

Parad-OX

Rapport de projet (Soutenance finale)

Constant MALANDA
“Ghakizu”

Cyril RIDEAU
“Pokkihju”

Axel GRUNENBERGER
“PixL”

Table des matières

1 Reprise du cahier des charges	4
1.1 Origine et nature du projet	4
1.2 Etat de l'art	6
1.3 Découpage du projet	7
1.4 Fonctionnalités de notre jeu	12
2 Avancement du projet partie par partie	13
2.1 Game Design	14
2.2 LevelDesign	16
2.3 Modélisation 3D	21
2.4 Level Build	25
2.5 Gameplay	28
2.6 Intelligence Artificielle	33
2.7 Network	37
2.8 Multijoueur	38
2.9 Gestion des collisions	40
2.10 Création des menus	41
2.11 Sound Design	43
2.12 Animations et particules	44
2.13 Création du site internet	45
3 Expériences personnelles	47
3.1 Axel "PixL" GRUNENBERGER	47
3.2 Cyril "Pokkihju" RIDEAU	48
3.3 Constant "Ghakizu" MALANDA	50

Introduction

Un héros sachant voyager dans les rêves ? Un équilibre à maintenir entre monde réel et monde des songes ? Voici comment a commencé notre projet : un scénario un peu fou pour un jeu vidéo d'autant plus loufoque.

Au départ, nous, la *TeamCheall*, comptions un total de quatre membres : Maya Douadi, Axel Grunenberger, Cyril Rideau et Constant Malanda. Cependant, peu après la première soutenance, l'ancienne chef de projet Maya a décidé de quitter *EPITA*, et a donc cessé de travailler sur le projet. Elle était principalement chargée de la modélisation 3D de tous les objets. Cependant, ayant rencontré des problèmes d'importation de son travail sur *Unity*, nous n'avons pas pu utiliser ses modèles. Après son départ, elle a complètement arrêté de travailler avec nous. Elle n'a donc presque rien apporté au projet, si ce n'est son enthousiasme et ses bonnes idées. C'est pourquoi nous n'en parlerons que très peu dans ce rapport.

Pour rappel, notre jeu est un jeu de type action-aventure qui mêle des phases très calmes, le joueur résolvant des énigmes ou recherchant des indices, à des phases de jeu beaucoup plus mouvementées, le joueur se déplaçant dans des niveaux de plateforme et combattant divers ennemis. Chaque niveau est ainsi constitué de trois parties : une phase d'enquête dans le monde réel, une phase de jeu plus active qui diffère selon les différents niveaux, celle-ci se déroulant dans le monde des rêves, et enfin l'affrontement d'un boss, permettant de conclure le niveau en beauté.

Comme précisé dans le cahier des charges, notre projet a été en grande partie réalisé à l'aide du moteur de jeu *Unity 5*, qui nous a permis de créer notre jeu, et du logiciel de modélisation *Blender*, nous permettant de réaliser nos propres modèles en 3D. Cependant, d'autres outils ont également été indispensables pour mener à bien notre projet : par exemple, le service d'hébergement *GitHub* nous a permis de

travailler en équipe sans trop de problèmes, et nous a ainsi fait gagner en productivité.

1 Reprise du cahier des charges

1.1 Origine et nature du projet

Idée de départ

Chaque membre de l'équipe, avant la création du groupe, se faisait une idée personnelle du projet sur lequel il voulait travailler. Certains se plaisaient à imaginer un jeu rétro, aux dynamismes et couleurs épileptiques quand d'autres y voyaient du rythme ou encore de la stratégie. Cependant, le point commun à toutes nos idées était le jeu, qui devait être original, et non un sujet déjà traité des centaines de fois auparavant.

Notre rencontre nous a ainsi amené à une idée manquant de clareté et de précision, continuellement peaufinée selon l'apport de chacun ainsi que les différents points de vue que cela implique. Cette mouvance collective a eu pour moteur un objectif commun, résumé en un mot : *Originalité*.

Scénario

Euren, petite ville reculée, est victime d'une série de vols inexpliqués. Ces crimes sont exécutés avec une aisance déconcertante qui rend les habitants anxieux et soupçonneux. Le joueur incarne une jeune personne de 23 ans, vivant seule avec son frère jumeau Daniel. Tous deux survivent grâce aux vols qu'ils perpètrent, en s'a aidant d'une curieuse capacité : celle de voyager dans les rêves des autres et, de surcroît, de les manipuler. Cet avantage leur permet d'obtenir de nombreux coffres et codes sur leur chemin.

Néanmoins, lors d'une Chimerne (intrusion volontaire dans un rêve), l'opération tourne mal en raison d'un corps étranger sommeillant dans l'inconscient de la victime des jumeaux. Réveillée, cette entité tue Daniel et s'attaque à notre héros, qui parvient à s'enfuir tant bien que mal. Se retrouvant seul dans un monde où il ne peut révéler sa vraie nature, il découvre alors l'existence d'un parasite, à l'image du meurtrier de son frère, qui inhibe l'inconscient de l'individu parasité. Lui-même touché par ce qui semble être une épidémie, notre héros est entraîné dans une lutte contre ce mal, qui ne semble pas être apparu par hasard.

Gameplay

Comme indiqué dans l'introduction, le niveau est composé de trois phases, une ayant lieu dans le monde réel et les deux autres dans le monde chimérique. Le Gameplay change selon la phase dans laquelle le joueur se trouve de telle manière que :

Monde Réel : Le joueur se retrouve dans une petite ville, de type *Open World* (Scène d'un jeu où l'on peut se déplacer librement, sans suivre un chemin prédefini) minime, où il sera amené à interagir avec les objets et les PNJ (Personnages Non Jouables), afin de récolter des indices. Ceux-ci le feront réfléchir au scénario, et lui donneront envie de résoudre le mystère. Il devra, pour ce faire, être minutieux et examiner l'environnement dans lequel il évolue.

Monde fictif : Le joueur est plongé dans un univers où le gameplay est plus dynamique. Son principal but est de parvenir jusqu'à la salle du boss, surmontant des obstacles, énigmes et ennemis de toutes sortes. Cette deuxième partie, qui est le niveau à proprement parler, demande plus au joueur de montrer ses talents ou ses réflexes.

Boss : Après avoir parcouru le niveau dans son intégralité, le joueur parvient à un boss, qu'il doit combattre dans une lutte acharnée. Les boss ne sont pas que de simples ennemis, mais ils ont chacun une mé-

canique particulière, que le joueur se doit de comprendre afin d'espérer pouvoir les vaincre.

Multijoueur : Endormi, le joueur a accès à un ou plusieurs niveau(x) bonus, jouable(s) en multijoueur via internet. Ces niveaux sont facultatifs : le joueur n'est pas obligé de les terminer pour pouvoir continuer l'aventure. Les deux personnages doivent, dans un univers semblable à celui du monde fictif, surmonter des obstacles nécessitant une certaine coopération afin de trouver le boss dans le but de le combattre.

En résumé

Ainsi, notre jeu a pour but de découvrir les secrets que cachent ces fameux parasites, en s'aidant pour cela des indices récoltés au fur et à mesure des niveaux, pour finalement combattre différents boss dans des combats mêlant stratégie et créativité de la part du joueur face à son adversaire. De plus, **Parad-OX** propose des parties se jouant à deux, afin de satisfaire davantage les utilisateurs.

Nous espérons donc pouvoir combler le joueur en l'amenant à vivre une histoire passionnante, au travers d'aventures, de combats et de réflexion, tout en ayant la possibilité de s'amuser à deux, entre amis.

1.2 Etat de l'art

Inspirations principales du scénario

Notre jeu **Parad-OX** a pour inspiration principale le film d'animation japonais *Paprika* : ce film se dresse dans un monde alternatif où un appareil, nommé DC Mini, permet de visualiser les rêves du porteur. Cette invention, à la base créée pour aider les patients dans leurs séances thérapeutiques, est volée par un individu non identifié.

De multiples cas de démence sont notés, dus au fait que les personnages sont plongés dans un rêve éveillé, les rendant dangereux pour la société ainsi que pour eux-mêmes. Le personnage principal, Paprika, personnalité chimérique d'une scientifique ayant aidé à la conception de l'objet, tente alors de découvrir l'identité du criminel et les objectifs qui l'animent.

Inspirations principales du Gameplay

La particularité de notre jeu est que celui-ci regroupe à la fois un gameplay d'action-aventure, mais également des phases de réflexion, plus posées. Grâce aux nombreuses variations de rythme dans le gameplay, le joueur ne se lasse pas.

L'un des premiers jeux de ce type est *Adventure*, de Warren Robinett, sorti sur Atari en 1979. Il mélange des combats contre des dragons, mais également de la réflexion, comme des passages dans des labyrinthes par exemple.

Ainsi, pour le Gameplay de notre jeu, nous nous sommes inspirés de différents jeux d'action-aventure, comme la fameuse série *The Legend Of Zelda* où le héros doit résoudre des énigmes tout en se battant contre diverses créatures pour arriver à la fin du jeu, mais aussi les franchises *Assassin's Creed* et *Tomb Raider*.

1.3 Découpage du projet

Répartition des tâches

Lors du rendu de notre cahier des charges, Maya était toujours présente parmi nous. Nous l'avions donc intégrée dans la répartition des tâches à réaliser.

Maya, qui était la plus motivée pour devenir chef de projet, a été élue à l'unanimité. Elle avait également pris la responsabilité de créer les modèles 3D, car c'est elle qui a le plus de talent artistique, ainsi que de gérer les collisions et de construire les niveaux directement sur *Unity*. Maitrisant un peu le HTML, elle avait également voulu être responsable du site internet.

Axel, plutôt doué en programmation, a pris en charge une partie du Gameplay, que nous avions scindé en deux responsables, cette partie étant de loin la plus imposante du projet. Il a également voulu s'occuper de l'intelligence artificielle, une partie qui lui plaisait beaucoup. Enfin, ayant envie de travailler à l'intérieur même de *Unity* et de développer sa créativité, il a également décidé de s'occuper de tous les menus du jeu.

Constant, de son côté, avait envie de s'essayer au Network. C'est pourquoi son rôle principal était la création et la maintenance du Network, ainsi que du niveau en multijoueur. Passionné de musique, nous lui avons également donné la responsabilité des sons et bandes sonores de notre jeu.

Enfin, Cyril, qui adore imaginer de nombreux jeux de rôles, s'est tout de suite vu gérant du Game Design et du Level Design. Il a donc eu pour mission de définir ce à quoi ressemblerait notre jeu, tout en respectant l'avis des autres membres du groupe. Il est également le deuxième responsable du GamePlay car, ayant déjà imaginé ce que le jeu serait, il était le plus à même de l'implémenter.

Chaque tâche à réaliser était donc dotée d'un responsable, mais également d'un suppléant. Le rôle de ce dernier était d'aider le responsable à réaliser ses tâches en cas de besoin.

Au commencement du projet, la répartition des tâches¹ ressemblait donc à celle-ci :

Tâches	Maya	Axel	Constant	Cyril
Modélisation 3D	R	S	-	-
Gestion des collisions	R	-	S	-
Level design	-	S	-	R
Level build	R	-	-	S
Sound design et Musique	S	-	R	-
Game design et gameplay	-	R	S	R
Création du site internet	R	-	-	S
Création menus	S	R	-	-
Multijoueur	-	-	R	S
Network	-	-	R	S
IA	-	R	S	-

R : Responsable.

S : Suppléant

Cependant, en raison du départ de notre chef de projet, nous avons dû repenser l'intégralité de notre cahier des charges dès la deuxième période, et parfois réduire nos ambitions, ceci afin d'éviter d'encourir le risque de ne pas finir le projet dans les temps.

Tout d'abord, il nous a paru impératif de désigner un nouveau chef de projet. Dans la mesure où il était le membre du groupe le plus investi, et s'était de lui-même proposé pour reprendre le rôle, nous avons choisi à l'unanimité de désigner Axel comme nouveau chef de projet. Sur le fond, le concept du jeu n'a absolument pas changé : l'âme de **Parad-OX** restera la même, avec son univers bien à lui. Cependant,

1. Pour information, l'explication de chacune des tâches se trouve dans la deuxième partie de ce rapport.

nous avons dû apporter de nombreuses modifications à la forme que celui-ci prendra, tant par sa qualité graphique que par les détails du scénario, Maya étant la personne dotée de la plus grande imagination et du meilleur sens artistique d'entre nous.

Les tâches dont était principalement chargée Maya étaient les suivantes :

- Modélisation 3D : Axel étant le suppléant pour cette tâche, nous avons décidé de le placer en tant que responsable, en nommant Constant comme suppléant, ce dernier ayant un sens artistique plus affûté que Cyril.
- Création du site internet : Constant ayant déjà un peu programmé en HTML, nous avons décidé de lui attribuer cette mission. En effet, nous ne pouvions prendre le temps d'apprendre un langage intégralement : cela nous aurait fait perdre un temps considérable. Cyril reste ainsi suppléant, désireux d'apprendre à créer un site par lui-même.
- Level Build : Cyril et Axel étant responsables du Level Design, il nous a paru logique qu'ils soient également responsables du Level Build : en effet, ayant déjà travaillé sur l'aspect qu'ils voulaient donner aux différents niveaux, ils étaient les plus à même de les mettre sur pieds, y ayant déjà réfléchi au préalable.
- Collisions : Pour les mêmes raisons que précédemment, Cyril et Axel ont été nommés nouveaux responsables des collisions. Bien que Constant était désigné comme suppléant pour cette tâche, nous pensions que lui rajouter la création du site était déjà un travail conséquent, et ne voulions par conséquent pas lui attribuer une tâche supplémentaire.

Ainsi, l'intégralité du cahier des charges et de la répartition des tâches a dû être modifiée. En voici la nouvelle version :

Tâches	Axel	Constant	Cyril
Modélisation 3D	R	S	-
Gestion des collisions	S	-	R
Level design	S	-	R
Level build	S	-	R
Sound design et Musique	-	R	S
Game design et gameplay	R	S	R
Création du site internet	-	R	S
Création menus	R	-	-
Multijoueur	-	R	S
Network	-	R	S
IA	R	S	-

Répartition du travail au cours des périodes

Afin de terminer le projet dans les temps, il était également important de définir quels objectifs devaient être atteints à quel moment du semestre. Voici donc le tableau récapitulant l'avancement que nous voulions atteindre durant chaque période et pour chaque item.

Tâches	Période 1	Période 2	Période 3
Modélisation 3D	30%	90%	100%
Gestion des collisions	50%	80%	100%
Level design	50%	80%	100%
Level build	20%	70%	100%
Sound design et Musique	0%	60%	100%
Game design et gameplay	40%	80%	100%
Création du site internet	10%	50%	100%
Création menus	10%	30%	100%
Multijoueur	20%	60%	100%
Network	30%	70%	100%
IA	40%	90%	100%

1.4 Fonctionnalités de notre jeu

Objectifs

Afin de parvenir aux objectifs que nous nous étions fixés concernant le contenu de ce projet, nous devions remplir les conditions suivantes afin de garantir son bon déroulement :

- Un menu au lancement du jeu.
- Un menu lorsque le joueur désire mettre le jeu en pause.
- Une interface complète du jeu pour que le joueur puisse connaître le nombre d'indices en sa possession, ses équipements, etc.
- Des collisions réalistes.
- Des animations permettant d'indiquer les actions des PNJ, etc.
- Au moins un niveau multijoueur.
- Au moins 3 niveaux solos différents.
- Plusieurs boss, ayant des différences dans le gameplay.

Bonus

Dans le cas où les éléments de base de notre projet seraient terminés, nous avions déjà plusieurs idées que nous comptions rajouter au jeu si le temps restant avant de rendre le projet nous le permettait. Parmi ces idées :

- Réaliser des cinématiques pour la cohérence de certains événements du scénario.
- Réaliser ou choisir des bandes sons en cohérence avec le jeu et son univers.
- Réaliser un niveau qui servirait de Tutoriel.
- Ajouter un animal de compagnie, qui nous suivrait partout.
- Créer plus d'un niveau multijoueur.
- Créer plus de 3 niveaux solos.

Technologie et méthodologie

Unity est le moteur graphique que nous avons utilisé, avec *Visual Studio* ou *Rider*, pour la programmation des scripts – entièrement en *C#*. Nous avons également utilisé *Blender* pour créer nos propres modèles et les animations qui les accompagnent. Enfin, nous avions pensé à utiliser *Fl Studio* pour réaliser la bande sonore, si le temps nous le permettait.

Nous avions à notre disposition au moins un ordinateur chacun, nous permettant de bonnes conditions de travail. Nous possédions également une assez bonne connexion nous permettant de participer activement à la réalisation de notre projet, à toute heure, grâce à la plate-forme *Github*.

Opérationnel

Sachant que les logiciels que nous avons utilisé sont gratuits ou nous offrent une licence étudiante, le coût du projet en a été amoindri. De plus, considérant le fait de devoir créer un site internet et un serveur pour notre jeu, nous avons dû prévoir de louer un VPS et un nom de domaine afin de les héberger.

2 Avancement du projet partie par partie

Pour réaliser notre projet, nous l'avons découpé en plusieurs parties, que nous nous devions d'accomplir pour un rendu global réussi. Pour chaque soutenance, nous nous donnions des objectifs à atteindre pour chacun de ces items, afin d'avoir des buts concrets, et pour éviter de ne trop nous éparpiller sans savoir exactement où nous allions. Ainsi, nous décrirons dans les prochaines pages les différents avancements de chacun de ces points au fil des soutenances. Le projet peut ainsi être découpé en trois périodes : la première se situe avant

la première soutenance, la deuxième est celle entre la première et la deuxième soutenance, et enfin la troisième et dernière période est celle précédant le rendu final du projet.

2.1 Game Design

Le Game Design est la mise au point de toutes les règles, mécaniques et autres éléments qui constituent le jeu. Il doit être fait avant de commencer quoi que ce soit d'autre : à la fin de cette partie, le but est d'avoir une idée la plus précise possible de ce à quoi ressemblera le futur jeu.

Avant la première soutenance, le Game Design avait mis l'emphase sur le personnage principal et les attributs et armes qu'il devait posséder, le but étant d'avoir un personnage complet, permettant de varier le Gameplay des niveaux sans problème. Ainsi, nous avions déjà décidé qu'il devait, à l'instar de nombreux personnages d'autres jeux, être capable de sauter, se déplacer, attaquer, ou même lancer des sorts. Pour cela, il devait donc posséder des attributs tels que la Vie, lui permettant de se battre, l'Endurance (ou Stamina), lui permettant de courir ou encore le Mana, lui permettant de lancer des sorts. Nous avions également réfléchi à la manière dont ceux-ci seraient implementés dans le jeu. De plus, nous avions déjà décidé de lui donner différentes armes tels qu'un Katana, un couteau, ou encore ses poings. Ces armes ont donc été désignées une par une, et nous avons réfléchi aux effets et aux statistiques qu'elles pourraient avoir chacune séparément. Nous savions à cette époque que nous voulions rajouter des sorts, mais n'avions pas encore eu l'occasion de nous y intéresser plus en détails.

Suite à cela, nous avons décidé que, pour accéder à ses objets, le joueur aurait deux raccourcis pour ses armes et deux autres pour ses sorts. Le clavier numérique semblait être la meilleure des solutions pour utiliser ces raccourcis. De plus, il nous fallait stocker tous les autres objets, que ce soit les indices, ou bien même les armes et

sorts supplémentaires ne pouvant rentrer dans les raccourcis, ceux-ci étant déjà occupés. C'est pourquoi il fallait que notre joueur ait un inventaire. Après mûre réflexion, nous nous sommes accordés sur la manière dont celui-ci serait implémenté. En appuyant sur une touche du clavier, l'inventaire apparaît et le joueur peut assigner ses armes aux raccourcis, ou bien seulement observer les objets qui lui appartiennent. Par défaut, nous pensons que la touche *tab* du clavier est un bon choix car située près des touches de direction habituelles. D'autres éléments devaient bien sûr être implémentés, mais nous n'y avions pas réfléchi suffisamment et devions nous y repencher dans les semaines à venir, tels que par exemple les différents sorts, un système d'argent, ou encore différentes vitesses qu'il faudrait gérer (lorsque notre personnage marche, court, et éventuellement s'accroupit).

Pour la deuxième soutenance, le Game Design a tout d'abord insisté sur les différents indices qui devaient être implémentés pour mener le joueur aux différents niveaux. Ainsi, nous avons réfléchi sur les liens que nous pourrions mettre en place entre le scénario d'une part, et les indices ainsi que les niveaux d'autre part. C'est ainsi que nous avons décidé que chaque boss aurait un lien avec une phobie, ces phobies étant en réalité responsables de l'apparition des parasites dans les rêves des personnages. Nous avons ensuite rapproché cette phobie avec les particularités dont ferait preuve chaque boss. Par exemple, le premier boss, qui représente le vertige, ne sera atteignable que par les hauteurs. Les indices ramassés dans le monde réel peuvent permettre de comprendre plus facilement la façon dont le boss fonctionne, et donc de le vaincre avec moins de difficultés.

Ensuite, nous avons dû réfléchir aux statistiques des différentes armes et ennemis, qui devaient être rééquilibrés afin de maintenir son équilibrage, et de parvenir à quelque chose de beaucoup plus réaliste. Cependant, malgré beaucoup de temps passé à y réfléchir, elles n'étaient pas encore parfaites, et du travail restait à effectuer avec des tests directement dans le jeu. De plus, Nous avons designé les différents sorts que le joueur pourrait lancer. Ils sont au nombre de six, et ont

chacun un fonctionnement très différent, permettant une variété de Gameplay plus vaste et par conséquent plus intéressante. Ci-dessous, une liste des sorts qui devaient être créés :

- Freeze : Ce sort gèle l'adversaire et l'empêche d'effectuer une quelconque action pendant un certain laps de temps.
- Heal : Ce sort soigne le personnage d'un montant prédéfini de points de vie en échange de son mana.
- EarthSpike : Ce sort inflige des dégâts à l'adversaire qui le subit.
- Fireball : Ce sort inflige des dégâts dans une zone devant le joueur.
- Airwall : Ce sort protège le lanceur contre tous les projectiles.
- Flash : Ce sort désoriente l'adversaire pendant un temps donné.

Pour la troisième et dernière soutenance, le Game Design était presque intégralement terminé. Il ne restait ainsi plus qu'à rééquilibrer certaines caractéristiques, tel que les points de vie, les dégâts des armes, la durée des sorts, etc., ceci dans le but de créer un jeu à la fois amusant et intéressant, où chaque item a sa place, et non à l'inverse où une arme trop puissante permettrait de finir tout le jeu beaucoup trop facilement.

2.2 LevelDesign

Le level Design est la partie visant à inventer les différents niveaux du jeu. Pour cela, nous avons réalisé de nombreux plans sur papier, que nous avons ensuite améliorés, jusqu'à ce que le résultat nous convienne à tous.

Le Level Design a été une véritable difficulté puisque différents types de niveaux n'ayant rien à voir les uns avec les autres ont été créés dans notre jeu. En effet, on peut y trouver un niveau de plateformes, un autre de labyrinthes et enfin un dernier d'énigmes avant de pouvoir affronter d'innombrables ennemis sur la route du dernier boss.

Pour la première soutenance, le Level Design a été relativement lent à démarrer. En effet, nous avons dû observer et apprendre à partir de nombreux jeux, en les étudiant, les codes à respecter pour obtenir un niveau fonctionnel, ceci dans le but de pouvoir créer notre propre style de niveaux et décider de ce que nous pourrions faire. Le premier niveau étant un niveau de plateformes, il a fallu déterminer les meilleurs endroits pour positionner murs et obstacles. Dans une première partie, nous avons fait un plan approximatif de ce que nous semblait être un bon niveau (**Cf. annexe 1**). Cependant, en le construisant, nous nous sommes vite rendus compte que certaines parties étaient trop simples, tandis que d'autres étaient trop complexes, voire infaisables. Il nous a donc fallu tester le niveau en construction à de maintes reprises, pour pouvoir obtenir quelque chose de constant en termes de difficulté, et redesigner chaque partie dès que nous faisions face à un problème. C'est pourquoi le design de ce niveau a demandé beaucoup de temps et de patience, de par son effet répétitif.

Suite à ce niveau, il était nécessaire de placer un boss afin d'amener un petit peu de relief dans le paysage des plateformes. Afin de rester cohérent vis-à-vis du scénario, ce boss est Centaurus (**Cf. annexe 2**). Sa particularité est qu'il ne peut recevoir des dégâts que d'une attaque venant d'une zone surélevée, ayant le vertige. Ainsi, il sera vital pour le joueur de comprendre cette stratégie et de trouver un moyen de prendre de la hauteur, tout en restant à proximité de son adversaire. Ce boss est lié à la personnalité et aux phobies du personnage dans lequel il vit. Si le joueur ne prête pas attention à l'histoire, il pourrait l'apprendre à un prix non négligeable.

seconde période a demandé plus de travail en amont au niveau du Level Design, car il a fallu créer le niveau multijoueur qui est un niveau de labyrinthes. Son fonctionnement est le suivant : deux joueurs se retrouvent dans une même scène, mais ils ne peuvent pas se voir. En effet, l'un se trouve à l'intérieur même d'un labyrinthe tandis que l'autre se retrouve plusieurs mètres en hauteur, de nombreux leviers ainsi que de multiples plateformes auxquelles il n'a pas encore accès devant lui. En fait, les joueurs ont tous les deux des portes les empêchant d'avancer, ainsi que des leviers ne déverrouillant que les portes de l'autre joueur. Leur but est de coopérer afin que celui se trouvant dans le labyrinthe parvienne à la sortie de celui-ci, affrontant de nombreux pièges et ennemis, et ramassant de nombreux trésors sur son chemin. Cependant, un échange des joueurs est possible, ces derniers pouvant échanger leurs places dans la scène plusieurs fois, ceci dans le but d'équilibrer les Gameplays, et d'éviter qu'un joueur ne s'ennuie.

Il a donc fallu décider de l'emplacement des salles, des différents couloirs et même des pièges et portes bloquant l'avancement du joueur à l'intérieur du labyrinthe. De plus, la création des phases de plateformes pour le second joueur a pris un peu de temps, car il fallait les rendre à la fois intéressantes et suffisamment courtes pour ne pas casser le rythme du jeu. Enfin, Le placement de passages secrets et autres trésors a également requis de la réflexion afin de les placer de manière la plus logique qui soit. Toutes ces étapes se sont faites en grande partie sur papier, afin d'avoir une vue d'ensemble sur ce que nous produisions (**Cf. annexe 3**).

Le boss de ce niveau est encore plus particulier que le précédent. Il s'agit d'un hypogriffe, claustrophobe, et enfermé dans un labyrinthe, le pauvre. Le joueur devra exploiter cette phobie afin de rendre le combat plus facile, en enfermant l'ennemi dans des cages situées de part et d'autre de la salle. En effet, celui-ci sera paralysé par la peur et ne pourra alors plus se déplacer pendant un certain laps de temps, avant de réussir à s'échapper, détruisant la cage. Par rapport au scénario, la peur de ce boss s'explique par la claustrophobie du personnage dont il

est issu. Il pourra donc être très difficile à vaincre, car possédant des dégâts élevés. Cependant, avec un peu de talent, il pourra être vaincu grâce à l'absence de dégâts qu'il inflige lorsqu'immobilisé.

En plus du niveau de labyrinthe, le niveau d'éénigmes a également été préparé avant la deuxième soutenance (**Cf. annexe 4**). Celui-ci consiste en la résolution successive d'éénigmes, avec une difficulté croissante. La première partie de ce niveau requiert la résolution de 9 éénigmes. En cas de mauvaise réponse, le joueur sera transporté dans une salle "piège" dans laquelle il devra soit répondre à une autre éénigme soit traverser une phase de labyrinthe ou une phase de plateformes. La résolution de ces neufs éénigmes donnera accès aux trois suivantes, qui elles-même donneront accès à la dernière éénigme. Les trois appartenant à la deuxième partie ne sont pas de simples questions mais requièrent de l'observation et de l'attention envers l'environnement qui sera présenté. En effet, ce sont des éléments de la mise en scène proposée au joueur qui lui permettront de répondre à la question de manière efficace. En cas de mauvaise réponse, le joueur devra faire face à la colère des personnages du décor qui feront apparaître des ennemis. Ces mises en scène peuvent décrire une situation de tous les jours dans laquelle il faudra faire une déduction ou encore une situation inhabituelle comme un crime. Dans ce dernier cas, le joueur devra mener l'enquête pour trouver le coupable.

