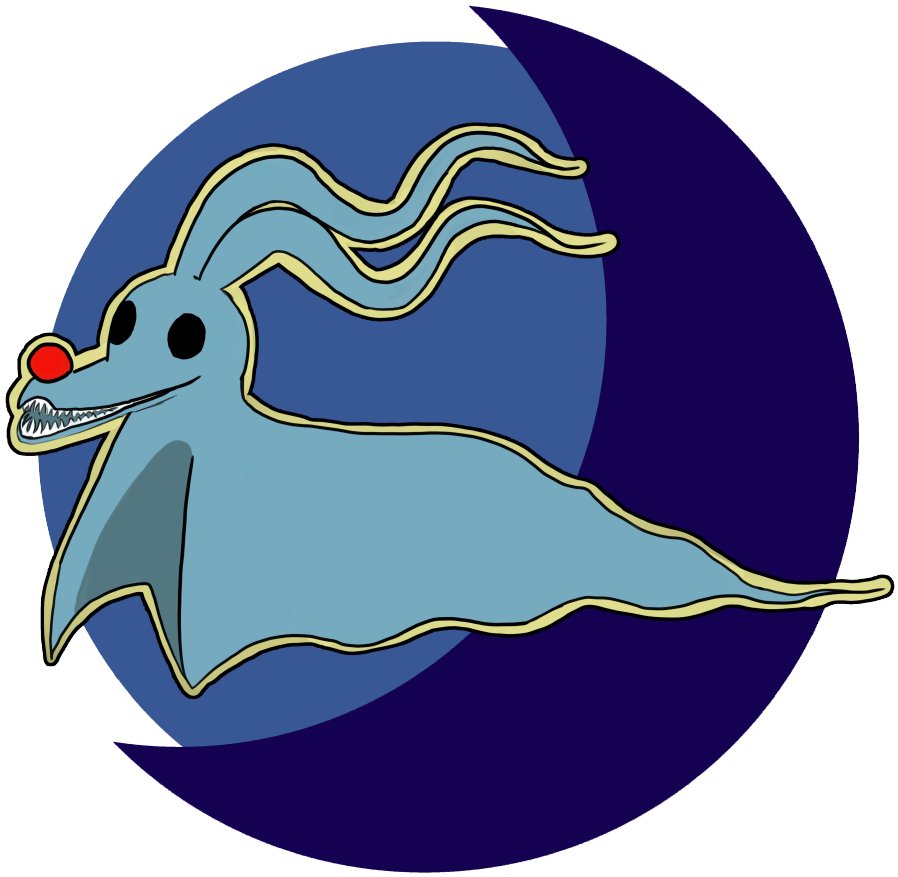
Rapport de projet final
Version lite



par

TAAG TEAM

composé de :

Thomas Solatges
Arnaud Corcione
Axelle Destombes
Gwennan Jarno

Table des matières

1 Histoire du projet

1.1 Origine du projet

La création d'un jeu vidéo fut une envie commune dans le groupe. L'idée d'un jeu semblable à The Binding of Isaac est ensuite apparu comme une évidence au sein du groupe.

Toddlers Are Afraid est donc un jeu de type roguelike. Le joueur affronte des monstres et parcourt des niveaux générés de façon procédurale.

À chaque fois qu'il perd une partie, le joueur doit recommencer depuis le début dans une nouvelle, sans aucun inventaire.

1.2 Objectif du projet

D'un point de vue collectif, ce projet a eu pour but de nous faire travailler en groupe, ainsi que de nous faire développer un esprit d'équipe et d'entraide face aux problèmes rencontrés. Objectif qui s'est avéré réussi.

De plus, la création d'un jeu nous a permis d'acquérir à la fois de nouvelles connaissances en développement mais aussi en organisation.

D'un point de vue individuel, chacun souhaitait découvrir et approfondir les tâches et parties du jeu qui lui avaient été confiées. Cet objectif a lui aussi été atteint.

L'objectif commun de répondre aux attentes et contraintes imposées et d'avoir une bonne note semble atteint. Nous sommes satisfaits de notre jeu, qui respecte notre cahier des charges, et de nos notes jusqu'à présent.

1.3 Inspiration et références

1.3.1 The Binding of Isaac

The Binding of Isaac est un jeu paru en 2011 et développé par Edmund McMillen.

C'est à ce jour un des roguelike les plus connus, puisqu'il est toujours mis à jour plus de 10 ans après.

Ce jeu ainsi qu'Enter the gungeon, dont nous parlerons ensuite sont des twin stick shooter, ce qui signifie que le joueur contrôle le mouvement du personnage d'une main et avec l'autre, la direction vers laquelle les projectiles.

L'origine du jeu est une Game Jam sous flash, pendant laquelle le créateur voulait développer un jeu qui n'attirerait personne. Une mécanique principale du jeu est les combinaisons d'items, qui sont à ce jour au nombre de 1200. Cela permet une quasi-infinité de combinaisons et une re-jouabilité poussée à l'extrême.

La communauté alimente aussi le jeu de contenus, allant du simple mod pour avoir des descriptions des items, à des histoires complètes. Certaines de ces histoires ont même réussi à se faire implémenter dans des extensions du jeu.

1.3.2 Enter the gungeon

Enter the gungeon est un roguelike qui rentre aussi dans le genre du bullet hell.

Ce dernier est un genre de jeu qui se caractérise par une grande quantité de projectiles présents à l'écran que le joueur doit éviter.

Ce jeu a été développé par Dodge Roll en 2016. Enter the gungeon est radicalement différent dans sa manière d'être joué de The Binding of Isaac notamment à cause de l'aspect très rapide et réactif du jeu.

Une mécanique importante du jeu et qui améliore son rythme est l'esquive (en anglais dodge roll, comme le développeur), qui est primordiale à la réussite du jeu, permettant de sauter par-dessus des obstacles ou des projectiles inévitables autrement.

Beaucoup de salles sont équipées de téléporteurs qui permettent une navigation plus rapide des niveaux. Ces derniers sont plus longs que les étages de The Binding of Isaac. Par exemple, il faut environ 10 à 15 minutes pour finir un niveau d'Enter the gungeon contre 5 à 7 minutes environ pour chaque étage de The Binding of Isaac.

2 Présentation générale du projet

2.1 Répartition des tâches

	Thomas	Arnaud	Axelle	Gwennan
Level Design	Responsable	Suppléant		
Gameplay			Responsable	Suppléante
IA		Responsable	Suppléante	
Immersion	Responsable		Suppléante	
Gestion des Contrôles			Responsable	Suppléante
Graphisme	Suppléant			Responsable
Réseau	Suppléant	Responsable		
Communication		Suppléant		Responsable

2.2 Le Level Design

2.2.1 La génération

Par Thomas

Le level design est un point essentiel dans la conception d'un jeu. C'est grâce à cette étape que le jeu est jouable et intéressant pour le joueur.

Dans un jeu « classique », chaque niveau est réfléchi avec grand soin pour ne pas frustrer le joueur tout en lui offrant un défi intéressant. Mais nous ne faisons pas un jeu classique, nous faisons un Rogue Like, ce qui veut dire que le niveau est généré aléatoirement à chaque nouvel essai. Quid donc d'un niveau parfaitement orchestré, il faut que le niveau soit intéressant à chaque nouvel essai. Cette façon de faire à l'avantage de créer une re jouabilité presque infinie, au détriment d'un petit peu de cohérence dans le monde. Ainsi, l'intérêt pour Thomas sur toute la partie du level design a été de trouver le bon équilibre entre cohérence et fun.

Au début de ce projet, Thomas a donc naïvement tenté une méthode en suivant les éléments qui lui venaient en tête. Malheureusement pour lui, procéder de la sorte est peut-être la pire des choses à faire car on ne prend pas en compte tous les éléments au début. On se retrouve donc plutôt à jeter chaque élément les uns sur les autres, jusqu'à avoir quelque chose de fonctionnel.

De plus une fois que Thomas a obtenu une génération « satisfaisante » (et par cela il veut dire que beaucoup d'éléments ne fonctionnaient pas comme il le souhaitait, mais fonctionnait quand même), il s'est retrouvé de face au reste du jeu.

En effet, la méthode avec laquelle il était parvenu à générer le niveau s'apparentait à un cellular automata (un automate cellulaire). Cela veut dire que tout se faisait à la suite, sans conserver aucune information sur le niveau. Cela pose beaucoup de problèmes pour tout ce qui arrive après la génération d'un niveau. Thomas a donc rayé cette méthode.

Il a ensuite pensé à une autre méthode, prenant en compte tout ce dont il avait besoin avant de commencer. Pour ce qu'il a pu omettre, il s'est assuré de pouvoir toujours ajouter des fonctionnalités dans le futur sans aucun souci.

C'est donc après avoir regardé des conférences et lu des papiers de recherches (très intéressants par ailleurs) qu'il s'est lancé dans une approche matricielle. Cela veut dire qu'il a accès à une matrice qui enregistre pour chaque case toutes les propriétés d'une salle. Cela permet notamment de savoir si une salle a déjà été visitée, si le boss doit apparaitre dedans, etc. . ..

Nous allons maintenant entrer dans les détails techniques de la génération du niveau.

