

EPITA

Rapport de Soutenance n°3 - projet S2

THE OTHER SIDE

Membres : Hugo Schlegel, Antoine Riquet, Angelina Kuntz, Mathéo Romé

Groupe : Hanabi



Table des matières

The Other Side	5
Origine et nature du projet	5
Présentation du groupe	5
Origine	5
Scénario	5
Objet de l'étude	5
Système et mécanique du jeu	6
Mécanique de vie	6
Système économique	6
Zone et niveaux	6
Classes	8
Ennemi	8
Tableaux des répartitions des tâches	8
Hugo	11
Présentation	11
Première soutenance	11
Création d'assets pour le jeu	11
Bases du site Web	12
Items	12
Cooldown des items actifs	14
Main Menu / Options Menu	14
Animation du cœur (système de vie)	14
Seconde soutenance	15
Création d'assets pour le jeu	15
Gestion de l'évolution du stress	16
Menu pause	16
Implémentation des derniers Items	17
Site web	17
Dernière soutenance	18
Création d'un manuel de jeu	18
Création d'une jaquette pour le jeu	18
Site web	19
Histoire	19
Améliorations permanentes	19
Inventaires	20
Spawn des items actifs	20
Correction d'éventuels bugs et autres	20
Mathéo Romé	21
Présentation	21
Première soutenance	21

Déplacement et Dash	21
Animation	22
Multijoueur	23
Scénario	23
Seconde Soutenance	24
Les Difficultés et les résolutions de bug de la seconde soutenance	24
Feu de camp portatif	24
Dispositif anti-chute infinie (DACL)	25
Sauvegarde	25
Salle torche	25
Don't Destroy on load	25
Amélioration du multijoueur	26
Animation de dash	26
Dernière Soutenance	27
Solo	27
Jarre à pièces	27
Raspberries	28
Resolution de bug	28
Angelina	30
Présentation	30
Première soutenance	30
Début du projet	30
Assets du projet	30
Musique	31
Implémentation de six salles différentes et changement de salle	31
Implémentation d'un PNJ et d'un tableau interactif	35
Seconde soutenance	35
Début du deuxième rendu	35
Correction de bugs	35
Système capteur	36
Génération aléatoire	36
Musique	37
Création de toutes les salles des zones 1 et 2	38
Montage vidéo	39
Dernière soutenance	39
Implémentation des salles défis	39
Création de toutes les salles des zones 3 et 4	40
Approfondissement du lore	40
Amélioration du personnage	41
Scène de crédit	42
Cheatcode	42
Antoine	43
Présentation	43

Première soutenance	43
Début du projet	43
Les objets et le système d'inventaire	45
Système de Stress	48
Seconde soutenance	48
Effets visuels du stress	48
Récapitulatifs de l'ensemble des objets	49
Intelligence artificielle	50
Dernière soutenance	51
Finition de l'intelligence artificielle	52
Système d'améliorations et création du PNJ d'améliorations	52
Les améliorations	53
Ressentis sur le projet	55
Conclusion	55
Annexe	56

The Other Side

Origine et nature du projet

Présentation du groupe

Le groupe Hanabi est mené par Hugo Schlegel et est constitué des membres Mathéo Rome, Angelina Kuntz du groupe 2 et Antoine Riquet du groupe 1. La dénomination *Hanabi* vient du japonais fleur de feu qui se traduit par feu d'artifice. Notre équipe se veut d'une ambiance festive et cette appellation se tient en référence aux traditionnels festivals japonais et plus précisément à une scène de l'anime *Kaguya-sama : Love is war*.

Origine

Ce projet est né d'une passion commune à l'ensemble du groupe pour le jeu *Celeste* et des jeux de type Roguelike. *Celeste* est un jeu vidéo de plate-formes mélangeant action et puzzle sorti en 2018. Dans celui-ci, chaque ensemble d'actions doit être réfléchi et organisé pourachever un tableau. L'histoire du jeu est basée sur le modèle Kübler-Ross, expliquant les cinq phases du deuil.

Notre but, au travers de ce projet, est de réaliser un jeu mélangeant jeu de plate-formes et Roguelike.

Scénario

Le nom du projet *The other side* est fondé sur l'intrigue du jeu, plus précisément sur l'aspect du dédoublement de la personnalité du personnage principal.

L'histoire se déroule à une époque similaire à la nôtre, peu avant le nouvel an, et ainsi, des feux d'artifices. Le personnage principal est un homme qui manque de confiance en soi et qui cherche sa place dans le monde, un but à son existence. Le personnage souffrant du dédoublement de la personnalité, cherche à gravir une montagne afin de se réunir avec son double et d'accomplir un exploit pour se redonner de la valeur et obtenir la reconnaissance des autres.

Le personnage de base sera visible de tous contrairement à son double. Cependant, cela permettra au double de percevoir ce qui est invisible pour le commun des mortels. Le personnage doute de sa personne et se veut visible et invisible à la fois, créant ainsi un double qui existe et n'existe pas en même temps. L'entité du double se base sur le théorème de superposition quantique émis par Schrödinger.

Objet de l'étude

L'objectif au travers de ce projet est de créer un jeu permettant de divertir, mais également d'apprendre les principes fondamentaux d'une "mini-entreprise" comme par exemple la rédaction d'un cahier des charges.

Il nous est également important de développer davantage certaines qualités comme par exemple l'organisation et la répartition du travail au sein du groupe, mais également de travailler de manière équitable et efficace.

Notre but est aussi de développer nos connaissances en informatique comme par exemple l'acquisition de la capacité à réaliser certaines choses comme un site web ou encore l'utilisation de logiciels comme Unity. Mais surtout de développer nos connaissances en programmation afin d'être, plus tard, capables de réaliser des projets en entreprise et de pouvoir se projeter dans l'avenir afin de se donner des objectifs réalistes.

Ainsi, vous l'aurez compris, développer les qualités d'un futur ingénieur informaticien, est notre principal objectif, mais nous n'oublierons pas, tout de même, de nous amuser et, le point le plus important, de prendre du plaisir.

Système et mécanique du jeu

Mécanique de vie

La vie est symbolisée par les pulsations du cœur de notre personnage, son stress, représenté par un cœur battant dans le coin supérieur gauche de notre écran. Plus il est stressé, plus celui-ci se sent en danger, allant jusqu'à halluciner et percevoir un monstre, ayant pour but de mettre fin à son ascension de la montagne. La barre de stress est commune aux deux joueurs. L'unique moyen de connaître le niveau de stress de notre personnage est la vitesse à laquelle bat son cœur, il n'y aura aucun chiffre présent à l'écran pour représenter le taux de stress du personnage.

Si le taux de stress de notre personnage dépasse les 200 (la valeur maximale étant de 220), l'écran se trouble et la musique devient distordue. Un ennemi apparaît alors et suit notre personnage en permanence. