

Rapport de la deuxième soutenance



par
TAAG TEAM

composé de :

Thomas Solatges
Arnaud Corcione
Axelle Destombes
Gwennan Jarno

Table des matières

1	Introduction	3
2	Avancement du projet	4
2.1	Le Level Design	4
2.1.1	Remise en contexte	4
2.1.2	Avancement	4
2.2	L'IA	5
2.2.1	Remise en contexte	5
2.2.2	Poursuite du développement de l'intelligence artificielle	5
2.2.3	Pour la prochaine soutenance et Conclusion	6
2.3	Le Gameplay	6
2.3.1	Remise en contexte	6
2.3.2	Attaque du joueur	6
2.3.3	Récupération d'items	7
2.3.4	Attaques des ennemis	7
2.3.5	Pour la prochaine soutenance	7
2.4	Les Graphismes	8
2.4.1	Les personnages	8
2.4.2	Les ennemis	9
2.4.3	Les items	11
2.4.4	Bilan des graphismes	12
2.5	Réseau	13
2.5.1	Le multijoueur	13
2.5.2	Multijoueur sur Unity	13
2.5.3	Découverte de Mirror	13
2.5.4	Scène en arrière-plan et développement avec Mirror	14
2.5.5	Planning pour la prochaine soutenance	14
2.6	Communication	15
2.6.1	Le site web	15
3	Ressentis des membres	17
3.1	Arnaud Corcione	17
3.2	Axelle Destombes	17
3.3	Gwennan Jarno	17
3.4	Thomas Solatges	17
4	Conclusion	18
5	Bibliographie	19

1 Introduction

Depuis cinq mois, TAAG Team travaille sur le jeu *Toddler Are Afraid of Ghost*. Nous avons réussi à atteindre les objectifs que nous nous étions fixés durant la rédaction du cahier des charges et nous nous estimons dans les temps par rapport à notre planning.

Pour rappel *Toddler are afraid of ghost* est un roguelike 2D, à un ou deux players, où le joueur affronte des monstres et parcourt des niveaux générés de façon procédurale.

Pour cette deuxième soutenance, le but principal était de présenter un jeu fonctionnel, mais non final. TAAG TEAM a donc continué son avancée dans le projet, en reprenant de zéro ou en continuant le travail déjà fourni pour la soutenance précédente.

Ainsi, tout en respectant le cahier des charges, nous avons avancé sur les principaux points suivants :

- Level Design
- Intelligence Artificielle
- Gameplay
- Graphisme
- Réseau
- Communication

Nous sommes satisfaits de nos progrès et surtout de nos réussites. Nous arrivons avec plus de facilité à développer notre jeu, même si certains points n'en restent pas moins complexes.

2 Avancement du projet

2.1 Le Level Design

2.1.1 Remise en contexte

Lors de la précédente soutenance, nous avions un niveau qui se générait de façon à peu près cohérente, mais avec des défauts et surtout aucun moyen de modifier les propriétés des salles une fois le jeu lancé.

2.1.2 Avancement

Le problème majeur de l'ancien système est que les salles apparaissaient, mais on n'avait aucun moyen de conserver les informations de ces dernières. A cause de cela, il a donc fallu approcher la génération des étages sous un autre angle. Thomas a donc tout repris de zéro en pensant à l'étage comme une matrice plutôt que de juste placer des salles. Cela permet de donner des propriétés spécifiques à ces salles avant même de les faire apparaître. Cette méthode peut par exemple nous servir à mettre en place une mini map et à augmenter la modularité du niveau (simplicité de changer la taille de l'étage, des salles présentes, etc...).

De plus, avec l'ajout de différentes salles, le besoin de gérer le chargement des prefab au runtime est apparu. Grâce à ce processus, il n'y a plus besoin de mettre les salles dans une liste à la main. Cela permet aussi de conserver les propriétés de toutes les salles, comme leur position ou leurs ouvertures.