Buckle up!!!

Pour commencer, on crée une matrice vide. On y place ensuite quatre culs de sacs, qui serviront de salles spéciales.

Ces culs de sacs sont positionnés de la sorte : un en haut à gauche, un en haut à droite, un en bas à gauche, un en bas à droite de la matrice.

On prend bien soin de stocker leurs coordonnées. On va ensuite tracer le chemin qui va des culs de sacs vers le milieu, qui est le point départ du joueur. Pour cela, on part des coordonnées précédemment stockées, et on se dirige vers le milieu, en se déplaçant d'une case à la fois horizontalement ou verticalement (choisis aléatoirement).

A partir de là, on définit la spécialité de chaque cul de sac : ils peuvent être une salle d'item, un shop, un réel cul de sac, ou bien la salle du boss.

Une fois cela fait, il faut maintenant indiquer la direction du chemin, donc où sont les salles situées autour d'une salle.

Maintenant, presque toutes les informations sont prêtes, il ne manque plus que de donner au salles leurs coordonnées à l'intérieur d'Unity. Pour cela on prend les coordonnées x et y dans la matrice que l'on soustrait à la valeur du milieu de la matrice. Le tout est multiplié par la taille d'une salle dans Unity. Cette méthode nous permet de modifier la taille des salles sans causer de problème.

Maintenant que tout est fait, on instancie les prefabs de salles, choisies parmi les différentes disponible pour chaque type de salles, aux coordonnées unity récupérées précédemment. Ainsi, voilà, le tour est désormais joué.

Cela peut sembler très dense, mais Thomas pense qu'il est essentiel de comprendre le fonctionnement de quelque chose pour en apprécier la complexité.

2.2.2 Les portes

Par Thomas

Nous allons maintenant passer au fonctionnement des portes, essentielles pour contenir les fantômes dans les salles...

Quand le joueur arrive dans une salle qu'il n'a pas encore visité, les portes se ferment. Le joueur est donc coincé avec les fantômes jusqu'à ce qu'il les ait tous vaincus. Lorsque cet objectif est atteint, les portes disparaissent aux yeux du joueur.

Mais en réalité, la porte est séparée en deux éléments. Le premier est la porte dite réelle : c'est celle qui est présente en jeu pour bloquer les déplacements du joueur. Mais cachée en dessous se trouve le second élément, la porte dite virtuelle. Celle-ci n'est pas visible pour le joueur, mais est essentielle au bon fonctionnement du jeu. En effet, elle permet au joueur de se déplacer entre les salles, et surtout, que la caméra le suive.

2.2.3 Le trou

Par Thomas

Le trou est la plateforme entre deux niveaux. C'est un objet qui apparait après avoir affronté et triomphé d'un boss.

Il fonctionne de la sorte : quand le joueur entre dans le trou, il recharge la scène. Malheureusement, faire cela recharge tout, même notre joueur. Cela retirerait les améliorations que l'on aurait pu trouver, mais aussi toucher à la vie. Comme par exemple rendre la vie perdue. Bref un reset total du joueur.

Pour pallier ce problème, on se sert de la classe PlayerPrefs d'Unity, qui permet de stocker des données de joueur entre plusieurs parties. Nous détournons cette utilisation en stockant les informations avant de recharger le niveau et en créant un nouveau personnage avec les statistiques de l'ancien. Enfin, pour que ces informations ne façonnent pas le joueur de la partie suivante, nous effaçons toutes ces informations.

2.2.4 La caméra

Par Thomas

Elément essentiel au bon fonctionnement du jeu, la caméra permet au joueur de se voir dans la scène.

Dans notre jeu, la caméra ne se comporte pas comme la majorité des caméras que l'on connait. La nôtre est statique, et ne se déplace que lorsque le joueur change de salle.

Cela permet d'éviter que le joueur puisse voir les salles suivantes avant d'arriver dedans car la caméra fait pile la taille d'une salle.

2.2.5 La Mini Map

Par Thomas

Liée à la caméra principale, on retrouve une caméra un petit peu spéciale.

En effet, cette dernière ne voit pas exactement la même chose que son parent : elle ne voit que les contours des salles et un point en guise de joueur.

Cette camera est la responsable de la mini map, qui sert au joueur à se repérer à l'aide des salles aux alentours.

Pour la faire fonctionner, on crée des sprites qui sont sur un layer différent des autres. Ensuite, à l'aide des culling masks de la caméra, on peut indiquer à la première camera de ne pas afficher les éléments de cette couche, et à la seconde d'afficher uniquement cette couche.

2.2.6 Problèmes rencontrés

Par Thomas

La façon dont Thomas a implémenté les salles est la suivante : 95 prefab sont réparties dans différents dossiers pour les différencier. Elles sont ensuite appelées grâce à la méthode ressource fournie par monobehaviour. Ceci implique que pour tout changement impliquant l'intégralité des salles, Thomas a dû passer sur toutes les prefab une par une à la main. Heureusement pour lui, ce contretemps ne s'est produit que 3 ou 4 fois pendant tout le projet. Mais, il n'empêche que cela prenait entre 2 et 3 heures à chaque fois. S'il devait refaire le projet, Thomas prendrait cela en considération afin d'éviter ce genre d'inconvénient

Les portes virtuelles ont posé à Thomas un problème majeur.

Au contact de la porte, la caméra se déplaçait dans la direction où la porte se trouvait en bougeant à chaque fois de la taille d'une salle. Le problème était que le système de collision de Unity détectait beaucoup de collisions avec la porte et déplaçait donc la caméra plusieurs fois.

Pour pallier ce problème, au contact de la porte, le joueur se téléporte de l'autre côté de la porte. Il n'est donc désormais en contact seulement un instant.

Pour ce qui est du mouvement de camera, le mouvement souhaité était une transition fluide.

Malheureusement, la fonction Lerp fournie par Unity et qui sert spécifiquement pour cela n'est pas pensé pour des déplacements non prévus à l'avance. Les mouvements de caméra sont donc gérés salle par salle.

2.3 Le Gameplay

Lorsque notre groupe a choisi de créer un jeu vidéo, le premier aspect auquel nous avons réfléchi fut le gameplay. Nous voulions créer un jeu de type roguelike, semblable à The Binding Of Isaac. Pour y parvenir, nous avions besoin, en dehors de niveaux générés de façon procédurale, de différents éléments.

Il nous fallait tout d'abord un personnage jouable capable de se déplacer et de tirer des projectiles selon les commandes du joueur. Nous avions également des ennemis capables également de se déplacer et d'attaquer. Nous avons choisi de nous occuper de la gestion de leur comportement dans une autre section, l'IA, gérée par Arnaud.

Afin de rendre le jeu plus intéressant et moins complexe, nous voulions également ajouter des objets récupérables par le joueur. Leur but est de rajouter de la vie ou de booster les compétences du joueur. Nous avons également choisi où seraient accessibles ces derniers : une salle leur serait dédié ainsi que dans un magasin. Le joueur aurait également une chance récupérer un des deux objets les plus basiques lorsqu'il aura tué tous les adversaires d'une salle.

2.3.1 Les déplacement du personnage jouable

Par Axelle

Dans un premier temps, la personne responsable du gameplay, Axelle, créa une classe personnage afin d'y implémenter en priorité les déplacements du joueur. En effet, ses collègues avaient besoin pour commencer leur partie d'un personnage capable de se déplacer.

Un attribut speed fut ajouté à la classe Player, permettant de déterminer sa vitesse.

Dans la toute première version du déplacement, nous avions trois problèmes.

Le premier était lié à la méthode pour déterminer la direction du déplacement lorsque plusieurs touches étaient pressées simultanément.

Le deuxième était causé par l'incompatibilité du script avec un clavier QWERTY, en effet, il récupérait comme information si les touches ZQSD étaient appuyées. Ces touches sont celles couramment utilisées dans les jeux vidéo pour se déplacer lorsque le joueur utilise un clavier AZERTY. Dans le cas contraire, les touches à utiliser par défaut sont WASD. Au sein de notre groupe, les deux claviers étaient présents ce qui nécessitait une modification du script pour faciliter les tests.

Le troisième et dernier problème était le plus important, bien que nous aurions pu le remarquer lors de l'implémentation du multijoueur. Il était causé par la méthode que

nous utilisions : la vitesse de déplacement du joueur dépendait du nombre d'image affichées par secondes. Bien que cette erreur puisse rendre le jeu plus dur ou plus simple, le plus problématique serait en multijoueur, d'avoir un personnage se déplaçant bien plus vite qu'un autre.

Afin de résoudre ce dernier problème, nous avons simplement utilisé le paramètre Time.deltaTime de Unity afin de calculer la distance que le personnage doit parcourir en fonction du temps écoulé depuis la dernière image affichée.

En ce qui concerne les deux autres problèmes, ils furent réglés lorsque Axelle choisi d'utiliser l'input manager fourni par Unity. Elle créa deux axes, Horizontal et Vertical, ayant respectivement pour valeur négatives Q ou A pour l'un, S pour l'autre et pour valeur positive Z ou W pour le premier, D pour le second.