Avant la dernière soutenance, nous avons dû finaliser le design du niveau d'éénigmes. En effet, la dernière salle restait un mystère pour nous car il nous fallait trouver quelque chose d'original, mais restant cohérent avec le reste du niveau. Nous avons finalement décidé de faire un mélange des premières et secondes salles. En effet, le personnage devra répondre à une série de 3 éénigmes courtes, puis fera face à une scène, liée à ces éénigmes, dans laquelle il devra trouver une clé à l'aide des déductions qu'il aura faites. Les talents d'observation du joueur seront une fois de plus mis à rude épreuve, et celui-ci devra rester attentif s'il veut réussir. La ligne directrice de ce niveau s'étant confirmée, il était temps de choisir ces éénigmes. Ainsi, plusieurs ont été retenues.

Pour la première partie, les énigmes choisies sont plus des questions requérant l'attention du joueur, mais également une logique imparable afin de ne pas tomber dans certains pièges. Elles jouent donc sur les mots et les concepts. Plusieurs réponses seront à chaque fois présentées au joueur afin de l'aider dans sa réflexion. Bien que cela puisse paraître une aide trop importante, nous voulions éviter qu'il ne se retrouve bloqué trop longtemps au même endroit. Attention cependant, il ne faut pas trop mal répondre, car les malus encourus pourraient être fatals.

Le boss de ce niveau d'énigmes est, à l'image du mythe d'Oedipe, un Sphynx. Il commencera par nous poser trois questions, puis nous attaquera tel un ennemi classique. Les réponses que le joueur donnera affecteront ses statistiques et sa puissance en combat. Pour rajouter une touche d'humour, nous avons pensé que créer des questions en lien avec l'EPITA était un bon clin d'oeil pour les membres du jury, le jeu n'ayant de toutes manières pas pour but d'être distribué. La dernière question restera tout de même la question très connue présente dans le mythe d'Oedipe, "Qu'est-ce qui a 4 pattes le matin, 2 pattes le midi et trois le soir ?". Il faudra bien sûr répondre 42, réponse absolument absurde mais bien plus comique, la réelle réponse étant bien trop connue.

Le lien entre ce boss et le personnage qu'il occupe est sa psychologie : en effet, il s'agit d'un historien paranoïaque admirateur de Sherlock Holmes et de la mythologie grecque. Il apprécie particulièrement le mythe d'Oedipe, pour des raisons familiales, sa mère était également sa grand-mère (ce qui se rapproche plus ou moins du mythe d'Oedipe).

Enfin, Le dernier niveau ne sera en réalité, comme nous l'avions prévu, que le monde réel revisité avec de nombreux ennemis et un gameplay centré sur la fuite. Le joueur devra sauter pour passer des parties de plateforme et s'enfuir à travers des rues presque labyrinthiques. Certaines parties du terrain seront supprimées et transformées en vide, tandis que d'autres seront remplacées par des objets aléatoires et décousus. Le personnage devra parvenir à rentrer chez lui

avant d'affronter le boss final, qui représentera la fin de notre jeu, car il est en réalité la source du mal qui atteint les rêves des gens. Pour cela, il devra faire usage de toutes les armes et autres objets qu'il aura récupérés. Ce boss sera en quelque sorte un mélange des trois boss précédents dont il adoptera les faiblesses mais également les forces.

2.3 Modélisation 3D

La Modélisation 3D est la partie qui regroupe tout ce qui traite de la modélisation des objets, et de l'esthétisme de notre jeu. Ils sont pour la plupart réalisés à l'aide du logiciel *Blender*, qui permet une réalisation plutôt simple et intuitive des modèles.

Au départ, la personne en charge de nos modèles était Maya. Elle s'est d'ailleurs très vite prise au jeu, et a tout de suite tenté de prendre en main le logiciel *Blender*. Axel, en tant que suppléant, s'est aussi instruit sur la façon dont fonctionne le logiciel.

La première période a ainsi été celle durant laquelle le travail sur la modélisation a été le moins visible : il a fallu regarder de nombreux tutoriels sur différents sites et des contenus vidéos-ludiques explicatifs, afin de savoir comment réaliser de manière efficace et réussie tous nos modèles.

Ceci étant fait, Maya a commencé à réaliser différents modèles constituant les éléments principaux habillant le décor du monde réel. En effet, nous avons privilégié ce monde en début de projet car il représente la passerelle entre les niveaux, et c'est par conséquent la scène où l'utilisateur passera la plus grande partie de son temps, à la recherche de divers indices par exemple. Au moment de la première soutenance, nous n'avions alors qu'un bar, un immeuble, une maison, et également un petit kiosque à journaux (**Cf. annexe 5**), mais ceux-ci n'étaient pas implémentés dans le jeu car nous faisions face à un problème conséquent : l'importation de ces modèles sur *Unity* ne pre-

nait pas en compte les textures. Ainsi, tous nos objets se retrouvaient blancs.

C'est ainsi que nous avons présenté lors de la première soutenance une ville constituée exclusivement de cubes blancs, totalement temporaires, dans le but de montrer que notre jeu était fonctionnel, bien que très peu esthétique (**Cf. annexe 6**).

De plus, nous avions décidé dès le début que nous réaliserions le personnage en dernier, car notre jeu étant en vue à la première personne, les seuls et uniques moments où nous le voyons se situent dans le niveau en multijoueur. Nous voulions également prendre le temps de nous documenter davantage sur les armatures, outil intégré dans *Blender* très utile pour faciliter la création des squelettes des personnages et pour permettre des animations fluides.

Enfin, afin de tester les scènes de combat et l'efficacité de notre intelligence artificielle, nous avons réalisé très rapidement un Boss affectueusement surnommé "Mr Centaurus le Moche". Ce personnage ayant vite trouvé une place sentimentale importante à nos yeux, tant par son originalité que par son histoire, nous avons décidé de le garder en tant que boss du premier niveau, et avons également décidé que tous les boss seraient de la même qualité graphique, ceci rajoutant une petite touche d'humour dans le jeu.

Durant la deuxième période, comme énoncé plus tôt, Maya étant partie, c'est Axel qui a repris le rôle de responsable de la modélisation. Malgré de nombreux efforts pour tenter de récupérer le travail réalisé durant la première période, nous n'avons pas réussi à sauver ce qui avait déjà été fait : les modèles ne pouvaient être texturés. Cependant, ayant décidé de réaliser nos modèles par nous-même, nous n'avons pas eu recours aux assets de Unity. Nous sommes ainsi restés motivés et avons donc dû recommencer tous les modèles, en adoptant une nouvelle méthode : les modèles n'auraient à partir de ce jour que des couleurs unies. Nous pouvons penser qu'il s'agit d'une solution de

facilité, mais en y réfléchissant et après avoir essayé le rendu directement sur *Unity*, nous en sommes arrivés à la conclusion que cela donnait un style à part à notre jeu, une sorte "d'âme" qui lui est bien spécifique, et cela nous a beaucoup plu.

Nous avons donc une fois de plus considéré que l'esthétisme du monde réel était prioritaire par rapport à celui des autres niveaux, car le joueur y passe la plupart de son temps. D'autre part, le monde doit apparaître le plus réaliste possible, de par sa définition. Les niveaux du monde des rêves resteront assez simples, avec des couleurs unies et un esthétisme très abstrait. Cela leur donnera un certain style, et le joueur saura, quoi qu'il arrive, dans quel monde il se trouve.

L'accent a d'abord été mis sur les objets les plus volumineux. La plupart des modèles des bâtiments ont été réalisés avant la seconde soutenance. Tout d'abord, les bâtiments résidentiels ont tous une base commune, mais diffèrent tant par leurs coloris que par leurs formes, certains ayant des côtés recoupés, pour mieux se fondre dans l'ensemble de la scène. Les bars et les restaurants sont en fait placés au rez-de-chausée d'un immeuble commun, recoupés, et redesignés de manière à réellement ressembler à un restaurant (**Cf. annexe 7**). De plus, un supermarché a été réalisé, avec un parking et quelques voitures, afin d'éviter que tous les décors ne se ressemblent (**Cf. annexe 8**). Des bancs et des arbres ont également été intégrés à l'intérieur d'un parc pour apporter un peu de verdure, sans quoi la ville était plutôt triste. Une zone de marché était mise en évidence, mais pas terminée le jour de la deuxième soutenance : les stands étaient donc remplacés par de simples tables. Enfin, les armes de notre personnage étaient temporairement de simples objets créés directement sur *Unity*, et non colorés.

Une fois de plus, nous avons mis à l'écart le design de tous les personnages. Un squelette avait été commencé, mais il n'était pas suffisant pour une démonstration, c'est pourquoi nous avons laissé notre capsule, objet de base de *Unity*, pour représenter notre joueur. Réaliser un personnage humanoïde ne relève pas de la même difficulté que

de créer un bâtiment. C'est pourquoi Maya aurait été bien plus qualifiée que nous pour le réaliser. Cependant, même si sa qualité était moindre que ce qu'elle aurait pu être, nous voulions tout de même réaliser notre propre modèle.

Avant la dernière soutenance, différents stands ont été créés et ajoutés à la place des tables citées auparavant. Ils ont tous la même base, mais contiennent différents aliments modélisés assez rapidement, ceux-ci relevant du détail. Nous avons par exemple des stands de fruits, de légumes, etc. (**Cf. annexe 9**). De plus, nous avons très vite réalisé les armes des différents personnages, et les avons ensuite mis à la place des anciennes armes temporaires qui étaient déjà présentes.

Un personnage a finalement été créé (**Cf. annexe 10**). Nous avons décidé, de manière totalement arbitraire, que ses mains, ses jambes, et sa tête seraient détachés. Cela lui donne un effet très particulier, et nous rappelle également qu'il a été intégralement conçu par nos soins. Le personnage est donc découpé en six parties différentes, ce qui facilite grandement les animations. Il est ainsi devenu la base de tous les personnages qui existent dans notre jeu, que ce soit le personnage principal, un ennemi, ou un PNJ. Les seules différences notables qu'il pourrait y avoir entre eux seraient les coloris, les différentes coupes de cheveux, ou encore les différents vêtements. Enfin, nous avons décidé que le joueur pourrait choisir d'incarner un homme ou une femme, en prenant une option dès le début du jeu.

Aujourd'hui, notre jeu est tel que nous l'avions espéré esthétiquement parlant. Il reflète parfaitement l'âme que nous voulions lui donner, et nous sommes fiers de n'avoir utilisé aucun asset, bien que cela représentait une charge de travail supplémentaire.

2.4 Level Build

Le level build est le processus mettant en application les décisions prises lors du Level Design. On place ainsi sur une scène de *Unity* les différents modèles réalisés au préalable, de façon à ce qu'ils respectent les plans que nous avons décidés d'appliquer.

Lors de la première période, le niveau que nous avions construit était celui de plateformes (**Cf. annexe 11**). Celui-ci, comme indiqué dans la partie dédiée au Level Design, a été long et fastidieux à élaborer, car nécessitant de très nombreuses modifications pour le rendre suffisamment dynamique et complexe, sans pour autant le rendre impossible. Nous avons donc commencé par créer des plateformes spéciales, que nous avons enregistrées sous forme de préfabriqués, n'ayant plus qu'à les replacer sur la scène par la suite. En effet, leur nombre étant extrêmement conséquent, les recréer une par une aurait nécessité beaucoup trop de temps et aurait été une perte d'énergie inutile. De plus, afin que certaines plateformes puissent bouger comme désiré, nous avons dû créer de petits scripts permettant à celles-ci de bouger seules. Nous avons également ajouté des points de sauvegarde à des endroits stratégiques permettant d'éviter au joueur de recommencer les même sauts en permanence.

Pour la deuxième soutenance, notre objectif était de construire le niveau entier du labyrinthe (**Cf. annexe 12**). Cela s'est avéré plus fastidieux encore que de bâtir le niveau de plateforme, et ce pour plusieurs raisons :

- La taille du niveau était beaucoup plus étendue, celui-ci étant réalisé pour deux joueurs.
- Le nombre de murs et d'obstacles à placer était très conséquent. Nous avons donc dû les placer un par un, en faisant attention à ce qu'ils ne gênent pas le bon déroulement du jeu.

- La mise en application du Gameplay du level en multijoueur a également dû être prise en compte. La création de tous les objets particuliers, tels que les boutons permettant d'avancer dans le niveau, ou encore les phases de plateforme placées pour le joueur situé en dehors du labyrinthe, a pris un certain temps. En effet, la mise en compatibilité avec le mode multijoueur n'était pas aussi simple que nous l'avions imaginé.
- Il a été nécessaire de séparer les différentes parties du labyrinthe afin de savoir quand échanger les positions des deux joueurs, et de cacher les différents trésors et pièges de manière optimale.

Afin d'être prêt pour la dernière soutenance, il fallait avoir finalisé l'ensemble des niveaux. Nous nous y sommes donc attelés, en nous occupant en priorité du dernier niveau qui n'était pas encore implémenté. Pour ce faire, il a fallu créer les neuf salles d'énigmes, ainsi que celles dans lesquelles le joueur est transporté lorsqu'il donne une mauvaise réponse. Nous avons décidé de placer ces dernières au dessus de la salle dans laquelle se trouve l'éénigme correspondante. Ainsi, les neuf premières sont organisées autour de celle où se trouve l'éénigme principale, au rez-de-chaussée. En cas de mauvaise réponse, le joueur est transporté à l'étage supérieur, afin d'y passer une épreuve supplémentaire, telle qu'une épreuve de labyrinthe, de plateforme ou une seconde énigme. Dans ce dernier cas, chaque mauvaise réponse fera apparaître des ennemis qu'il devra vaincre avant de pouvoir s'enfuir.

