, une attraction basique qui attire les objets suffisamment proches vers le joueur. Ensuite, une attraction plus forte qui donne beaucoup de vitesse aux projectiles. Enfin, une attraction également forte mais couplée à un cercle de collision invisible autour du joueur qui force les projectiles à prendre une orbite circulaire même à grande vitesse sans frapper le joueur. Cependant, les projectiles ennemis peuvent toujours atteindre le joueur sans autre obstacle que les objets orbitant autour de lui. De plus, les collisions et les dégâts induits sont dépendants de la vitesse des projectiles sans dépendre de la vitesse des joueurs et des

particules de vitesse apparaissent lorsqu'un projectile est assez véloce pour infliger des dégâts.

5.3 Contrôles / Multijoueur

Le multijoueur dans notre projet est local et non pas en ligne, ce qui veut dire que nous n'utilisons pas de connexion externe pour que deux ordinateurs puissent jouer au même jeu. Bien que ce choix nous ait permis de ne pas avoir à implémenter un système de communication entre les différentes machines, il a cependant soulevé d'autres problèmes, notamment il a demandé l'implémentation d'un système de contrôle permettant le contrôle de nos personnages par plusieurs manettes connectées à un seul ordinateur.
Ainsi, le jeu se joue donc de 2 à 4 personnes, en utilisant jusqu'à 4 manettes en plus du clavier.
Tout d'abord, la navigation des menus (première chose à laquelle on est confronté dans le jeu) se fait au clavier ou avec n'importe quelle manette connectée à l'ordinateur, c'est ensuite dans le menu de sélection de personnage que seront créés les paramètres des joueurs pour chaque manette se connectant, correspondant à la texture choisie par l'utilisateur et l'identifiant contrôleur qui lui est connecté. Enfin, en jeu, chaque personnage existant est contrôlé par la manette qui l'a "créé" dans le menu de sélections des personnages.
Pour ce qui est des contrôles en soi, le *stick* gauche de la manette est utilisé pour le déplacement des joueurs, les quatre boutons et la gâchette droite servent à la manipulation d'objets, (leur accélération , les différents modes d'attraction etc.) et la gâchette gauche est utilisée pour l'esquive. De cette façon, tous les boutons liés au déplacement sont sur la partie gauche de la manette, et tous les boutons liés aux objets sont sur la partie droite, pour rendre les contrôles plus instinctifs. Notamment la gâchette droite sert à baisser l'attraction avec les objets sans annuler leur attraction, afin qu'elle soit utilisée en jeu pour lancer un objet, ce qui correspond aux normes des jeux vidéo dans lesquels la gâchette droite et souvent utilisée pour attaquer (dans les *First-Person Shooter* par exemple). Pour le clavier, ce sont les touches z, q, s, d, espace, gauche, shift et les flèches directionnelles qui sont utilisées.

Afin de faciliter la compréhension des touches, nous avons créé en manuel en jeu expliquant les contrôles (Fig. 1).

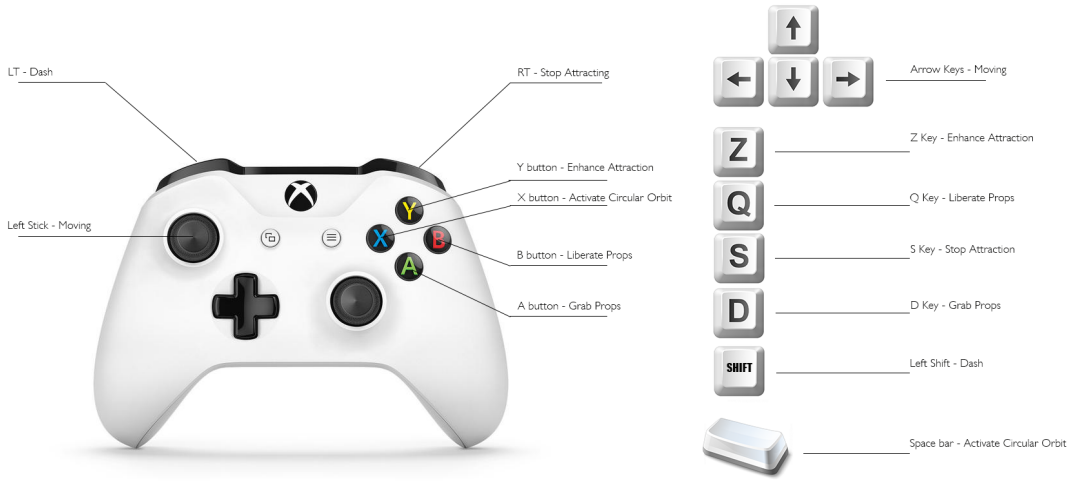

FIGURE 1 – Tutoriel sur les contrôles visibles en jeu

5.4 Graphismes

Dès le début du projet, nous tenions à ce qu'un maximum des images utilisées dans le jeu soient réalisées par nous-mêmes. Nous avons donc mené un projet de recherche pour aboutir à une charte graphique, que nous nous sommes efforcés de suivre lors de la réalisation des différents graphismes du jeu. Pour rappel, nous avions décidé de travailler en *pixel art* car c'est un style que l'on retrouvait parmi toutes nos références, qu'il retransmet bien l'aspect "rétro" que nous voulions pour notre jeu et que nous le pensions rapide à mettre en place. Nous nous sommes également accordés sur l'aspect de certains éléments, comme la présence d'un *CyberPunk* à crête ou d'un objet explosif un peu "cartoonesque" dans sa représentation (ce qui deviendra la bombe).

Ainsi pour chaque élément du jeu nous avons réalisé une image, voire plusieurs dans le cas des objets nécessitant une animation, ou ayant la possibilité de changer d'état.

5.4.1 Les personnages

Nous avons créé plusieurs personnages (CyberPunks, THEUS et Spazon) ayant chacun un style particulier pour les différencier les uns des autres, mais également plusieurs poses pour signifier l'activation d'un pouvoir (l'attraction, l'orbite circulaire, etc ...). Ces personnages sont issus de l'histoire que nous avons mise en place pour le jeu, ils représentent les trois factions qui s'y affrontent.
Pour aider le joueur à repérer son personnage et les objets qui sont attirés par lui, nous avons mis en place un système de contour. Chaque joueur se voit attribuer une couleur en début de partie (nous avons repris les couleurs utilisées sur le logo du groupe, le jeu se jouant jusqu'à quatre joueurs), et à chaque fois qu'un objet rentre dans son orbite, il sera lui aussi entouré de la même couleur (Fig. 2).

FIGURE 2 – Le contour du joueur et des objets.

5.4.2 Les cartes

Chaque carte du jeu est accompagnée par son lot d'obstacles et un fond servant de décor à l'affrontement des joueurs. Nous avons essayé pour chacune d'elles de créer une ambiance particulière pour diversifier l'expérience de jeu. Ces différences permettant également d'incorporer de manière logique de nouveaux éléments, comme le dispositif de sécurité de la carte "Prison" (les *bumpers*), ou les lianes de la carte "Jungle".

5.4.3 Les objets

Plusieurs objets sont disponibles sur le champ de bataille. Pour aider le joueur à déterminer leur dangerosité, nous avons joué sur les matériaux des éléments (et donc leurs couleurs). La caisse, étant l'objet le plus fragile, est en bois avec des couleurs pâles. Le caillou, petit et rapide, est légèrement bleuté, pour rappeler le fait que le jeu prend place sur une planète extraterrestre. Le boulet est l'objet qui inflige le plus de dégâts après la bombe, mais il est lourd et lent à déplacer, il est donc représenté plus gros que les autres objets, dans des tons très sombres. Certains objets infligent des dégâts pendant une certaine durée aux joueurs, nous leur avons donc donné une représentation particulière, comme s'ils étaient des objets un peu mystérieux et/ou futuristes. C'est le cas de l'objet empoisonnant le joueur et de la boule de feu (Fig. 3).

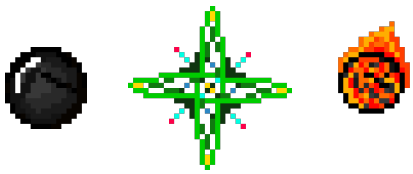

FIGURE 3 – Le boulet, le poison et la boule de feu.

5.4.4 Les animations

Pour pouvoir bien différencier la bombe du boulet, nous n'avons pas fait la grosse bombe noire que l'on peut voir dans certains dessins animés. Nous sommes partis sur une forme plutôt atypique de crâne et nous lui avons fait arborer un sourire inquiétant. Ensuite, nous avons créé une animation découpée en six images où l'on voit la bombe se déformer et rougir pour montrer qu'elle s'apprête à exploser. On passe ensuite à l'animation d'explosion, qui est composée de seulement trois images mais qui doublent de volume pour montrer la zone affectée par l'explosion. La flamme surmontant la boule de feu a également été animée (Fig. 4).

FIGURE 4 – Les images composant l'animation de la bombe.

5.4.5 Les systèmes de particules

Certains éléments du jeu ont nécessité l'implémentation de systèmes de particules. Les objets, lorsqu'ils vont suffisamment vite pour infliger des dégâts aux joueurs, émettent des particules linéaires dans le sens inverse à leur mouvement, pour rappeler les lignes de vitesse typiques des bandes dessinées. Certains objets peuvent empoisonner ou mettre en feu les joueurs, qui émettent alors des particules pour indiquer qu'ils sont en train de prendre des dégâts. Ces dernières sont vertes et circulaires lorsque le joueur est empoisonné, rouges et triangulaires lorsqu'il est en feu (Fig. 5).

FIGURE 5 – Les particules de poison.

5.4.6 Les menus

— **L'écran titre et le menu principal** : Pour que, dès la séquence d'introduction, le joueur soit plongé dans une ambiance "rétro", nous avons ajouté une vidéo de VHS en train d'être lue. L'écran titre, quant à lui, montre au joueur que l'aventure aura lieu dans l'espace. Une fois ces deux étapes passées, le joueur arrive sur le menu principal. On change alors d'ambiance pour se rapprocher du *synthwave*, qui est notre seconde source d'inspiration. Néons violets et métal sont de mise pour retranscrire ce genre.

— **Le menu pause et le menu de fin de partie** : Dans une approche totalement différente, le menu pause est très épuré, avec un fond quasiment transparent pour que les joueurs puissent toujours voir l'action derrière. Idem pour l'écran qui s'affiche à la fin d'une partie, le but étant d'éviter au maximum de sortir le joueur de l'ambiance du jeu.

5.4.7 L'affichage en jeu

En plus des retours visuels sur les personnages et les objets que nous avons pu voir, nous avons dessiné quelques interfaces pour donner des informations aux joueurs. La barre de vie, par exemple, va prendre la même couleur que le contour du personnage, pour que le joueur l'identifie facilement, elle est également accompagnée d'une barre plus petite mais à la couleur vive pour être plus visible, et qui indique au joueur qu'il peut utiliser son esquive. Nous avons également ajouté quelques éléments graphiques simples pour guider le joueur dans les menus du jeu, en lui indiquant les actions qu'il peut effectuer.

Au total, ce sont soixante-dix images qui ont été produites pour le jeu, auxquelles s'ajoutent les deux logos du groupe et les illustrations du site web.

5.5 Site Web

Le site web est là pour servir de vitrine au projet, c'est là que les potentiels joueurs vont se rendre pour en apprendre plus sur le jeu, et peut-être le télécharger. C'est en nous basant sur l'aspect *synthwave* de notre projet que nous avons trouvé le style du site web, en particulier au niveau des couleurs : orange, rouge et violet. Le violet, en tant que couleur sombre, servira pour le fond, alors que l'orange et le rouge seront là pour accentuer les éléments importants du site.
Après avoir trouvé le style général, nous avons organisé le site web pour pouvoir présenter tous les éléments que nous voulions. Nous nous sommes beaucoup inspirés de deux sites web différents dans leur style pour créer le nôtre. Dans un premier lieu, celui du jeu *Hytale* dont le style riche retranscrit parfaitement l'ambiance du jeu un peu médiévale et romanesque, nous nous sommes donc inspirés de ses éléments marquants comme la barre de navigation ou les en-têtes des parties pour notre site. Mais nous voulions aussi garder un style simple et peu chargé, nous avons donc analysé le site d'*Alto's Odysseys* qui ne présente qu'une seule page, très épurée, mais qui arbore toutes les informations du jeu. L'organisation que nous avons retenue est la suivante :

— Une page principale avec une description brève du jeu et qui redirige vers toutes les autres pages du site.
— Une page consacrée au jeu, présentant le concept, l'histoire et les mécaniques de notre projet, le tout accompagné par des captures d'écran du jeu ainsi que quelques extraits de la bande originale.
— La présentation du groupe OGAML et de chacun de ses membres, avec la répartition des tâches dans l'équipe.
— L'historique du projet qui montre toutes les étapes importantes du développement de Purest Atom Exam, telles que les dates des soutenances, les publications des versions jouables ou la mise en place de nouvelles fonctionnalités.
— On peut également trouver sur le site la liste des outils que nous avons utilisés tout au long de ce projet, comme par exemple Overleaf ou Sublime Text.
— Le site dispose également d'une page regroupant tous les documents produits

pour les différentes soutenances, c'est-à-dire les rapports, les plans et diaporamas des oraux ainsi que la version du jeu correspondant à la soutenance.