L'input manager de Unity permet la sélection d'une touche principale ainsi que d'une touche alternative pour chaque valeur de chaque axe. Nous pouvions ainsi faire cohabiter les configurations pour clavier QWERTY et AZERTY.

Le fonctionnement de l'input manager est simple. Si une touche est appuyée, l'axe prend la valeur de cette touche (positive ou négative). Si les deux sont appuyées simultanément, il prend alors la valeur 0. Ceci solutionnait notre problème en cas d'appui de plusieurs touches.

Pour effectuer le déplacement du joueur, nous avions donc simplement à lui appliquer un vecteur de déplacement prenant en coordonnées X et Y la vitesse du joueur (son attribut speed) multipliée par Time.deltaTime et par la valeur de l'axe en question (Horizontal pour X, Vertical pour Y).

2.3.2 Les attaque du personnage jouable

Par Axelle

Afin de permettre au personnage contrôlé par le joueur de pouvoir attaquer, il fallut rajouter des attributs à la classe personnage. Notre joueur ne peut attaquer qu'à distance en lançant des projectiles. Axelle a donc ajouté les attributs suivants :

- attack, la valeur de l'attaque du personnage
- shotSpeed, la vitesse de ses projectiles d'attaques
- cooldown, la durée minimum entre deux attaques
- attackRange, la durée maximale d'existence des projectiles, équivalent à la distance maximum à laquelle ses projectiles peuvent être envoyés.

Du côté du personnage jouable, il fallait créer une méthode permettant de lancer une attaque. Pour y parvenir, Axelle utilisa une fois encore l'Input manager fourni par Unity avec deux nouveaux axes : FireHorizontal et FireVertical. La direction des attaques est contrôlée avec les flèches directionnelles du clavier.

Contrairement aux déplacements qui peuvent être effectués dans toutes les directions, y compris en diagonale, les attaques elles ne doivent se diriger que dans quatre directions : haut, gauche, droite, bas.

En cas de conflit, le script fera en sorte que l'attaque se dirige dans la première direction demandée et non contrée par la direction opposée, dans le sens des aiguilles d'une montre, en partant du haut.

Ainsi, si deux flèches directionnelles de sens contraire sont enfoncées en même temps, elles sont alors ignorées. Si la flèche haut et une des flèches gauche ou droite sont pressées, l'attaque aura la direction haute. Si la flèche droite et la flèche bas sont pressées, la direction droite prendra la priorité. Et si les flèches bas et gauche sont pressées, l'attaque se dirigera vers le bas.

Le joueur doit cependant patienter avant de pouvoir attaquer à nouveau pendant la durée cooldown.

Lors de l'attaque du joueur, une instance de projectile allié est créée aux coordonnées du joueur.

L'étape suivante fut de créer les projectiles en question. Ces projectiles héritent des attributs utiles pour l'attaque du player (attack, shotSpeed, cooldown et attackRange), et possèdent également deux autres attributs : une position initiale et une direction.

Le comportement de ces projectiles est le suivant :

Ils apparaissent aux coordonnées du joueur, se déplacent à la vitesse shotSpeed dans la direction indiquée pendant la durée attackRange, si aucun obstacle n'est rencontré, ils sont détruits au bout de cette durée. Dans le cas d'une collision avec un autre objet, plusieurs cas sont alors possibles. Si l'objet en question est un ennemi, celui-ci perd de la vie proportionnellement à la valeur de attack et quelle que soit la nature de l'objet rencontré, qu'il s'agisse d'un ennemi, d'un allié, d'un autre projectile, d'un mur ou d'un obstacle, le projectile est détruit.

2.3.3 Les ennemis

Par Gwennan

Les ennemis sont tout aussi importants que les personnages. Sans ennemis, le jeu n'aurait plus de sens. Il faut donc, comme le personnage, créer l'ennemi et ensuite lui attribuer ses actions : se déplacer de façon autonome, attaquer lorsque le personnage rentre dans son champ d'attaque et mourir en fonction des dégâts subis.

Gwennan s'est chargée lors de la première soutenance de la création du script de la classe ennemi.

Ce script contient plusieurs variables tels que le nom du monstre, permettant d'identifier son type afin de lui associer plus facilement son type de déplacement et d'attaque.

Tout comme le script player, le script ennemy contient aussi la variable health, représentant la vie actuelle de l'ennemi et donc permettant de connaître son état : vivant ou mort. Cet état rend alors possible d'effectuer les actions correspondantes. Une autre variable utile de ce script est speed, la vitesse de l'ennemi. Les quatre autres variables (attack, attackRange, fireRate et shotSpeed), reprenant le même principe que le script player, sont utiles pour les attaques.

2.3.4 Les attaques ennemies

Par Axelle

Lors de la création des projectiles du joueur, Axelle a également créé en parallèle les projectiles ennemis qui possèdent les mêmes caractéristiques afin de les fournir à Arnaud pour l'IA .

La seule différence est que les projectiles ennemis infligent des dégâts uniquement aux personnages joueurs.

Pour gérer les dégâts infligés au joueur, il fallut lui rajouter un attribut : health, qui indique sa vie. Ainsi, si celle-ci tombe en dessous de 0, le joueur meurt.

2.3.5 Les items

Par Axelle

Le troisième élément principal de la section gameplay consiste en la création des objets récupérables. Afin de réduire légèrement la complexité de notre jeu, nous

avons choisi de créer des items capables d'augmenter les statistiques du personnage. Axelle a ainsi créé la classe item, comportant les mêmes attributs que le player.

Nous avons aussi voulu ajouter un objet permettant de rajouter de la vie au joueur, au-delà de celle avec laquelle il commence la partie. Il fallut donc rajouter les attributs maxHealth et bonusHealth au personnage. Cela permet nous permet de créer deux items distincts : les potions de vie rouges qui permettent au joueur de récupérer les points de vie qu'il a perdu, dans la limite de la barre de vie qu'il avait au départ, et les potions de vie bonus donnant des points de vies supplémentaires à la jauge de vie "classique". La vie bonus ne possède pas de maximum, le joueur peut l'augmenter jusqu'à l'infini tant qu'il trouve des potions bonus.

Le script des items appelle, lorsqu'un joueur rentre en contact avec un objet, une fonction du joueur avec en paramètre les caractéristiques de l'objet qui ajoute aux statistiques du joueur les caractéristiques de l'objet.

Les objets présents dans le jeu sont :

- une potion rouge qui ajoute de la vie classique (health) dans la limite du maximum maxHealth
- une potion bleue qui ajoute de la vie bonus (bonusHealth), un aspirateur qui rajoute des dégats (attack), une fleur qui augmente la distance d'attaque (attackRange)
- un champignon qui réduit la durée entre deux attaques (cooldown)
- un pistolet qui augmente la vitesse des attaques (shotSpeed)
- des baskets qui augmentent la vitesse de déplacement du personnage (speed)
- une étoile multicolore qui combine une version améliorée de tous les objets cités précédemment à l'exception des potions.

Suite à la décision de l'implémentation d'un marchand, Axelle a créé un dernier item : une pièce et ajouté un attribut gold au joueur. Cela constituera ainsi la monnaie du jeu.

2.3.6 Le shop

Par Gwennan

Elément implémenté essentiellement par Gwennan, Axelle et Arnaud n'ont pas hésité à l'aider et la conseiller.

Même si ce n'était pas dans le cahier des charges de base du jeu, Gwennan, du fait qu'elle ait passé énormément de temps sur les graphismes, a décidé de se rattraper en quelque sorte pour cette dernière soutenance en codant plus.

Ainsi, voici comment marche le système de shop.

Le player rentre dans la salle d'item et se rapproche du marchand. En se rapprochant suffisamment, et en cliquant sur la barre d'espace, le canva du shop s'ouvrira. Ainsi, le joueur sera capable, dans la limite de son argent, de dépenser ses pièces afin de posséder des boosters qui pourront l'aider grandement à finir le niveau. Pour quitter ce shop, il suffit juste de presser de nouveau la barre d'espace. Le canva du shop disparaitra ainsi et le joueur reviendra dans la salle d'item où il était précédemment.

De plus, grâce au script d'Arnaud pour le menu Pause, Gwennan a pu relativement facilement faire apparaitre et disparaitre le canva du shop. Mais un problème s'est tout de même posé de son côté car même avec le code juste et l'implémentation juste sur Unity, le shop ne voulait pas apparaître, même sous les conseils d'Arnaud.

Pour ce qui est du fonctionnement du shop, Gwennan n'a pas eu de difficultés à l'implémenter seule. Il suffisait seulement de gérer l'ajout des capacités et les retrait de l'argent du player en fonction de sa monnaie et de là où il cliquait.

Ainsi, c'est avec joie que Gwennan, avec l'aide d'Arnaud et les conseils d'Axelle, a implémenté entièrement le système du magasin du marchand.

2.4 L'IA

L'intelligence Artificielle, développé par Arnaud, a été pour notre jeu une des parties qui a pris beaucoup de temps, d'efforts mais aussi qui a causé de nombreux problèmes tout au long du projet.

Il y a eu beaucoup de modifications et de reprises à zéro du code dues à de nombreux bugs persistants. Mais après de longues heures de travail acharné, nous avons enfin obtenu une IA basique mais qui marche convenablement.

2.4.1 Les ennemis

Par Arnaud

En plus de la map, les ennemis constituent une partie importante de notre jeu. Il fallait donc impérativement créer une première version d'ennemis capables de se déplacer et de réagir en fonction de la position du joueur. Pour se faire Arnaud a divisé ce thème en 2 parties : le déplacement et le comportement des ennemis.

2.4.2 Les déplacements

Par Arnaud

Dans notre jeu, le déplacement d'un ennemi est quelque chose de fondamental. En effet sans ce déplacement le jeu serait beaucoup trop simple.

Les ennemis doivent donc se déplacer dans leur salle uniquement de façon plus ou moins libre et sans être prévisible par les joueurs. Il y a eu plusieurs changements au début du projet sur cet aspect mais la version finale est maintenant optimale pour ce qu'on veut réaliser.

Dans Unity, il y a une classe appelée « NavMesh Agent », il s'agit d'un outil qui permet de simplifier grandement les calculs d'itinéraire. Il suffit pour l'utiliser d'avoir une zone où les ennemis peuvent se déplacer. Cette zone est la «walkable zone», (espace où l'on peut marcher). Pour la première version de la méthode de déplacement, nous voulions utiliser la classe NavMesh Agent. Cela nous aurait fait gagner un temps monstrueux.

Mais hélas, plusieurs problèmes sont apparus. Notre map, étant générée procéduralement, il aurait été compliqué d'adapter à chaque génération le NavMesh. De plus, le NavMesh doit fonctionner uniquement dans une salle à la fois et pas dans toutes les salles en même temps.

Nous avons donc dû réfléchir à une autre méthode qui serait, elle, parfaitement adaptée à nos salles. Et c'est après plusieurs heures de réflexion qu'Arnaud trouva une autre idée.

Pour faire simple, chaque ennemi génère un point invisible vers lequel il se déplace jusqu'à qu'il arrive sur ce point. Il va ensuite recommencer en boucle tant que l'ennemi ne détecte pas le joueur dans sa zone de détection. Le point fort de cette méthode est donc que nous sommes capables de changer les méthodes de déplacement de chaque ennemi comme nous le souhaitons.

Avec cette méthode, nous pouvons donc rendre le jeu plus intéressant avec une variété, pour le joueur, de différents ennemis.

Au total nous avons 4 différentes méthodes de déplacement, ce qui vous nous permet d'assigner un type d'ennemi avec une méthode de déplacement.

2.4.3 Le comportement des ennemis

Par Arnaud

Le deuxième caractère important de notre intelligence artificielle est le comportement des ennemis. Cette partie a elle aussi posé plusieurs problèmes.

Pour commencer, bien que notre jeu soit en 2D, il faut que l'IA détecte le joueur dans un espace en deux dimensions. Nous avons donc utilisé plusieurs méthodes afin de déterminer où sont les joueurs.

En effet, nos ennemis doivent être en mesure de tirer sur les joueurs sur deux axes, horizontal et vertical. A l'aide d'un algorithme, qu'Axelle à fait, cela est désormais possible. Ainsi, les ennemis sont capables de tirer vers le joueur sur l'axe vertical ou horizontal.

Mais, bien que nos ennemis soient capables de tirer en direction des joueurs, ils doivent quand même avoir un comportement spécifique.

C'est pour cela, qu'à l'aide des méthodes de déplacement, nous avons donc conçu et mis en place un comportement spécifique pour chaque IA. Nous avions pour idée de rendre chaque déplacement ennemi désagréable de façon à ne pas rendre notre jeu trop facile.

Voici les 4 types d'ennemis disponibles en jeu :

Le patrouilleur :

Cet ennemi possède deux types de déplacement :

- Le mode patrouille : il va se déplacer vers un point, attendre quelques secondes puis repartir vers un autre point.

- Le mode attaque : le patrouilleur possède une zone autour de lui. Dès que le joueur passe dans cette zone, il va se faire poursuivre et tirer dessus par le patrouilleur tant qu'il est dedans.

A partir du moment où le joueur sort de cette zone, l'ennemi se rend vers la dernière position connue (l'endroit où le joueur est sorti) et attend quelques secondes avant de reprendre son mode patrouille.

Le Tourneur :

Cet ennemi ne possède qu'un seul type de déplacement. Il tourne en cercle autour du joueur et tire dans les quatre directions tant qu'il est en vie, sans jamais s'arrêter.

Bien que le déplacement soit simple à comprendre, le créer a été un peu plus compliqué. Il a fallu concevoir un algorithme qui produit un cercle puis l'adapter pour que ce dernier se génère autour d'un point et qu'il reste quoiqu'il arrive autour de ce point.

Le Follower :

Un ennemi plutôt simple mais peut être assez perturbant et prise de tête pour le joueur. En effet, le follower a pour but de suivre le joueur quoiqu'il arrive et lui tirer dessus en continu.

Le Spawner :

Le Spawner est un ennemi assez fourbe qui fait apparaître des ennemis toutes les x secondes.

Concernant son mode de déplacement, le Spawner reste à un point précis tant que le joueur ne rentre pas dans sa zone. Dès que le joueur est dedans, le Spawner va se rendre vers un autre point et continuer à faire apparaître des ennemis.

2.4.4 L'IA liée au gameplay

Par Arnaud

Pour avancer dans notre jeu, il nous faut tuer les ennemis. L'IA est donc fortement liée au gameplay et à la génération procédurale.

Afin d'éviter que les joueurs ne passent dans toutes les salles d'un coup, nous avons un système de porte dans chaque salle. Ces portes s'ouvrent uniquement lorsqu'il n'y a plus d'ennemis dans la salle.

Mais, cette logique spécifique s'applique aussi la salle du boss.

Lors de la génération de la map, nous détectons la salle du boss et dès lors que le joueur rentre dedans, aucun ennemi basique ne peut spawn à l'intérieur. Seulement le boss du niveau a le droit d'y spawn.

Point important à souligner, chaque salle à son propre nombre d'ennemis à spawn par salle.

Afin d'éviter une redondance dans le jeu, plusieurs petits détails comme ceci sont présents.

Notre IA ne se contente pas juste de gérer les ennemis. Elle est directement liée avec la génération des ennemis, la génération de la map et le gameplay tout entier.

2.4.5 Conclusion sur L'IA

L'intelligence Artificiel a été éprouvante à faire. En effet, beaucoup de méthodes nous ont amené à des culs de sac. Passer plusieurs journées sans aucun aboutissement est toujours décevant et jugé comme une perte de temps.

Mais, malgré tous les problèmes rencontrés, nous avons tenu bon et notre IA est fonctionnelle !

2.5 L'immersion

2.5.1 L'histoire

Par Gwennan

Même si le plot de l'histoire peut vous sembler assez court, la TAAG TEAM estime que les jeux avec une histoire simple et facile à comprendre sont les plus attrayants pour le public.

Ainsi, voici un résumé de l'histoire du jeu Toddlers Are Afraids of Ghosts : Dans un orphelinat sombre et humide, des enfants se retrouvent chaque nuit prisonniers du même cauchemar. Ils sont enfermés dans un donjon dont le seul moyen d'en sortir est d'affronter le dernier boss, leur surveillant.

Ainsi, l'histoire remet en contexte les événements (la nuit dans un orphelinat), annonce le statut des joueurs (orphelins), le boss final (le surveillant) et comment gagner le jeu.

Même si le plot de l'histoire est assez simpliste, notre jeu ne l'est pas. Et c'est justement ce qui permet de dire que notre jeu est qualitatif.

Le player n'est pas inondé complètement dans l'histoire, il en saisit les bouts importants et est dans un premier temps concentré à réussir son niveau. C'est ensuite qu'il fait le lien entre les graphismes, les maps et l'histoire.

2.5.2 L'écran de chargement

Par Gwennan

Gwennan s'est chargé d'implémenter l'écran de chargement directement dans Unity.

Elle a donc tout d'abord dessiné l'élément de la barre de chargement. Puis elle a ensuite construit le canva dans la scène Menu de Unity. L'écran était nécessaire lorsque le joueur cliquait sur Play afin de laisser un petit délai de chargement à la scène principale du jeu.

Pour ce qui est de l'implémentation, Gwennan n'a pas rencontré de problème. Il lui suffisait de relier les scènes, d'afficher en fonction du pourcentage de chargement une fraction représentative sur la barre correspondante et de donner en même temps le pourcentage du chargement.

Ce sont des éléments simples à implanter, qui n'ont pas généré de problème et qui

fonctionnent parfaitement.

2.5.3 La musique

Par Thomas

La musique a été réalisée à l'aide du logiciel Incredibox. Ce dernier permet de composer des musiques de manière intuitives à l'aide de samples et de boucles.

J'ai commencé par imaginer une musique dans un style dystopique. Pour cela, je me suis servi d'une basse sinusoïdale, d'un bruit de radar, d'un bruit s'apparentant à celui d'une vielle horloge, et enfin d'un sample vocal qui lie l'ensemble.

La musique suivante est beaucoup plus joyeuse et absurde. Celle-ci est composée de beatbox en guise de percussion, de son générés par ordinateur, et d'un sifflement en guise de mélodie.