Par la suite, les trois salles suivantes ont été élaborées. Elles ont suivi un plan très similaire aux précédentes. Elles contiennent des mises en scène que le joueur doit observer, celles-ci étant effectuées à l'aide de préfabriqués et de modèles créés par Axel dans l'étape de modélisation (**Cf. annexe 13**). Après observation de son environnement, le joueur doit répondre à une question en lien avec celui-ci. Il peut aussi bien s'agir de déterminer le coupable s'il s'agit d'une scène de crime, ou d'observer un détail très précis de la scène actuelle. Chaque mauvaise réponse entraîne l'apparition d'ennemis, que

le joueur doit affronter. Attention cependant, être confronté à trop d'ennemis à la suite pourrait s'avérer fatal.

Suite à la construction de ces niveaux, il a fallu créer les salles dans lesquelles les boss peuvent évoluer en fonction de leurs particularités :

- La salle du premier boss, *Centaurus*, a nécessité la mise en place de plateformes surélevées afin de rendre possible les attaques envers celui-ci, n'étant vulnérable qu'aux attaques aériennes comme expliqué dans la partie du Level Design. Cela a requis un placement particulier de ces plateformes, afin d'empêcher le boss de se retrouver coincé entre deux d'entre elles, amenant à son immobilité.
- La salle du deuxième boss, l'hippogriffe claustrophobe, doit disposer de plusieurs cages. Il a donc fallu créer un script les refermant quand ce dernier y entre mais aussi l'immobilisant, celui-ci étant effrayé. De plus, ces cages ont dû être positionnées à plusieurs endroits, afin de ne pas trop limiter les possibilités du joueur et de lui laisser plus de liberté dans sa manière de jouer.
- Le troisième boss, le sphinx, a comme particularité de poser des questions avant d'attaquer. Cela a été plus simple à réaliser que les autres boss, étant donné qu'il suffisait simplement d'afficher les questions et les réponses au moment opportun, puis de lancer un combat plutôt classique, avec seulement des statistiques différentes.
- Enfin, le dernier boss, représentant les parasites OX doit, comme expliqué dans le Level Design, apparaître sur la scène du monde réel. Nous avons donc repris cette scène, puis l'avons modifiée de telle sorte que l'on voit la différence entre les deux. Finalement, nous y avons rajouté les différents ennemis devant être combattus par le joueur (**Cf. annexe 14**).

2.5 Gameplay

Le Gameplay est la partie la plus importante de notre projet. Elle traite de l'implémentation de toutes les décisions prises lors de la phase de Game Design dans des scripts, afin que les différents objets réagissent comme nous le souhaitons.

Pour la première soutenance, notre but était de donner une base solide à notre jeu, de telle manière que toutes nos actions soient facilitées par la suite. Nous avons donc implémenté de multiples classes très larges, qui sont en fait des bases d'héritage pour les différents objets que nous devons instancier tout au long de notre jeu. Nous avons par exemple créé la classe "*_Characters*", qui est la base de tous les personnages : ennemis, PNJs, ou même personnage principal. En nous appuyant sur les héritages, nous avons évité de multiples recopiages de code inutiles et fastidieux. En effet, beaucoup de fonctions n'ont alors plus qu'à être instanciées, puisque déjà présentes dans la classe parent.

Ainsi, dès la première soutenance, nous avions déjà de bonnes bases de Gameplay, tels que par exemple les mouvements. De ce fait, tous les personnages pouvaient alors se déplacer sur les axes horizontaux, sauter, ou même courir, ceci de manière limitée grâce à une jauge de Stamina se vidant au fil de la course, avant de se recharger lorsque le joueur recommençait à marcher normalement. Notre jeu étant en vue à la première personne, choix arbitraire sur lequel nous sommes vite tous tombés d'accord, les mouvements de la caméra ont tout de suite été bloqués de telle manière que la vue sur l'axe vertical soit limitée, ceci afin d'éviter que celle-ci ne se retrouve à l'envers et ne gène malencontreusement le joueur.

Pour ce qui est des statistiques relatives au personnage, entre autres *Points de vie*, *Attaque* et *Mana*, nous les avons vite implémentées. Elles ne demandaient alors qu'à être utilisées. Cependant, n'ayant ni sorts ni attaques, elles n'avaient alors pas grande utilité. De plus, elles n'étaient pas du tout équilibrées, certaines étant beau-

coup trop élevées tandis que d'autres étaient beaucoup trop faibles.

Nous avons également géré tout ce qui est relatif aux *Checkpoints* et aux *Respawns*. Ainsi, si le personnage meurt, il revient à son dernier *Checkpoint*, pour lui éviter de recommencer le niveau dans son intégralité. Ceci était très important pour le niveau de plateforme : si le joueur perd, il n'a pas envie de le recommencer dans son intégralité. C'est d'ailleurs le seul endroit où, avant la première soutenance, les *Checkpoints* étaient réellement utiles, les autres ayant pour valeur par défaut le centre de la scène et ne changeant pas encore, car l'utilité que cela nous apportait était moindre.

Enfin, concernant les objets, de multiples scripts dérivant d'une classe que nous avions nous-même créée *_Objects* ont été ajoutés pour nous faciliter la tâche. Tout d'abord, certains objets ont pour but de nous téléporter dans une autre scène. Nous avons donc commencé par un script qui réalisait ceci, mais n'avions pas encore trouvé comment récupérer les statistiques de notre personnage. Ce dernier était détruit à chaque changement de scène, ce qui n'était absolument pas l'effet désiré. De plus, nous devons dans notre jeu collecter des objets, mais ceux-ci ne doivent pas réapparaître une fois collectés, même en cas de changement de scène. Cela n'était une fois de plus pas encore réglé, et nous faisions donc face à quelques problèmes.

Pour stocker ces objets collectables, nous avons également donné au joueur un inventaire, qui lui permet de stocker ces différents items, à savoir armes, indices, sorts et potions. Comme dit précédemment, le personnage peut affecter deux items de chaque sorte à des raccourcis (par défaut les touches du clavier numérique). Ainsi, en cliquant sur la bonne touche, il peut changer d'arme active, la remplaçant par celle du raccourci. De plus, pour assigner de nouveaux objets aux raccourcis, nous avons créé une sorte de roue des sorts, qui apparaît si le joueur appuie sur la bonne touche (par défaut le bouton tab comme expliqué plus haut). Il n'a alors plus qu'à assigner ses items à l'aide de la souris. Cette version de la roue des sorts était une version totalement provi-

soire qui n'avait pour but que de montrer la diversité des armes que le joueur pouvait avoir, et la manière dont fonctionnerait notre jeu. Lors de la première soutenance, seul le changement des armes était implémenté, mais celle des autres items n'était plus qu'une question de temps.

Enfin, pour faciliter le développement, nous avons aussi créé des fonctions spécialement dédiées aux développeurs, très utiles afin de gagner un maximum de temps durant les tests, telle que la possibilité de voler, ou encore de réapparaître instantanément au dernier checkpoint visité. Ces fonctions n'auront qu'à être supprimées lors de la remise du projet, car elles n'affectent en rien le reste du jeu.

Lors de la deuxième période, le Gameplay a énormément avancé, et a même pratiquement été achevé. Tout d'abord, la classe *_Characters* citée au-dessus a été grandement améliorée et complétée. Ce script gère toutes les statistiques communes, tels que les points de vie, les points d'attaque, les différentes vitesses, ou encore les différents statuts qu'ils peuvent avoir (nous y reviendrons un peu plus tard). Enfin, des fonctions de base permettant de bouger, lancer des sorts ou attaquer par exemple y sont implémentées, pour nous aider par la suite. Comme expliqué précédemment, ce script peut se diviser en trois grandes sous-parties :

- Le personnage principal : il a bien plus de statistiques qu'un simple personnage : en effet, il sera le seul à avoir une caméra qui lui est attachée, ou encore le seul à pouvoir courir ou se baisser. C'est dans ce script que sont gérées la plupart des actions que le joueur peut effectuer : se déplacer à l'aide du clavier, courir à l'aide de la touche lui permettant de sprinter, sauter, faire bouger la caméra, attaquer, ou même lancer des sorts. Il peut également mettre le jeu en pause à tout moment, ce qui arrêtera le jeu en arrière plan, sauf si une partie avait été lancée en mode multijoueur bien évidemment. Après ceci, nous avons également amélioré le script gérant l'inventaire du personnage

principal : comme nous l'avions prévu, l'intégralité de la roue des sorts a été revue, de manière à ce qu'elle soit d'une part plus esthétique, et d'autre part plus intuitive pour le joueur. Plusieurs fonctions permettent ainsi d'afficher à l'écran tous les objets possédés, et de les sélectionner, pour les assigner aux différents raccourcis. De plus, si la souris passe par dessus un item que l'on possède, toutes ses statistiques sont affichées à l'écran. Cela pourra donc aider le joueur à se décider sur quel objet utiliser. Enfin, nous sommes restés sur l'idée de garder deux raccourcis par type d'objets, assignés au clavier numérique. Bien évidemment, ne sont pas compris les indices qui n'ont pas à être utilisés couramment. On peut ainsi passer sans problème de l'arme assignée à notre premier raccourci à celle assignée à notre second raccourci en un clic, comme nous le pouvions déjà à la première soutenance, mais ceci était devenu possible avec les sorts et les consommables pour la deuxième soutenance.

- Les Ennemis : en plus des comportements hérités par les personnages de base, ils doivent éventuellement détecter les joueurs rentrant dans leur champ de vision, les suivre, et les attaquer. C'est ici qu'entre en jeu l'intelligence artificielle, que nous développerons plus tard. Ce script se spécialise cependant en plusieurs autres : les différents types d'ennemis que l'on peut avoir. Chaque ennemi a ainsi une arme avec laquelle il peut nous attaquer, une zone de détection, une vitesse mais également une manière de fonctionner qui lui est propre : par exemple, un archer s'arrêtera beaucoup plus loin qu'un fantassin, et pourra également nous frapper en hauteur. C'est pourquoi chaque script dérivant de la classe *_Ennemis* a des méthodes bien particulières, que lui seul peut utiliser. Les boss sont également de simples ennemis, mais ils possèdent des attributs bien particuliers : par exemple, notre premier boss ne peut être attaqué que par les airs. Si le joueur n'est pas assez élevé, il ne lui fait aucun dégât.

- Les PNJs : ils se rapprochent également des intelligences artificielles, mais ont un comportement beaucoup plus aléatoire, et ne chercheront jamais à attaquer le joueur : ils ne font que se déplacer sur la scène, mais ne réalisent pas d'actions importantes. Tout comme pour les ennemis, il y a plusieurs types de PNJS : certains, comme les voitures par exemple, devront se déplacer de manière ordonnée sur un circuit bien défini, d'autres devront réellement avoir des déplacements aléatoires, et certains d'entre eux pourront même nous donner des indices si on leur parle, ce qui nous ferait avancer dans l'histoire.

En plus des personnages, les principales autres instances de notre jeu sont des objets. Nous avons donc également consacré beaucoup de temps à les implémenter durant la deuxième période. Comme dit précédemment, on peut par exemple retrouver des armes, possédant toutes des statistiques différentes et enlevant des points de vie à la cible lorsqu'elles sont utilisées. Les sorts, quant à eux, peuvent aussi bien affecter une cible, en lui infligeant des dégâts, ou en réduisant ses forces : c'est le cas du sort *Freeze*, qui gèle l'adversaire pendant un certain temps, ou du sort *Flash*, qui le désoriente, et lui fait perdre sa cible. Enfin, les sorts, tout comme les consommables, peuvent être utilisés sur nous-même, pour nous redonner de la vie comme le font les sorts de *Heal* et la potion de vie, ou encore booster notre attaque pendant un temps donné. Les indices, quant à eux, ne sont que stockés, et permettent de déverrouiller l'accès à la suite des niveaux. Nous pouvons y accéder et voir quelles informations ils renferment, mais rien de plus. Au moment de la deuxième soutenance, seuls deux sorts et deux consommables étaient complètement implémentés, les autres n'étant pas parfaitement terminés.

Durant la troisième période, nous avons seulement réarrangé certains détails qui manquaient ou qui ne fonctionnaient pas correctement. Entre autres, nous avons implémenté la gestion des sauvegardes. En effet, jusqu'alors, le jeu devait se faire en une seule fois, sous peine de perdre toute sa progression. Cela est vraiment dommage pour un

jeu de ce type. C'est pourquoi il nous semblait impératif de rajouter ce système de sauvegarde de partie.

Le menu de Game Over, lui, avait été implémenté de manière très rapide pour la deuxième soutenance. Ainsi, lorsque l'on reprenait le jeu à la suite d'une mort, toutes nos statistiques étaient restaurées, mais le jeu restait dans l'état où nous l'avions laissé. Ainsi, tous les ennemis n'étaient pas remis à zéro, il restaient à la même place, avec les mêmes statistiques. Ceci n'est bien évidemment pas le principe d'une fin de partie : le personnage, bien que mourrant, reviendrait à la vie tout le temps, comme si rien ne s'était passé, et serait en quelque sorte invincible, il ne servirait donc à rien de faire apparaître le menu de Game Over. Nous avons donc également réglé ce problème.

Enfin, nous avons terminé l'implémentation des différents sorts et consommables. Ils sont maintenant tous fonctionnels, et le joueur peut choisir d'utiliser tout ce qu'il désire, à partir du moment où il le possède dans son inventaire. Comme expliqué dans le Game Design, le personnage a maintenant un maximum de 6 sorts qu'il peut collecter, ainsi que 3 potions. Pour terminer, nous avons rééquilibré toutes les statistiques des objets et des personnages afin de rendre l'expérience du joueur la plus agréable possible, sans que le jeu ne lui paraisse ni trop simple ni trop complexe.

2.6 Intelligence Artificielle

L'intelligence artificielle regroupe la gestion de tous les événements devant être réalisés par le jeu lui-même. Entre autres, on peut nommer les Ennemis, les PNJs, ou encore les boss, qui doivent être capables par exemple de se déplacer, d'attaquer, etc.

