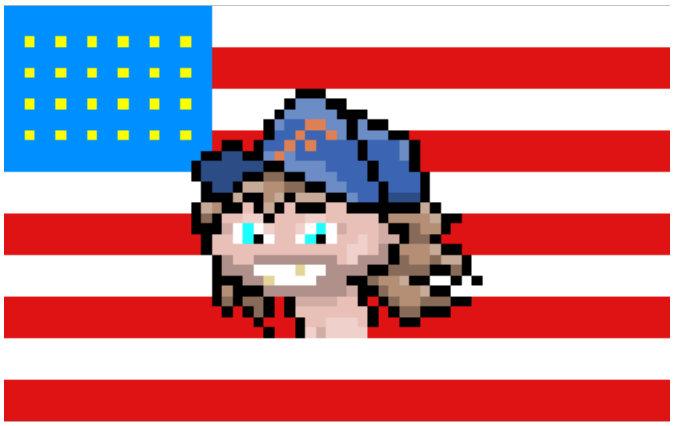
Deter_Birb

Projet de S2

Florida Man VS the Berds : Rapport de projet



Fatou **TOURE**
Simon **ZAKEYH**
Daniel **MIGUEL**
Léo **TRIPIER**

Chef d'équipe : Simon **ZAKEYH**

Sommaire

Chapitre 1

Introduction

Depuis le début du second semestre de cette première année à EPITA Lyon, 4 étudiants se sont réunis afin de créer le groupe Deter_Birb et le projet **Florida-Man VS The Berds**. Ainsi, pendant ce semestre, nous avons travaillé pour vous donner un projet de qualité, représentant ce qui fait rire la nouvelle génération sur ce monde spécial qu'est internet.

Au programme de ce rapport final, un rappel du cahier des charges puis l'historique, la chronologie, les réussites et les difficultés de chacun des membres du groupe.

Rappel : L'équipe Deter_Birb est composée de 4 membres :
— Le chef de projet : Simon ZAKHEYH aka MrZed
— Fatou TOURÉ aka The Queen
— Daniel MIGUEL aka Barbarossa
— Léo TRIPIER aka L'Otarie

Chapitre 2

Ce qui était prévu...

Durant le mois de janvier, nous avons rendu un cahier des charges regroupant nos prévisions pour notre projet.

2.1 Ce que disait le cahier des charges...

2.1.1 La présentation du jeu

Florida Man VS the Berds est un jeu de type Rogue-Like en deux dimensions basé sur des déplacements et un gameplay nerveux. Le thème de base, origine du meme "Florida-Man", un meme basée sur un homme de Floride se retrouvant en permanence dans des situations pour le moins cocasses. Le jeu se veut humoristique dans son fond et précis dans sa forme, associant légèreté de ton et "tryhard".

Le Rogue-Like, la base de notre projet

Le rogue-like est un style de jeu-vidéo extrêmement addictif du fait de son gameplay à la prise en main simple mais à la maîtrise complexe, et de sa rejouabilité virtuellement infinie. Cet archétype ne date pas d'hier mais il a largement évolué depuis les débuts du premier Rogue, un jeu d'exploration de donjon au tour par tour. Le gameplay actuel préfère le live action au tour-par-tour, appelant désormais plus aux speedrunners qu'aux stratèges.

En quoi ça consiste ?

Un rogue-like, est un jeu qui place le joueur dans un environnement généré aléatoirement, selon des règles précises, tout en lui donnant les moyens d'arriver à un objectif. Généralement un roguelike se caractérise aussi par un système de donjon qui constitue l'environnement, l'objectif étant d'en sortir, généralement après un combat de Boss exceptionnellement difficile, le parcours de diverses salles et le ramassage de nombreux objets et armes permettant au joueur d'améliorer et d'optimiser ses propres statistiques. La progression peut se faire en dehors du donjon, comme dans le jeu Rogue-Legacy de Cellar Doors Games, où le joueur obtient des améliorations constantes valable dans toutes ses prochaines parties du, ou à l'intérieur du donjon, où le joueur débloque des objets éphémères qu'il pourra ramasser dans ses parties. L'aléatoire est en effet un aspect majeur des rogue-likes, car bien qu'il soit possible de finir un donjon sans objets, les bonus aléatoires que le joueur ramasses au cours de sa partie lui permettent de le finir plus facilement, rapidement et, ou de créer une expérience unique à chaque partie. De cette manière, si à la découverte du jeu, le but est de finir une "run", le joueur expérimenté trouvera son plaisir dans l'amélioration de son meilleur temps/score, à l'aide de combinaisons d'objets et de mouvements toujours plus optimisés. Encore une fois, la rejouabilité est un point-clé.

Des exemples

Le Rogue-like étant un genre ancien et vaste, il est difficile de trouver un jeu-type parfait pour servir d'exemple, il serait donc plus judicieux de donner plusieurs exemples, tous cousins dans cette famille étendue.

Il est impossible de parler de Rogue-like sans parler de "Rogue" (1980), c'est écrit dans le nom ! Ce jeu qui va sur ses 40 ans a posé les bases du genre, et même si son aspect tour-par tour le distingue de ses dérivés plus récents, tout est là : la mort permanente, le donjon aléatoire, le boss final, les objets...

Grand phénomène à son apogée, le très célèbre "The Binding of Isaac" (2011) popularise l'archétype par son gameplay dynamique action-RPG, mi-shooter, mi-roguelike, et inspire de nombreux jeux à la recette similaire, tels que "Enter the Gungeon" (2016).

Enfin, Il nous faut évoquer "Risk of rain" (premier du nom, 2014), basé sur un gameplay en 2D, avec un aspect platformer bien exploité, grande source d'inspiration pour notre projet.

Des exemples

Le Rogue-like étant un genre ancien et vaste, il est difficile de trouver un jeu-type parfait pour servir d'exemple, il serait donc plus judicieux de donner plusieurs exemples, tous cousins dans cette famille étendue.

Il est impossible de parler de Rogue-like sans parler de "Rogue" (1980), c'est écrit dans le nom ! Ce jeu qui va sur ses 40 ans a posé les bases du genre, et même si son aspect tour-par tour le distingue de ses dérivés plus récents, tout est là : la mort permanente, le donjon aléatoire, le boss final, les objets...