Dans la théorie, tout cela doit bien s'emboîter. Malheureusement, Unity a été capricieuse et la méthode dont Thomas se servait pour charger les prefab ne voulait pas lui renvoyer de Game Object. Cela rendait donc impossible l'affichage des salles dans la scène. Heureusement, Arnaud, notre chef de projet, a trouvé la solution et maintenant un nouveau problème est apparu : les coordonnées ne s'appliquaient pas et toutes les salles étaient ajoutées les unes sur les autres. Une fois ce problème résolu, les salles apparaissaient avec un mauvais offset et donc le décalage entre les salles n'était pas constant. De plus, certaines salles, nécessaires pour ne pas avoir d'ouvertures sur l'extérieur, n'étaient pas là.

Une fois ce problème résolu, il ne restera plus qu'à ajouter une sortie de niveau dans la salle du boss et une salle d'item. Grâce à la modularité de ce nouveau système, cela devrait être assez simple.

2.2 L'IA

2.2.1 Remise en contexte

Pour la dernière soutenance, nous avions réussi à mettre en place quatre types différents d'IA avec leurs spécificités, que ça soit au niveau de leur déplacement, de leur capacité ou bien même de leur façon d'attaquer le joueur. Donc actuellement les ennemis s'adaptent au joueur et même aux déplacements de ce dernier.

Pour cette deuxième soutenance, l'objectif fixé était de continuer sur cette lancée le reste de l'IA afin d'avoir un jeu jouable, et de pouvoir finir une partie de Toddlers Are Afraid of Ghost, du lancement de la partie jusqu'à la mort du joueur ou du boss.

2.2.2 Poursuite du développement de l'intelligence artificielle

Pour permettre au joueur de finir une partie, nous avons besoin de plusieurs éléments :

- Générer un certain nombre d'ennemis par salle.
- Etre capable de tuer les ennemis.
- Prendre des dégâts par les ennemis.
- Faire apparaître un boss uniquement dans la salle du boss

Afin de régler un grand nombre de problèmes, nous utilisons un objet invisible, le *GameManager*, qui va s'occuper de gérer la partie surtout au niveau de la génération des monstres.

Arnaud a également approfondi les scripts de chaque ennemi.

Voici une liste des éléments approfondis dans les scripts :

Pour le Patrouilleur, Followers et Tourneur :

- Mise en place de la distance d'attaque, résistance, attaque, temps d'attente avant attaque.
- Adaptation du script de déplacement afin d'éviter les obstacles dans la salle.

Pour le Spawner :

- Mise en place de sa fonctionnalité principale, faire apparaître des ennemis.
- Adaptation du script de déplacement afin d'éviter les obstacles dans la salle.

En développant ces fonctionnalités essentielles au jeu, de nombreux problèmes sont apparus et ont pu être résolus assez rapidement. Cela a donc permis de faciliter le développement du jeu du point de vue de la programmation du côté de l'intelligence artificielle.

2.2.3 Pour la prochaine soutenance et Conclusion

La prochaine soutenance étant la dernière, l'intelligence artificielle devra être entièrement terminée. Il est possible que plusieurs ennemis voient le jour. Tout dépendra de l'inspiration. Pour conclure sur l'IA pour cette soutenance, on pourrait dire que nous avons largement fini l'IA pour le projet. Evidemment, quelques modifications pourront survenir, mais aucune mise à jour majeure n'est prévue, excepté l'implémentation de nouveaux ennemis.

2.3 Le Gameplay

2.3.1 Remise en contexte

Pour la première soutenance, nous avions créé les classes personnages et ennemis et implémenté la première action fondamentale du personnage, le déplacement de celui-ci à l'aide des touches ZQSD. Nous avions aussi implémenté le déplacement des ennemis et commencé la deuxième action principale du personnage et des ennemis, l'attaque.

2.3.2 Attaque du joueur

Les attaques sont dans notre jeu des attaques à distance. Le joueur émet un projectile qui inflige des dégâts à un ennemi lorsqu'il rentre en collision avec ce dernier. Le joueur peut ainsi utiliser les quatre flèches directionnelles du clavier pour choisir la direction de ses attaques, permettant ainsi de dissocier la direction des déplacements et des projectiles.