Il y a autant de manières d'implémenter une intelligence artificielle que de personnes voulant l'implémenter, certaines étant meilleures que d'autres. Notre but était bien évidemment de créer la meilleure IA pos-

sible, tout en restant conscients de nos capacités. De plus, si elle avait été mal implémentée, le rythme du jeu aurait pu être brisé et celui-ci aurait fini par perdre tout son charme aux yeux du joueur, ce d'autant plus que notre type de jeu impose des ennemis capables de "réfléchir" par eux-mêmes, ainsi que des PNJs capables de se déplacer seuls dans la ville de manière la plus ordonnée possible. Sans cela, notre jeu serait vite devenu répétitif et ennuyeux. C'est pourquoi cette partie a été traitée avec une attention toute particulière.

Nous nous sommes donc concentrés en premier lieu sur la qualité de celle-ci, et non sur la diversité des comportements qu'elle pourrait avoir, en cherchant à trouver les meilleures méthodes possibles pour la faire agir, de manière fluide, mais pas forcément prévisible, ce qui rendrait le jeu rébarbatif.

Durant la première période, nous avons donc réalisé une multitude de tests sur des personnages et des scènes indépendants du projet, en conservant les meilleurs parmi ceux-ci. Pour la démonstration de la première soutenance, nous n'avions donc qu'un seul personnage que l'on pourrait qualifier d'intelligent, le fameux boss du premier niveau Centaurus. Cependant, celui-ci ne faisait que nous suivre, et nous attaquer lorsqu'il était assez prêt de nous.

Au départ, nous pensions qu'il serait simple de réaliser une intelligence artificielle par nous même. Cependant, après de multiples essais peu conséquents, nous nous sommes intéressés aux *NavMesh*, système d'intelligence artificielle directement intégré dans *Unity*. Ceux-ci permettent de créer un terrain et de définir où un personnage peut aller et où il ne le peut pas. Ensuite, des algorithmes déjà implantés vont chercher automatiquement le passage le plus optimisé qui soit pour permettre à notre objet de rejoindre la destination que nous lui indiquons. Bien sûr, tout ceci est fait en évitant toutes sortes d'obstacles sur son passage. C'est cette technique que nous avons conservée pour la base de notre IA, mais nous l'avons bien évidemment beaucoup modifiée, de sorte que notre personnage ne fasse pas que se déplacer

d'un point à un autre, mais qu'il réalise de nombreuses autres actions.

Durant la deuxième période, nous avons bien avancé notre intelligence artificielle. Celle-ci se décomposait alors en deux grandes sous-parties :

- La première, la plus simple à implémenter, devait simplement se déplacer à l'intérieur du monde réel, de manière plus ou moins ordonnée, afin de donner un côté plus réaliste à la scène : en effet, qui a déjà vu une grande ville sans aucun habitant ? Pour cela, nous avons créé un *NavMeshAgent*, dispositif créé par *Unity*, et lui avons donné des coordonnées plus ou moins aléatoires à suivre : nous voulons que le personnage bouge de manière arbitraire, mais qu'il réussisse à utiliser tout le terrain. Ainsi, dès que le personnage arrive à sa destination de base, nous utilisons une fonction qui recalcule de nouvelles coordonnées en fonction de la position de l'objet, sans pour autant qu'il puisse faire sans cesse demi-tour, ce qui serait particulièrement étrange.
- La deuxième, bien plus complexe, joue le rôle de notre ennemi : son unique but est de nous vaincre, quel qu'en soit le prix. Pour ce faire, nous avons commencé par lui définir une aire d'où elle est capable de détecter ses ennemis : un champ de vision en quelque sorte. S'il n'y a aucun obstacle entre les deux personnages (nous le vérifions avec des *Raycast*, fonctions implementée par *Unity* agissant un petit peu comme un champ de vision), le joueur devient alors la cible de l'IA, et se fait poursuivre. Les seuls moyens de lui échapper sont de se cacher ou de disparaître de son champ de vision, celle-ci arrêtant de nous poursuivre après un certain délai sans nous avoir aperçu. Un autre moyen existe également, mais il nécessite de charger la partie en multijoueur : en effet, si plusieurs joueurs sont en même temps dans la zone de détection de l'ennemi, celui-ci va choisir automatiquement la cible la plus proche, sauf bien sûr

si elle est cachée. Enfin, lorsque l'ennemi n'a plus de cible, il se met à patrouiller : il cherche à sa gauche et à sa droite, en tournant autour de lui-même, puis avance à un endroit positionné autour de sa zone de recherche choisi aléatoirement. Il répètera ces mouvements tant qu'aucun joueur ne vient le déranger.

Avant d'obtenir nos modèles actuels, nous avons réalisé de multiples tests, notamment avec de nombreuses instances de la même intelligence sur une seule scène. Cela nous permettait d'observer les ralentissements qui pouvaient survenir, le jeu ayant beaucoup de scripts à faire tourner par seconde. Nous ne gardions ainsi que les intelligences qui nous paraissaient convenables, mais qui ne prenaient pas trop de ressources : il s'agissait de faire un choix entre rendu et optimisation.

Ce que nous avons présenté à la deuxième soutenance nous convenait plutôt bien, mais il restait encore quelques détails à améliorer. Tout d'abord nous avons commencé par optimiser certains points, qui ne l'étaient absolument pas. De plus, un problème que nous n'avions pas vu est apparu en faisant des tests sur les ennemis : si nous passions derrière celui-ci alors qu'il était assez prêt, il continuait de regarder droit devant lui, et essayait de nous attaquer, bien qu'il nous tourne le dos. Il a donc fallu régler ce bug mineur, mais qui aurait vite pu être utilisé par les joueurs pour terminer le jeu beaucoup plus facilement.

Ensuite, nous avons spécialisé les IA des boss, qui n'étaient pas encore tous implémentés. Chaque boss ayant un comportement spécifique, nous avons dû refaire diverses fonctions pour imiter les agissements qu'ils devaient avoir. Aujourd'hui, tous les boss sont fonctionnels : ils descendent tous, comme expliqué plus haut, de la classe *_Enemies* mais ne se dirigent pas aux mêmes endroits en fonction des différents événements. De plus, il a fallu immuniser le Premier Boss de toutes les attaques au sol, celui-ci ne pouvant être attaqué que par les airs.

Enfin, nous avons rajouté des voitures dans la scène du monde réel,

afin que celle-ci ait un effet encore plus réaliste. Il a cependant fallu faire attention à certains détails : les voitures ne doivent en aucun cas écraser les piétons. C'est pourquoi nous avons redéfini les endroits où ceux-ci pouvaient voyager, indiquant la route comme un endroit plus dangereux. Les voitures sont ainsi instanciées en dehors de la scène, et dessinent un chemin aléatoire, défini à l'avance, sur les routes qui leur sont destinées.

2.7 Network

Le Network regroupe toute action en rapport avec le réseau : il s'agit aussi bien de créer les serveurs, de les maintenir, et de gérer les différentes parties se déroulant en multijoueur.

Le Network est une partie assez compliquée à implémenter. Dès la première période, nous avons commencé à faire des tests sur des scènes à part sans rapport avec le projet. Nous utilisions alors *Unet*, service de Network directement implémenté dans *Unity*. En s'instruisant sur différents sites, nous avons donc repersonnalisé ce service afin qu'il comble nos besoins.

La prise en main du Network peut être déconcertante et assez repoussante au début : à chaque test, un nouveau problème apparaît, tel que par exemple le fait que les deux personnages instanciés sur la scène soient liés et bougent ensemble. Cependant, à force de persévérance, les problèmes ont peu à peu disparu. Un problème persistait cependant avec la synchronisation entre le client et le serveur. Toutefois, pour la première soutenance, le jeu pouvait être lancé en multijoueur sans problème, un joueur hébergeant la partie pendant que les autres la rejoignaient.

Durant la deuxième période, n'arrivant pas à résoudre le problème de synchronisation auquel nous faisions face et ayant eu des bons retours venant d'autres camarades sur le framework *Photon*, nous avons

décidé d'abandonner *Unet* pour ce dernier. Cette migration a été plutôt complexe, du fait d'une implémentation très différente d'architecture entre les deux : en effet, *Unet* passe par une implémentation en "Peer to Peer" tandis que *Photon* passe par un serveur géré par les créateurs de librairie.

Cependant, après nous être documentés et avoir étudié comment cela fonctionnait, les choses nous ont été grandement simplifiées. Avec *Photon* il a ainsi été plus simple d'implémenter les RPC (Remote Procedure Call), des protocoles réseaux sans lesquels le jeu en multijoueur serait impossible. De plus l'implémentation d'un lobby aurait été bien plus complexe sur *Unet*, *Photon* les gérant nativement de manière simple et intuitive. Malgré cela, de nombreux problèmes ont encore été rencontrés pendant l'implémentation, comme notamment l'incompatibilité de certains scripts, n'ayant pas été prévus de base pour être utilisés dans un mode multijoueur.

Le Network étant presque terminé lors de la deuxième soutenance, la troisième période n'a servi qu'à régler les différents problèmes qui subsistaient au sein même de notre jeu. Ceux-ci étaient minimes, mais nous nous devions de les corriger pour rendre un produit correct.

2.8 Multijoueur

Cette étape consiste en la création d'un mode de jeu pouvant être joué en ligne à plusieurs.

A la première soutenance, le jeu en multijoueur permettait d'héberger deux joueurs simultanément sur la même partie. Cependant, aucun niveau spécifique n'était prévu pour ce mode de jeu. Nous avons donc placé les deux personnages sur la scène du monde réel, scène principale. Chacun d'entre eux pouvait alors récupérer des indices, avant d'obtenir l'accès au premier niveau. Ce dernier n'avait toutefois aucun réel intérêt, puisqu'il s'agissait d'un niveau conçu pour un seul joueur.

Le multijoueur a d'abord, tout comme le Network, été testé sur des scènes à part, le temps de se familiariser avec les fonctionnalités que propose *Unity*. C'est ainsi qu'en le réimplémentant directement dans le projet, quelques problèmes sont apparus, car certains scripts n'étaient pas du tout adaptés pour ce mode. Lors de la première soutenance, l'accès au multijoueur se faisait via le menu principal, et l'on ne pouvait le lancer qu'au démarrage du jeu, sans quoi il n'était alors plus accessible.

De plus, bien que le mode de jeu fonctionnait plutôt correctement, quelques problèmes connus étaient encore à régler. Par exemple, tous les joueurs avaient la même interface quoi qu'il arrive, les scripts ne dissociant pas le joueur local du joueur hébergé. Ainsi, lorsque l'un d'entre eux voulait faire apparaître son inventaire, l'autre le voyait également, ce qui n'était bien sûr pas l'effet désiré.

Pendant la deuxième période, avec le passage de *Unet* à *Photon*, le multijoueur a été complètement retravaillé et les problèmes de synchronisation auxquels nous faisions face ont pu être corrigés. De plus, les soucis connus tel que les bugs d'interfaces cités ci-dessus ont été résolus. Le système de *Matchmaking* a également été modifié. Ainsi, pour accéder à une partie, il était alors nécessaire de passer par un *Lobby*.

Pour lancer une partie à plusieurs, le joueur peut créer une *Room*, en choisissant le nom qu'il veut lui donner, et en appuyant sur le bouton "Create Room". Il est aussi possible de rejoindre toutes celles déjà créées, affichées sur le côté droit du menu. (**Cf. annexe 15**) Après avoir créé ou rejoint une *Room*, l'écran de pré-chargement apparaît. Le joueur peut alors y lancer la partie (uniquement s'il en est le créateur), la quitter ou encore changer la visibilité de cette *Room* (cela pouvant permettre dans certains cas à d'autres de le rejoindre). La partie lancée, les deux personnages se retrouvent directement téléportés sur la scène du niveau multijoueur, à savoir le niveau du labyrinthe.

Dans ce niveau, les deux joueurs doivent s'entraider afin d'avancer jusqu'au bout du labyrinthe, et ainsi débloquer le boss. Comme expliqué dans le Level Design, il y a deux rôles principaux : le premier se trouve à l'intérieur même du labyrinthe, et doit l'explorer, pendant que l'autre se trouve sur une plateforme en hauteur, sans aucun moyen de le rejoindre, et doit l'aider en désactivant des portes à l'aide de leviers. Le labyrinthe est jonché de pièges et d'ennemis en tout genre, mais on y trouve également de nombreux coffres, pouvant aider le joueur à débloquer de nouveaux items une fois revenu dans le monde réel. Bien sûr, les deux personnages peuvent échanger leurs rôles à certains moments, ceci afin de diversifier l'intérêt du jeu.

Avant la troisième et dernière soutenance, nous n'avions plus qu'à régler quelques problèmes encore existants. Nous avons également amélioré le niveau du labyrinthe, pour que celui-ci prodigue une expérience de jeu encore plus grande au joueur. Enfin, nous avons implémenté le boss, qui n'était à cet instant qu'à l'état de plan. Celui-ci doit être battu par les deux joueurs, unissant leurs forces dans un combat acharné.

2.9 Gestion des collisions

Cette partie est l'une des plus petites du projet, bien que tout de même importante. Il s'agit du processus consistant en la gestion des collisions, aussi bien pour les items ramassables, que pour les objets que l'on ne peut pas traverser, et qui doivent donc produire une collision la plus réaliste possible.

Ainsi, nous nous devons d'utiliser des *Colliders* qui ne ralentissent pas trop le jeu, mais qui restent tout de même crédibles. Au tout début, nous avions utilisé des *MeshColliders*, *Colliders* les plus précis de *Unity*, ceux-ci épousant la forme de l'objet à la perfection. Cependant, après plusieurs recherches et documentations, nous nous sommes rendus compte des mauvais aspects que cette façon de faire impliquait,

notamment par sa gourmandise en ressources. Nous avons donc finalement décidé d'utiliser, dans la mesure du possible, des *BoxColliders*, *CapsuleColliders* et *SphereColliders*, quitte à en superposer plusieurs sur un seul objet. En effet, ceux-ci sont moins gourmands mais tout de même efficaces : ils n'imitent pas parfaitement le monde réel mais s'en rapprochent un maximum.