Grand phénomène à son apogée, le très célèbre "The Binding of Isaac" (2011) popularise l'archétype par son gameplay dynamique action-RPG, mi-shooter, mi-roguelike, et inspire de nombreux jeux à la recette similaire, tels que "Enter the Gungeon" (2016).

Enfin, Il nous faut évoquer "Risk of rain" (premier du nom, 2014), basé sur un gameplay en 2D, avec un aspect platformer bien exploité, grande source d'inspiration pour notre projet.

Florida Man VS the Berds, Comment en est-il arrivé là ?

L'histoire

Dans un monde parallèle, une grande entreprise journalistique s'est installée dans le haut du podium des entreprises les plus influentes du monde : Berds Incorporation. Cette entreprise n'est pas comme les autres. Tout d'abord, son PDG n'est autre que BERD, un oiseau tout blanc potelé à l'apparence plus simpliste qu'une addition niveau CP.

Leur slogan : "0% Fake News, 100% optimal for academic use."

Et pourtant...
Il se trouve que les articles de cette boite ne sont pas aussi simplistes que leur gérant. En effet, les articles rédigés sont obligatoirement réalisés par les personnes que BERD veut. Et c'est ainsi que BERD commença un business plus sombre qu'on ne le pense...

La plus grande entreprise de journalisme tient en captivité les acteurs des articles de journaux et les force à rendre les articles véridiques.

Fake it until you make it..., comme diraient certains.

Florida-Man, un être humain de Floride d'apparence lambda fait partie des captifs depuis plusieurs années. Et après toutes ces années de captivité, il a décidé de prendre son destin entre ses mains et s'enfuir, quitte à risquer une fin terrible...

Les personnages

De par l'aspect aléatoire de la génération de niveau, le nombre de personnages notables du jeu se réduit au personnage principal, au boss final, et au(x) vendeurs que l'on croisera dans les niveaux. Notre héros est évidemment Florida-man, preux chevalier au M16 de la liberté, conquérant des tours journalistiques, et vainqueur des Berds.

Le boss final est Le patron de Berd Inc., bien déterminé à ne pas laisser son plan tomber à l'eau, il emploiera tous les moyens à sa disposition pour monétiser vos action.

Les vendeurs sont des rescapés de la tour de l'information infernale. Ils rasent les murs et risquent leur peau pour amener au joueur de quoi augmenter ses chance de réussite. ils méritent bien une petite compensation.

Ceci étant dit, le jeu ne se limitera pas à cela. dans chaque niveau, une multitude d'ennemis attend l'aventurier, chacun avec sa manière de se battre, de se déplacer et de se protéger...

2.1.2 Le Gameplay

Notre jeu pourra se jouer avec le clavier et la souris. Il y aura de multiples options de mouvements : doubles sauts et dash viendront donner de la fluidité au gameplay, tandis que l'aspect stratégique sera assuré par le choix des objets à chaque nouveau trésor. Florida-Man étant américain, il est évidemment un roi de la gachette et se battra à moyenne distance de ses ennemis, en restant mobile. Il est prévu que la difficulté augmente proportionnellement au timer, donnant au joueur le choix entre la recherche d'équipement et

l'avancée, mais permettant également une arrivée en douceur pour les nouveaux joueurs. Le jeu disposera aussi d'un didacticiel afin de permettre au joueur de se familiariser avec les mécaniques du jeu. Les items se veulent "stackable" pour un gameplay varié et amusant.

Mais comment on fait ?

— **Génération des niveaux**
Les niveaux seront générés aléatoirement de telle sorte que à chaque fois que nous changeons de niveau, une nouvelle map apparait et reste la même pour toute la partie. Et si le joueur meurt, le nouvelle sera régénéré et donc complètement différent de la partie d'avant.

— **Colliders et mouvements du personnage**
Les lois de la physique s'appliquent à notre joueur et son envirronement. Il lui sera impossible de sortir d'un immeuble sans dégats, de bondir sur un toit ou de traverser des flammes sans aucun dégat. Et il en est de même pour ses interactions avec les ennemis. Il y aura du damage boost, c'est-à-dire que Florida-Man pourra profiter des dégats qu'il s'est pris pour foncer dans le niveau et échapper à une horde de crocodiles affamés, par exemple.

— **Les mobs et les boss**
Les boss et les mobs seront des IA, que nous créeeront et que nous perfectionneront tout le long de notre projet.

— **Les items**
Les items restent en permanence en possession du joueur jusqu'à la mort de celui-ci ou sa réussite.

Le Game Design et l'expérience de jeu, made in Deter Birb

Puisque notre gameplay se veut difficile et demandant, il est impératif que notre game design soit juste, millimétré. L'injustice doit être rare voir inexistante, afin d'éviter toute frustration. L'utilisateur doit savoir qu'il perd par sa faute et non par celle du jeu et le jeu doit pouvoir être terminé sans prendre un coup (au moins en théorie). Par exemple, nous souhaitons éviter les déplacements "type mario"

où le personnage semble glisser à chaque accélération/décélération.

Il est également important que le jeu ne soit pas trop facile, ce qui le rendrait à notre avis inintéressant. La difficulté attire les bon joueurs, et leur bon gameplay attire les viewers d'éventuels streams, tout spécialement si le jeu a la réputation d'être compliqué.

Les outils aboutissant à notre projet

Le jeu utilisera le moteur Unity3D, rendant ergonomique le level-design. les mécaniques propres à notre jeu seront codées en c# sur des fichiers lus par unity. Nous avons choisi un style d'animation pixel-art, dans une volonté de ne pas utiliser d'assets préfaits. Les sprites seront donc dessinés et animés sur piskel, logiciel gratuit.

2.1.3 Coûts et Délais

Notre approche au développement

Notre volonté est de produire la plus grande partie possible du projet par nous-même, par exemple, les sprites, artworks et effets sonores. Nous nous attendons donc à un coût réduit.

Cependant, créer un jeu vidéo demande un domaine de compétences très vaste, et il pourrait s'avérer impossible pour nous d'assurer certaines parties de la production. Par exemple l'expérience musicale du groupe est très limitée, et acheter des musiques peut demander un investissement conséquent. Le coût du projet dans sa globalité reste pour le moment largement indéterminé.

Concernant les délais, nous nous étions donné une avancée de 30% à chaque soutenance jusqu'à atteindre les 100% à la soutenance finale.

Chapitre 3

Ce qu'il s'est finalement passé...