Les attaques sont possibles dans les quatre directions. Nous avons ainsi géré le cas où le joueur essaierait d'appuyer sur plusieurs directions à la fois :

- 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 haut.
- Si la flèche droite et la flèche bas sont pressées, la direction droite prendra la priorité.
- Si les flèches bas et gauche sont pressées, l'attaque se dirigera vers le bas.

Lorsque le joueur attaque dans une direction, une instance de projectile est créée aux coordonnées du joueur. Celle-ci possède comme attribut la direction vers laquelle elle se déplace, la distance sur laquelle elle peut infliger des dégâts au premier ennemi rencontré et le nombre de dégâts qu'elle peut infliger.

Dans le cas où le projectile entre en collision avec un ennemi, le nombre de points de vie de celui-ci diminue du nombre de dégâts infligé par le projectile et ce dernier est détruit. Si aucun ennemi n'est rencontré avant d'atteindre la distance maximale que le projectile peut parcourir, ce dernier est simplement détruit.

2.3.3 Récupération d'items

La troisième action du personnage est la récupération d'items. Au cours du jeu, le joueur pourra obtenir différents objets afin d'augmenter ses statistiques. Afin de récupérer ces derniers, il devra appuyer sur la barre espace en étant à proximité de l'item qu'il souhaite récupérer.

Les premiers objets que nous avons implémentés sont des potions de vie et vie bonus, une paire de chaussures qui augmente la vitesse du joueur ainsi qu'un aspirateur permettant d'augmenter les dégâts infligés par les projectiles du joueur.

Les potions permettant de récupérer de la vie se classent en deux catégories : 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 jaude de vie "classiques". La vie bonus ne possède pas de maximum, le joueur peut l'augmenter jusqu'à l'infini tant qu'il trouve des potions bonus.

2.3.4 Attaques des ennemis

De la même façon que le joueur, les ennemis peuvent également attaquer en utilisant des projectiles. Nous utilisons un système de tags afin que les projectiles ennemis ne puissent infliger des dégâts qu'au joueur. Les projectiles ennemis possèdent les mêmes attributs que ceux du joueur : une direction, un nombre de dégâts, une distance maximale, et ils apparaissent également à la position de l'ennemi qui attaque. Tout comme les projectiles alliés, si aucun joueur n'est rencontré avant d'avoir parcouru la distance maximale, le projectile ennemi est détruit. Mais, s'il rentre en collision avec un joueur, il lui inflige alors des dégâts avant de se détruire.

2.3.5 Pour la prochaine soutenance

Du point de vue du gameplay, nous avons jusqu'à maintenant plus avancé que prévu. Pour la prochaine soutenance, l'objectif est simplement d'implémenter un marchand permettant au joueur d'obtenir des objets et de diversifier ceux-ci.

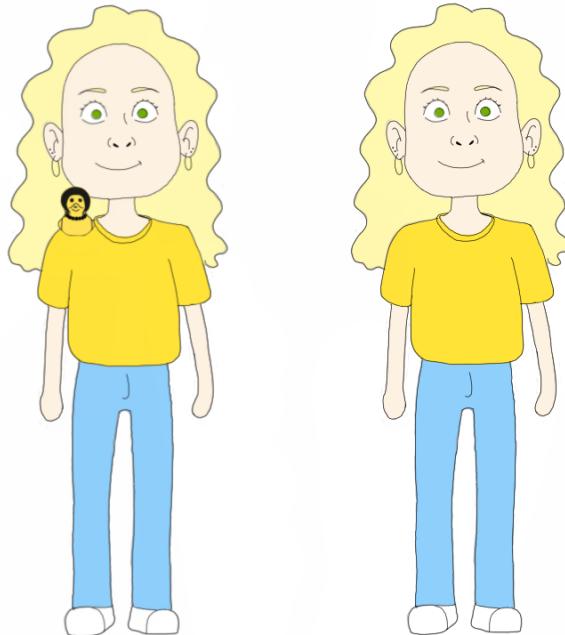
2.4 Les Graphismes

Pour cette deuxième soutenance, Gwennan et Thomas ont réalisé la plupart des graphismes nécessaires au fonctionnement du jeu.