— La page la plus importante du site reste bien sûr celle permettant au visiteur d'installer le jeu de plusieurs façons et sur plusieurs plateformes. La dernière version est disponible sous forme d'installeur ou d'archive zip pour Windows (7 et plus), et également téléchargeable en archive pour Linux et Mac OS X. Toutes les anciennes versions du jeu peuvent également être téléchargées sur cette page (Fig. 6).

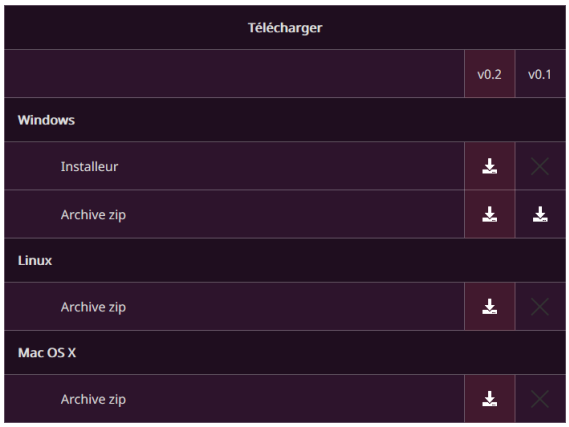

FIGURE 6 – Les différentes versions téléchargeables.

5.6 Interfaces / Menus

5.6.1 Menu principal

Tout d'abord, il a fallu confectionner un menu principal. Celui-ci offre les trois fonctionnalités de base d'un menu de jeu vidéo classique : lancer, régler et quitter. Ce menu se présente sous une forme très basique : quatre boutons alignés à l'horizontale sur un fond simple avec une petite musique qui tourne en boucle. Les quatre boutons sont, de gauche à droite : jouer, réglages, aide et quitter. Avant de lancer la partie, il faut passer par deux sous-menus. Vu que le jeu se joue en multijoueur local, le premier sous-menu est indispensable : il s'agit de celui dans lequel les joueurs se "connectent" au jeu avec leur manette ou autre outil de contrôle. Ils peuvent y choisir leur personnage et ensuite, après avoir validé, passer dans le second sous-

menu où ils choisissent le niveau dans lequel se déroulera la partie. À partir de là, la partie peut être lancée.

5.6.2 Menu pause et de fin

Ensuite, comme dans la plupart des jeux vidéo, il doit être possible de mettre le jeu en pause au milieu d'une partie. Le "menu pause" permet de revenir au menu principal ou de directement quitter le jeu. Une variante de ce menu est utilisée à la fin d'une partie pour indiquer le joueur gagnant (ou une éventuelle égalité) et proposer de rejouer ou retourner au menu principal.

5.6.3 Affichage tête haute

L'affichage tête haute (ATH ou HUD en anglais pour *Head-Up Display*) constitue les éléments graphiques qui viennent se superposer à la scène de jeu pour donner des informations en temps réel. Il y a d'abord un élément principal qui indique le statut de la partie. Ensuite, il y a un élément par joueur qui indique :

— sa santé au moyen d'une barre de vie
— la disponibilité du pouvoir d'esquive avec une icône qui s'affiche lorsqu'il est complètement rechargé et disponible à l'utilisation ainsi que l'état du rechargement de ce pouvoir
— le nombre de rounds gagnés par le joueur

Les éléments sont situés à chaque coin de l'écran et leur nombre est limité à quatre (comme les joueurs). La couleur de la barre de vie correspond à celle du joueur, qui elle, est unique.

5.6.4 Lancement du jeu

Lorsque le jeu se lance, il y a au début une petite animation qui se lance imitant l'introduction d'une cassette vidéo dans un lecteur VHS. Passé cette animation, il y a le traditionnel écran titre qui invite l'utilisateur à appuyer sur une touche pour commencer et, une fois la tâche accomplie, le ramène sur le menu principal.

5.7 Mécaniques de jeu

La majorité des mécaniques du jeu tournent autour du système d'attraction que nous avons mis en place ainsi que des différents objets. Les personnages contrôlés par les joueurs sont en lévitation dans une "arène" et peuvent se déplacer de gauche à droite et de bas en haut. Les joueurs ont plusieurs pouvoirs à leur disposition, le principal étant d'attirer les objets qui se trouvent dans l'arène. Le but du jeu est d'utiliser ces objets pour les lancer contre ses adversaires et leur faire perdre de la vie jusqu'à ce qu'ils n'en aient plus.

5.7.1 Les pouvoirs

Pour remplir ce but, les joueurs ont à leur disposition quelques pouvoirs simples pour lancer les objets et/ou les esquiver.

— **Attrape des objets** : Tout d'abord, pour pouvoir appliquer les pouvoirs d'attraction, les joueurs doivent "attraper" les objets. Cela signifie que le joueur est associé à l'objet et que ses pouvoirs s'appliquent dessus. Lorsque le joueur active ce pouvoir, il attrape tous les objets proches dans un petit périmètre restreint. Pour savoir quel objet est associé à quel joueur, les objets attrapés brillent d'une petite aura de la même couleur que le joueur auquel ils sont associés.

— **Attraction passive** : Tous les objets attrapés sont légèrement attirés par le joueur et se rapprochent doucement vers lui quelle que soit la distance qui les sépare. Cette attraction est présente en permanence sauf quand le joueur appuie sur la gâchette de droite pour l'annuler et "relâcher" les objets.

— **Attraction renforcée** : Ce pouvoir, lorsqu'il est activé, augmente fortement l'attraction sur tous les objets attrapés et leur donne une accélération considérable et ce depuis n'importe quel endroit de l'arène.

— **Orbite circulaire rapprochée** : Celui-ci attire aussi les objets attrapés vers le joueur mais seulement ceux qui sont dans un rayon proche. De plus, ce pouvoir active une zone de collision qui vient protéger le joueur de recevoir ses propres objets. Cette sorte de bouclier ne l'empêche pas de recevoir les objets des autres mais elle permet de donner une accélération aux objets en

les faisant tourner autour de soi à la manière d'une orbite circulaire.

— **Esquive** : Aussi appelé *dash* dans les jeux vidéo, l'esquive consiste à donner une forte mais courte impulsion au joueur dans la direction dans laquelle il se déplaçait. Elle lui permet de se déplacer très rapidement pour, par exemple, traverser l'arène ou éviter un objet qui lui serait destiné. Après l'avoir activé, ce pouvoir met un certain temps à se recharger avant de pouvoir être réutilisé.

5.7.2 Les objets

Les objets font partie intégrante du système de jeu puisqu'il s'agit des seuls éléments qui permettent d'infliger des dégâts aux joueurs. Ils apparaissent de manière aléatoire grâce à un élément situé au centre de chaque niveau et qui s'appelle le **prop spawner**. Tous les objets ont besoin d'une vitesse minimale pour provoquer les dégâts ou leur effet mais chaque objet possède sa petite spécificité :

— **La caisse** est l'objet le plus basique et le premier à avoir été imaginé. Elle fait plus de dégâts que le rocher et rebondit mieux que les autres projectiles. Par contre, elle se détruit au bout de 3 chocs.

— **Le rocher** est l'autre objet de base avec la caisse. Il inflige moins de dégâts mais est un peu plus lourd et surtout indestructible. Il porte le rôle d'objet le plus faible et le plus basique.

— **Le boulet** est l'autre objet indestructible du jeu. Il est plus lourd que le rocher mais aussi inflige plus de dégâts. Il n'y en a qu'un seul par partie. Il est donc relativement puissant, et attire les convoitises. Mais il reste néanmoins difficile à manipuler correctement dû à son poids.

— **La flèche**, quant à elle, ne peut être utilisée qu'une seule fois. Lorsqu'elle heurte avec suffisamment de vitesse un joueur, elle se plante dans celui-ci et lui inflige de lourds dégâts (l'équivalent de la moitié de la vie pleine du joueur, pour être précis). Elle est donc un objet très puissant.

— **La boule de poison** se comporte de manière un peu différente. Son effet quand elle touche un joueur n'est pas de lui infliger des dommages directs mais de l'empoisonner. Cet effet va enlever petit à petit de la vie au joueur pendant une courte durée avant de se dissiper. En soi, ses dégâts totaux ne

sont pas mauvais, mais le temps que met le poison à les causer rend l'objet moyennement efficace.

— **La boule de feu** fonctionne un peu de manière similaire à la boule de poison : elle met en feu le joueur contre lequel elle rentre en collision et un joueur en feu perd progressivement de la vie. La différence se trouve dans le fait que lorsqu'un joueur rentre en collision avec un autre joueur, ce dernier se retrouve en feu aussi. Il porte donc un rôle semblable à celui de la boule de poison, mais avec une couche de complexité supplémentaire.

— Le compte à rebours avant l'explosion de **la bombe** s'active à partir de la première fois qu'un joueur l'attrape. Avant que les précieuses secondes se soient écoulées, la bombe se comporte comme un rocher et inflige les mêmes dégâts à l'impact mais son explosion provoque la perte de l'équivalent des trois quarts de la santé pleine d'un joueur, ce qui en fait l'objet le plus destructeur et puissant du jeu.

5.7.3 Environnement

Il y a aussi quelques petites mécaniques propres aux niveaux qui interviennent principalement sur les déplacements des personnages dans l'environnement.

Tout d'abord, il y a tout simplement les blocs et les obstacles qui viennent gêner l'évolution des joueurs mais qui leur permettent aussi de se cacher derrière pour se protéger des projectiles par exemple.

Ensuite, on trouve dans le niveau "Prison" des rebondisseurs (*bumpers* en anglais) qui renvoient dans la direction opposée tout joueur qui aurait le malheur de se cogner contre, tout en l'étourdissant pendant quelques secondes, ce qui l'empêche de réaliser n'importe quelle action et le rend totalement vulnérable face à ses adversaires.

La dernière mécanique se trouve dans le niveau "Jungle". Ce sont les lianes qui séparent le niveau en deux. La particularité de ces lianes est qu'elles laissent passer les objets d'un côté à l'autre de l'arène mais bloquent les joueurs et les empêchent de passer. Cela crée une situation pouvant s'apparenter à une partie de volley-ball ou autres sports possédant un filet au milieu.

5.8 Construction des niveaux

Les niveaux peuvent être mis dans deux catégories distinctes, les niveaux avec et sans éléments de décor particuliers. En effet, les niveaux ont été pensés pour offrir une expérience de jeu variée demandant pour chacun un usage différent des mécaniques de jeu offertes par les projectiles et la physique.
Parmi les quatre cartes proposées nous avons d'une part deux cartes simples. Ces cartes sont les niveaux nommés "Arena" et "Space".
La première est une version finale du prototype ayant servi au développement du jeu. Elle est simple et ne comporte que quelques obstacles basiques comme des murs et des petites plateformes. Son ambiance visuelle garde la même simplicité avec un fond assez sombre et peu détaillé qui concentre l'attention sur les joueurs, les projectiles et les obstacles. Les obstacles ont quant à eux une texture de brique basique cohérente pour une carte par défaut (Fig. 7).

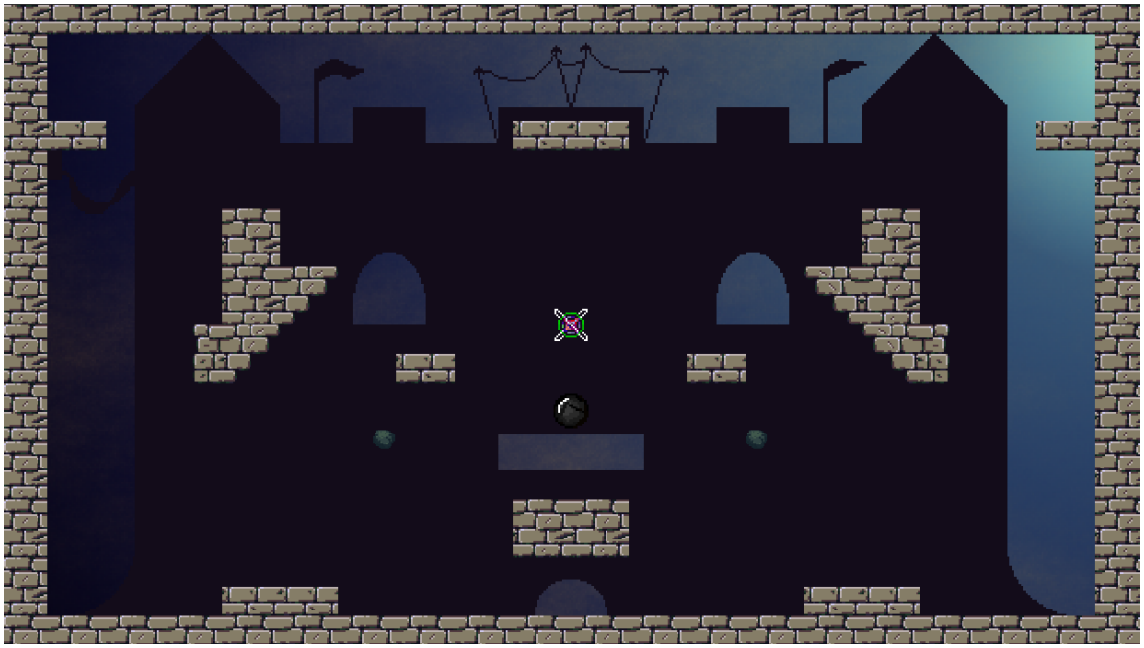

FIGURE 7 – Aperçu de la carte "Arène"

La seconde, "Space", n'est rien de plus qu'un décor vide permettant de perfectionner le maniement des projectiles et des pouvoirs sans être gêné par divers obstacles. Elle ne comporte qu'un fond uni avec une texture de ciel étoilé et des bordures dans un style néon futuriste.

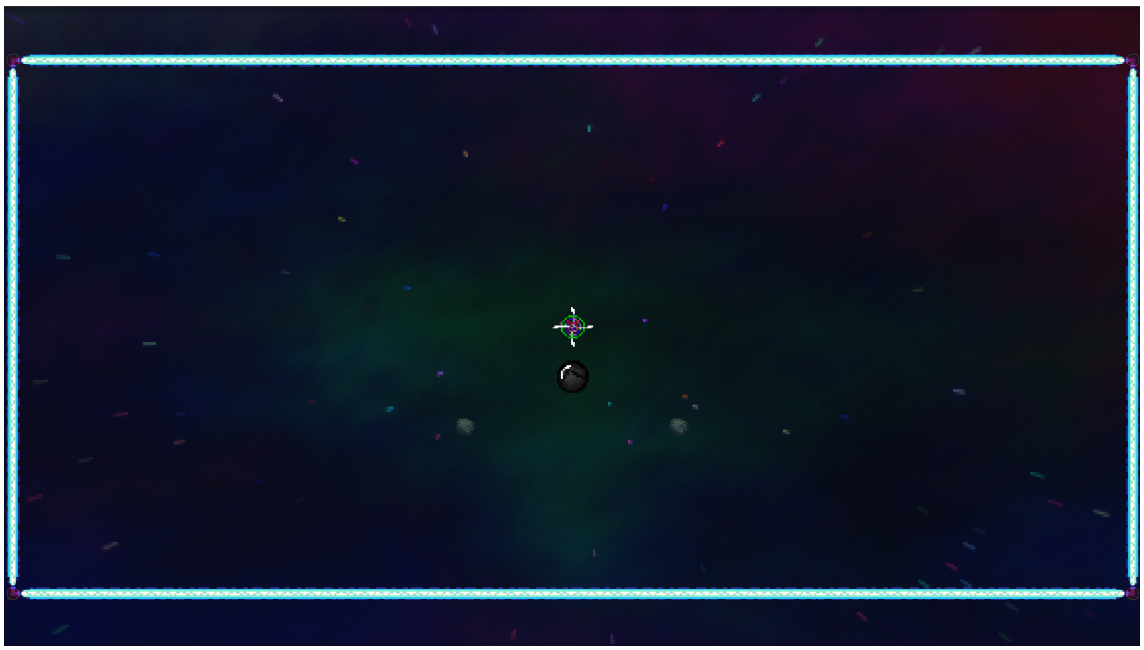

FIGURE 8 – Aperçu de la carte "Space"

D'autre part, les cartes dites spéciales que sont "Prison" et "Jungle" comportent des éléments de décor avec des particularités modelant le style de jeu en fonction des contraintes et avantages qu'apportent ces obstacles spéciaux.
"Prison", initialement appelée "Bumpers" est issue de l'ajout d'un élément circulaire rappelant les flippers qui peut repousser les projectiles et joueurs en cas de contact. Ces repoussoirs sont placés afin de rendre les manches sur ce niveau plus dynamiques bien qu'un peu plus complexes. Le fond de ce niveau reste comme la carte par défaut très sombre pour contraster avec les repoussoirs très colorés. L'aspect visuel bien que voulu comme presque oppressant fait ressortir les repoussoirs et ajoute au dynamisme.

Enfin, le dernier niveau est semblable à "Space" offrant un grand espace de combat. Pourtant cette carte se distingue des précédentes par la présence de l'élément liane qui ne laisse passer que les projectiles et non les joueurs. Ces lianes sont placées sous la forme d'une colonne centrale qui est le seul obstacle de cette carte. Les joueurs apparaissent de part et d'autre des lianes ce qui encourage un jeu à distance et un jeu en équipe au-delà de 2 joueurs bien qu'il n'y ait tout de même qu'un vainqueur.

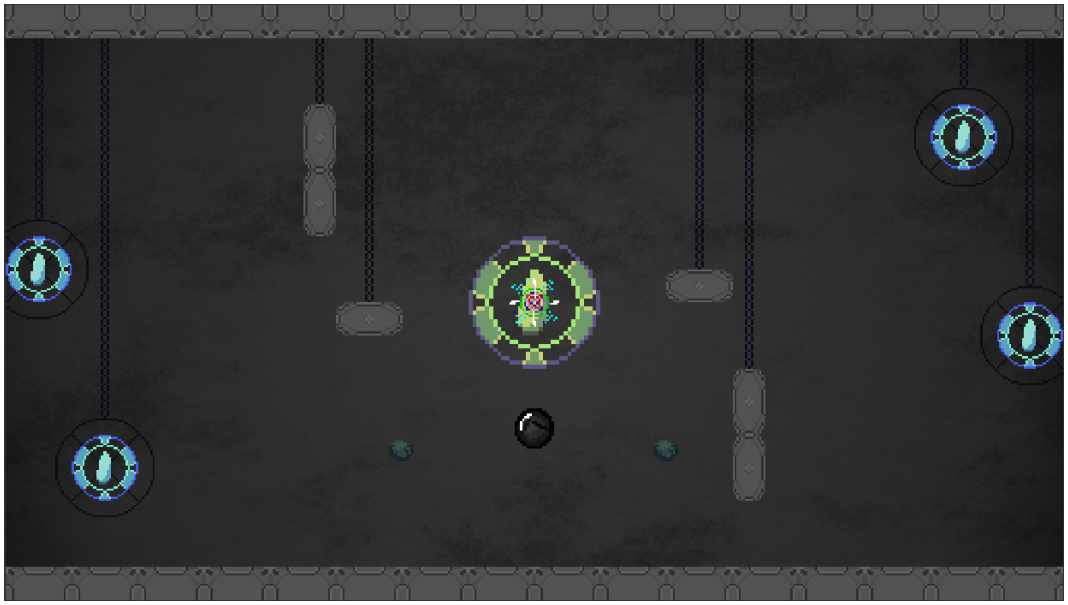

FIGURE 9 – Aperçu de la carte "Prison"

L'esthétique de cette scène se distingue également des autres avec une végétation abondante et des tons qui tendent plus vers le pastel. En effet, ce fond représente une canopée qui laisse entrevoir un ciel rosé et une planète géante gazeuse. Cette carte enrichit donc l'univers graphique initial du jeu mêlant nature, poésie et aventure spatiale aux combats survoltés.

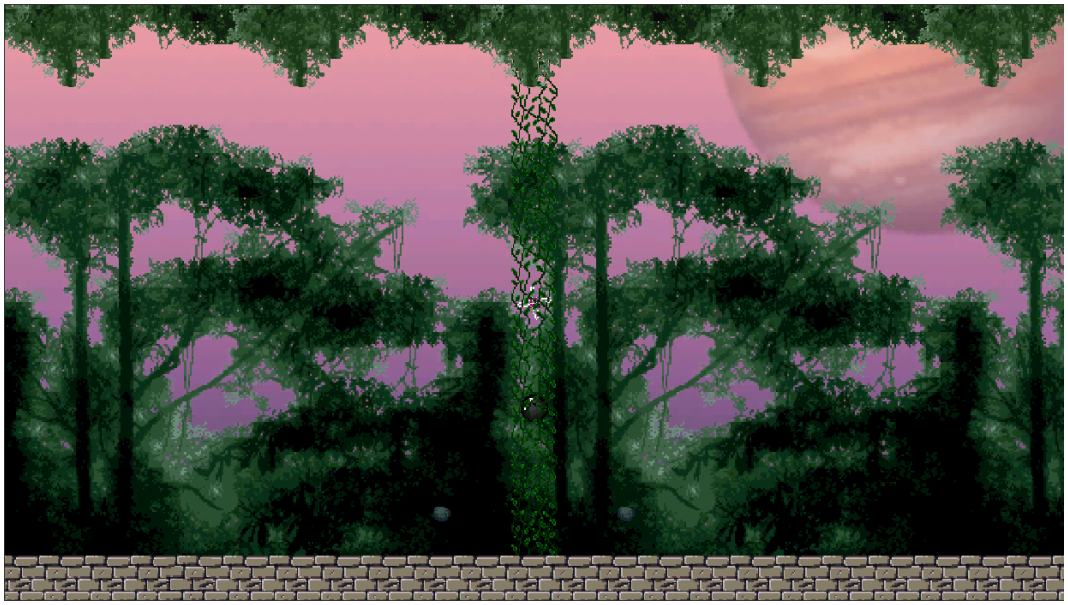

FIGURE 10 – Aperçu de la carte "Jungle"

5.9 Audio

La partie Audio concerne tout ce qui a été fait dans le projet et qui est audible, donc les musiques et les différents bruitages réalisés. Notamment, la bande-son du jeu contient six musiques originales, composées pour le jeu par nos soins, et environ une vingtaine de sons et bruitages différents peuvent être entendus dans le jeu.

5.9.1 Bande Originale

Les différentes musiques, ayant chacune une longueur d'environ 3 minutes, sont toutes dans un style synthwave afin adhérer à l'ambiance du jeu. Les instruments utilisés dans ces morceaux sont donc très souvent des synthétiseurs et autres accessoires nécessaires à la composition de musique électronique. Néanmoins, les émotions que cherchent à procurer nos morceaux, elles, varient. Ainsi, par exemple, tandis que la musique de la Carte Arène essaye de mettre en place une ambiance stressante de combat, la musique de la Carte Espace sera plus calme et aérienne et la musique du Menu Principal sera elle plus minimaliste et servira plutôt de support de fond, à l'instar d'une musique d'ascenseur.
Néanmoins les points communs essentiels à chacun de ces morceaux sont leur simplicité, leur constance en terme de volume (rares sont les moments de silence complet) ainsi que leur capacité à pouvoir se répéter à l'infini. Il est primordial que la bande originale respecte ces caractéristiques puisqu'une fois en jeu, la musique ne doit pas être trop complexe pour ne pas détourner l'attention du joueur, ne pas être trop variable en intensité pour ne pas cacher ni être recouverte par les différents bruitages et doit pouvoir se répéter sans que le morceau s'arrête puis redémarre de façon évidente.