Première partie

Nos actions et ressentis

3.1 Première soutenance

3.1.1 MrZed

Lors de la première période de travail, il m'a été difficile de réaliser l'ampleur du projet, ainsi que les attentes particulières de l'école.

Ce que j'ai fait

J'ai malheureusement pensé que le travail dans le logiciel, par exemple la création de tilemaps, un outil de game design, serait du travail valorisé. J'ai donc passé plus de temps que de raison à tenter de créer des niveaux, des salles, des sprites...

Pour cette première soutenance, j'ai essayé de coder les déplacements du joueur, en coopération avec Daniel, qui a plus tard récupéré cette tâche dans son ensemble, pour plusieurs raisons : premièrement, il était responsable de cette partie, Il était également plus inspiré en terme de code, il a été jugé que sa version était supérieure, et je me devais de m'occuper de ma partie du projet.

Mes joies

Je n'avais jamais touché, ne m'étais jamais intéressé à des outils tels que unity3D. La découverte du déroulement du projet, et de la manière dont un jeu était créé m'ont permis de me rendre compte que le procédé nécessitait des compétences dans plusieurs domaine que j'apprécie : musique, dessin, mise en scène, modélisation (même si ces deux derniers éléments ne sont pas pertinents dans le cadre du projet Florida Man vs The Berds.), et bien sûr programmation. La manière dont unity3D traite les scripts m'a également appris plus avant la logique et le fonctionnement de la programmation orienté objet en c, ce qui a été d'une grande aide pour les examens sur ce langage.

Il est toujours satisfaisant de voir quelque chose dans lequel on a investi du temps et de l'énergie prendre vie, et un semblant de jeu vidéo ne fait pas exception. Pouvoir contrôler les mouvements d'un sprite, voir que les collisions sont fonctionnelles sont de petites victoires que je garderai en mémoire.

enfin, le travail d'équipe, s'il est moins confortable et flexible qu'un projet en solo, a apporté une rigueur nouvelle a mes horaires

et méthodes. Pouvoir et devoir faire confiance a autrui pour un travail complémentaire au sien est un saut de la foi, mais est souvent un soulagement. Le sentiment de responsabilité induit par la confiance d'autrui est aussi une source de motivation supplémentaire (en ce qui me concerne, ma propre motivation, même pour un projet que j'apprécie, n'est parfois pas suffisante pour me mettre en mouvement).

Mes peines

j'ai réalisé lors de cette première soutenance que la création d'un jeu était un projet intéressant et enrichissant, mais également fastidieux, et également que le travail en équipe avait parfois ses désavantages. Cependant, malgré de légers différends, l'équipe est restée soudée et nous avons pu travailler en collaboration plutôt qu'en conflit.

3.1.2 The Queen

Cette première soutenance était un peu éprouvante pour moi. La dure réalité de faire un jeu entier m'a frappé de plein fouet.

Ce que j'avais à faire

Pour la première soutenance, je devais faire un tiers du code des IA et un tiers du code des items. Ainsi, lors de la présentation, j'ai pu montrer un ennemi bougeant de droite à gauche frénétiquement et expliquer le fonctionnement du pathfinding.

Mes joies

Mes premières joies furent lors d'un rush, quelques jours avant la soutenance. J'ai enfin pu voir mon IA patrouiller (animations non inclues) et même la voir suivre le joueur ! Ce fut relaxant.

Mes peines

Mes peines se trouvaient surtout dans l'apprentissage de Unity à ce moment-là. Le logiciel paraissait compliqué, dur à prendre en main et tellement complet qu'il en devenait trop bien pour le comprendre et le maitriser. En plus d'essuyer de nombreux échecs quant

au fonctionnement de mes IA. J'avais donc créé des classes multiples et variées (Entity, Enemy, Item et j'en passe) qui n'ont même pas été utiles. Cela m'a fait perdre beau beaucoup de temps et d'énergie pour rien.

3.1.3 Barbarossa

Ce que j'ai fait

Pour la première soutenance je me suis principalement documenté sur Unity, j'ai d'abord essayé de comprendre comment marche les vecteur dans Unity et comment prendre en compte les input du joueur. J'ai au final fini par faire un script "place holder" grâce à l'aide de plusieurs Tuto sur youtube et Unity learn. Celui-ci permettait au joueur de courir,sprinter,sauter et faire des doubles sauts, ces features se veraient réimplémenter dans mes future script de mouvement pour le joueur.

Mes joies

J'ai appris beaucoup de chose qui me seront utile plus tard dans le projet et commencé à bidouiller le moteur de jeu s'est vue être très amusant.

Mes peines

Ma principal peine vis-à-vis de la première soutenance était que les scripts que j'avais produit ne venait pas entièrement de moi, ce qui est normal puisque je n'avais aucune notion sur comment utilisé Unity. Une autre chose que j'ai trouvé pénible est l'apprentissage sur Unity, en effet malgré tout les tutos disponible sur Internet, on se retrouve vite perdu et c'est très frustrant quand on passe des heures à implémenter une fonction pour au final se rendre compte que Unity offre des moyens bien plus simple pour la faire.

3.1.4 L'Otarie

Ce que j'ai fait

Pour la première soutenance, en tant que responsable de la génération des niveaux, j'ai commencé la conception d'un algorithme

destiné à générer des niveaux aléatoirement a partir de salles préfabriquées. Mon travail s'est concentré essentiellement sur cet algorithme pour la première soutenance. Mon but était de l'avancer le plus possible pour pouvoir partir d'une base solide pour construire mes niveaux.

Mes joies

La conception d'algorithme est ma partie favorite dans un projet informatique comme celui-ci. J'apprécie réfléchir à la conception d'algorithme, essayant de les optimisés par la suite. Travailler sur cet algorithme m'a certes apporté une certaine satisfaction mais m'a surtout permis de m'améliorer.

Mes peines

Mon principal remord de la première Soutenance c'est de ne pas avoir pu tester mon algorithme, en effet, n'ayant pas encore de salles préfabriquées ni de joueur se déplaçant aisément il fut compliqué de tester mon code.

3.2 Deuxième soutenance

3.2.1 MrZed

Avec le printemps est arrivé le virus, et considérer sa chambre d'adolescent comme un espace de travail a demandé un effort cognitif tous les matins. Heureusement, les outils à notre disposition nous ont permis d'être en communication régulière et de partager tout ce dont on avait besoin

Ce que j'ai fait

Lors de la deuxième période du projet, je me suis concentré sur le système d'inventaire, et la gestion des objets du jeu, de leurs effets, de leur stockage et de leur distribution. J'ai également créé les classes Player et Entity, qui avaient pour buts respectifs de définir les statistiques du joueur et des entités, mais elles ont été abandonnées au profit des classes similaires créées par Fatou et Daniel, celles-ci étant plus compatibles avec leurs travaux.

Mes joies

malgré les conditions sans précédent de cette période de développement, j'ai tout autant apprécié le codage et la création de plus en plus de sprites pour le jeu qu'auparavant.

mes peines

La motivation a parfois été compliquée à trouver, et s'installer dans un rythme de travail également. J'avoue sans fierté avoir fait la plus grande parti de mon travail en 3 sessions de 12 heures, proche de la deadline, alors qu'il aurait été profitable de multiplier les courtes sessions afin d'avoir une perspective différente et du temps de réflexion latente, état dans lequel les idées affluent souvent.

3.2.2 The Queen

Cette deuxième soutenance était assez spéciale, étant donné que nous étions tous confinés chez nous.

Ce que j'ai fait

J'ai décidé de recommencer complètement mes IA, pour commencer sur de meilleures bases. J'ai donc fait des animations pour mes ennemis, des colliders et de quoi les faire attaquer et prendre des dégats. Pour les menus, que j'avais aussi à faire, j'ai pu bien avancer en faisant le menu de pause et le menu principal.

Mes joies

Faire les menus était bien plus simple que les IA étant donné qu'il s'agissait principalement de coder des boutons qui exécutent certaines actions. Et comme j'avais bien pris la main sur Unity, j'ai mieux vécu le codage des IA que pour la première soutenance (surtout quand je les voyais fonctionner).

Mes peines

Avec le confinement, on pourrait se dire qu'on a plus de temps pour avancer dans nos projets. Néanmoins, on se laisse facilement avoir par le confort du lit et des plats préparés avec amour. J'ai eu du mal à trouver un rythme de travail.

3.2.3 Barbarossa

Ce que j'ai fait

Pour la deuxième soutenance je me suis donné comme objectif de refaire totalement le script mouvement du joueur, j'ai d'abord commencé par un simple saut, le personnage pouvant sauté à l'infini, j'ai rajouté un simple test pour savoir si le personnage est sur le sol, si oui il peut sauté, sinon il ne peut pas. Ensuite j'ai implementé des simple mouvement latéraux pour pouvoir se déplacer une fois sur une plateforme. La base des mouvement de mon personnage maintenant implémenté, il était temps de le faire attaquer, j'ai créé un simple projectile et attaché au joueur objet vide qui servira de canon, une référence pour savoir ou faire apparaître les projectile et dans quelle direction. J'ai ensuite permis au joueur de se retourner afin de pouvoir tirer dans les deux sens, et enfin j'ai implémenté un moyen pour le joueur de tirer vers le haut quand il est au sol et vers le bas quand il est en l'air. J'ai ensuite créé un script de Stat qui gère les point de vie du personnage et lui permet de prendre des dégâts.

Mes joies

Cette soutenance était pour moi l'occasion de faire les chose bien et de refaire de A à Z mes scripts, je me sentais particulièrement fier lorsque ça marchait car je l'avais fait de mes propre mains sans trop de Tuto, juste un peut de documentation Unity sur comment fonctionne certaine fonction et lesquelles je pouvais implémenter.

Mes peines

J'étais assez frustré du fait que l'on ait pas réussi a avoir un build jouable du jeu et je n'avais pass réussi à supprimer certains bug dans mon code comme le fait que le personnage se collais au mur si on sautait dans le mur .

3.2.4 L'Otarie

Ce que j'ai fait

Pour la Deuxième Soutenance je me suis d'abord concentré sur le site web, créé de A à Z par mes soins. Je me suis d'abord replongé

dans les cours d'OpenClassRooms sur les langages html et css afin de me rafraîchir la mémoire. Ayant déjà visionner ces cours en terminale la mémoire m'est vite revenue. J'ai donc débuter la création du site que j'ai codé intégralement. Pour la Seconde Soutenance j'ai atteint mon objectif de terminer le site à 50% environ. Je me suis ensuite tourné vers la génération des niveaux, plus précisément sur la façon dont les niveaux aillaient être affichés durant le jeu. J'ai envisagé plusieurs possibilités et cherché la plus optimale, soit celle qui serait la moins coûteuse en puissance de calcul durant le jeu. J'ai d'abord penser construire une matrice de salle représentant le niveau.

Mes joies

Ce fut intéressant de réapprendre l'html et le css.

Mes peines

J'ai été un peu déçu de ne toujours pas pouvoir tester réellement mon algorithme.

3.3 Soutenance finale

3.3.1 MrZed

Le résultat final n'est pas forcément celui que l'on avait espéré, cependant il est ce à quoi on pouvait s'attendre.

Ce que j'ai fait

Lors de cette dernière période de travail je me suis concentré sur la gestion de plusieurs fonctionnalités dédiées au confort de l'utilisateur, ainsi que sur l'ajout de quelques fonctionnalités de gameplay :

— la barre de vie : La barre de vie est de toute évidence une fonctionnalité clé dans tout jeu qui inclut un système de combat, puisque elle permet au joueur de connaître en permanence son état de santé, et d'en déduire les actions nécessaires (fuite, attaque...).
Par chance, unity3D possède une option de "remplissage" d'images, qui permet à l'aide d'un float de 0 à 1, de savoir quelle proportion de l'image est affichée.

afin d'afficher de manière dynamique la barre de santé, on utilise un script qui calcule le rapport (vie actuelle/vie max) du joueur, puis utilise ce rapport comme facteur de remplissage. Ces statistiques sont codées par le script "Stat" de Daniel, et l'implémentation de la barre de santé a demandé concertation :

```
healthb.fillAmount = Convert.ToSingle(stats.HP) / Convert.ToSingl
```