Pour pouvoir gérer le ramassage des objets, ou même toutes les interactions avec ceux-ci, nous avons également utilisé des fonctions telles que *OnTriggerEnter*, *OnTriggerExit*, *OnMouseDown*, et beaucoup d'autres, qui sont des fonctions intégrées à *Unity* qui ne se déclenchent que dans des conditions bien particulières. Celles-ci ont été essentielles pour interagir avec les objets correctement.

Enfin, certains *Colliders Triggers* ont été utilisés pour faciliter l'implémentation des ennemis. En effet, ceux-ci possèdent tous une *SphereCollider*, qui modélise leur aire de détection. Si un joueur y entre, il devient alors une cible potentielle.

Aujourd'hui, les collisions ne sont certes pas parfaites, mais convenables pour un jeu de notre niveau. Tous les objets réagissent comme espérés, et nous n'avons pas décelé de problèmes majeurs en testant notre jeu.

2.10 Création des menus

Cette partie consiste en la création de tous les menus du jeu : ils doivent bien évidemment être fonctionnels, mais également être le plus esthétiques possibles. Cependant, mieux vaut se concentrer sur l'esthétisme à l'intérieur même du jeu, car c'est l'endroit où le joueur passera le plus clair de son temps. Il faut préciser que nous considérons comme Menu tout ce qui est créé à parti des *UI*, outil de *Unity* permettant de gérer tous les éléments en 2D.

Pour la première soutenance, nous nous sommes limités à un menu sommaire et brut. Il était muni d'un bouton pour lancer le jeu, un autre pour afficher les paramètres *Volume sonore*, *difficulté du niveau* et *sensibilité de la souris*, qui ne fonctionnaient alors pas, ainsi qu'un dernier affichant les crédits, à ce moment vides (**Cf. annexe 16**).

Nous avions également un menu de changements d'armes temporaire, comme précisé plus haut, celui-ci n'ayant pas pour but d'être esthétique, mais d'être fonctionnel. En effet, notre but était de pouvoir tester notre jeu facilement, ainsi que de faire la démonstration de tout ce que nous avions implémenté le jour de la soutenance.

Lors de la deuxième période, nous avons beaucoup plus travaillé sur l'avancée des menus. Nous avions ainsi, le jour de la deuxième soutenance, plusieurs menus fonctionnels :

- Un écran titre.
- Un menu pause, qui ne peut apparaître si le jeu est lancé en mode multijoueur.
- Un écran de GameOver, laissant au joueur le choix de retenter sa chance ou de quitter le jeu.
- Une interface s'affichant en jeu.
- Le menu de l'inventaire qui était grandement amélioré.

Tous les menus étaient provisoires, leur apparence pouvant largement être améliorée, mis à part l'inventaire qui est resté exactement tel qu'il était. Ce dernier a en effet nécessité beaucoup de temps, mais contient de nombreuses fonctionnalités. Tout d'abord, il affiche tous les items que le joueur possède. Il est bien sûr possible de décider quels éléments afficher, en cliquant sur les boutons *Weapons*, *Spells*, *Consumables* et *Clues*. Ensuite, si le joueur passe sa souris par dessus un item qu'il possède, toutes ses statistiques sont affichées. Il peut alors cliquer dessus, pour l'affecter à un raccourci. Normalement, les images des armes devaient se mettre automatiquement sur les boutons. Cependant, nous n'avions pas encore réalisé les images pour les y placer,

car les armes n'étaient pas modélisées (**Cf. annexe 17**).

L'interface du joueur contenait également une barre de vie, de mana et de stamina, se mettant à jour automatiquement, afin de suivre sans difficulté les statistiques de notre personnage. Celle-ci était temporaire, mais a été très utile pour tester notre jeu (**Cf. annexe 18**).

Pour la dernière soutenance, nous n'avions plus qu'à améliorer le rendu de nos menus. Nous ne nous sommes ainsi attardé qu'à leur esthétisme : par exemple, les Sprites des armes ont été rajoutés. Ceux-ci ne sont en réalité que des screenshots des armes que nous avons modélisées, pris directement sur Blender. L'interface du jeu a également été améliorée, afin qu'elle ne fasse pas brouillon. Nous pouvons ainsi voir à tout instant les armes et sorts que nous avons équipés, ainsi que les statistiques de notre personnage.

2.11 Sound Design

Le Sound Design regroupe le choix ainsi que l'implémentation de tous les effets sonores et bandes musicales qu'il y aura dans notre jeu.

Cette partie étant pour nous secondaire durant tout le début du projet, le son n'a été implémenté que très tardivement. La première soutenance s'est en effet déroulée sans son, et la seconde n'avait pour seul effet sonore que des bruits de pas, qui différaient en fonction de la vitesse de déplacement du personnage et du sol sur lequel il marchait, ainsi qu'une musique de fond sur le menu principal. Nous n'avions cependant pas d'autres musiques.

Durant la troisième période, ayant plus de temps pour travailler sur les peaufinages, nous avons pu nous consacrer pleinement aux bandes sons du jeu. Nous avons ainsi amélioré les bruits de pas, ceux-ci ayant quelques problèmes, tel que par exemple le fait qu'ils ne s'arrêtent pas durant un saut. De plus, nous avons rajouté des bruits lorsqu'un per-

sonnage attaque, qu'il jète un sort, ou encore lorsqu'un piège est activé.

Enfin, la dernière partie importante du Sound Design était l'implémentation des bandes musicales du jeu. Nous avons donc rajouté des musiques lorsque le jeu est lancé, de telle manière qu'elles se mêlent parfaitement à la situation. Par exemple, si le personnage est ciblé par un ennemi, la musique sera bien plus stimulante. Il a également fallu régler les transitions, pour éviter que les musiques ne changent trop brusquement.

2.12 Animations et particules

Cette partie est certes la moins importante pour le fonctionnement du jeu, mais sûrement une des plus essentielles pour le rendu de celui-ci. En effet, les animations représentent tous les mouvements que réaliseront notre personnage de manière automatique, comme bouger les bras lorsqu'il avance ou attaque. Les particules, elles, seront présentes pour que le jeu soit plus réaliste : elles permettent de rajouter des effets, aussi bien physiques que lumineux.

Pendant longtemps, nous avons ignoré cette partie. Ne connaissant pas *Unity*, nous ne savions même pas que ces éléments existaient à la remise du cahier des charges, et n'en avions ainsi pas fait mention. Nous avons donc passé la première soutenance sans aucune animation ni particules, cela étant selon nous un élément d'esthétisme à rajouter à la fin, et non quelque chose de vital pour notre projet.

Pour la deuxième soutenance, nous nous sommes retrouvés face à un problème assez malencontreux : nous pouvions attaquer, mais rien n'indiquait que notre personnage le faisait. Ainsi, nous ne pouvions être sûrs en tant que joueurs que ce que nous faisions servait à quelque chose. C'est pourquoi nous avons implémenté assez rapidement des animations d'attaques, ainsi qu'un mouvement continu qui est présent lorsque le personnage se déplace : il est différent si celui-ci marche, court, ou saute.

Pendant la troisième période, comme dit précédemment, nous avions plus de temps pour nous occuper des détails. Par exemple, les animations ont toutes été refaites, et ont été recalibrées pour aller avec le nouveau personnage. De plus, chaque arme possède sa propre animation d'attaque afin que celles-ci restent cohérentes. En effet, la manière d'attaquer avec un Katana, qui n'a qu'une seule lame, n'est pas la même qu'avec une épée, qui en a deux, ni qu'avec ses poings ou un couteau, bien trop court pour effectuer les mêmes mouvements.

Les particules, elles, ont été rajoutées sur plusieurs objets :

- La fontaine, devant imiter l'eau.
- Des torches, devant imiter le feu.
- Les personnages, s'ils se déplacent, attaquent ou utilisent des sorts.

De plus, les effets de particules sont tous différents, et représentent les effets de sorts le plus logiquement possible. Par exemple nous avons utilisé des particules jaunes pour représenter le sort de soin, ceci étant une sorte de code universel dans le monde du jeu vidéo.

2.13 Creation du site internet

Pour la première soutenance, le site etait a peine commence. Cependant, une partie du travail avait djà t  faite, celle concernant le plan original du site. Celui-ci se pr sentait ainsi :

- Accueil
- Parad-OX :
 - Sc nario
 - Gameplay
 - Inspirations
- TeamCheall (a propos des dveloppeurs)
 - Pr sentation des membres

- Formulaire de Contact
- Téléchargement
- Support
 - Guide d'installation du jeu
 - Problèmes et solutions associées
 - Soumettre les bugs
- Mentions légales

Une ébauche du site avait également été préparée afin de pouvoir la présenter. Celle-ci avait été réalisée par Maya en PHP, HTML et CSS, dont elle avait appris les bases.

Pour la seconde soutenance, le site avait été étoffé, quelques pages et de petites fonctionnalités rajoutées (**Cf. annexe 19**). Cependant, nous avons éprouvé des difficultés suite au départ de Maya, car c'est elle qui s'était occupée de la phase initiale du site, sans nous y impliquer.

Pour la dernière soutenance, nous avons finalement décidé de le redesigner dans son intégralité. Une touche de modernité a été ajoutée, et les menus sont plus accessibles qu'auparavant. Nous avons cependant gardé le même plan, qui nous paraissait correct. Le site est ainsi plus fluide et intuitif que l'ancien. Il est maintenant hébergé sur *GitHub Pages*, un service permettant d'héberger des sites géré par *GitHub*. Cette prise de décision résultait d'un manque d'informations pour continuer le site web initialement créé par Maya. Grâce à cela, le site a alors été beaucoup plus simple à modifier, et nous avons réellement l'impression qu'il nous appartient.

3 Expériences personnelles

3.1 Axel "PixL" GRUNENBERGER

Etant de nature réservé, j'ai mis beaucoup de temps à me rapprocher des autres membres de ma classe. A l'heure de la création des groupes, ceux-ci étaient tous déjà formés. C'est à ce moment que j'ai rencontré Maya, Constant et Cyril, qui cherchaient tous les trois à intégrer un groupe. C'est ainsi qu'était créée la TeamCheall, notre groupe de projet.

N'ayant jamais réalisé de projets de la sorte, et les applications *Unity*, *github* et *Blender* m'étant toutes inconnues, j'étais au départ particulièrement inquiet de ne pas être à la hauteur et de ne pas savoir m'adapter pour aider mon groupe. J'ai donc très vite commencé à me documenter sur les différentes bibliothèques que contiennent *Unity*, prenant des notes sur les fonctions qui pourraient nous être utiles, et à regarder des tutoriels pour utiliser *Blender*, même si la modélisation n'était à la base pas ma tâche principale.

Lorsque nous avons réellement commencé à mettre en œuvre notre projet, j'ai tenté d'implémenter différentes fonctions. Cependant, la plupart de mes scripts menaient à des échecs. A force de persévérance, de nouvelles recherches et de nombreux essais, mes fonctions ont commencé à montrer des résultats de plus en plus prometteurs. Je me suis alors pris au jeu, et ai travaillé dur pour avancer du mieux que je pouvais.

En charge d'une très grande partie du Gameplay et de l'IA de notre jeu, je pensais passer la majeure partie de mon temps à coder. J'ai d'ailleurs beaucoup appris et mûri, à force d'erreurs et de corrections de mes programmes. En développant l'intelligence artificielle, j'ai également pu tester de nombreux algorithmes que je créais, et j'ai donc pu voir l'avancée de ma création au fil du temps, ce que j'ai trouvé assez plaisant. Plus le temps passait, plus j'avais l'impression

de toucher au but.

Cependant, avec le départ de Maya, j'ai également dû prendre en charge la partie Blender du projet. J'ai toujours sous évalué mes capacités artistiques, ne me considérant ni comme un bon dessinateur, ni comme un bon artiste en général. Mes camarades ayant la même image d'eux, nous nous sommes demandés si nous n'allions pas changer d'avis au dernier moment et prendre des assets sur unity. Toutefois, j'ai fait le choix de relever ce défi, quitte à ce que ce soit une perte de temps, afin de voir ce que je pouvais produire. Aujourd'hui, je ne regrette absolument pas ce choix, car je trouve mes modèles plutôt réussis, et me suis ainsi rendu compte que l'on pouvait réussir tout ce que l'on voulait entreprendre, à force de volonté et si l'on s'en donne les moyens.

Œuvrer sur ce projet m'a également permis de découvrir une façon de penser qui m'est plutôt peu familière. En effet, n'étant pas quelqu'un de très sociable, je n'ai pas pour habitude de travailler en groupe. Aujourd'hui, je me rends compte des avantages à travailler collectivement : que ce soit avec les membres de mon propre groupe, ou avec d'autres étudiants que l'on peut croiser en salle machine, nous avons tous une manière de penser différente, et rassembler nos idées permet de créer quelque chose qui n'aurait pas été possible seul. De plus, une ambiance conviviale et bienveillante régnait en salle machine lorsque nous y travaillions.

3.2 Cyril "Pokkihju" RIDEAU

Originellement, ce projet était un peu mon dernier recours pour trouver un groupe, ne m'y étant pas pris à l'avance. Je m'étais retrouvé sans groupe et c'est le groupe d'Axel, Constant et Maya qui a accepté de me prendre comme dernier membre. Le style du jeu et les différents éléments le composant ont fait l'objet de quelques débats dans les premières réunions du groupe, mais il a fallu faire un choix

et c'est ainsi que nous nous sommes orientés sur le projet actuel. Ce projet m'a beaucoup plu car il m'a laissé expérimenter de nombreux types de jeux différents : l'énigme, le labyrinthe et la plateforme. Cela m'a permis de m'essayer au design de ces différents styles.

Ce projet m'aura ainsi rendu possible de m'essayer au Game Design d'un jeu qui, certes, ne sera pas publié, mais qui aura tout de même atteint un stade final de développement. De plus, il a fallu apporter beaucoup de modifications au plan initial, principalement sur des détails, mais cela a beaucoup affecté le rendu final, le rendant plus satisfaisant encore. Ces modifications ont été faites de concert avec toute l'équipe. Le Design de tous ces niveaux différents aura permis à mon propre style d'évoluer. Bien que je sois encore très loin des Game Designers d'entreprises comme Ubisoft ou Bethesda, je pense avoir bien progressé dans ce domaine.