Enfin la troisième musique disponible en jeu est faite en se servant d'un xylophone, de castagnettes, rythmé par une cloche, et quelques respirations, le tout au-dessus d'un bruit blanc à très basse fréquence. Cela nous donne une musique très pesante pour visiter les niveaux.

La musique du shop et des salles d'items est une petite musique d'ambiance faite à base de beaucoup de samples vocaux, dans un style sud-américain.

2.5.4 Les effets sonores

Par Thomas

Les effets sonores ont tous été fait à la main, excepté pour un qui a été généré à l'aide du site https ://sfxr.me/ .

Afin de les créer, je me suis servi du logiciel Audacity et d'un microphone Neat BeeCaster.

Les bruits ont été réalisés à la voix, en faisant des onomatopées.

Une fois les sons enregistrés, des effets ont été appliqués dessus afin de les rendre plus immersif.

Un effet qui a été utilisé parmi est la réverbération. Ce procédé permet la prolongation du son et engendre une résonnance. Ainsi, puisque le jeu se déroule dans un donjon, on ne s'attend pas à une acoustique de qualité mais plutôt à ce sentiment

de résonnance.

2.5.5 Musiques et effets en jeu

Par Arnaud

Grâce à Thomas, nous avons pu avoir la musique et les effets sonores pour le jeu. Cependant il restait à les implémenter en jeu, et pour ça Arnaud s'en est chargé.

Dans un premier temps, Arnaud a mis en place la musique que nous avons sélectionné dans le menu principal en la reliant la sortie audio afin qu'elle soit réglable dans les options.

Puis dans un second temps, Arnaud a implémenté les musiques en jeu directement. Pour éviter que l'on écoute tout le temps la musique, nous avons donc mis en place un script qui évite de jouer la même musique 2 fois d'affilé. Arnaud a aussi fait en sorte que quand la musique arrive à la fin de sa bande sonore, on change de musique.

Pour ce qui est des effets sonores, l'implémentation est quasiment la même. Ils s'activent uniquement lors d'une action spécifique.

Afin de permettre d'entendre au joueur tous les effets, même quand ils se cumulent, le script instancie pour chaque son un nouveau "Audio Source" et le détruit lorsque le son est fini. Cela demande évidemment que chaque son joué soit lié au "Audio Mixer" respectif des effets sonores.

2.5.6 Les menus

Par Arnaud

En commençant le jeu, la possibilité de faire un menu est vite devenue une évidence afin qu'il contienne la gestion de plusieurs choses telles que la gestion des scènes, du volume, et encore...

Cette partie est donc divisée en cinq sous parties :

- Le menu principal, une fois le jeu Toddlers Are Afraid of Ghosts lancé.
- Le menu de sélection des persos.
- Le menu de pause, en jeu, qui s'active à l'appui de la touche échap.

- Le menu des réglages, Options.
- Le menu de gestion des contrôles.

2.5.7 Menu Principal

Par Arnaud

Le menu principal permet d'accueillir le joueur sur le jeu. Il est constitué de 3 boutons :

Le bouton "Jouer/Play" redirige vers un sous-menu permettant de rejoindre une partie ou bien d'en héberger une.

Le boutons “Options” : ouvre un sous-menu constitué de différentes possibilités de réglage telles que : plein écran, Volume, Contrôles

Un bouton “Quitter” qui comme son nom l'indique quitte le jeu directement.

2.5.8 Menu de sélection des persos

Par Gwennan

Gwennan s'est chargée d'implémenter dans le menu, un système de sélection des persos. Attention, elle n'a pas implémenté cette sélection dans le jeu. Seulement dans le menu.

Ainsi, lorsque le joueur accède au menu, juste avant de lancer le jeu, il a accès à un menu de sélection de son personnage.

Il peut alors aisément à l'aide des flèches faire le tour des personnages disponibles mais aussi choisir un personnage, qui restera sauvegardé dans le menu peu importe s'il change de scène. Cela permet par exemple lorsqu'il veut revenir en arrière, de pouvoir ensuite retrouver le personnage sur lequel il était l'instant d'avant.

En ce qui concerne l'implémentation, elle s'est encore avérée assez simple.

Gwennan a tout d'abord eu besoin de créer une scène à part. Puis elle a ensuite fait une liste des skins disponibles pour le joueur. Il ne manquait plus que des flèches pour changer l'élément de la liste qui apparaissait et le tour était joué.

Bien sûr les scènes sont reliées entre elles, que ce soit la scène de menu classique ou la scène du jeu même.

Le plus complexe reste tout de même à implémenter ce choix dans la scène du jeu même, car il faut aussi gérer les animations et avec le multi joueur cela s'annonce d'une certaine difficulté.

2.5.9 Menu Pause

Par Arnaud

Il est très ressemblant à celui du menu principal. Il est constitué de 3 boutons :

Le bouton "Jouer" afin de quitter le menu pause pour reprendre la partie là où le joueur en était.

Le bouton "Options", qui a la même utilité que celui du menu principal.

Le bouton "Revenir au menu principal", qui permet de quitter la partie et de retourner sur le menu principal.

2.5.10 Menu des options

Par Arnaud

Le menu Options permet de modifier les réglages du jeu. Il possède 3 possibilités :

Le Plein écran : en cochant une case, le jeu passe en plein écran, et, en la décochant, l'effet inverse se produit.

Un bouton "Volume" qui ouvre un autre menu afin de régler avec précision le son entre la musique ou les effets sonores ou encore régler le son global du jeu.

Un bouton "Contrôles" qui ouvre lui aussi un autre menu détaillé dans la section Gestion des contrôles.

2.5.11 Menu de gestion des contrôles

Par Axelle

Tel qu'indiqué dans la section gameplay pour les déplacements et les attaques du joueur, nous utilisions auparavant l'input manager de Unity. Celui-ci nous a permis une compatibilité AZERTY et QWERTY conforme à notre cahier des charges. Cependant, au vu du temps supplémentaire obtenu grâce à l'avance prise sur le projet,

nous avons décidé de perfectionner encore le système des contrôles.

Axelle a donc créé, dans le menu des options, un nouveau bouton donnant accès à un nouvel écran : celui de la gestion des contrôles. Sur cet écran, le joueur se retrouve face à dix boutons. Un texte est présent à coté de chacun des boutons pour indiquer à quelle action ils correspondent.

Les huit des boutons correspondent aux déplacements et attaques dans les quatre directions : haut, gauche, droite, bas. Un est pour l'ouverture du shop et le dernier sert à revenir à l'écran des options.

Le joueur a ainsi la possibilité de cliquer sur la case indiquant la touche actuelle d'une action puis n'importe quelle touche de son clavier afin de choisir cette nouvelle touche pour l'action en question. Un texte est présent à coté de chacun des boutons pour indiquer à quelle action ils correspondent.

Des touches sont cependant assignées par défaut avant que l'utilisateur n'utilise ce menu, pour les déplacements ce sont les touche ZQSD, pour les attaques les flèches directionnelles et pour le shop la barre espace.

2.6 Les Graphismes

2.6.1 Les personnages

Par Gwennan

Commençons par l'élément le plus important pour le joueur : son personnage. Non seulement il doit être quelque peu esthétique, mais il faut aussi que le joueur dispose d'une variété de choix, avec des capacités différentes.

Gwennan avait ainsi réalisé pour la première soutenance six personnages proposés en version visible pour le menu.

Le design était passé par plusieurs réflexions afin de créer un personnage enfantin.

Avec la première proposition, en dehors du fait que c'était un simple croquis, la forme de la tête était un peu trop ronde et les éléments rappelaient un peu trop les anciens designs de jeux. Il fallait quelque chose de plus vif et plus moderne pour le joueur.

Dans la deuxième, la forme de yeux différait de la première version dans un but de comparaison afin de déterminer quel design semblait être le plus adapté. La forme de la tête, un peu moins ronde, du nez, de la bouche et des oreilles différaient aussi. Mais, le résultat n'était pas assez abouti et était peut-être toujours trop simple.

Puisque les précédents designs ne convainquaient pas. Gwennan avait fait des recherches de design sur internet et avait sélectionné des éléments distincts permettant le design final. Ainsi, dans la dernière proposition, les traits du modèle étaient plus développés, la forme du visage n'était pas ronde et le reste des éléments diffèraint des précédentes versions mais l'esthétique souhaité de design était là. Le joueur pouvait avoir devant lui un personnage assez expressif et agréable d'esthétique.

Pour la soutenance intermédiaire, Gwennan avait ajouté deux autres players au menu.

De plus, sur les huit personnages proposés à ce moment-là, quatre étaient animés et donc jouables.

Au niveau de l'apparence des joueurs, même si leur différence pour certains est à peine visible, on pouvait apercevoir un petit objet sur les épaules droites de deux des quatre joueurs dessinés pour cette soutenance.

Pour ce qui est de cette dernière soutenance, c'est avec fierté que Gwennan vous annonce que tous les players, que TAAG Team avait prévu et souhaité implémenter, sont finis. Aussi bien leurs animations, que leur intégration finale dans le jeu. Elle a

ainsi fini de réaliser les quatre players restants mais a aussi décidé d'en rajouter un, un chat.