— le système de monnaie : malheureusement, le temps jouant contre nous, nous n'avons pas eu le temps d'implémenter une utilité à l'argent dans la version rendue du jeu. Cependant, la monnaie est parfaitement fonctionnelle en tout autre aspect, et le joueur en gagne un nombre aléatoire en tuant un monstre. la monnaie existe dans deux formes : les pièces et les billets. L'idée était d'imiter le système de beaucoup de rogue-likes modernes, où une monnaie s'utilise pendant la partie, et disparaît à la fin de celle-ci, et l'autre s'utilise entre les parties et ne disparait pas, permettant de créer un sentiment de progression. L'implémentation de la partie non-visuelle du système monétaire se fait dans la classe Inventory, celle-ci possède 2 nouvelles variables : money et currrency. on a également ajouté une nouvelle fonction : VaryMoney, qui permet de retirer ou d'ajouter de l'argent dans l'inventaire du joueur.

```
public void VaryMoney(int varcoin, int varcurrency)
{
    coins += varcoin;
    currency += varcurrency;
}
```

afin d'afficher la monnaie du joueur sur l'interface utilisateur, on utilise des zones de textes unity, dont on contrôle le texte via un script : on introduit la classe moneydisplay, possédant comme attributs les zones de texte money et currency. dans la fonction Update de ce script, on ajoute les lignes suivantes :

```
money.text = Convert.ToString(inv.coins);
currency.text = Convert.ToString(inv.currency);
```

où inv est l'inventaire que l'on lie au script dans unity.

— affichage de l'UI :
servant à la base à cacher ou montrer l'intégralité de l'UI,

ce script à été modifié pour simplement cacher ou montrer l'inventaire à chaque activation de la touche B sur le clavier. Il a été décidé de cacher l'inventaire et de le rendre grand, plutôt que de l'afficher en tout temps et le rendre minuscule et décentré. Pour afficher ou cacher l'inventaire, on utilise le script UItoggle, que l'on attache à la caméra principale :

```
public class UItoggle : MonoBehaviour
{
    public GameObject Panel;
    private bool is_inventory_showing;


    public void Start()
    {
        is_inventory_showing = true;
    }

    public void Update()
    {
        if (Input.GetKeyDown(KeyCode.B))
        {
            is_inventory_showing = !is_inventory_showing;
            ToggleInventory();
        }
    }


    public void ToggleInventory()
    {
        Panel.gameObject.SetActive(is_inventory_showing);
    }
}
```

— La DataBase
la classe database ayant désormais une taille conséquente, on ne la montrera pas dans son intégralité. Cette classe database stocke tous les objets codés jusqu'ici et permet à tous les autres éléments du jeu d'y accéder. Les objets sont codés sous le format suivant :

```
new ItemClass("Yeezys", "lightest shoes in the west", 3,
        new Dictionary<string, int>
        {
            {"triple jump", 1},
            {"price", 500}
        }),
```

les arguments sont respectivement : le nom, la description, l'id, et les statistiques. On remarquera que la description n'est utilisée nulle part. Cependant, elle aurait dû s'afficher lorsque l'on passe la souris au dessus de l'objet dans l'inventaire, comme en témoignent les deux fonctions squelette à la fin de la classe Slot.

— Le vendeur

La classe vendorscript, permet de coder le vendeur, un élément qui n'a pas pu être implémenté, léo l'ayant jugé trop compliqué à générer aléatoirement dans les salles. D'après mon design initial, il devrait y avoir un vendeur par niveau, et chaque vendeur devrait avoir 3 objets dans son inventaire. C'est ce vendeur qui devait donner une utilité à l'argent du jeu. Le personnage ayant été abandonné en cours de route, la classe ne contient pas tout le code pour en permettre le bon fonctionnement, mais voici le fonctionnement théorique imaginé :

le vendeur devait être un objet statique, non tangible par le joueur. cependant, il devait posséder un collider2D, marqué comme déclencheur. Quand les colliders déclencheurs du joueur et du vendeur entrent en contact, l'inventaire du vendeur s'ouvre, et affiche les objets et leur prix. Pour ce faire, on utiliserait une classe vendorinventory, proche de la classe Inventory, mais avec seulement 3 cases au lieu de 25 et des prix afficher. Si le joueur cliquait sur un slot(en utilisant la version complétée de la fonction squelette qui détécte les clics dans Slots), l'objet serait alors acheté, c'est à dire que :

— le coût en coins serait retiré de l'inventaire du joueur
— l'objet serait supprimé de l'inventaire du marchand et remplacé par l'objet empty (id = 0)
— l'objet acheté serait ajouté à l'inventaire du joueur, et commencerait à affecter ses statistiques

si le joueur achète les 3 objets du marchand, celui ci disparait.

Mes joies

avec l'implémentation de l'UI, j'ai pu essayer de créer quelque chose que j'appréciais théoriquement parlant, mais aussi visuellement parlant. J'ai également pu créer de nombreux sprites en pixel art pour mes camarades de projet, activité qui, malgré sa cronophagie, reste pour moi un plaisir.

Mes peines

à mon grand regret, plusieurs des choses que je voulais créer pour le jeu n'ont soit pas vu le jour, soit pas été implémentées dans la version finale, faute de temps dans le premier cas, et rapport à la compatibilité dans le deuxième. Mon travail principal visible se résume donc à l'UI et aux musiques.

3.3.2 The Queen

Pour cette soutenance finale, je n'avais qu'une mission et pas n'importe quelle mission : finir mes IA et mes menus, Simon et Daniel ayant décidé de s'atteler aux items avec joie et plaisir.

Ce que j'ai fait

Pour les IA, j'ai décidé qu'elles ne suivraient plus le joueur quand il s'approche, tout simplement car l'expérience de jeu n'aurait pas été très agréable si une horde de crocodiles nous poursuivait. Ainsi, avec les méthodes Patrolling() et PlayerAttack() dans le script Patrol, j'ai pu créer des crocodiles qui patrouillent tout en faisant attention à leur environnement :

```
void PlayerAttack()
{
    if (isAttacking)
    {
        rb.constraints = RigidbodyConstraints2D.FreezeAll;
        stat.TakeDamage(damage);
    }
    else
    {
        rb.constraints = RigidbodyConstraints2D.FreezeRotation &
        RigidbodyConstraints2D.FreezePositionY;
    }
}

 void Patrolling()
{
    transform.Translate(Vector2.right * speed * Time.deltaTime);

    RaycastHit2D sol = Physics2D.Raycast(groundDetectLeft.transform.
    position, Vector2.down, 2f);
    RaycastHit2D gauche = Physics2D.Raycast(groundDetectLeft.transform.
    position, Vector2.left, 1f);
    RaycastHit2D droite = Physics2D.Raycast(groundDetectLeft.transform.
    position, Vector2.right, 1f);

    if (sol.collider == false)
    {
        if (goingRight)
        {
            transform.eulerAngles = new Vector3(0,-180,0);
            goingRight = false;
```

```
            }
            else
            {
                transform.eulerAngles = new Vector3(0,0,0);
                goingRight = true;
            }
        }
        else if (droite.collider)
        {
            transform.eulerAngles = new Vector3(0,-180,0);
            goingRight = false;
        }
        else if (gauche.collider)
        {
            transform.eulerAngles = new Vector3(0,0,0);
            goingRight = true;
        }

    }
```

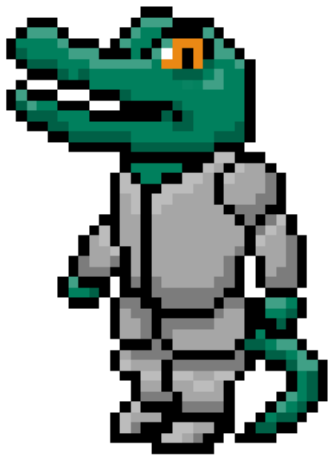
Et dans la méthode Update(), je gère les attaques et animations :

```
private void Update()
    {

        if (Math.Abs(target.transform.position.y -
        transform.position.y) <= 3f)
        {
            if (Math.Abs(target.transform.position.x
            - transform.position.x) <= 2f)
            {
                LookAtPlayer();
                anim.SetBool(Attack, true);

                PlayerAttack();
            }
            else
            {
                anim.SetBool(Attack, false);
                Patrolling();
```

```
            }
        }
        else
        {
            anim.SetBool(Attack, false);
            Patrolling();
        }

    }
```

Dans cette méthode, avec le booléen "Attack", je change l'animation de l'ennemi : Si Attack est à true, alors l'ennemi va attaquer. Sinon il avance.

Le script Patrol() marche pour les ennemis qui sont des crocodiles :

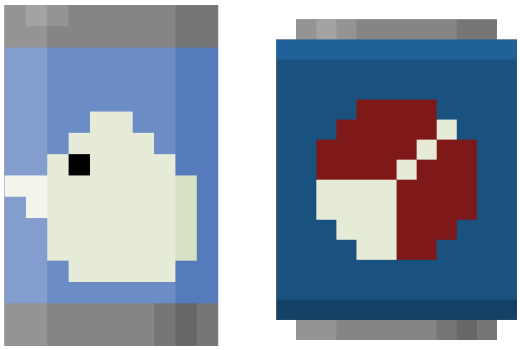


Pour les ennemis un peu moins animaliers, j'ai créé un autre script, fonctionnant de la même manière que Patrol, sans Patrolling() : Mecha.

Ce script est reservé aux machines à café qui versent du café sur le joueur dès qu'il est à portée :

Pour les boss, j'en ai créé deux de plus : la cheffe des machines à café et la secrétaire.

La cheffe des machines à café est, en apparence, une simple machine à café quelque peu hostile. Néanmoins, cette machine-là ne verse pas de café : elle lance des collations.



Son code est dans BossMachine, qui est assez spécial de par sa

simplicité :

```
public int vie = 250;
public int damage = 3;
public float speed = 3f;
private float fireRate;
private float nextFire;

public GameObject firePoint;
public GameObject bullet;

// Start is called before the first frame update

void Start()
{
    fireRate = 2f;
    nextFire = Time.time;
}

// Update is called once per frame

void Update()
{
    if (Time.time > nextFire)
    {
        Instantiate (bullet, firePoint.
        transform.position, Quaternion.identity);
        nextFire = Time.time + fireRate;
    }
}
```

Les flottants fireRate et nextFire permettent de réguler le lancer de cannetes. Les GameObjects firepoint et bullet permettent d'instancier nos projectiles dans Update() à la ligne "Instantiate". Le firepoint est placé juste à coté de notre machine, c'est de là que les projectiles viennent. Et GetHurt() permet de lui faire des dégats.

Pour que les projectiles soient fonctionnels, j'ai créé un prefab "Projectile". Ce prefab possède un script : Bullet.

Dans ce script, preque tout se passe dans la méthode Start(), comme tout doit s'instancier quand le projectile est créé (cf. Up-

date() dans BossMachine) :

```
    public float moveSpeed = 1f;

private Rigidbody2D rb;
private GameObject target;
private Vector2 moveDirection;

private int anim = 0;

private static readonly int Can1 =
Animator.StringToHash("can1");

private static readonly int Can2 =
Animator.StringToHash("can2");

// Use this for initialization

void Start ()
{
rb = GetComponent<Rigidbody2D>();
target = GameObject.FindGameObjectWithTag("Player");

moveDirection = (target.transform.position
- transform.position) * moveSpeed/2f;

rb.velocity = new Vector2 (moveDirection.x, moveDirection.y);

if (anim % 2 == 0)
{
gameObject.GetComponent<Animator>().SetBool(Can1,true);
gameObject.GetComponent<Animator>().SetBool(Can2,false);
}
else
{
gameObject.GetComponent<Animator>().SetBool(Can1,false);
gameObject.GetComponent<Animator>().SetBool(Can2,true);
}

anim = (anim + 1) % 10;
```

```
}
```

L'integer anim est un compteur permettant de choisir quelle animation va se lancer entre une première canette géante et une deuxième, tout aussi grosse.

Ces animations (et celles de tous les autres Boss et ennemis) ont été faites grâce au component Animator et les Animation controller de Unity. Dans ce cas précis, cet Animation controller a 2 booléens "can1" et "can2" qui passent true ou false selon la valeur de anim.

La secrétaire est une femme assez classique mais très hargneuse. Elle n'hésitera pas à poursuivre le joueur pour lui mettre des coups de téléphone :

Elle a deux scripts qui lui permettent de fonctionner : Boss et BossAttack. Grâce deux GameObjects, Attack Check et Phone, étant les fils du GameObject pricipal Boss2, elle peut mettre des coups dès que le circle collider d'Attack Check est en contact avec le joueur.

Boss est composé de plusieurs méthodes : Start(), Update(), LookAtPlayer() (qui permet de regarder le joueur ou de se tourner vers lui), Attack() (qui fonctionnent comme PlayerAttack() dans Patrol), GetHurt() (qui lui permet de prendre des dégats et de charger un niveau quand elle meurt) et Following().

Following() permet de poursuivre le joueur :

```
void Following()
{
    if (Math.Abs(player.transform.position.x
    - transform.position.x) >= 1f)
```

```
    {
        var position = transform.position;

        position = Vector2.MoveTowards
        (position, new Vector2(player.
        transform.position.x, transform.position.y) ,
        speed * Time.deltaTime);

        transform.position = position;
    }
}
```

Cette ligne permet d'éviter que le boss s'envole si le joueur saute :

```
new Vector2(player.
transform.position.x, transform.position.y)
```

Et le script "attack" permet de faire des dégats au joueur.
Pour les menus, j'ai créé une autre scène : WinMenu.
Cette scène s'active quand tous les boss on été battus et elle nous propose de retourner au menu principal.
Quand on perd, nous sommes directement ramené au menu principal.

Mes joies

Voir mon travail enfin terminé est extrêmement satisfaisant et fait vraiment plaisir.

Mes peines

Le sommeil n'a pas vraiment été au rendez-vous.

3.3.3 Barbarossa

Ce que j'ai fait

Tout d'abord, pour régler le bug mentionné auparavant, j'ai rajouté un matériaux physique sans friction pour le joueur afin de glisser le long des mur. J'ai ensuite commencer à utiliser des Coroutine pour que le joueur ne puisse pas "spammer" les projectiles à toute les frames.

```
IEnumerator primary_fire()
{
    if (can_fire)
    {
        can_fire = false;
        anim.SetBool("Is_Attacking",true);
        Instantiate(Projectile, FirePoint.transform.position, FirePoint.tr
        yield return new WaitForSeconds(0.2f);
        can_fire = true;
    }
}
```

J'ai ensuite implémenter un tir secondaire qui fait reculer le joueur quand il est utiliser :

```
IEnumerator secondary_fire()
{
    if (can_fire)
    {
        anim.SetBool("Is_Attacking",true);
        can_fire = false;
        if (orientation == "right")
        {
            rb2d.velocity = Vector2.left * knockback;
        }
        else
        {
            rb2d.velocity = Vector2.right * knockback;
        }
        Instantiate(Projectile2, FirePoint.transform.position, FirePoint.t
        yield return  new WaitForSeconds(0.25f);
        rb2d.velocity = Vector2.zero;
        yield return new WaitForSeconds(0.25f);
        can_fire = true;
    }
}
```

J'ai ensuite rajouté les animations que j'avais pour le joueur et j'ai commencé a m'intéresser a l'inventaire. Grâce à Simon j'ai pus implémenter de manière simple l'inventaire, avec un script par objet qui applique l'effet de l'objet et qui le rajoute dans l'inventaire :.

Voici l'exemple des Yeezys :

```
    public class Yeezy : MonoBehaviour
{

   private void OnTriggerEnter2D(Collider2D other)
   {
      Player_Control player = other.GetComponent<Player_Control>();
      if (player != null)
      {
         player.Inventory.AddItem(3);
         Destroy(gameObject);
      }
   }
}
```

Ceci m'a permis d'implémenter un triple saut et un sprint si on a certains items dans l'inventaire. J'ai ensuite créé d'autres projectiles pour l'effet de certains objets comme le casque de Storm Trooper qui remplace le tir secondaire par un laser qui a le même script que le projectile de base mais plus rapide et avec plus de dégâts. J'ai ensuite gérer la mort du personnage qui fait revenir le joueur au menu principal. Enfin pour finir j'ai créé un prefab de coffre pour pouvoir obtenir des items autrement qu'en tuant les boss et j'ai aider les autres membre de mon groupe à debugger leur code et set up le jeu.

Mes joies

J'ai appris beaucoup de chose lors de ce projet, malgré le fait que ma partie soit centré sur le joueur le rush de fin de projet m'a permis de comprendre une partie de ce que mes partenaires ont fait. J'ai beaucoup aimer le processus de conception de jeu vidéo et je continuerai sûrement a l'avenir à faire des petits jeux dans mon temps libre.

Mes peines

Le rush de fin s'avéra compliqué notamment lors de la mise en place de la génération de niveau mais dans l'ensemble le projet m'a

surtout permis d'acquérir des bases en Unity afin de pouvoir faire des jeux plus abouti dans le future.

3.3.4 L'Otarie

Ce que j'ai fait

J'ai créer les 81 tilemaps qui ont été utilisées pour créer les niveaux du jeu. J'ai finalisé mon algorithme de génération pour finalement me rendre compte qu'il ne fonctionnait pas. Le constructeur ne remplit apparement que la première salle (celle ou le joueur apparait) mais pas les autres. J'ai passé plusieurs heures à tenter de régler ce problème mais je n'ai pas réussi ne serait-ce qu'a comprendre d'où vient le problème. En parallèle au developpement de l'algorithme j'ai également créé des objet possédant un BoxCollider2D avec un trigger, ce qui permet lorsque le joueur les touches d'éxécuter des instructions données dans un script c. J'ai donc créer 4 objets de ce type afin qu'ils servent de "portes" pour passer d'une salle a l'autre. Malheureusement ces scripts étant liées directement a l'algorithme de génération ils n'ont pas été utilisés non plus. Pour finir j'ai finis le site web.

Mes joies

Pas de réelle joie, je suis déçue que mon travail n'ai pas pu être utilisé pour ce projet. J'ai néanmoins essayer tant bien que mal de le faire fonctionner, je n'ai pas de regret de ce coté la.

Mes peines

Je reste assez frustré que la partie la plus intéréssante de mon travail sur ce projet soit inutile et manifestement incomplète. Il y a tout de même une trace de ma participation au projet avec les 81 salles que j'ai créer mais je suis déçu car ce fut un travail long et répétitif mais pas complexe dans le fond.