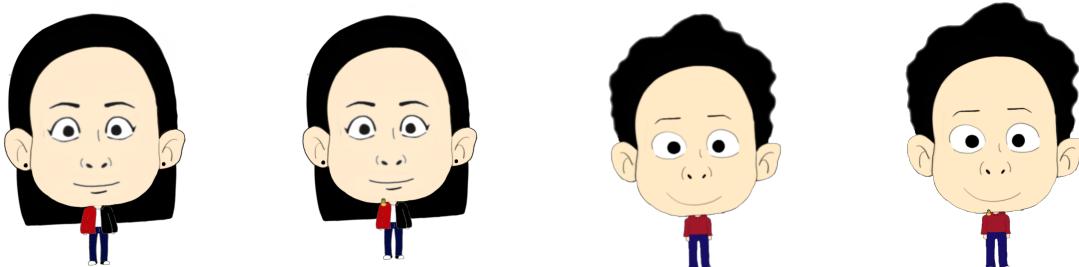
2.4.1 Les personnages

Pour la première soutenance, Gwennan avait dessiné 6 des huit players disponibles au choix dans le futur menu.

Pour cette deuxième soutenance, elle a donc fini les 2 autres players disponibles.



De plus, sur les huit personnages, quatre sont désormais animés et donc jouables pour le moment.



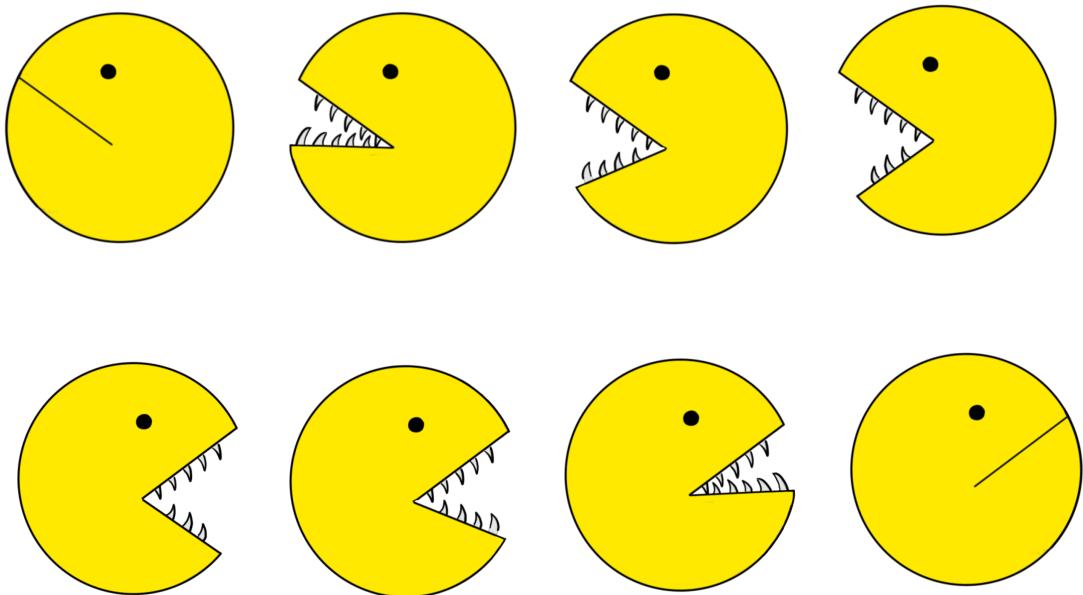
Même si leur différence est à peine visible, si vous regardez bien, vous apercevrez un petit objet sur les épaules droites de deux des quatre joueurs

2.4.2 Les ennemis

Dans la première soutenance, Gwennan avait réalisé 4 mobs ennemis ainsi que leurs déplacements. Elle voulait aussi pour cette nouvelle soutenance réaliser un boss ennemi inspiré de Pacman.