2.6.2 Les ennemis

Par Gwennan et Thomas

Lors de la première soutenance, le but premier a été de se concentrer sur les personnages, et ce, quelque peu au détriment des ennemis.

Ainsi, à ce moment-là, Gwennan et Thomas s'étaient laissé un peu plus de temps de réflexion et c'est pour cela que les éléments présents étaient peu nombreux.

Ils étaient tombés d'accord sur le fait que le design devait être plus simple à animer mais aussi en concordance avec le thème du jeu : des monstres pour enfants. L'équipe avait donc sélectionné des ennemis basés sur les méchants des jeux de leur enfance.

Gwennan avait donc tout d'abord repris le design des fantômes du fameux jeu Pacman et l'avait adapté selon son style pour répondre à ces exigences, tout en respectant des contraintes comme le fait que les formes devaient être simples et faciles à modifier afin de faciliter l'animation. C'est pourquoi le mouvement de vague était le plus simple à dessiner.

Le design général des fantômes n'a pas changé. Les seuls éléments qui puissent les différencier sont leur couleur ainsi que la forme de leurs sourcils.

Pour la soutenance intermédiaire, Gwennan avait continué dans l'esprit du jeu Pacman. Elle avait ainsi dessiné le premier boss ennemi inspiré de Pacman lui-même.

Après réflexion et par soucis de coller avec le personnage de Pacman du jeu classique, Gwennan avait aussi décidé de ne pas le dessiner du point de vue face et derrière.

Thomas avait aussi dessiné un autre boss, provenant de son imaginaire cette fois et non d'un jeu. Son boss nommé Shade's Wrath possède de multiples inspirations.

Il a donc commencé par un visage qui avait comme éléments distincts un sourire gigantesque, rempli de dents acérées.

Pour ce qui est de la partie supérieure, on retrouve huit yeux rappelant ceux d'une araignée.

Au centre de ce visage, on retrouve un œil ressemblant à celui de Sauron dans la

saga Le Seigneur des Anneaux.

Au-dessus de tout cela, on retrouve des cornes de bouc, qui est une des représentation Satan dans la bible.

Pour la couleur violette, Thomas voulait une teinte assez synonyme du mal ou du malheur. Ainsi, quoi de mieux que la couleur vers laquelle on vire lorsque l'on s'étouffe.

Enfin le corps de cette bête est fantomatique, car le thème du jeu reste les fantômes.

Si Thomas devait résumer ce monstre, il a été pensé pour être la créature imaginaire la plus horrible possible et hanter les cauchemars de nos protagonistes, mais aussi de nos joueurs...

Ce boss est aussi pour Thomas un bon exemple de coopération. En effet, certaines idées viennent de personnes externes au groupe.

Ainsi, les deux boss dessinés à ce moment-là respectaient notre critère principal, être en concordance avec l'histoire du jeu, des montres qui font peur aux enfants.

De plus, afin de suivre l'avancement programmé du projet, Gwennan avait aussi tenu à réaliser tous les autres mobs ennemis.

Ainsi, de seulement quatre mobs ennemis à la première soutenance, le jeu était passé à vingt mobs ennemis au total.

Ainsi, Gwennan avait notamment dessiné quatre Mickey quelque peu troublants, avec des yeux qui reprennent les couleurs des yeux humains, afin à la fois de transformer le fameux personnage de Disney, mais aussi dans le but, en quelque sorte, "d'humaniser" ces fantômes.

Le personnage emblématique de Super Mario, le fantôme Boo, avait également été redessiné et animé.

De plus, enfant, tout le monde a fait de bonhommes bâtons. Ainsi, en souvenir de cette époque, Gwennan a réalisé quatre petits bonhommes, aux allures à la fois enfantines mais aux déplacements quelque peu perturbants.

Enfin, en mémoire d'un jeu populaire sorti en 2017, Hollow Knight, Gwennan a dessiné les derniers mobs dessinés et animés.

Pour la soutenance finale, contrairement à ce que Gwennan pensait, TAAG team avait décidé de n'avoir que trois boss.

Ainsi, c'est avec du soulagement et de la contradiction avec ses plans, que Gwennan a dessiné un boss final, au lieu de trois.

Puisque ce dernier était donc le boss final, il fallait avoir de la cohérence avec l'histoire de Toddlers Are Afraid of Ghosts.

Ainsi, il fallait que ce boss possède une apparence humaine mais aussi qu'il ait l'air mauvais. Après concertation avec son équipe, TAAG Team avait décidé de s'inspirer d'une personne réelle, Adolf Hitler, parce que même s'il est mort, trop de personne ont souffert à cause de lui. A travers notre jeu, on peut en quelque sorte dire que nous attaquons cet homme, même si cela peut sembler un peu déplacé.

Afin tout de même de ne pas le représenter entièrement, Gwennan s'est donc inspiré pour son visage d'une carricature de lui, mais elle a préféré le rendre chauve, par soucis de ne pas donner l'impression que le jeu se concentrait sur Hitler.

Des éléments rappellent tout de même ce dernier : le col un peu d'uniforme, la moustache et aussi les couleurs kaki de ses habits.

2.6.3 Le marchand

Par Gwennan

Suite à la volonté de Gwennan de pouvoir disposer d'un marchand dans le jeu, il a été décider que l'apparence de ce dernier reprendrait l'apparence du logo de la TAAG Team, de façon à incorporer un peu plus les développeurs dans le jeu.

Ainsi, Gwennan a choisi de ne le représenter que de deux cotés, rendant à la fois plus facile son animation mais aussi respectant le logo de base.

2.6.4 Les items

Par Gwennan

Elément lui aussi relégué au second plan lors de la première soutenance, la TAAG Team s'était laissé un temps de réflexion de réflexion sur cette partie.

Lors de la soutenance intermédiaire, la TAAG team avait donc réussit à cibler dans un premier temps, les principaux boosters et objets nécessaires au fonctionnement du jeu.

Tout d'abord, il fallait créer des items permettant de redonner de la vie au personnage. Deux potions de vie avaient été créées : en rouge la vie normale et en bleu la vie bonus.

De plus, comme dans tous les jeux, le personnage devait avoir accès à des boosters.

Ainsi les chaussures de Sonic, qui permettent au joueur de se déplacer plus rapidement, et l'aspirateur de Luigi, qui booste l'attaque, avaient été créés.

Enfin, afin de rendre plus visuelle l'attaque, Gwennan avait aussi dessiné deux armes. Une pierre banale, utilisé lorsque l'attaque n'était pas boostée et une pierre enflammée, pour l'attaque boostée.

De plus, Gwennan a aussi réaliser le projectile ennemi.

Puisque la majorité de nos ennemis sont représenté sous la forme de fantômes, Gwennan a tenu à ce que leur projectile soit en accord avec leur nature.

Ainsi, leurs projectiles sont censés être des boules d'ectoplasme, un résidu vert, semblable à de la gelé, sécrété par les fantômes.

Même si le dessin peut être assez abstrait et bizarre, voilà l'idée derrière : être en cohérence avec les ennemis.

Pour la soutenance finale, Gwennan a donc créé quatre autres items :

- un révolver, qui augmente la cadence de l'attaque.
- une fleur, inspirée de la fleur arc-en-ciel de Mario Kart mais redesigné dans un style plus simpliste, qui augmente la portée de l'attaque.
- un champignon, lui aussi inspiré directement du jeu Mario Kart, qui diminue le cooldown du player.
- une étoile, elle aussi reprise de Mario Kart, qui agit comme un booster un peu partout.

Une pièce a aussi été dessiné, afin de correspondre avec la boutique du marchand. Elle ressemble, elle aussi, aux pièces disponibles dans Super Mario.

2.6.5 Les scènes

Par Gwennan

Délibérément laissées de côté lors des soutenances précédentes, Gwennan a décidé d'accorder plus d'attention à l'aspect général du jeu.

Tout d'abord, afin de rendre le jeu plus attractif, il fallait customiser certains éléments tels que la barre de vie mais aussi l'apparence des boutons.

Pour ce qui est de la barre de vie, Arnaud avait pour la soutenance intermédiaire pris le haut d'un des fantômes afin d'avoir un semblant de barre de vie. Face à cet affront pour son fantôme, Gwennan a décidé de dessiner une barre de vie, assez classique mais beaucoup plus esthétique et parlante qu'une simple colline rouge.

En ce qui concerne l'apparence des boutons, Arnaud et Gwennan ont décidé de choisir une police de caractère en accord avec l'histoire du jeu. Ainsi, la police Black Shirt Slime Trail est présente sur à peu près tous les éléments textuels du jeu. Elle a été choisie principalement pour son design se rapprochant des textes dans les histoires d'horreurs.

Mais, ce n'est pas le seul élément différent des boutons. La forme, propre, lisse et carré, a cédé place à un design un peu plus "sale" et surtout plus stylisé. Pour être en concordance avec l'histoire, Gwennan voulait une touche qui donnait l'impression qu'on l'avait dessiné en vitesse, sans aucun souci de détails. Ainsi, au lieu d'un bloc, Gwennan a réalisé à partir d'un trait, une forme se rapprochant de ce concept, sans pour autant que l'ensemble ne paraisse trop peu soigné.