Deuxième partie

Chronologie et historique

3.4 Chronologie

Janvier :
— Découverte de Unity
— Arrivée de Léo dans le groupe
— Rédaction du cahier des charges et mise en commun des idées et envies pour le projet

Février :
— Premiers assets créés
— Premières lignes de code écrites
— Prise en main (un peu faiblarde) de Unity

Mars :
— Ennemis créés
— Personnage jouable codé en partie
— Items pris en compte dans le script du joueur
— Génération de niveau imaginée et mise sur papier
— Première soutenance et conclusion des 3 premiers mois de travail : beaucoup de travail encore
— Menus créés
— Site iternet créé

Avril :
— Prise en main sérieuse de Unity
— Binge-watching de tutoriels Unity et lecture intensive de forums pour être efficace.
— Nombreuses tentatives de faire un répertoire Git fonctionnel à 100%
— Création de niveaux pour la génération
— Joueur encore plus fonctionnel
— Menus continués
— Filmage de la vidéo
— Deuxième soutenance et conclusion du travail en Avril : on avance bien
— Site internet normalement hébergé (en vain)

Mai :
— Ennemis terminés
— Joueur terminé
— Items designés, codés
— Salles créées
— Git troqué pour les mails, Discord et WeTranfer

— Nouvel hébergeur trouvé pour le site
— Préparation pour la dernière soutenance

3.5 Historique

Janvier à milieu Mars : Le projet débute.
L'équipe s'attelle à l'apprentissage de Unity et sa maîtrise. Les ennemis arrivent, les mouvements du joueur sont reliés aux touches du clavier, la génération de niveaux est mise sur papier est imaginée, les items sont imaginés aussi.

Milieu Mars à Fin Avril : La première soutenance est passée. L'équipe commence les menus et le site internet. Les ennemis sont artificiellement intelligents, le joueur est bien plus fluide, les items avancent, la génération de niveau prend forme.

Fin Avril à fin Mai : Le projet entre dans sa phase finale.
Les ennemis sont finis, les items sont fonctionnels, la génération de niveau fonctionne, le joueur tire des projectiles et reçoit les effets des items ainsi que les attaques des différents ennemis, les menus sont fonctionnels, le jeu possède des musiques.

Chapitre 4

Les problèmes que nous avons eu et leurs solutions

Au début, nous nous étions focalisé sur nos parties respectives, ce qui a causé des problèmes d'organisation assez important.

La solution : S'appeler bien plus régulièrement, partager notre travail bien plus régulièrement, de toutes les manières possibles.

En voulant rassembler notre travail, nous avons remarqué une chose que nous avions oublié, qui nous était pas venu à l'esprit : la version de Unity que nous comptions utiliser. Certains étaient sur du 2018, d'autres du 2019 et cela causait des problèmes.

La solution : Choisir une version de Unity : la version 2019.3.4f1.

Nous avions créé un repo Git pour notre projet au début du semestre. Nous avions créé des branches et nous pushions tous sur notre branche attitrée. Mais dès que nous voulions merge sur la branche master, les conflits se révélaient, mettant tout notre Git à mal. Nous avons créé pour repo pour qu'ils finissent tous de la même manière : abandonnés quand le git merge ne marchait pas sur la branche master.

La solution : Les mails, Discord et de très longs appels nous ont permis de mettre notre jeu en place, de leur rendre fonctionnel.

Le site était originellement hébergé grâce à un site : Hostinger. Néanmoins, le site a été mystérieusement retiré.

La solution : Faire jouer ses contacts et trouver un autre serveur.

Chapitre 5

Ce que ce projet nous a apporté

5.1 MrZed

Ce projet m'a permis de pratiquer le travail en groupe dans un cadre plus sérieux et durable que celui d'un oral de TE ou d'anglais à plusieurs. Cette notion est quelque chose que l'on acquiert rarement durant le lycée, et je suis content de cette expérience, certain qu'elle est positive.

En termes de programmation, coder des éléments d'un jeu m'a beaucoup aidé à concevoir le fonctionnement de la programmation orienté objet, ainsi que dans une certaine mesure, la logique de programmation dans on ensemble.

J'ai appris la fonction de game dev, et gagné un respect nouveau pour la profession. Je pense que je saurai d'autant plus apprécier l'originalité d'un jeu et la quantité de travail que certains éléments auront demandé.

Finalement, ce que j'ai le plus fait pendant ce projet, c'est chercher sur internet, trier les résultats pertinents, tenter d'exprimer mes questions en anglais pour plus de réponses... Tous ces aspects de la recherche sur le web sont de petites compétences auxquelles je suis sûr d'être devenu meilleur. Je trouve en moyenne beaucoup plus rapidement ce que je cherche.

5.2 The Queen

Ce projet m'a fait prendre conscience de beaucoup de choses techniques : mes capacités à apprendre vite, à coder des IA de A à Z par exemple. Mais il m'a permis d'en apprendre plus sur moi-même, savoir ce qui va me faire stresser et ce qui va rien me faire : la perte d'un repo Git ne me fera pas grand chose, mes capacités à rester éveillée me surprennent à chaque seconde mais m'inscrire sur un site pour héberger un site qui n'est au final pas hébergé me met dans une colère noire.J'ai aussi vu ce que je devais améliorer pour rendre mes prochains projets groupés encore plus efficaces et agréables. Je me suis lancée dans ce projet sans savoir à quoi m'attendre et j'en ressors grandie. Certes, ça n'a pas été simple. J'avais l'impression qu'on nous jetait dans une fosse aux lions avec un livre comme seule arme pour survivre. Mais au final, j'en sors que plus forte et plus mature.

5.3 Barbarossa

Ce projet m'a permis de développer des base dans un secteur que j'ai toujours aimer qui est le développement de jeu vidéo, j'ai toujours voulu faire des petits jeu mais je n'ai jamais eu le courage de me lancer dedans mais avec ce projet j'ai pu apprendre les bases de Unity, qui certes ne me permette pas de faire d'excellents jeux pour l'instant mais sont une porte d'entrée pour le développement de futurs jeux.

5.4 L'Otarie

En conclusion ce projet m'aura beaucoup appris sur le travail en équipe. Il m'aura aussi permis de prendre conscience que le développement de jeu vidéo ne m'intéresse absolument pas, du moins pas sous cette forme. La seule partie qui m'a vraiment intéressé dans ce projet fut la réflexion pour la création d'un algorithme. Je n'ai vraiment pas apprécier l'outil Unity. Ce projet reste tout de même une expérience enrichissante tant pour le travail en équipe que pour affiner mes objectifs et mieux comprendre ce que j'aime vraiment faire dans l'informatique. Ce projet m'a aussi fait prendre conscience qu'il est urgent que j'apprenne à utiliser GitHub.

Chapitre 6

Conlusion