Après réflexion et par soucis de coller avec le personnage de Pacman du jeu classique, Gwennan a décidé de ne pas le dessiner du point de vue face et derrière. Elle a donc seulement dessiné deux côtés du personnage, de même dans son animation.

Voici donc le résultat des sprites bruts :



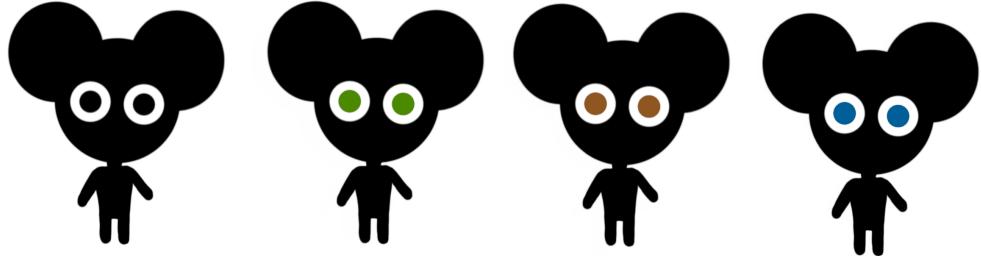
Thomas a aussi dessiné un autre boss :



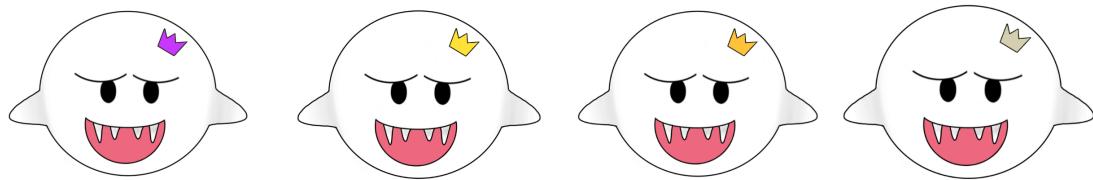
De plus, afin de suivre l'avancement programmé du projet, Gwennan a aussi tenu à réaliser tous les autres mobs ennemis. Ainsi, de seulement 4 mobs ennemis à la première soutenance, le jeu est passé à 20 mobs ennemis au total.

Ainsi, en plus des 4 fantômes colorés inspirés, eux aussi, du jeu Pacman, Gwennan a notamment dessiné 4 Mickey quelque peu troublants, avec des yeux qui reprennent

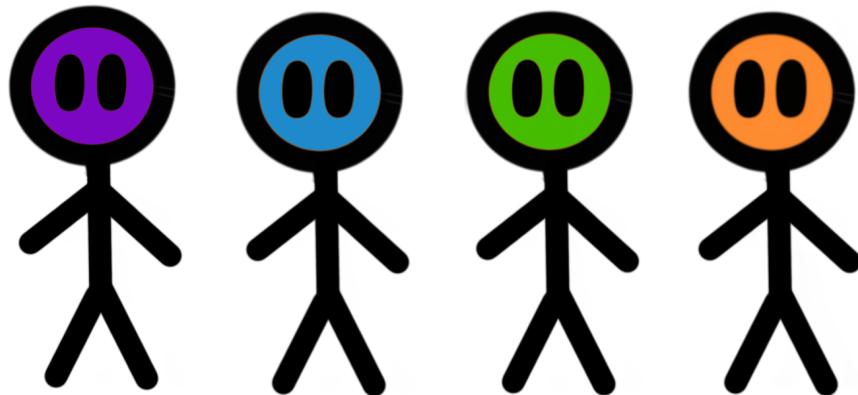
les couleurs des yeux humains, afin à la fois de transformer le fameux personnage de Disney, mais aussi dans le but, en quelque sorte, “d’humaniser” ces fantômes.