Gwennan a aussi souhaité dessiner, par égard au marchand, la bulle de dialogue du marchand, visible uniquement dans la boutique. Cette bulle est assez classique. Elle reprend un design connu mais pas copié collé.

Pour en revenir aux besoins du marchand, vous pouvez aussi apercevoir en arrière-plan de sa boutique, une sorte de fond gris avec quelques petits éléments plus sombres vers le centre et les extrémités. Gwennan a essayé de donner une impression de caverne, avec au milieu une zone plus sombre, comme si la caverne était plus profonde qu'on ne le croie, et à ses rebords, des petites formes plus sombres, pour donner un effet de premier plan, afin de renforcer la profondeur de l'image.

Par soucis que le fond ne soit pas trop chargé graphiquement, Gwennan a essayé de minimiser ces détails afin que le player puisse pleinement se concentrer sur les items plutôt que le fond de la boutique.

De plus, suite à un besoin de Thomas, Gwennan a réalisé une barre de chargement, ainsi que son écran, afin de donner plus de professionnalisme au jeu. On peut notamment y distinguer les silhouettes du marchand, logo officiel de TAAG team, mais aussi celles de quatre petits canards, chacun appartenant à un des développeurs du jeu.

Enfin, comme dernier élément non essentiel mais plus professionnel pour le jeu, Gwennan a dessiné deux scènes, visibles au tout début du jeu.

Ces deux scènes, bien que superficielles niveau fonctionnement du jeu, ont pour rôle de donner plus de sérieux et de professionnalisme au jeu. En effet, de nos jours, presque tous les jeux utilisent des animations ou des images afin de remettre en contexte leur histoire ou juste de styliser les actions.

La première scène visible dans le jeu est celle où l'on distingue de loin d'ensemble d'un bâtiment à priori assez ancien.

Sur cette image, plusieurs détails sont à noter comme l'apparence délabrée du lieu, renforcé par cette masse noire semblable à du lierre, les toiles d'araignées mais aussi par l'apparence générale de la bâtisse. Cette dernière, semblable à un manoir, représente l'orphelinat. Elle est assez grande pour accueillir des dizaines d'enfants mais sur elle repose une aura sombre, presque maléfique, ce qui coïncide avec le plot de l'histoire. De plus, la présence de croix catholiques, semblables à celles présentes sur les pierres tombales, renforcent cette impression.

Mais vous pouvez aussi distinguer la silhouette de ce qu'il pourrait être un enfant à gauche, silhouette qui semble incité le joueur à le suivre.

Sur la seconde image, la scène se déroule dans le dortoir de l'orphelinat. Un lieu vaste, impersonnel, sombre et miteux. Les rangées de lits, les lampes minuscules accrochées au plafond, les toiles d'araignées ainsi que la profondeur de la scène permettent de réaliser ces impressions.

De plus, plus discrètement que dans la première image, une silhouette d'enfant est encore apercevable au fond de la scène. Sa silhouette est entourée d'un halo noir, comme s'il était terré dans les ténèbres ou en train de se faire aussi petit que possible.

Ainsi, à travers ces deux images et leur texte, Gwennan tente d'inciter le lecteur à la fois à se plonger dans le contexte de l'histoire du jeu mais aussi à le motiver à persister jusqu'à vaincre le boss final, le surveillant, un homme aussi méprisable qu'Hitler.

Mais, ces images ont aussi pour but de donner plus de cachet au jeu, en ne donnant pas l'impression que c'est un simple jeu banal pour commencer mais un jeu plus réfléchi.

2.6.6 La couverture du jeu

Par Gwennan

Afin de donner envie d'acheter le jeu, il est indispensable de faire une couverture attractive.

Bien que la couverture qu'a dessiné Gwennan est assez simple et banale, nous pouvons apercevoir les éléments essentiels du jeu : les enfants et le surveillant.

Mais en regardant bien, vous pouvez apercevoir dans un coin, un canard jaune, énième rappel aux canards des développeurs du jeu.

Le fond, composé d'un couleur sombre, est aussi un rappel du contexte sombre, basé sur la peur, de l'histoire.

Les rebords plus clairs permettent de voir un contraste avec l'image de couverture et les informations annexes. Gwennan a jugé important de rajouter le mois de sortie du jeu, mais aussi le nom du groupe de développeurs derrière.

Aussi, pour donner encore plus d'emphase sur les développeurs du jeu, Gwennan a tenu à écrire chacun des noms et des prénoms, afin que l'on s'aperçoive réellement qui est derrière la TAAG team.

2.6.7 Logo

Par Gwennan

Dessiné dans les premiers temps du projet Toddlers Are Afraid of Ghosts, le logo s'inspire de Zero de l'étrange noël de monsieur Jack, un chien fantôme incroyablement mignon et attachant.

Ce personnage, du commun accord de tous les membre de TAAG team, correspond en quelque sorte à l'esprit du jeu. Vous avez à la fois la combinaison du fait que ce soit un fantôme mais aussi un personnage emblématique de l'enfance. En effet, qui n'a jamais vu ce film fantastique ?

Ainsi, qui de mieux qu'un Zero customisé pour être la mascotte de la TAAG team.

2.6.8 Textures

Par Thomas

Eléments secondaires dans le jeu, mais important pour l'aspect professionnel du jeu, les textures ont présenté un défi : celui de l'imprévisibilité des salles.

Chaque salle est construite de plein de murs carrés qui, mis ensemble, enferment

les joueurs. Ces blocs doivent donc avoir leur cotés identiques pour ne pas briser la cohérence établie jusqu'ici. C'est donc grâce aux outils de symétrie axiale que les textures (ou au moins les contours des textures) ont été fait. Cela permet à la texture de se répéter sur tous les côtés sans imperfections.

Pour ce qui est de la texture du sol/fond, une texture été appliquée à une Skybox, ce qui fait que n'importe où regarde la caméra, il y a toujours la même texture.

Les textures initiales présentées pour la première soutenance avaient une résolution trop élevée, ce qui les rendaient très peu agréable à regarder. Thomas les a donc toutes refaites afin de ne pas heurter l'expérience et les yeux de l'utilisateur.

2.6.9 Icône du launcher

Par Gwennan

Suite à la création du launcher par Arnaud, il a été demandé à Gwennan de réaliser une icône.

Pour répondre à cette sollicitation, Gwennan a défini comme objectif que l'icône ne devait pas être chargé, lisible et simple.

C'est pour cela qu'elle a choisi le logo de la TAAG Team, qui est aussi le marchand, pour représenter pleinement le jeu.

De plus, les initiales TAAG, diminutif pour Toddlers Are Afraid of Ghosts, sont pleinement visibles et permettent de détecter plus rapidement le jeu.

Enfin, la couleur de fond, rappelant les couleurs utilisées dans le jeu, est un énième rappel à son atmosphère sombre et mauvaise.

Ainsi, Gwennan trouve, avec seulement ces 3 éléments, que son objectif est atteint et que l'icône est visible et lisible.

2.6.10 Les animations

Par Gwennan

Les animations se sont développées au fur et à mesure de l'apparition des personnages, des ennemis et du marchand.

Même s'il se peut que les animations ennemies ne soient pas forcément en accord avec le sens de mouvement, les animations des personnages rattrapent ce problème.

Gwennan, qui s'est chargée des animations, n'a pas rencontré de problème particulier en dehors de les lier aux déplacements de leur porteur. Cette tache, quoique simple du fait des outils disponibles sur Unity (Animation et Animator), est en fait très répétitive et lassante. Devoir animer tous les caractères du jeu a pris pas mal de temps à Gwennan mais au moins le résultat est là. Ainsi, même si les animations sont une tache simple, c'est une tache répétitive et longue sur la durée dont Gwennan s'est chargée.

Pour rappel, l'outil animation permet d'ajouter plusieurs sprites (images) d'un caractère sur une timeline et l'outils animator permet de gérer la cohésion des mouvements avec les déplacements grâce à des BlendTree où les axes et la vitesse sont essentiels à l'implémentation des mouvements.

2.6.11 Bilan des graphismes

Par Gwennan

Gwennan a trouvé les graphismes relativement simples mais long dans l'ensemble.

Le temps de comprendre l'utilisation de sketchbook, outil quelque peu infâme qui lui a quelque fois fait perdre sa progression en s'arrêtant brusquement.

Aussi, le fait de gérer les calques peut être un peu complexes, surtout quand l'on pense gommer mais que l'on est plutôt en train de dessiner au blanc. Chose horrible quand après on doit faire disparaitre le fond blanc et qu'on se retrouve avec des traits blancs.

De plus, c'est assez simple de dessiner mais beaucoup moins lorsque l'on copie colle le dessin et on s'aperçoit que la qualité diminue. Il est donc parfois nécessaire de tout reprendre car les pixels sont trop visibles ou les traits se sont trop atténués.

Mais, Gwennan préfère tirer de cette expérience tous les apprentissages qu'elle en a retenu. Notamment, comme le fait que pour synchroniser une animation, il ne suffit pas de juste changer l'emplacement des jambes. Cela peut au contraire perturber entièrement l'animation et gâcher une partie du travail si l'on est trop perfectionniste.