5.9.2 Bruitages et Sons

Concernant les sons, ils ont la particularité d'être pratiquement tous semblables aux sons joués par les anciennes consoles de jeux 8-bit telles que la Nintendo Entertainment System.
Ce choix de notre part s'explique notamment par la difficulté de faire ou de récu-

pérer nous-mêmes des sons plus réalistes mais aussi et avant tout pour adhérer au style graphique en Pixel Art du jeu, style qui s'inspire lui-même des graphismes utilisés sur ces anciennes consoles comme la N.E.S. Leur utilité dans le jeu est de signifier les actions que nous jugeons importantes pour la bonne compréhension du jeu, par exemple, l'évènement de se faire enflammer par un objet déclenchera un son reconnaissable et très différent d'un son d'empoisonnement, d'impact ou encore d'activation de pouvoir par notre personnage. Il est ainsi plus facile de comprendre ce qui se passe en jeu.

5.10 Comparaison avec les objectifs

Tout d'abord, on peut heureusement constater que la majorité de nos objectifs initiaux ont été réalisés. Mais il existe cependant quelques différences. Dans le contrôle du personnage, par exemple, les capacités spécifiques des personnages n'ont pas été implémentées pour simplifier l'expérience de jeu, et la capacité de rappel a été remplacée par un simple renforcement de l'attraction afin d'augmenter la maniabilité. Le concept des cartes ayant des changements de pesanteur a aussi été abandonné, puisque trop gênant une fois en jeu. Par contre nous avons implémenté un nombre conséquent d'ajouts non prévus lors de l'écriture du cahier des charges, tel que les boules de feu et les virus empoisonnés ainsi que les concepts de cartes Prison et Jungle. Enfin et bien évidemment énormément de petites modifications ont été apportées sur nos idées principales afin de perfectionner l'expérience de jeu. Nous sommes donc plutôt satisfaits du jeu dans son état actuel qui, sans correspondre parfaitement à nos idées originales, réunit néanmoins tout ce que nous jugions essentiel au projet et répond totalement à nos attentes.

6 Réalisation

6.1 Physique

Un des éléments centraux des mécaniques de jeu du projet est de permettre des combats à base d'attractions et de déplacements d'objets flottant autour de personnages en lévitation. Ce but donne une forte place à la physique, poussant à chercher des cas concrets pour simplifier le problème que pose son implémentation.
La base fut donc de simuler une gravité considérant les personnages comme des astres pour ensuite chercher à mettre facilement en orbite les objets puis casser cette orbite tout aussi facilement. Le principe fondamental d'attraction était facile à implémenter grâce à certains tutoriels expliquant comment créer une simulation astronomique qui n'avait qu'à être adaptés en 2D. Ce premier script a servi de base pour y ajouter certaines fonctionnalités nécessaires au jeu comme les limites d'effet d'attraction selon la distance ou les éléments concernés par les interactions. Les principales difficultés débouchant de cette approche restent les implémentations liées au principe d'orbite qui en restant sur un fort réalisme seraient trop complexes à maîtriser pour les joueurs.
Le but suivant était donc de réussir à simuler une orbite stable autour d'un personnage qui soit suffisamment simple à tenir pour les joueurs mais sans empêcher les collisions entre les personnages et les projectiles. Afin d'affiner les orbites l'idée principale est de mettre en place une "zone tampon" autour des personnages pouvant opposer une force aux objets orbitant autour pour stabiliser l'orbite. La complexité ici repose sur l'équilibre entre forcer une pseudo-orbite stable et suffisamment facile à manier pour les joueurs sans pour autant rendre impossibles les collisions entre le personnage et les projectiles qu'il attire. La première piste était d'utiliser la simulation de fluide 2D de Unity sur un second *collider* représentant la "zone tampon". Cependant cette idée n'aboutit pas, la simulation de fluide par défaut n'étant qu'horizontale. Finalement une simple attraction mais de force variable et la possibilité de faire apparaître un cercle de collision maintenant l'orbite a suffi à donner un résultat convaincant.
Suite à cela, l'attention s'est portée sur l'ajustement et l'optimisation du travail

existant puis le nouvel objectif qu'est la gestion des collisions ainsi que quelques problèmes mineurs. Nous avons ajusté les paramètres liés aux différents modes d'attraction au fil des tests selon la maniabilité et la fluidité de l'expérience de jeu. Pour les collisions, les méthodes d'Unity ont été faciles à appréhender mais ont ensuite nécessité de nombreux ajouts de conditions liées au type d'objet, la vitesse, les déplacements et l'état.
Les quelques ajouts mineurs sont principalement les particularités des différentes cartes comme les *bumpers* (terme pris au flipper pour des éléments repoussant ce qui les touche) qui vont récupérer le vecteur mouvement de n'importe quel objet ou joueur le touchant pour lui appliquer l'inverse de ce vecteur en amplifié.
Enfin le dernier ajout concerne l'explosion des bombes. En effet pendant tout le temps d'une explosion, un cercle de collision de la taille de l'explosion récupère tous les objets ou joueurs avec lesquels il se superpose et leur applique un vecteur de magnitude arbitraire, dirigé du centre de l'explosion vers le centre de l'objet/joueur. De cette manière plus un objet est proche du centre de l'explosion quand la bombe se déclenche, plus de fois ce vecteur sera appliqué à l'objet avant qu'il ne soit plus dans la zone de collision de l'explosion, et donc plus il sera propulsé loin et fort du centre de l'explosion.

6.2 Contrôles / Multijoueur

Pour ce qui est de la gestion des commandes des personnages, nous avons vite compris que le jeu serait bien plus agréable à jouer avec une manette qu'avec un clavier et une souris. Nous nous sommes donc focalisés sur l'implémentation de contrôles à la manette. Pour cela nous nous sommes intéressés à l'*input manager* natif de Unity, qui permet de reconnaître la pression des touches et l'inclinaison des *JoySticks* des manettes en fonction d'un identifiant qui leur est attribué à leur connexion (Ref. 11).

Nous avons donc pu implémenter avec succès ces contrôles sur le script des joueurs, en utilisant ces actions pour coefficienter les vecteurs qui composeront la vitesse du joueur (à travers l'utilisation du composant *Rigidbody2D*) et déclencher

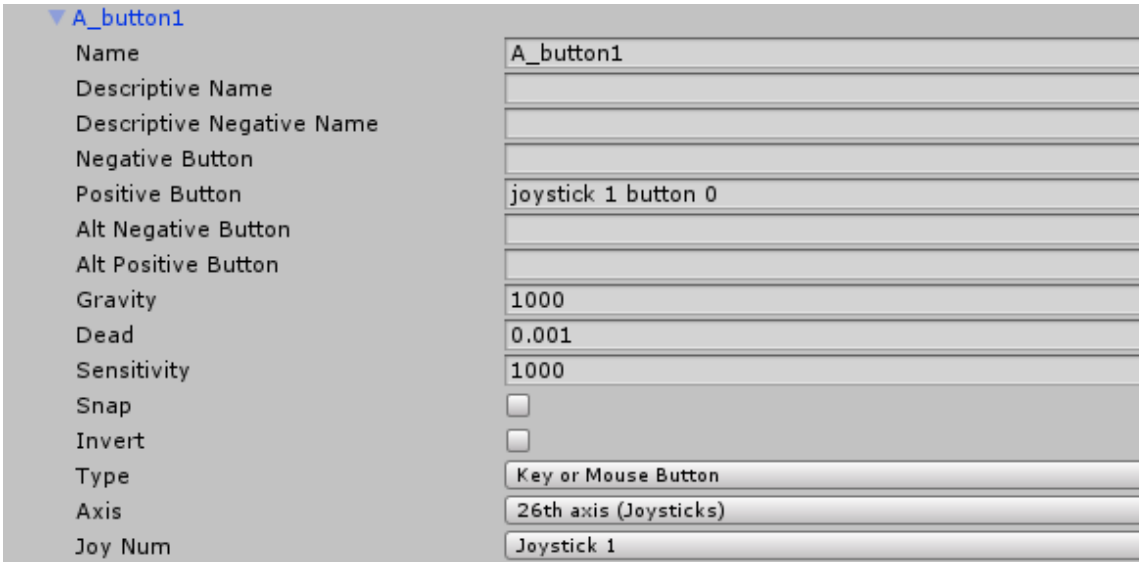

FIGURE 11 – Input Manager natif de Unity

des évènements. Nous avons alors pu rapidement rendre fonctionnels les contrôles sur un seul joueur mais nous avons cependant rencontré un problème lorsque nous avons voulu augmenter le nombre de joueurs.

En effet nous avons constaté que le numéro affecté aux manettes que l'on connecte et permettant de les identifier semblait aléatoire, et nous avons donc dû implémenter un système permettant aux différents joueurs d'être contrôlés par les manettes non pas en fonction de ces numéros de façon arbitraire, mais en fonction d'un ordre choisi par l'utilisateur. En effet à chaque nouvelle pression du bouton "Start" sur une manette n'ayant pas encore pressé ce bouton, cette dernière prendra le contrôle du prochain joueur qui n'a pas de manette affectée. Ainsi, même si le numéro de la manette est aléatoire et ne suit pas d'ordre prévisible, il est tout de même possible de contrôler tous les joueurs avec plusieurs manettes relativement facilement.

Des modifications ont néanmoins ensuite été apportées à ce système. Puisque nous avions un fonctionnement de jeu où les quatre personnages sont d'abord initialisés, puis reçoivent au fil des connexions une manette, il n'était pas possible de jouer avec seulement 2 ou 3 personnages, les autres personnages restant inertes mais présents. Dans le nouveau système de création de joueurs que nous avons ensuite conservé définitivement, les connexions de chaque manette sont récupérées dans le menu de choix de personnages (Fig. 12),

```
//Pour chaque contrôleur non connecté pouvant se connecter et si le sélectionneur de personnage est libre
for (int i = 0; i < buttonIndexes.Count && status == CharacterSelectorStatus.Waiting; i++)
{
    string idx = buttonIndexes[i];

    if (Input.GetButtonDown("A_button" + idx))
    {
        if (registeredPlayers == nbPlayersRequired)
        {
            //Si c'est au tour de ce sélectionneur de personnage de recevoir un contrôleur
            controller = idx;
            //1.75 - On enregistre le contrôleur, ce qui permettra lorsque le personnage sera créé de lui assigner ce dernier
            status = CharacterSelectorStatus.Selecting;
            imageRenderer.enabled = true;
            imageRenderer.sprite = charactersPreview[currentCharacterIndex];
            registeredPlayers++;
            buttonIndexes.Remove(idx);
            //1.81 - On retire le contrôleur de la liste des contrôleurs libres
        }
    }
```

FIGURE 12 – Exemple de code exécuté par les sélectionneurs de personnages, dont le rôle est de créer les futurs paramètres d'un personnage

il suffit ensuite de passer les paramètres choisis dans ce menu dans les attributs statiques de la classe GameManager pour permettre une fois le niveau lancé d'initialiser directement un nombre (qui restera ensuite figé) de personnages ayant tous des manettes pré-affectées. Cela permet de n'avoir que deux personnages en jeu s'il n'y a que deux joueurs, au lieu d'avoir des personnages inactifs affichés à l'écran.
En parallèle au contrôle des joueurs, un second système de contrôle a été rajouté pour la navigation dans les menus. En effet, en détectant les touches pressées sur les différents contrôleurs actifs, on change les différents attributs du système d'évènements affecté au menu, ce qui permet de pouvoir choisir quel contrôleur pourra naviguer dans les menus. Enfin bien que peu ergonomique, nous avons finalement implémenté des contrôles au clavier en le considérant comme une autre manette, afin de pouvoir plus facilement jouer à plusieurs (cela permet par exemple de jouer à deux au jeu même si on ne possède qu'une manette).

6.3 Graphismes

Pour commencer cette partie, nous allons parler de notre organisation vis-à-vis des graphismes, des choix que nous avons faits en matière de réalisation, puis nous verrons plus en détail le travail fait sur certaines parties relevant des graphismes.

6.3.1 L'organisation

Comme nous l'avons dit plus haut, nous voulions que le jeu soit en *pixel art* et que toutes les images soient réalisées par l'équipe. Dans cet objectif, et pour organiser notre travail, nous avons créé un tableur partagé où chaque membre du groupe pouvait ajouter les éléments visuels dont il avait besoin. Ce tableur était trié par "Type" (fonds, textures, objets, etc ...), et chaque élément ajouté était accompagné d'informations telles que ses dimensions en pixels, le nombre d'images nécessaires dans le cas d'une animation et si possible un croquis de référence. Avec cette méthode, n'importe quel membre du groupe souhaitant ajouter ou réaliser une image pouvait accéder facilement à une liste du travail à faire, et cela nous permettait également d'avoir un suivi précis de l'avancement de cette partie du projet(Ref. 13). N'ayant que quelques bases en dessin et en *pixel art*, nous avons dû suivre quelques tutoriels pour nous préparer, ceux du site *Les Forges* nous ont été d'une grande aide. Nous avons également fait de nombreux travaux de recherche pour trouver des références sur lesquelles nous appuyer lors de la réalisation des différentes images.

1							Nombre de sprite à réaliser :		Progression :
2							70		77%
3	Type	Dimensions	#	Description	Croquis	Fait	Asset fini		
10									
11	Texture	32x32	1	Mur de pierre (arène)	https://bit.ly/2XbiBMa	x			
12		-	1	"Triangle" en pierre (arène)		x			
13		-	1	Dalle (horizontale, verticale) en pierre		x			
14		-	1	Mur en métal (prison)	https://bit.ly/2TIUQw7	x			
15		-	1	"Triangle" métal (prison)		x			
16		-	1	Dalle en métal		x			
17		-	2	Bumper (genre champ de force)		x			
18		-	2	Délimitation pour map dans l'espace		x			
19		-	1	Délimitation carte jungle		x			
20									
21	Personnage	-	1	Manequin des personnages		x	https://bit.ly/2TQXY9D		
22		-	1	Une cyberpunk	https://bit.ly/2GvHdug				
23		-	1	Cyberounk avec crête					
24		-	1	Un THEUS	https://bit.ly/2TTgdbd	x			
25		-	1	Un gars de chez Spazon	https://imgur.com/e7jhVTV	x			
26									
27	Pose	-	1	Cyberpunk attire					
28		-	1	Bis attire					
29		-	1	THEUS attire		x			
30		-	1	Spazon attire		x			
31		-	1	Cyberpunk utilise pouvoir					
32		-	1	Bis pouvoir					
33		-	1	THEUS utilise pouvoir		x			
34		-	1	Spazon utilise pouvoir		x			
35									
36	Props	-	1	Caisse en bois		x			
37		-	1	Caillou		x			
38		-	1	Indiquateur quand props inflige dégat		x			
39		-	9	Une bombe (tete de mort REQUISE)		x			

FIGURE 13 – Tableau des éléments visuels lors du développement du jeu.

6.3.2 Les outils

Nous avons utilisé à la fois GIMP et Photoshop pour produire les images du jeu. Bien qu'ils ne soient pas spécialisés dans le *pixel art*, nous avions déjà des bases avec

ces deux outils et comme nous devions déjà apprendre à nous servir d'Unity, nous ne voulions pas nous rajouter la charge de devoir apprendre un nouveau logiciel tel que *GraphicsGale* ou *Aseprite*. Ne possédant pas de tablette graphique, nous avons travaillé à la souris, ce qui n'a pas posé de problème vu le style graphique que nous nous étions fixé.

6.3.3 Exemple de réalisation : les personnages

Nous avons accordé une attention toute particulière à la création des personnages qui seront les avatars des joueurs. Lors de la première phase de travail, nous espérions pouvoir réaliser leurs images avant la première soutenance. Mais le développement du site prenant plus de temps que prévu, nous avons été contraints de repousser ce travail à la deuxième phase. Mais grâce à cela, nous avions une meilleure vision du projet, et il fut plus facile de conceptualiser et d'intégrer les personnages.
Nous avons commencé par dessiner un modèle, en nous basant sur un style de *pixel art* simpliste, qui nous permettra de créer plusieurs personnages rapidement. Ce modèle ayant une résolution très basse, les personnages n'ont pas de visage et sont plutôt reconnaissables à leurs couleurs.

- **Les Cyberpunks :** Fiers habitants des Astéroïdes Troyens, ce peuple très pauvre se dresse contre l'oppression de Spazon. Ils portent du vert sombre pour pouvoir se dissimuler dans la forêt dense présente sur certains astéroïdes, et ont le visage masqué par un foulard. On pourrait se demander pourquoi ces précautions quand on sait que le port d'une crête iroquoise colorée fait partie de leur culture ...
- **Les THEUS :** Soldats d'élites créés par Spazon, ils sont les premiers clones génétiquement modifiés ayant développé des capacités télékinétiques. La modification de leur génome a entraîné un bleuissement de leurs peau, ce qui convient totalement au port de la salopette bleue des travailleurs de chez Spazon.
- **Les employés de Spazon :** Ces employés modèles et dévoués à leur entreprise en sont, pour la plupart, membres depuis le début. Malgré l'évolution de Spazon, ils ont tous gardé le costume du travailleur sérieux. Lorsque les

Cyberpunks prirent possession des THEUS, et s'approprièrent leur pouvoir, la quasi-totalité des haut gradés de chez Spazon se portèrent volontaires pour se faire implémenter une puce leur octroyant les mêmes pouvoirs, et partirent à la recherche des voleurs.

C'est en suivant ces éléments scénaristiques que nous avons élaboré chacun des personnages jouables, à savoir deux cyberpunks, un soldat THEUS et un employé de Spazon. Nous nous sommes également fait des avatars en suivant le même modèle, ils ont servi à la création d'un second logo pour le groupe et à illustrer le site web. Chaque personnage est décliné en trois poses : une basique, lorsqu'il ne fait rien, une où le personnage concentre sa force pour attirer à lui les objets, et une dernière où il est en train d'activer son pouvoir d'orbite. Pour ces deux poses, nous nous sommes inspirés respectivement de *Dragon Ball* lorsque les personnages chargent leur *Ki* et des films *Harry Potter* lorsqu'un sorcier lance un sort puissant.

FIGURE 14 – Exemple de personnage : le soldat THEUS.

6.3.4 L'importation des images dans Unity

Pour pouvoir utiliser nos images dans Unity, nous avons dû légèrement changer le processus d'importation. En effet, vu qu'il s'agit d'images avec une très basse résolution, du 32 par 32 pixels dans notre cas, nous devons l'indiquer à Unity. Pour ce faire, nous avons modifié trois paramètres :

— *Pixels per unit* qui indique combien de pixels de l'image correspondent à une unité dans l'éditeur, nous avons mis cette valeur à 32 pour faciliter le redimensionnement et le positionnement des images.

— *Filter mode*, ce paramètre indique l'algorithme utilisé pour recalculer l'aspect d'une image après un changement de taille ou de rotation, cela entraîne souvent un "flou" de l'image. Comme nous avons choisi de faire du *pixel art*, il faut que les pixels de nos images restent nets, nous avons donc indiqué "Point", ainsi aucun flou ne sera généré.

— *Compression* : nos images étant déjà très légères, une compression risquerait de les abîmer plus qu'autre chose, nous avons donc désactivé cette option.

Comme nous l'avons dit plus haut, certains éléments nécessitent plusieurs images, dans le cas d'une animation par exemple. Dans ce cas, toutes les images de l'animation sont compilées dans une seule image, qui est importée avec les mêmes paramètres que les autres, mais dont le paramètre *Sprite mode* est passé à "Multiple", ainsi nous pouvons indiquer à Unity les différentes images présentes dans le fichier de base (Fig. 15).

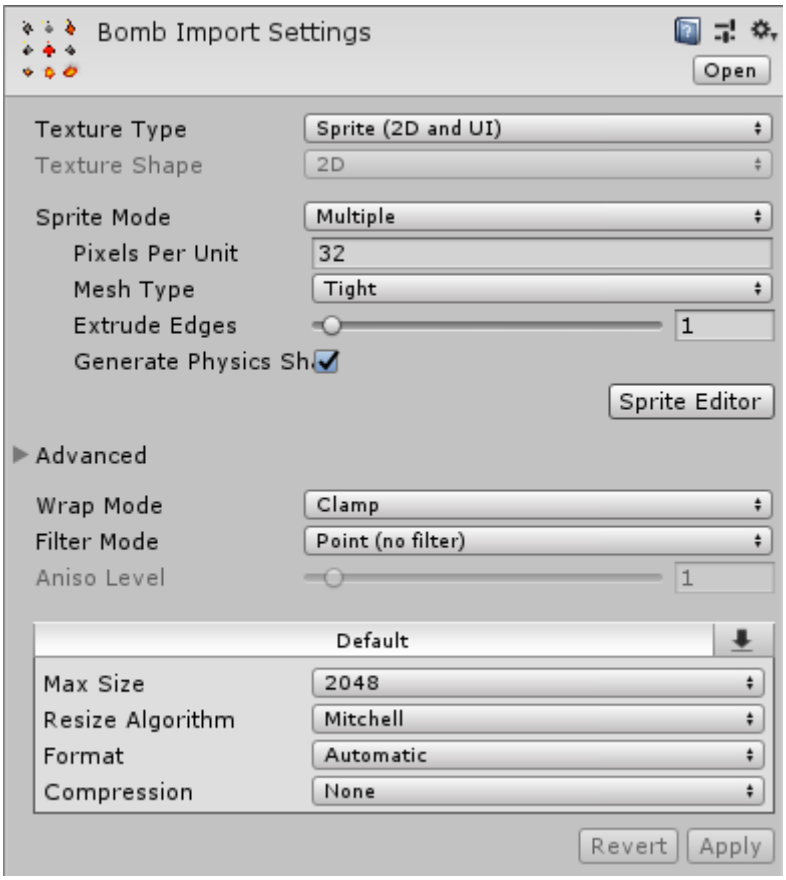

FIGURE 15 – Exemple d'importation d'image : la bombe.

6.3.5 L'implémentation

Une fois ce travail fait, il ne reste plus qu'à attacher les images à leur objet. On commence en ajoutant un composant *Sprite Renderer* à l'objet dans l'éditeur, on peut alors indiquer, de manière statique, à Unity quelle image il doit afficher lorsque cet objet est visible. Si nous devons modifier cette image affichée, nous avons deux méthodes dynamiques possibles :

— **Modifier directement l'image** : Le composant *Sprite Renderer* d'un objet est accessible depuis n'importe quel script attaché à ce même objet. Nous pouvons donc, après avoir récupéré les autres images, modifier facilement ce qu'il affiche directement dans le code.

— **Jouer une animation** : Avec Unity, il est possible de créer une animation avec plusieurs images. On place les différentes images sur une ligne de temps, on règle le délai entre chaque changement d'image, et il n'y a plus qu'à indiquer au composant *Animator* de l'objet quand jouer telle ou telle animation (Fig.16). Il est à noter que certains objets ne doivent jouer une animation qu'après un événement particulier, et le reste du temps afficher la même image, par exemple la bombe qui ne change pas tant qu'elle n'est pas activée. Dans ce cas de figure, l'animation par défaut de l'objet est un état "Vide" qui n'entraîne aucune modification de son affichage.

Le choix de telle ou telle méthode a été influencé par l'utilisation que nous faisions de ces changements d'images. Dans le cas du personnage, ils interviennent à intervalles très irréguliers car ce sont les actions de l'utilisateur qui amènent ces modifications, nous avons donc opté pour la première méthode. Pour ce qui est de la bombe, le changement doit être fluide et ne sera joué qu'une seule fois, nous avons donc opté pour l'animation.

```
1 // Définitions
2 private SpriteRenderer spriteRenderer;
3 public Sprite spriteStanding;
4 public Sprite spriteGrab;
5 // ...
6 // Récupération du composant SpriteRenderer et des images
```

```
7  spriteRenderer = GetComponent<SpriteRenderer>();
8  spriteGrab = sprites[0];
9  spriteStanding = sprites[4];
10 // ...
11 // Changement d'image
12 spriteRenderer.sprite = spriteGrab;
```

Modification directe de l'image.

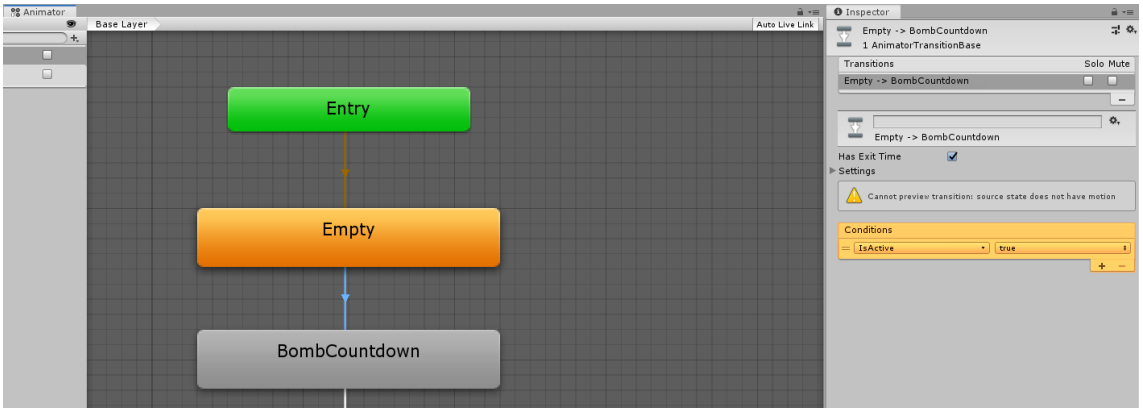

FIGURE 16 – La logique pour l'animation de décompte de la bombe.

```
1 // Définition
2 public Animator animator;
3 // ...
4 // On passe le booléen IsActive de l'animator à true -> l'animation de
      décompte se lance
5 animator.SetBool("IsActive", true);
```

Jouer une animation

6.3.6 Le contour

Comme nous l'avons expliqué dans la partie "État des lieux", nous avons ajouté un contour aux personnages pour que les joueurs puissent bien les différencier. Ce contour s'applique également aux objets qui sont attirés par le personnage. Pour arriver à ce résultat, nous avons dû ajouter un *shader* au personnage, qui va calculer, à l'aide de leur transparence, quels pixels correspondent au contour de l'image affichée. Il va ensuite modifier cette couleur selon celle définit pour le joueur. Lorsqu'un

objet est attiré par un personnage, on modifie également son *shader* pour que son contour soit visible et de la même couleur que celui du personnage.

```
1  // Définitions
2  public Color outlineColor;
3  public Material outlineMaterial;
4  private Material defaultMaterial;
5  // ...
6  // On récupère le material par defaut (pour enlever l'outline)
7  defaultMaterial = spriteRenderer.material;
8  // On fait une copie du outline material
9  outlineMaterial = new Material(outlineMaterial);
10 // On change la couleur du material outline
11 outlineMaterial.color = outlineColor;
12 // ...
13 // Si des objets sont en orbite et que le contour n'est pas actif
14 if (attractor.Attractors.Count > 0 && spriteRenderer.material !=
       outlineMaterial)
15 {
16     spriteRenderer.material = outlineMaterial;
17 }
18 else if (attractor.Attractors.Count == 0)
19 {
20     spriteRenderer.material = defaultMaterial;
21 }
22 // ...
23 // Lorsqu'un objet rentre en orbite
24 prop.GetComponent<PropScript>().ActiveOutline(outlineColor);
```

6.3.7 Les systèmes de particules

On peut ajouter le composant *Particule system* à n'importe quel objet dans Unity. On va pouvoir modifier la forme, la durée de vie, le comportement et plein d'autres paramètres de ces particules, et ainsi créer des effets visuels. C'est de cette façon que nous indiquons qu'un objet devient dangereux ou que le joueur prend

des dégâts de poison ou de feu. Pour les objets dangereux, nous voulions recréer des lignes de vitesse comme dans les bandes dessinées. Nous avons donc ajouté des particules longues, blanches et légèrement floues, qui s'activent lorsque l'objet dépasse une certaine vélocité (Fig. 17). Pour ce qui est de l'état des joueurs, nous avons créé une petite image triangulaire qui servira de modèle aux particules colorées en rouge qui indique que le joueur est en train de brûler. Le poison quant à lui sera représenté par une image circulaire et une couleur verte.

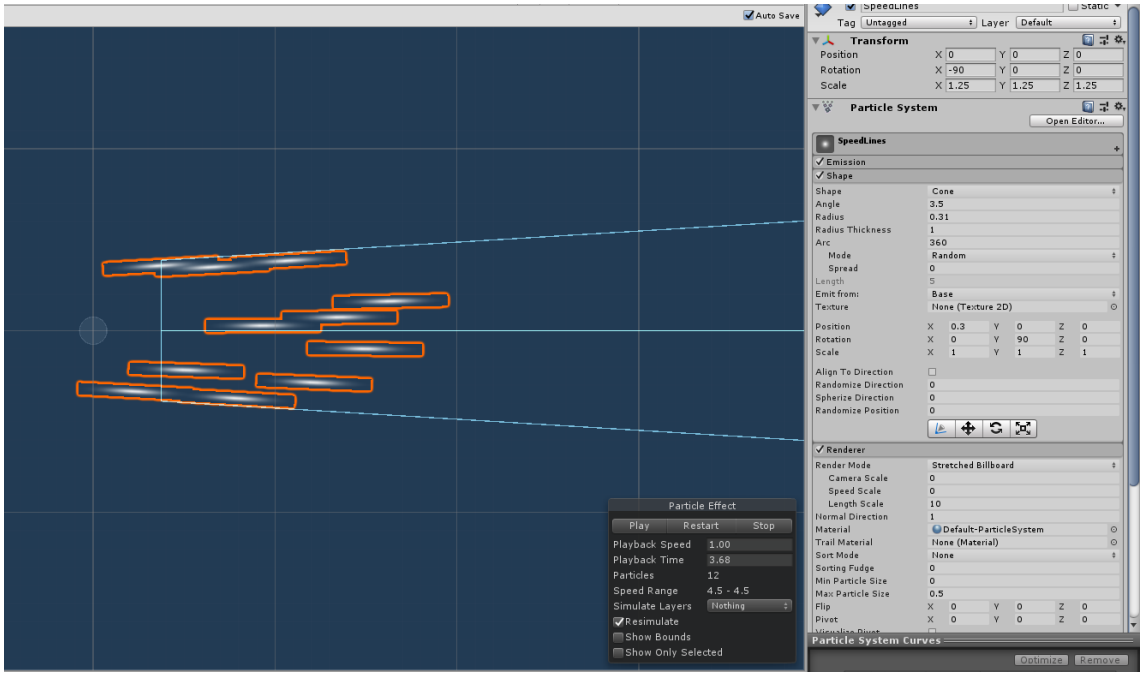

FIGURE 17 – Les particules de vitesse.

Force est de constater que tout au long du projet, ça n'a pas été chose facile que d'arriver à produire les images attendues, notre niveau en dessin n'étant pas suffisamment bon. Mais nous sommes plutôt fiers du résultat car seule une image a nécessité l'emploi de ressources appartenant à un autre jeu : le fond de la carte "Jungle". Nous en avons donc profité pour y glisser une référence en utilisant une image de forêt extraite du jeu *Donkey Kong Country* sur la *Super Nintendo Entertainment System*. Toutes les autres images ont été produites par nous-mêmes.

6.4 Site Web

La réalisation du site web n'a posé aucun problème car nous savions déjà ce dont nous avions besoin pour le faire. Nous avons commencé par les éléments de style généraux, qui sont présents sur toutes les pages comme le fond ou la barre de navigation (fichier *style.css*) avant de nous attaquer plus précisément à chaque page. La création, entre autres, de la barre de navigation a nécessité plusieurs ajustements, pour qu'elle s'affiche correctement quelle que soit la taille d'écran, cela passe par la définition d'une taille maximale fixe, qui ne sera modifiée que si le visiteur est sur mobile, auquel cas la barre de navigation est adaptée. Toutes les pages arborent un élément un peu particulier qui a nécessité un travail plus long pour obtenir le résultat escompté. Voyons deux exemples.

6.4.1 Exemple de réalisation : la page OGAML

Après les éléments communs à toutes les pages (barre de navigation et pied de page), nous avons ajouté un premier paragraphe simple pour présenter le groupe. Pour ce qui est des membres, nous voulions donner un petit côté "fiche de personnage" dans un jeu de rôles. Nous avions donc besoin d'un avatar, d'une classe et de statistiques (qui seront en fait les tâches de la personne). Pour l'avatar, il a suffi d'une balise image à laquelle nous avons attribué les images des personnages créés en même temps que ceux du jeu. Mais nous voulions ajouter un petit effet de lumière lorsque l'on survole la fiche d'une personne. Nous avons utilisé la méthode CSS *: :after*, qui permet d'ajouter des éléments autour de ceux déjà présents dans le code HTML. La "boîte" que nous avons ajoutée, fait la même taille que l'image et se place au-dessus d'elle. Elle dispose d'une ombre interne transparente et qui va changer de couleur selon la personne survolée, créant ainsi un effet de halo au-dessus de l'image (Fig. 18). Le nom et la classe sont des éléments de textes basiques. Les barres de progression des tâches, quant à elles, sont composées de deux éléments. Un conteneur, qui définit la taille maximale de la barre, et un contenu qui fait la même hauteur que le premier, mais dont la largeur est différente. Sa taille étant calculée en pourcentage de celle du conteneur, il nous suffit de recopier le pourcentage de progression de la tâche en question, et la barre prendra la bonne taille.

```css
.img-container::after
{
    content: ' ';
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    box-shadow: inset 0 -50px 30px -30px transparent;
    transition: all 0.2s ease;
}
```

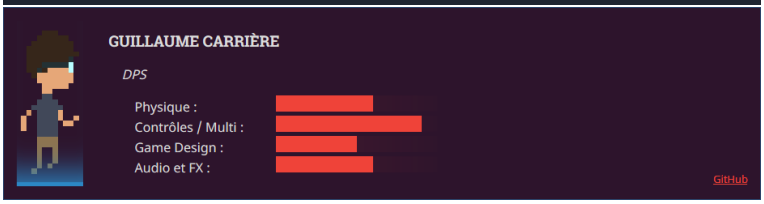

FIGURE 18 – Le code CSS et le rendu de l'effet halo.

6.4.2 Exemple de réalisation : la page Jouer

L'élément dont nous allons parler sur cette page en est le premier. Il s'agit du tableau proposant au visiteur de télécharger le jeu selon sa plateforme et les différentes méthodes disponibles. Nous sommes partis d'un tableau simple, regroupant les méthodes d'installation par système d'exploitation et par version. Si une méthode n'est pas disponible pour une certaine version, on laisse vide la case du tableau. Lorsqu'un téléchargement est possible, on place uniquement son lien (une balise *a*) dans la cellule correspondante. Pour ce qui est du style du tableau, nous avons modifié la couleur du fond des cellules titres, et lorsqu'une case de téléchargement est vide, on y ajoute une croix. Cette croix est créée grâce à la balise *: :after*, que nous avons vue plus haut, et à sa soeur *: :before*. Cette fois-ci, les "boîtes" ajoutées sont petites, sombres et on leur applique une rotation différente pour les croiser. Pour ce qui est des cellules contenant un lien, nous allons modifier la boîte de ce lien pour qu'elle ait la même taille que sa cellule. Son fond est également modifié pour y incorporer une icône de téléchargement.

6.5 Interfaces / Menu

Dans la scène du menu principal, chaque sous-menu est représenté dans un objet *canvas*. Les enfants de cet objet constituent les éléments du menu et sont des objets de la classe *UI* fournie par Unity. Ce sont des types d'objets faits pour élaborer des interfaces utilisateur, comme des images, du texte, des boutons, etc... L'action de chaque bouton se définit dans sa méthode *OnClick*. Pour naviguer d'un menu à un autre, le bouton qui change de menu se contente simplement de désactiver le *canvas* actuel et d'activer celui du menu sur lequel basculer. L'activation et la désactivation d'un *canvas* se font avec la méthode *GameObject.SetActive* en lui faisant passer en paramètre un booléen.
Le système d'évènements (lui aussi fourni par Unity) permet d'interagir avec les éléments au moyen d'une manette ou un clavier (alors qu'ils sont d'abord faits pour répondre à la souris) sans que nous ayons eu besoin d'implémenter quoi que ce soit par rapport aux contrôles.
Dans le sous-menu des réglages, ce sont des curseurs (objets *Slider* dans **Unity.UI**) qui règlent le nombre de manches à gagner et les volumes des effets sonores et de la musique.
Comme nous l'avons décrit plus haut, le sous-menu primordial est celui dans lequel les joueurs se connectent. Ce sous-menu est régi par un script qui s'occupe de récupérer les informations des manettes et d'y associer les joueurs. C'est ce même script qui modifie les éléments du *canvas* pour afficher les joueurs correspondant aux manettes connectées. Il permet aussi de modifier les images des objets *CharacterSelector* pour afficher les joueurs dans l'ordre dans lequel ils se sont connectés, de leur faire choisir entre différents personnages et de vérifier lorsque ceux-ci sont prêts, pour ensuite les faire valider et les amener sur le sous-menu dans lequel ils choisiront le niveau. Lorsqu'un niveau a été sélectionné et validé, pour le charger, le script appelle simplement une fonction de la librairie *SceneManager* de Unity pour charger la scène correspondant au niveau sélectionné.

Le menu pause fonctionne d'une manière assez similaire : il s'agit d'un *canvas* contenant les images, textes et boutons de ce menu mais il est désactivé pendant la

partie. Ce n'est que lorsqu'un joueur appuie sur la touche qui appelle à mettre le jeu en pause que le *canvas* est activé. Mais afficher le menu pause ne suffit pas, il faut aussi mettre le jeu en pause, sinon il continuerait de tourner avec le menu devant la scène. Pour mettre le jeu en pause, la fonction appelée lorsqu'un joueur appuie sur la touche pause met d'abord la fréquence d'actualisation du moteur de jeu à 0 avant d'activer le *canvas* du menu, ce qui a pour conséquence de ne jamais mettre à jour le jeu. Ainsi tous les éléments, que ce soient les joueurs, les objets ou les minuteurs sont figés et le jeu est techniquement bien en pause. Malheureusement, cela provoque l'inutilisation des *sticks* analogiques et l'impossibilité de s'en servir pour naviguer dans le menu pause. Pour pallier cet inconvénient, nous avons simplement fait en sorte que lorsque le joueur qui a lancé le menu pause appuie sur une touche spécifique (pour les manettes, la touche [B]), le script lui fait sélectionner le bouton suivant, ce qui lui permet de naviguer de bouton en bouton.

Le menu de fin a été réalisé après que le menu pause ait déjà été implémenté. Il est donc très similaire dans son fonctionnement en reprenant le principe de mettre le coefficient de déroulement du temps à 0. À vrai dire, il est même quasiment identique sauf pour deux choses qu'il a fallu changer. Tout d'abord le bouton *Resume* qui a été remplacé par le bouton *Restart*. Ce changement n'a pas été très complexe à implémenter puisque, son objectif étant de relancer la partie sans changer les paramètres des joueurs, il suffisait littéralement de recharger la scène sans toucher au reste, en n'oubliant pas de remettre le coefficient de temps à 1. L'autre ajout est le texte de victoire qui annonce le vainqueur en écrivant "Winner : Player <indice du joueur ayant gagné>". Pour cela il a fallu, depuis le code permettant d'initialiser le menu de pause, passer en paramètre l'attribut "id" du joueur vainqueur pour ensuite l'afficher dans le texte de victoire.

L'affichage tête haute est différent des menus. Il s'agit aussi d'un *canvas* qui contient des éléments d'images et de texte mais par contre aucun élément avec lequel on peut interagir. Contrairement aux autres *canvas*, il est activé en permanence. Il contient un élément principal qui affiche le numéro de la manche actuelle. Ce

numéro est mis à jour lorsqu'une manche se termine appelant une méthode depuis le *GameManager*. Il y a aussi un élément par joueur qui affiche des informations sur ce dernier. Chacun a un script qui possède un attribut référence vers l'objet joueur auquel il est associé et ainsi chaque script accède en temps réel aux attributs du joueur pour les afficher. Pour donner l'effet de remplir ou vider les jauges (de vie et de rechargement de l'esquive), on a simplement une image qui va bouger dans un cadre qui a pour particularité de rendre invisibles les objets qui en sortent. Ainsi, pour faire diminuer la longueur d'une barre, on la fait simplement se décaler et elle se trouve en dehors de son cadre, donc invisible. La position horizontale de cette image est calculée en fonction de sa position initiale, de sa taille et du rapport de remplissage que l'on veut lui donner.

6.6 Mécaniques de jeu

Le prototype du jeu, dès le départ, permettait déjà de contrôler des joueurs et interagir avec l'environnement via les quelques mécaniques d'attraction des objets, formant en soi la première et plus essentielle part de notre mécanique de jeu.
Néanmoins deux attractions principales différentes ont été implémentées. L'une active une attraction constante de l'objet associé. À n'importe quelle distance de celui-ci, le joueur peut le ramener rapidement vers lui pour le renvoyer. L'autre permet une attraction gravitationnelle, qui dépend donc de la distance entre l'objet et le joueur : si celui-ci est éloigné, elle sera peu efficace, par contre si l'objet est proche, il pourra être envoyé avec beaucoup plus de vitesse après avoir été accéléré en longeant le cercle de collision plus large, comme expliqué dans la partie sur le moteur physique.
Nous avons ensuite réfléchi sur ce qui devrait être modifié et apporté pour que ces mécaniques soient amusantes et satisfaisantes à maîtriser. C'est là aussi que nous avons constaté la différence de difficulté de maîtrise entre les deux mécaniques. Le mode renforcé étant toujours moyennement simple à utiliser tandis que le mode en orbite circulaire est, lui, plus simple à utiliser mais bien plus difficile à maîtriser complètement.
Après avoir peaufiné les deux modes d'attraction, nous avons travaillé sur les mé-

caniques des objets (caisses, rochers, etc...) qui ont été légèrement améliorées pour éviter des erreurs lors de situations particulières comme les blocages dans des coins ou d'autres collisions farfelues. La vitesse de déplacement des joueurs a aussi été réduite pour rendre l'action plus cohérente à l'échelle des personnages et des niveaux et nous avons donné aux joueurs des points de vie et la capacité de prendre des dégâts lorsqu'ils sont heurtés par des objets à haute vitesse, ce qui leur fait perdre de la vie. Enfin la dernière partie de notre travail fut de coder les différents comportements des objets. Pour la caisse, il a suffi de compter le nombre de chocs capable de donner des dégâts et de détruire l'objet au bout de trois chocs. Pour la flèche, il faut remplacer les dégâts d'impact par des dégâts fixes. La même solution a été utilisée pour la bombe lorsqu'elle explose. Enfin les boules de poison et de feu fonctionnent de la même façon : leurs dégâts d'impact ont été remplacés par le lancement d'une fonction dans le script du joueur qui va lancer l'état "Poison" ou "Feu". Le joueur va garder cet état pendant trois secondes et prendre de légers dégâts toutes les demi-secondes tant qu'il est présent.

6.7 Construction des niveaux

La construction des niveaux s'est divisée en trois phases différentes au fil de l'avancement. Tout d'abord, l'élaboration d'un prototype par des croquis papier rendant compte des formes, tailles et agencements prévisionnels des obstacles, joueurs et projectiles, puis par l'implémentation simple sans textures de ce prototype dans Unity. Ce prototype nous a servi à prendre nos marques sur Unity et à affiner l'idée du résultat attendu. Cette première phase s'est donc poursuivie en ajustant quelques aspects des niveaux comme la taille, la zone d'apparition des projectiles et l'adhérence des obstacles (Fig. 19).

Ensuite, une deuxième phase s'est caractérisée par l'ajout de texture au premier prototype et la création d'une deuxième carte vide mais directement texturée. Dans cette phase les deux niveaux présents ont été de très nombreuses fois testés et améliorés pour les rendre les plus fiables possibles. Ces deux cartes ont permis de tester au fil du temps l'ensemble des différents ajouts sur une base solide, pour se

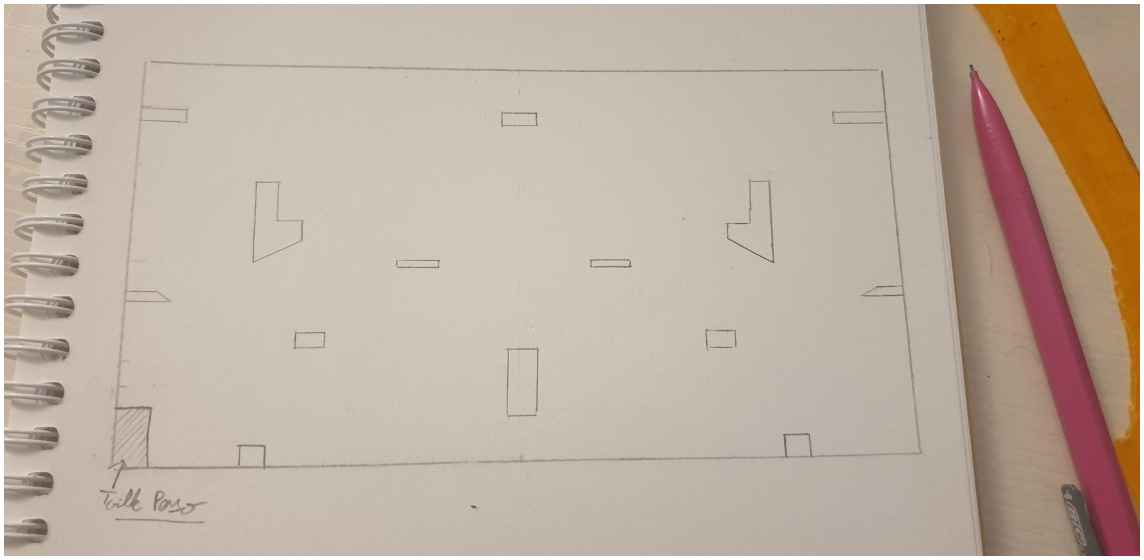

FIGURE 19 – Concept-Art typique pour les créations de cartes (Arène en l'occurrence)

rapprocher aisément et progressivement d'une version finale. Cette phase a permis de gérer une structure définitive pour les niveaux. De plus, deux cartes ont permis de normaliser la création des niveaux pour rendre simples la gestion des menus, le chargement des scènes et l'organisation générale des manches.

Enfin, la troisième phase est la plus riche car les bases étant déjà suffisantes, l'ajout de nouvelles cartes était bien plus simple. Cette simplicité d'implémentation a permis de développer des spécificités pour chaque nouvelle carte en introduisant deux nouveaux éléments que sont les vignes et les repoussoirs. Le premier est un obstacle semblable aux briques mais ne permettant aucune collision avec les projectiles qui passent à travers contrairement aux joueurs. Les repoussoirs appliquent une force inverse, amplifiée et proportionnelle à la magnitude du mouvement d'un élément le percutant et le paralysent quelques secondes si c'est un joueur, simulant un rebond. Un niveau a donc été construit autour de chacun de ces éléments spéciaux, respectivement les cartes "Jungle" et "Prison". Ces deux cartes ont comme pour le prototype été sous forme de croquis papier tout de suite implémentées, améliorées et texturées dans Unity. Cependant, leur aspect visuel et la répartition des obstacles ont été plus approfondis que pour les premières créations, les cartes étant plus riches

avec un plus grand intérêt que les anciennes cartes de tests par défaut.

6.8 Audio

Nous avons au démarrage vite compris que la gestion de l'audio devait être décomposée en deux sous-domaines à traiter : la création et l'utilisation de la musique, et celles des sons. Nous avons donc d'abord commencé en travaillant sur l'implémentation de bruitages trouvés sur internet en libre-accès, puisqu'il fallait d'abord créer les musiques afin de les implémenter, ce qui se révéla bien plus chronophage. Pour cela, nous avons utilisé le composant AudioSource, en l'attachant à différents objets du jeu, ce qui permet, dans les scripts de ces objets, de déclencher le fichier audio contenu dans le composant à un instant précis, nous avons donc pu facilement assigner un bruitage à un objet. Néanmoins nous avons alors rencontré quelques difficultés lorsque nous avons voulu faire jouer plusieurs sons par un seul objet de jeu. En effet notre système ne permettait que de jouer un seul son. Après nous être renseignés sur le fonctionnement de ce composant, nous avons alors pu, à travers le script des objets, utiliser différents sons en affectant le son nécessaire depuis les attributs du script avant de l'utiliser en fonction des évènements en jeu.

```
//On assigne le son que l'on veut jouer au composant Audio Source, puis on le joue
audioData.clip = dashSound;
audioData.Play(0);
```

FIGURE 20 – Exemple de code utilisé pour jouer un son (pour l'esquive en l'occurrence)

C'est donc avec ce système de fonctionnement que nous avons implémenté tous les bruitages nécessaires dans le jeu (20).
Ensuite pour la partie musique, nous avons commencé par composer le thème du premier niveau : l'Arène. Pour y parvenir, nous avons utilisé le logiciel FL studio, destiné à la composition, au mixage audio et particulièrement performant pour créer de la musique électronique (Fig. 21). Après familiarisation avec le logiciel, nous avons donc créé un thème de combat dans le style *CyberPunk*, en incorporant aussi des

sons de combat tels que des cris de guerriers. Cela nous a permis de commencer à prendre en main le logiciel FL studio et la composition de musique électronique de façon générale. En suivant cette voie nous avons continué à composer tout au long de l'avancement du projet 5 autres morceaux, en utilisant quasiment les mêmes outils (la seule petite différence étant l'ajout de différents *plugins* sur le logiciel FL Studio).

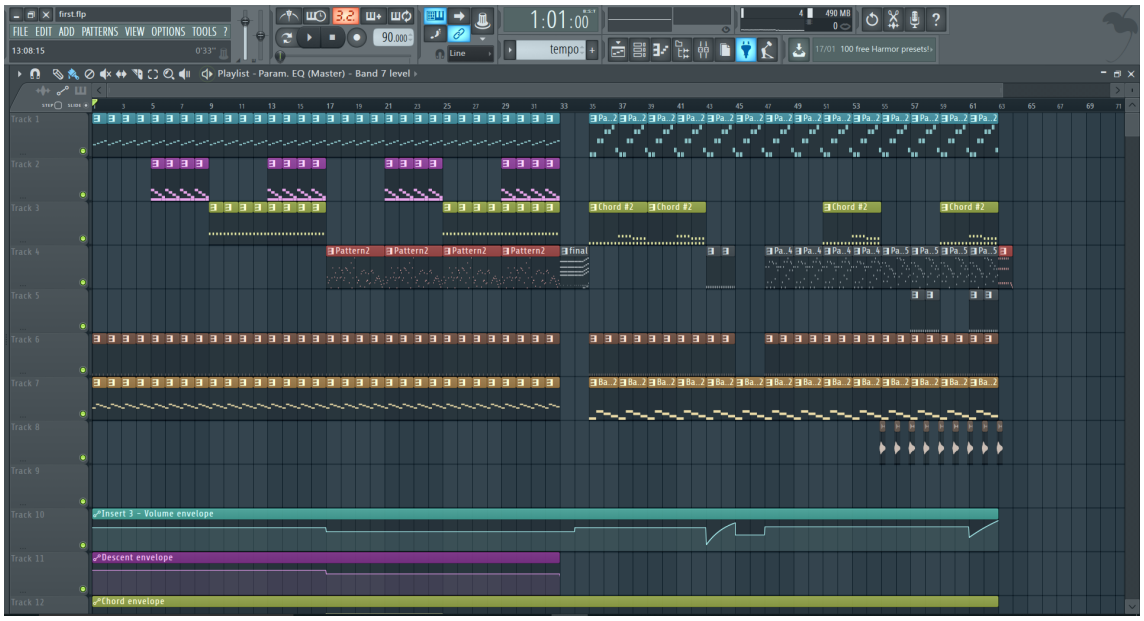

FIGURE 21 – Le plan de travail du logiciel FL Studio

6.9 Distribution

Pour pouvoir exporter notre projet sur Windows, Linux et Mac OS X, nous avons dû ajouter les modules *Linux Build Support* et *Mac Build Support* à notre installation Unity. Une fois cela fait, nous avons configuré l'exportation en personnalisant, entre autres, le logo de l'exécutable et l'animation jouée en introduction avec le logo d'Unity et celui d'OGAML. Une fois l'exportation des trois versions terminée, nous avons pu les héberger grâce à Unity Collab, mais il nous restait une étape pour la version Windows. En effet nous devions faire un installeur, c'est-à-dire un unique exécutable que l'on télécharge et qui va installer le jeu. Pour savoir comment faire, nous avons regardé la vidéo Youtube de la chaîne *Brackeys* à ce propos. Il suffit d'utiliser un logiciel comme *Inno Setup* avec lequel on peut paramétrer le comportement

de notre installeur. Nous avons personnalisé le logo, mis tous les fichiers nécessaires au bon fonctionnement du jeu, et indiqué que l'installation avec les droits administrateur n'était pas nécessaire. Nous avons également activé l'option qui ajoute une procédure simplifiée de désinstallation. Une fois ce travail effectué, nous avons pu écrire les instructions d'installation et de désinstallation, présentes sur le site et sur la clef USB remise avec une archive du projet et l'installeur.

7 Améliorations Possibles

Bien sûr aucun projet n'est jamais terminé et on peut toujours penser à des façons d'améliorer ou d'étendre son travail, mais, pour notre jeu, certaines améliorations nous semblent plus claires.

L'ajout le plus évident concernerait les cartes, le jeu n'en possédant que quatre pour l'instant. Entre autres, nous avons plusieurs idées et concepts, tels que par exemple une utilisation de la gravité compatible avec nos mécaniques de jeu, comme nous l'avions prévu à la base. D'autres ajouts éventuels seraient des images et des animations supplémentaires pour les personnages, des changements de *design* pour les menus ou autres améliorations graphiques. En dernier lieu, il pourrait être envisageable de rajouter un tutoriel complet ou un mode histoire jouable en solo (afin de développer le scénario du jeu pour l'instant relégué au second plan), mais ces derniers ajouts seraient bien plus complexes et chronophages à implémenter que les autres perspectives d'amélioration. Ultimement, afin de perfectionner notre projet, il nous faudrait aussi intensément tester le jeu pour pouvoir corriger les éventuels bugs, et modifier les règles et paramètres des parties de façon à proposer la meilleure expérience de jeu possible.

8 Ressentis Personnels

8.1 Guillaume Carrière

Pour ma part, les parties me concernant principalement étaient l'Audio ainsi que la Gestion des contrôles multijoueur. Tandis que la gestion des contrôles a été choisie assez arbitrairement, j'ai personnellement voulu m'occuper de la partie Audio car la gestion de fichiers audio et la composition sont des domaines qui m'attiraient. Puisque les parties du développement me concernant (principalement la gestion des inputs ainsi que l'Audio) demandaient moins d'efforts à gérer que les autres, j'ai bénéficié de plus de temps que mes camarades pour me pencher sur les autres parties plus conséquentes me concernant moins, afin d'aider à la progression de ces dernières au mieux. Ma participation dans le projet peut donc être qualifiée de "touche-à-tout", et le savoir que j'en retire est ainsi assez éclectique. Néanmoins malgré la charge de travail relativement légère de la partie Audio, en plus de l'expérience générale que j'ai pu acquérir, je me suis amplement familiarisé avec le domaine de la composition de musique électronique et la manipulation de fichiers audio. En somme le projet m'a permis d'apprendre beaucoup sur la programmation orientée objet, la manipulation de l'audio ainsi que sur le travail en groupe en informatique.

8.2 Marwan Dahou

Dans ce projet j'avais une responsabilité plus importante dans la gestion de la physique et la construction des niveaux. En tant que chef de groupe je me suis également concentré sur la coordination des différents domaines et sur la direction artistique globale du projet. L'aide de Guillaume en physique m'a été très utile pour implémenter et améliorer vite nos idées. Les implémentations faites par Guillaume étaient bien plus fiables et réussies que les miennes ce qui a permis pour la physique d'avoir un très bon résultat et m'a beaucoup aidé à progresser. Cette aide apportée m'a permis de mieux suivre l'avancement des autres domaines et de travailler plus sur la création des niveaux. Celle-ci m'a beaucoup plu et était donc au centre de mon travail mais demandait beaucoup de liens avec les mécaniques de jeu, les graphismes

et la physique ce qui m'a donné une approche assez complète dans ce projet et m'a permis de garder une vision globale du projet et de suivre des objectifs stables et clairs. Le travail en groupe et plus particulièrement en duo a été très bénéfique pour résoudre vite des problèmes et mettre en oeuvre nos idées. Cette cohésion du groupe nous a aidés à développer un projet cohérent mêlant harmonieusement les différentes visions personnelles en évitant les conflits qui auraient pu naître de la confrontation. Au final, cette expérience m'a permis de voir comment encadrer un projet et répartir les tâches pour passer d'une simple idée à un produit fini. De plus, concevoir un jeu vidéo m'a permis de voir comment établir des liens entre les aspects techniques et artistiques.

8.3 Louis Gasnault

Avant ce projet, je m'étais déjà investi dans le développement de petits jeux vidéo et cela souvent en équipe. Mais c'était à chaque fois sur une période courte, et avec une organisation hasardeuse. Participer à Purest Atom Exam a été pour moi l'occasion de me confronter à un projet long et complexe dans son organisation et sa réalisation. Heureusement toute l'équipe a su faire face, et c'est en travaillant de concert que nous avons pu mener à bout notre travail. La cohésion du groupe, aidée par l'amitié qui nous liait déjà, m'a beaucoup plu. Nous avons passé de longues soirées à travailler tous ensemble sur le projet, en échangeant avis et conseils, et les rires étaient souvent au rendez-vous. Le fait d'apprendre de nouvelles choses tout en créant a été ma motivation pendant tout ce projet. En effet, m'occupant principalement de l'aspect graphique, j'ai pu laisser libre cours à mon imagination pour créer un univers, chose que j'affectionne tout particulièrement et qui a donc été une source de motivation supplémentaire. Bien que l'on puisse penser que cette partie ne nécessite pas de programmation, cela se révèle vite faux. Pour intégrer tous les éléments graphiques dont nous avions besoin, j'ai dû passer plus de temps le nez dans le code que sur mon logiciel de dessin. De plus, certains éléments, comme les systèmes de particules et le contour des personnages étaient plus de la programmation que du dessin. J'ai également dû suivre de près le développement de chaque fonctionnalité du projet pour, le moment venu, savoir comment les illustrer. J'ai donc interagi avec

tous les membres de l'équipe sur toute la durée du projet. J'ai également participé directement à la construction de certains niveaux comme Arène et Prison, et j'étais en charge du développement du menu de pause. En ce qui concerne le développement du site web, c'est un domaine où j'avais déjà une certaine expérience, ce qui m'a permis de me concentrer sur une création artistique et fonctionnelle. Je sors de cette expérience avec des idées de projets plein la tête, mais surtout avec le désir de perfectionner mon utilisation d'Unity. Je garde précieusement en mémoire tout ce que nous avons dû mettre en place pour travailler ensemble, et je saurai le réinvestir dans de futurs projets.

8.4 Amayun Houéry

Pour ma part, ce projet m'a changé des autres projets de programmation auxquels j'avais participé auparavant dans le sens où cette fois j'ai dû faire avec un énorme environnement possédant plein de fonctionnalités pour résoudre les problèmes au lieu de tout faire à partir de zéro. J'ai aussi dû prendre beaucoup plus en compte les problématiques liées à l'ergonomie et au design dans un tel projet. Mes rôles principaux étaient sur les parties des mécaniques de jeu, des interfaces et des menus. Le travail de groupe et la répartition des tâches se sont tout de même révélés être beaucoup plus faciles à mettre en place par rapport à un projet où tout aurait dû être fait de zéro et les croisements des parties et autres collaborations se sont faits de manière plutôt naturelle. La cohésion du groupe et l'engouement global dans le projet étaient d'autant plus fort qu'il concernait des domaines qui nous passionnent tous, comme les jeux vidéo mais aussi, plus spécifiquement, les jeux multijoueur en local type *couch party* et la musique, entre autres. Au final, ce projet était beaucoup moins dans la programmation que ce que je me figurais, et beaucoup plus dans la recherche et prise en main d'un environnement logiciel complexe et complet. Il m'a fallu adapter le mode de pensée et la vision de l'organisation des objets dans un projet Unity qui, une fois qu'ils ont été acquis, permettent de produire des choses bien plus complexes et ce de manière plus efficace.

9 Conclusion

Notre projet, Purest Atom Exam, est donc désormais dans un état qui satisfait nos attentes à l'origine. Ce jeu, qui avait avant tout pour objectif principal de permettre à plusieurs personnes de pouvoir s'amuser au même endroit, sur le même ordinateur, correspond plutôt bien à l'image que nous nous en faisions au début du projet. Bien qu'il n'ait peut-être pas des mécaniques de jeu permettant une compétition rigoureuse ne laissant place qu'à la maîtrise pure, nous avons pu tous expérimenter le divertissement que procurent immédiatement les parties de jeu entre amis. Ainsi, à l'image de *Towerfall Ascension*, Purest Atom Exam repose sur son utilisation originale des orbites et ses concepts pour proposer une expérience de jeu simple et amusante tout en étant nerveuse et dynamique. Nous sortons tous de cette aventure satisfaits et plus expérimentés et considérons donc ce projet comme un succès.