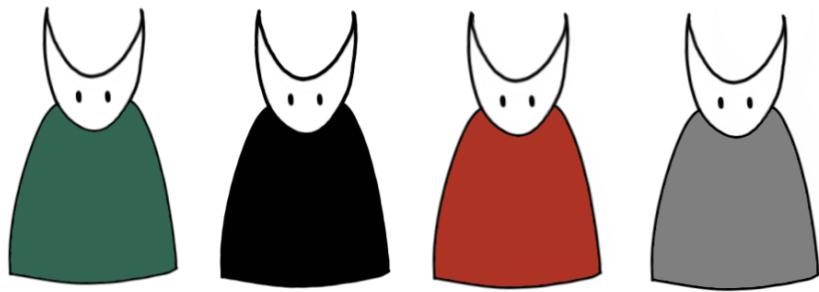
Personnage emblématique de Super Mario, le fantôme Boo a également été redessiné et animé.



De plus, enfant, tout le monde a fait des bonhommes bâtons. Ainsi, en souvenir de cette époque, Gwennan a réalisé ces 4 petits bonhommes.



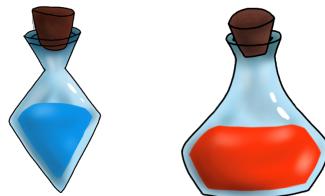
Enfin, en mémoire d'un jeu populaire sorti en 2017, Hollow Knight, voici les derniers mobs dessinés et animés.



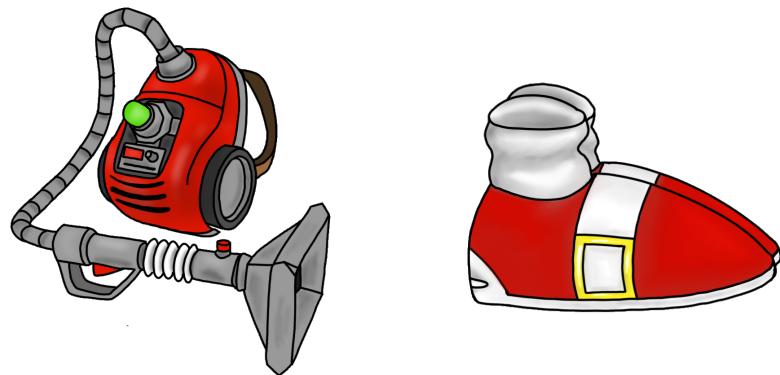
Ainsi, pour la troisième et dernière soutenance, le but sera de faire 3 autres boss ennemis.

2.4.3 Les items

Tout d'abord, il fallait créer des items permettant de redonner de la vie au personnage. Voici donc les deux potions de vie avec en rouge la vie normale et en bleu la vie bonus :

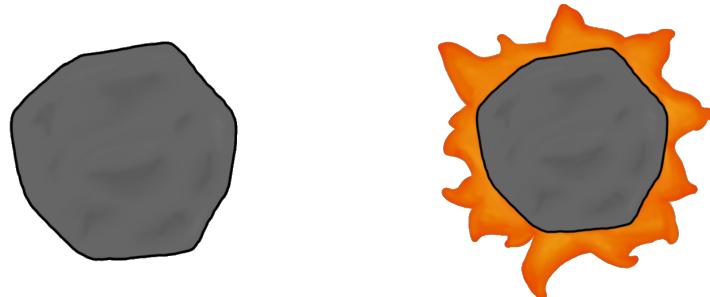


De plus, comme dans tous les jeux, le personnage doit avoir l'accès à des boosters. Ainsi les chaussures de Sonic permettent au joueur de se déplacer plus rapidement et l'aspirateur de Luigi booste l'attaque.



Enfin, afin de rendre plus visuelle l'attaque, Gwennan a aussi dessiné deux armes. La pierre de gauche sera utilisée pour l'attaque normale et celle de droite pour l'at-

taque boostée.



2.4.4 Bilan des graphismes

Il est tout d'abord important de souligner que Gwennan a coordonné les animations de chaque ennemi et joueur avec ses sens de déplacement. Ainsi, par exemple un joueur se déplaçant vers la droite aura l'animation de son personnage marchant du même sens. Il s'agit d'un détail important pour le jeu. En effet, la coordination des graphismes et des déplacements sont des éléments essentiels au bon fonctionnement d'un jeu.