2.7 Multijoueur

Le multijoueur en réseau est une contrainte imposée sur le projet du second semestre. Il fait donc partie intégrante de Toddlers Are Afraid of Ghost. Le multijoueur est une expérience assez particulière pour les joueurs donc nous avons décidé de leur laisser le choix de comment aborder la partie, nous en reparlerons plus tard.

2.7.1 Multijoueur sur Unity

Par Arnaud

Avant d'aborder le choix d'Arnaud pour gérer le multijoueur, revenons un peu sur l'histoire du multijoueur. Cela vous permettra de mieux comprendre sa décision.

Bien que le moteur de jeu multiplateforme de Unity soit sorti en 2005, c'est seulement 9 ans plus tard, soit en 2014, qu'un module permettant le multijoueur est annoncé par les équipes d'Unity. 'Unity Networking' ou plutôt UNet. Son premier but est de démocratiser le développement des jeux-vidéos. Malheureusement en 2018 UNet est déclaré obsolète : les technologies qu'il utilisait devenaient anciennes et posaient des problèmes de sécurité et de performance. Aujourd'hui encore, Unity n'a toujours pas sorti son petit frère malgré une annonce de leur part.

Cependant même si Unity n'a toujours pas sorti une version stable pour le multijoueur, les développeurs de jeux-vidéos multijoueur ont des alternatives : Photon Unity Networking (connu sous le nom de PUN) et Mirror. PUN est un module de gestion du multijoueur avec une partie payante et une version gratuite limitée. Au contraire, Mirror est open-source donc également gratuit. Mirror reprend le code de UNet, en corrigeant les défauts de performances et de sécurité, et ajoute de nouvelles fonctionnalités. Les deux possèdent des avantages et des inconvénients. Le choix s'offre donc à nous pour un projet comme le nôtre.

Avant même de commencer, Arnaud avait déjà entendu parler de Mirror, que ça soit dans les couloirs, en discutant des projets ou bien grâce à plusieurs vidéos sur YouTube. C'est ainsi pourquoi Mirror fut choisie pour le projet.

2.7.2 Découverte de Mirror

Par Arnaud

Le fonctionnement de Mirror est assez simple dans l'ensemble. Il utilise un protocole client-serveur, c'est-à-dire que chaque instance de jeu est un client. Pour jouer en multijoueur, chaque client se doit de se connecter à un serveur. Le serveur a le rôle de communiquer chaque information à chaque client actuellement présent sur le serveur. Lorsqu'un client se connecte ou se déconnecte de la partie, seul le serveur

est informé. Il se doit donc de mettre à jour tous les clients de cet événement et/ou déclencher divers scénarios.

Tout au long du développement, il est important de différencier ce qui appartient au serveur et au client (un élément important mais pas forcément facile à réaliser).

Pour compliquer la chose, nous allons introduire le principe d'hôte : la symbiose d'un client et d'un serveur. Afin d'éviter de louer un serveur, un des deux clients joue aussi le rôle de serveur, « l'hôte ».

La mise en place de Mirror est assez simple : on ajoute d'abord un NetworkManager (un objet invisible qui s'occupe du multijoueur), un script de Mirror qui permet pour le client de communiquer avec le serveur et pour le serveur de pouvoir répondre aux requêtes du client. Heureusement, il n'y a pas besoin d'implémenter des protocoles de communication ni de transport de paquet. Il suffit juste de configurer correctement le NetworkManager, d'adapter nos scripts pour le multijoueur avec Mirror et le tour est joué. Nous avons un multijoueur fonctionnel.

2.7.3 Scène en arrière-plan et développement avec Mirror

Par Arnaud

Notre jeu étant composé de deux scènes : une pour le menu principal et une pour jouer. Jusqu'ici, lorsqu'on voulait changer de scène, on demandait à Unity (à l'aide d'un script) de changer la scène. Gentiment, Unity déchargeait alors la scène actuelle et chargeait celle qu'on demandait. Mais cela pose problème pour le NetworkManager qui est un simple objet associé à la scène. A chaque changement de scène on perdait donc notre objet.

Afin d'assurer le lien entre le client et le serveur tout au long de la partie. Nous avons créé une scène en arrière-plan qui contient notamment le NetworkManager ainsi que d'autres objets importants peu importe la scène active. Grâce à ce système, on peut ainsi changer de scène sans toucher à la scène en arrière-plan. Une gestion du multijoueur inter-scène est ainsi possible.

2.7.4 Poursuite du développement avec Mirror

Par Arnaud

En vue de développer l'entièreté du multijoueur pour jouer en coopération à notre jeu, Arnaud s'est repenché sur Mirror pour implémenter quelques nouvelles fonctionnalités :

- changement du lieu d'apparition des joueurs
- synchronisation client server et server client
- correction des bugs liés aux déplacements

En intégrant tout cela avec Mirror, il a découvert des nouvelles méthodes et le constat est sans appel : ce qui avait été fait pour la deuxième soutenance a été largement amélioré, notamment au niveau des performances et de la stabilité.

Des changements ont aussi eu lieu sur l'organisation du code. Plus on avançait dans le développement du jeu, plus il y avait de choses à implémenter qui n'ont pas forcément été simples à faire.

Nous avons aussi pensé qu'il était nécessaire que les joueurs aient le choix de comment ils veulent aborder notre jeu en multijoueurs. Chaque joueur est ainsi indépendant et peut donc explorer la carte seul ou bien décider d'explorer la carte en équipe.

2.8 La Communication

2.8.1 Le site web

Par Gwennan et Arnaud

Elément principal de la communication de TAAG team, le site web a été entièrement réalisé par Gwennan sans utiliser de design déjà existant.

Pour la première soutenance, elle avait dans un premier temps décidé d'héberger le site web sur Github. Cet outils remplissait à la fois les conditions de gratuité d'hébergement mais aussi de facilité d'utilisation. Il y avait bien sûr d'autres moyens d'héberger son site web mais très peu qui allient gratuité et facilité d'usage.

De plus, le flou des copyrights des sites en France existe bel et bien. Faut-il acheter le copyright ou bien seulement se fier à la protection du droit d'auteur ? Par défaut de financement mais aussi par soucis de protection, nous avons voté pour un compromis, faire figurer de façon informelle la notion de copyright sans pour autant la posséder. Nous espérons que cela permettra de dissuader et de prévenir le plagiat le plus possible.

Ensuite, pour la soutenance intermédiaire, le but était de rendre le site web accessible et adaptable à tout type d'appareils.

Ainsi, pour cela, Gwennan a totalement changé l'apparence de la page d'accueil, qui était auparavant composée d'une image de fond. Cette image était en fait incompatible avec beaucoup d'appareils du fait de sa longueur.

Par conséquent, plutôt qu'une image de fond, il a été plus simple de créer une zone de texte qui s'adapte à la taille de l'écran de l'utilisateur, rendant ainsi le site mobile-friendly.

Même si la page d'accueil initiale était plus plaisante niveau design que celle d'aujourd'hui qui est beaucoup plus simple.

Mais, Gwennan ne regrette pas son choix. Mieux vaut avoir un site fonctionnel qu'un site cassé.

Pour cette dernière soutenance, Arnaud s'est chargé d'héberger le site et de lui trouver un nom de domaine. Il a ainsi organisé le changement d'adresse web ainsi que le basculement d'un hébergeur à l'autre.

Afin de nous faciliter le travail, Arnaud a mis en place le site de façon à ce qu'il lise chaque changement sur le git du site web et mette à jour automatiquement ses pages. Cela permet de push des modifications et de voir immédiatement les change-

ments actifs.

Bien sûr pour chaque soutenance, Gwennan a mis à jour le site, que ce soit pour les rapports de soutenance ou même pour le téléchargement depuis le site du jeu.

2.8.2 Le trailer

Par Thomas

Le trailer a été enregistré grâce au logiciel Open Broadcast Software et monté sur le logiciel Adobe Premier Pro.

Thomas a d'abord joué et enregistré plusieurs parties, avant de les mettre dans le logiciel de montage. Puis, il les a coupées et en a gardé uniquement les moments intéressants. Cela permet de révéler le gameplay et un petit peu d'histoire, mais sans pour autant tout en dévoiler.

Il a ensuite rajouté la musique et collé les extraits vidéos afin d'avoir un rendu diffusable pour faire la promotion du jeu.

3 Déploiement du jeu

Etant une obligation de la part d'EPITA mais aussi une question de logistique pratique, nous nous devions de mettre en place un installeur et un désinstalleur. Une méthode simple et efficace pour l'utilisateur.

Alors pour se faire, Arnaud s'est chargé de se renseigner de ce côté. Il a alors cherché ce qui permettait d'obtenir cette fenêtre, que tous développeurs connaissent, l'installeur.

Après diverses recherches, Inno Setup fut le meilleur compromis pour créer l'installeur. Simple d'utilisation et rapide, il suffit de renseigner l'exécutable du jeu avec ses fichiers. Puis ensuite remplir les champs que nous souhaitons, adresse du site, équipe de développement ect...

Ce qui est pratique avec Inno Setup, c'est qu'il met aussi dans votre solution, un désinstalleur qui supprime tout ce que vous avez installé.