La partie de construction de niveau m'aura également permis de développer ma persévérance. Etant un travail très répétitif et bien moins intéressant que de créer les niveaux dans son esprit, la construction de niveau sous *Unity* demande surtout du temps et de la préparation. Comme nous avions décidé de ne pas utiliser d'Assets, chaque objet ayant des particularités a du être réalisées à la main, et j'ai dû m'efforcer d'en comprendre les mécanismes.

La partie d'animations et de particules a été pour moi la partie qui m'a le plus détendu car elle ne requiert qu'une reproduction du réel ainsi qu'un déplacement logique des objets sous l'animateur de Unity. Ainsi, cela était une sorte de pause après de longues heures de travail passées sur le projet.

En dehors du travail sur le projet en lui-même, j'ai pu rencontrer de nombreuses personnes lors des sessions de travail à l'EPIITA, très sympathiques et toujours de bon conseil en cas de difficultés. La bonne ambiance régnant dans les salles machines à 23h est particulièrement agréable pour travailler et être productif. De plus, nous avions accès

en exclusivité aux jeux des autres groupes, et pouvions y rechercher différents bugs. Ce projet m'aura donc permis de me développer humainement et cela n'a pas de prix.

3.3 Constant "Ghakizu" MALANDA

Au départ j'ai décidé de rejoindre ce groupe car Maya, l'ancienne chef de projet, m'avais proposé de faire un jeu de rythme assez délivrant. Cependant, suite à plusieurs réunions avec les membres du groupe, l'idée finale a été complètement différente de celle de base. Malgré cela je suis quand même resté pour pouvoir expérimenter autre chose que les jeux de rythmes.

Grâce à ce projet, j'ai pu découvrir les joies du réseaux et de *GitHub*. Il m'a fallu beaucoup de patience et de persévérance pour pouvoir avancer. J'ai du faire face à de nombreux bugs et conflits en tentant d'implémenter le réseau, ce qui n'a pas été une tâche des plus faciles. De plus, suite à la perte de notre chef de projet Maya, j'ai du m'occuper du site internet, qu'il a fallu revoir dans son intégralité.

La refonte complète de ce site a été assez dure étant donné que je n'avais que très peu d'expérience avec le HTML et le PHP. Cependant, grâce à plusieurs documentations et à de l'entraide, j'ai pu réussir à concrétiser notre site.

Cette expérience a été pour moi très bénéfique car j'ai pu améliorer mes compétences en code tout en apprenant de nouveaux langage et le fonctionnement de nombreux logiciels comme par exemple *Git-Kraken*, *Wamp* ou encore *Audacity*. Elle m'a également permis de me lier d'amitié avec les personnes de mon groupe, ainsi qu'à d'autres étudiants rencontrés lors de session de travail intensif dans les salles machines.

Conclusion

Grâce à ce projet, nous avons tous gagné en maturité, et avons appris à travailler en groupe, avec des contraintes bien précises à respecter. Nous sommes ainsi fiers d'avoir réalisé l'intégralité de notre projet seuls, sans utiliser aucun Asset présent dans *Unity*.

Chaque jour passant, nous pouvions observer notre projet **Parad-OX** prendre forme, et évoluer au fil du temps. Bien que Maya soit partie au cours de l'aventure, nous sommes restés unis face aux problèmes rencontrés, et avons cherché à les surmonter un par un.

Nous avons également pu améliorer nos compétences en programmation en découvrant de nombreux algorithmes, mais également en expression, les soutenances nous préparant à de futurs entretiens dans le monde du travail. Enfin, il a fallu faire preuve d'une autonomie et d'un esprit d'équipe sans faille pour mener à bien notre jeu jusqu'à son terme.

Nous sommes ainsi très satisfaits de ce que nous avons été à même de réaliser, tant par les idées originales que nous avons apportées au jeu, que par leur implémentation à l'intérieur de celui-ci.

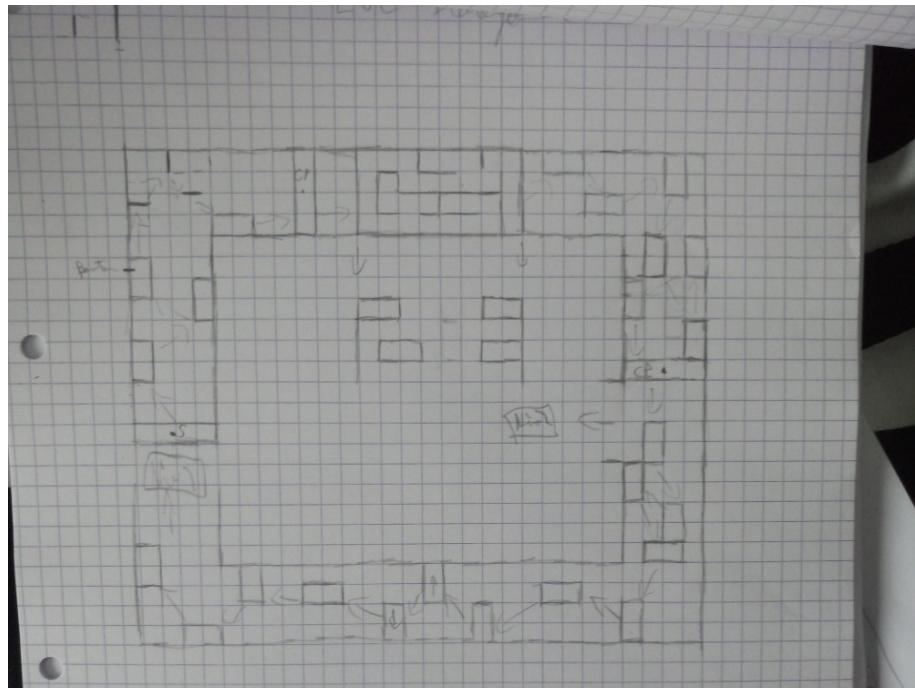


FIGURE 1 – Level Design du niveau de plateformes

Annexes

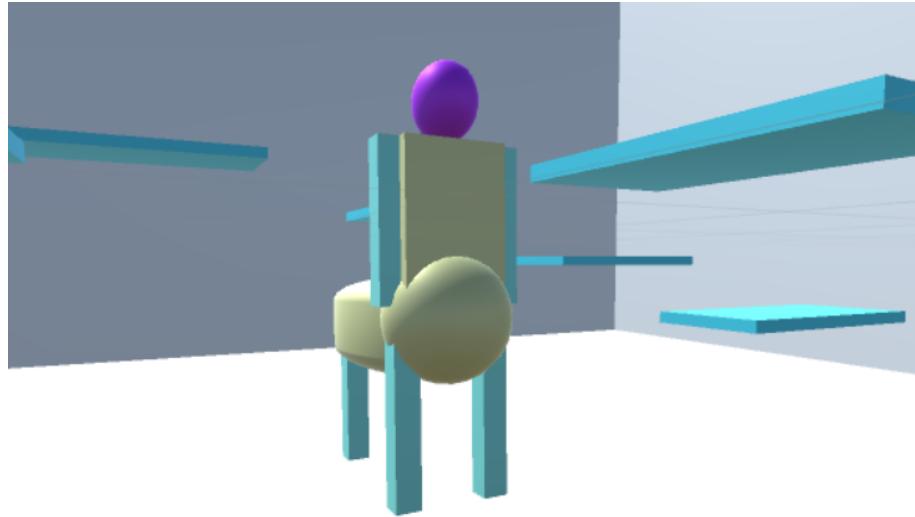


FIGURE 2 – Centaurus, boss du premier niveau

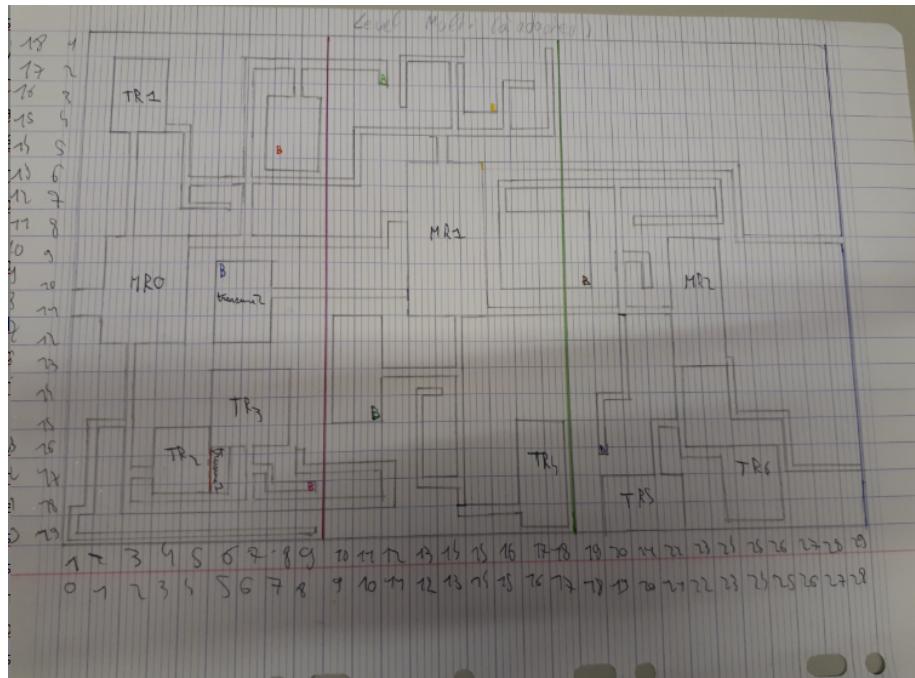


FIGURE 3 – Level Design du niveau du labyrinthe

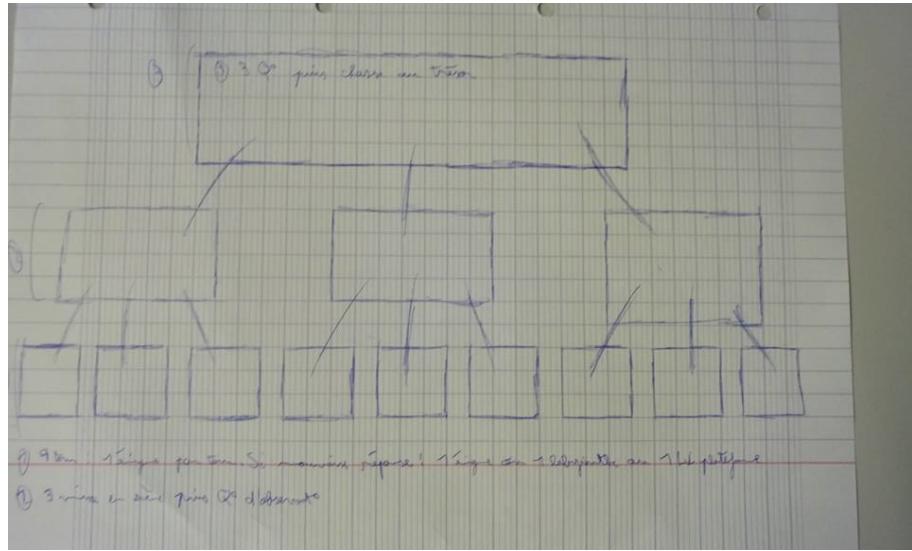


FIGURE 4 – Level Design du niveau d’énigmes

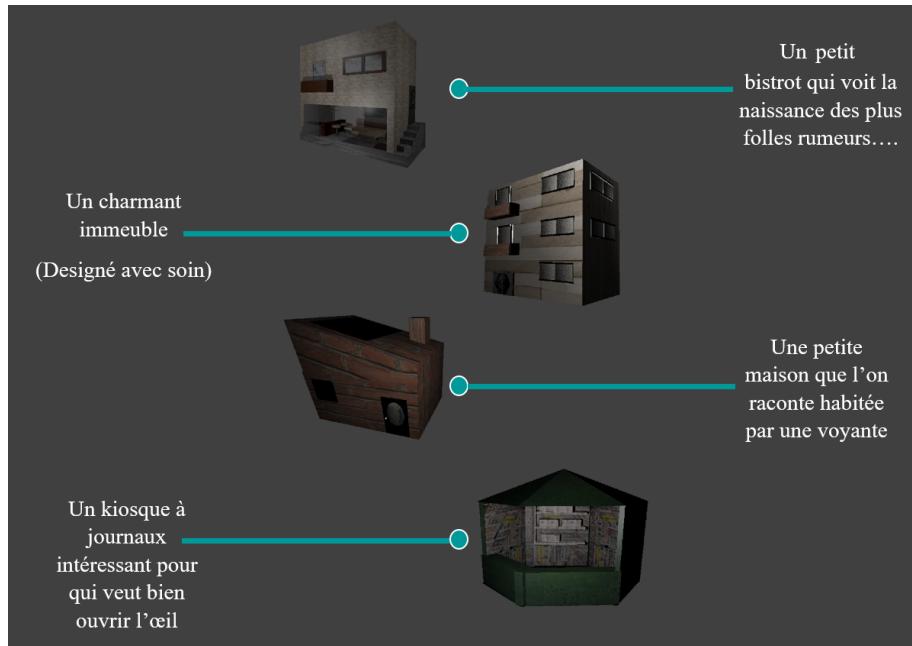


FIGURE 5 – Quelques modèles réalisés lors de la première période

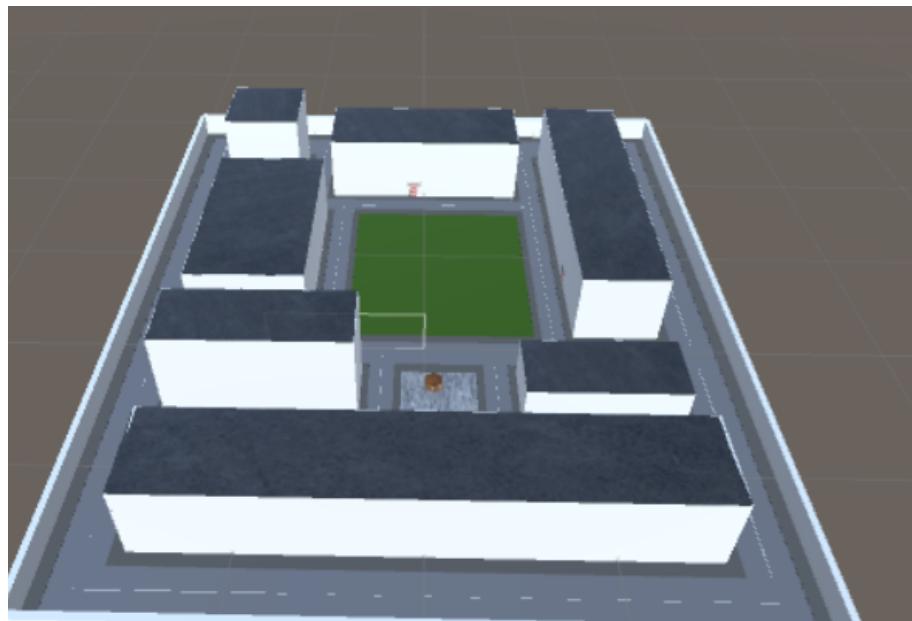


FIGURE 6 – Scène principale lors de la première soutenance

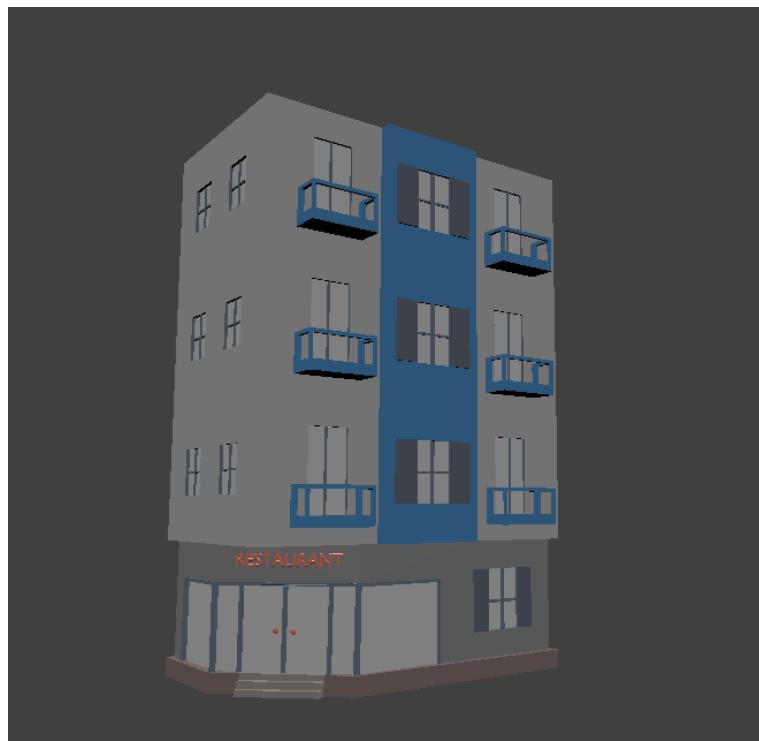


FIGURE 7 – Restaurant modélisé sous *Blender*



FIGURE 8 – Supermarché modélisé sous *Blender*

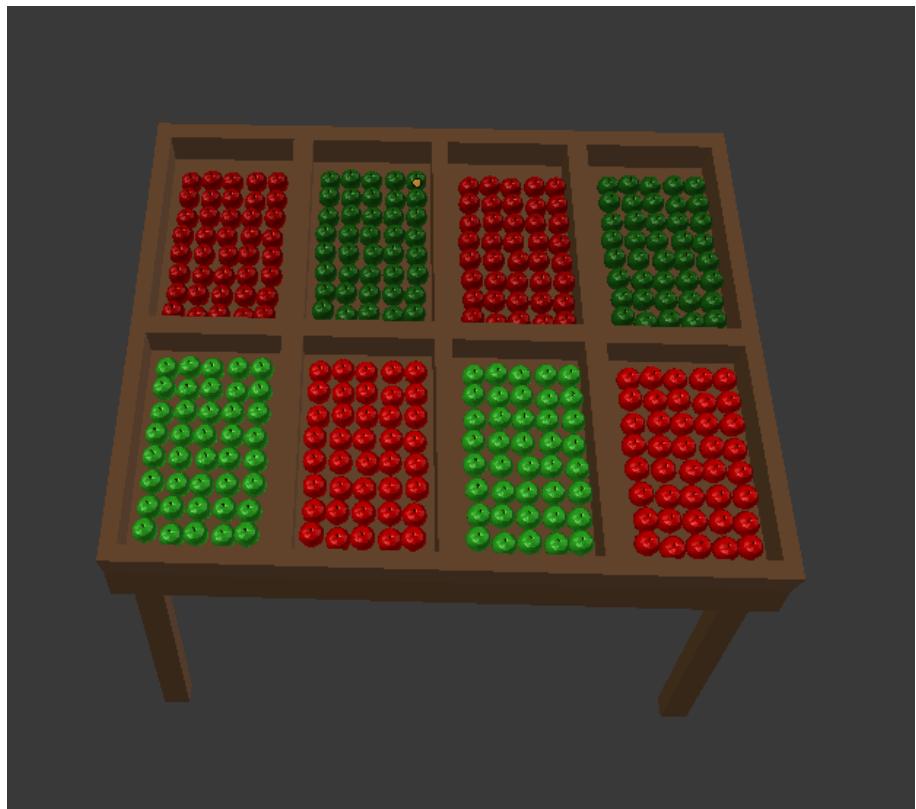


FIGURE 9 – Stand de fruits modélisé sous *Blender*



FIGURE 10 – Personnage modélisé sous *Blender*

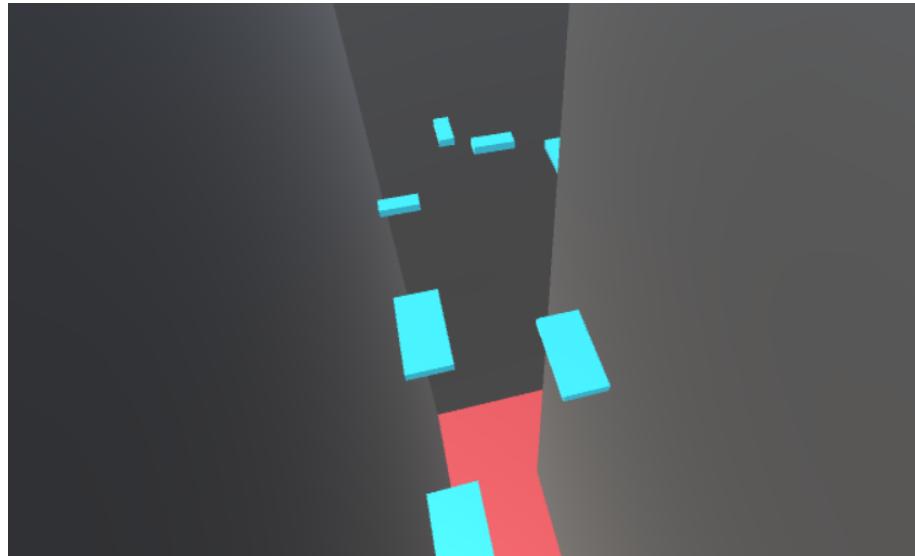


FIGURE 11 – Scène du level de plateformes

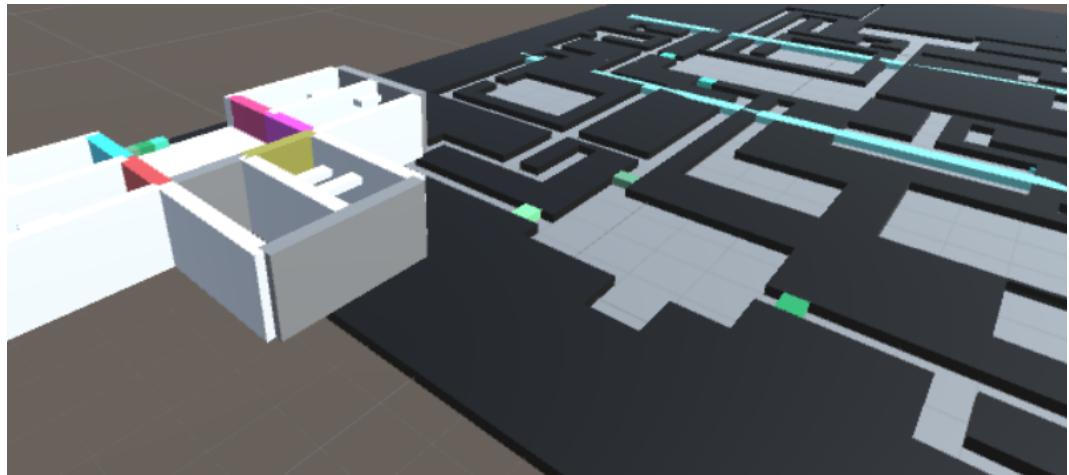


FIGURE 12 – Scène du level du labyrinthe



FIGURE 13 – Une énigme de la scène du troisième niveau



FIGURE 14 – Scène du monde réel



FIGURE 15 – Interface de lancement d'une partie en mode multijoueur

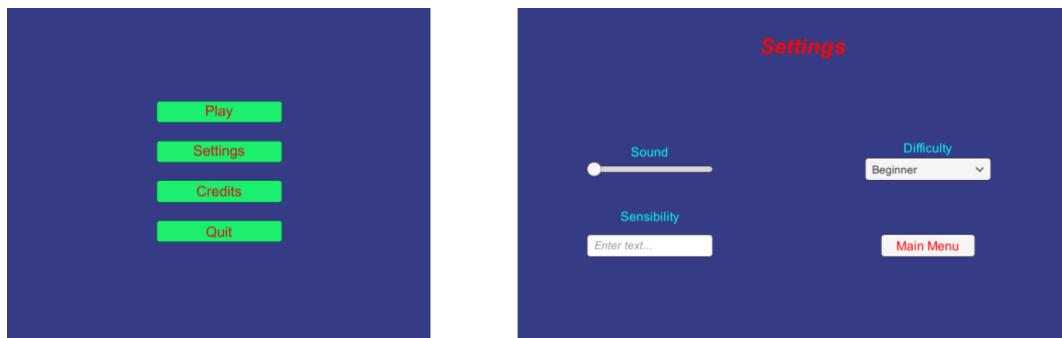


FIGURE 16 – Menu principal et Paramètres du jeu

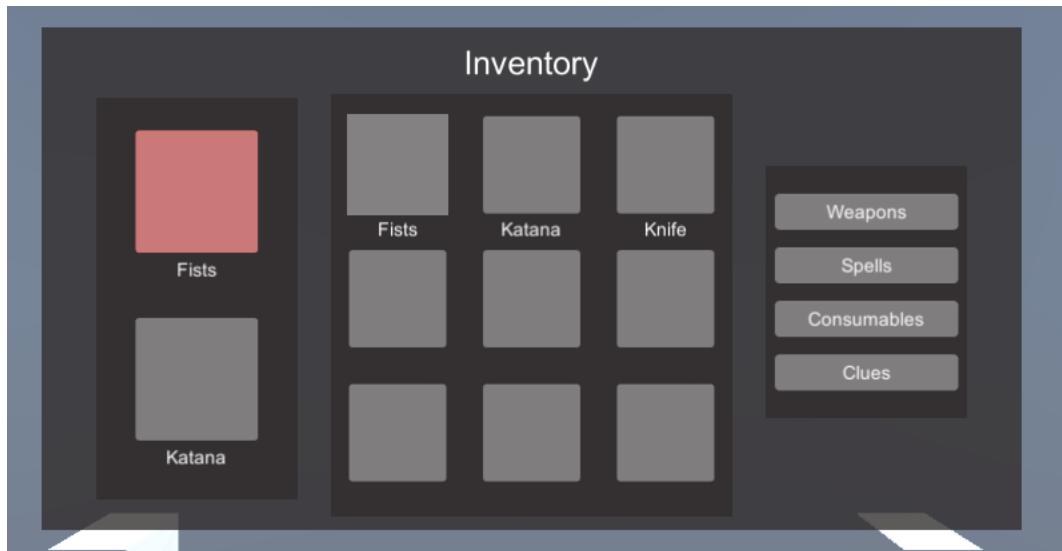


FIGURE 17 – Inventaire du joueur

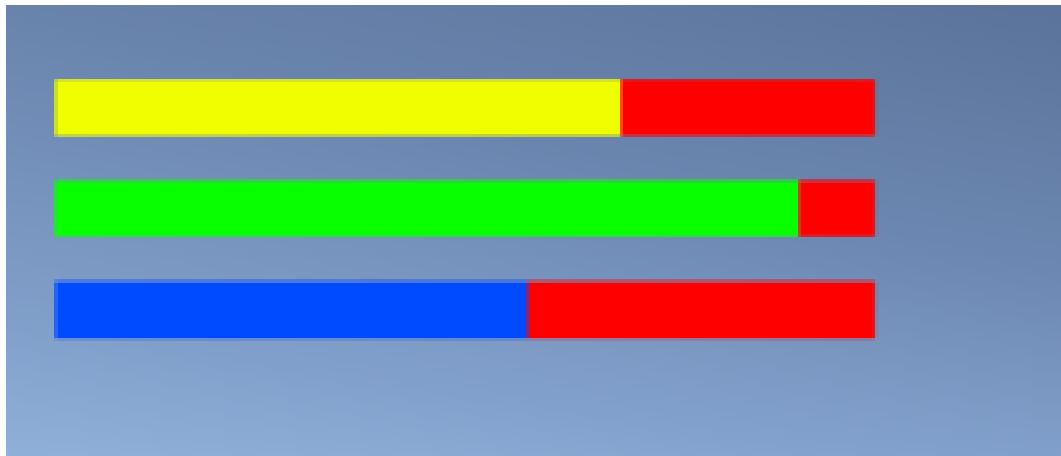


FIGURE 18 – Statistiques du joueur en jeu (deuxième soutenance)



FIGURE 19 – Site internet (deuxième soutenance)