Pour la troisième et dernière soutenance, il sera nécessaire de faire les trois derniers boss ennemis, de finir les animations des 4 players restants et de rajouter d'autres items si nécessaires. Il est aussi important de noter que les rôles de ces items peuvent encore changer, tout dépendra des nouveaux items créés.

Enfin, il est prévu de dessiner une couverture pour le jeu, ainsi qu'un écran de chargement.

2.5 Réseau

2.5.1 Le multijoueur

Bien que Toddlers Are Afraid of Ghost aborde de nombreux aspects de la programmation, une grosse partie de l'intelligence artificielle nécessite plusieurs fonctionnalités comme la génération procédurale ou la capacité de pouvoir tirer sur les ennemis. Ayant besoin du travail de tous pour continuer, Arnaud s'est occupé, pour garder de l'avance, de sa prochaine tâche selon le planning du cahier des charges : la partie gestion du multijoueur.

2.5.2 Multijoueur sur Unity

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 qu'est 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 sortie 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.

2.5.3 Découverte de Mirror

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.

2.5.4 Scène en arrière-plan et développement avec Mirror

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.

2.5.5 Planning pour la prochaine soutenance

Enfin pour finir sur le multijoueur avec Mirror, l'objectif pour la prochaine/dernière soutenance est d'avoir un jeu entièrement jouable en multijoueur local, et même pourquoi pas en réseau. Nous avons déjà conscience que l'implémentation complète du multijoueur provoquera plusieurs bugs que nous régleront évidemment.

2.6 Communication

2.6.1 Le site web

Élément indispensable à la communication du jeu, il fallait rendre le site web, déjà créé pour la précédente soutenance, accessible à tout les appareils.

Pour rappel le site web est accessible via ce lien :

<https://toddlers-are-afraid-of-ghosts.github.io/TAAG-Website/>

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

Ainsi, 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.

Voici une comparaison des deux pages d'accueil :

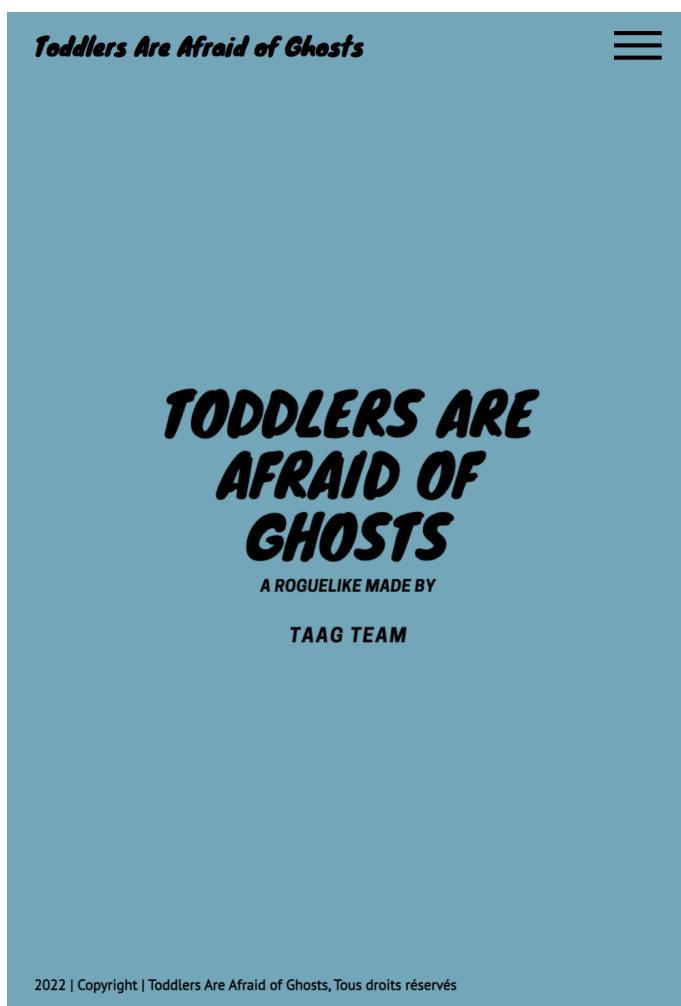
Ci-dessous, un aperçu de l'ancienne page d'accueil.



Même si cette page était plus graphiquement plaisante, elle n'en restait pas moins trop encombrante et mal pensée pour tous les appareils.



Plus sobre, cette page s'adapte à toutes les tailles d'écran.



3 Ressentis des membres

3.1 Arnaud Corcione

Nous avons actuellement créé un jeu jouable et j'en suis assez content. Il reste encore du travail dans certaines parties. Plusieurs problèmes ont surgis comme adapter l'intelligence artificielle ou le développement du multijoueur, bien que ce dernier fut assez simple à implémenter. Ces difficultés m'ont valu beaucoup de réflexion pour trouver des solutions. Nous devons absolument maintenir le rythme pour proposer un jeu entièrement fini et à la hauteur de nos attentes pour la prochaine et dernière soutenance.

3.2 Axelle Destombes

Depuis la dernière soutenance, de mon point de vue, nous avons bien progressé malgré que nous ayons fait face à plus de problèmes que précédemment. J'ai personnellement rencontré plusieurs difficultés liées à la compréhension du fonctionnement de Unity. Par rapport au planning annoncé, nous sommes en plus en avance que prévu sur certains points importants et avons moins avancé sur d'autres. Pour la prochaine soutenance, je compte ajouter avec Gwennan un marchand d'objets, diversifier les bonus disponibles et améliorer notre HUD. Pour celui-ci, nous pouvons par exemple afficher la vie actuelle du joueur, la valeur de son attaque en jeu. Nous allons aussi probablement modifier l'écran d'accueil, les menus et l'écran de pause.

3.3 Gwennan Jarno

Il est assez clair que pour cette soutenance je me suis principalement concentrée sur les graphismes et leur utilisation. Cela m'a pris du temps, mais je compte justement, pour la prochaine soutenance, passer si possible la moitié de mon temps consacré au projet à coder. Même si je préfère ne pas m'avancer encore sur mes plans, tout ce que je développerai sera considéré comme du "bonus" vis-à-vis de notre projet initial. Je suis satisfaite de mon avancement et je n'appréhende plus vraiment la dernière soutenance car je me sens dans les temps et j'ai l'impression que le plus long est derrière moi.

3.4 Thomas Solatges

Pour cette deuxième soutenance, j'ai du reprendre presque de zéro mes précédents avancements, mais je pense en avoir appris ainsi énormément par rapport à la compréhension du langage ainsi que du fonctionnement d'Unity. En réalité, la majorité de ce que je fait est hors Unity et est vraiment théorique. Je pense néanmoins avoir trop développé la théorie à défaut de la pratique. Je programmais à l'aveugle, ce qui m'a coûté beaucoup de temps à terme. Je retiens donc de cette expérience qu'il faut tester chaque fonctionnalité individuellement avant de tout regrouper, afin d'avoir un résultat cohérent.

4 Conclusion

En conclusion, chaque membre a réussi à avancer sur ses tâches, avec plus de facilité d'organisation que pour la première soutenance, même si des problèmes plus complexes nous sont apparus, et nous nous estimons à jour sur le planning pour cette deuxième soutenance, voir même en avance sur certaines parties.

Ainsi, avec toujours la volonté d'avancer et l'esprit d'équipe, TAAG TEAM est assez satisfaite de sa progression et enchantée de voir son avancée dans le projet.

5 Bibliographie

Voici les logiciels et sites que nous avons utilisé :

Pour rédiger ce document :

- Overleaf

Pour tout ce qui est création des images et dessins :

- Sketchbook
- Photoshop
- Unity

Pour tout ce qui est collaboratif et communication entre les membres de TAAG TEAM :

- Git
- GitHub du jeu : <https://github.com/Toddlers-Are-Afraid-of-Ghosts/TAAG>
- GitHub du site : <https://github.com/Toddlers-Are-Afraid-of-Ghosts/TAAG-Website>
- GitKraken
- Discord
- Google Docs

Pour coder :

- Visual Studio Code
- Rider

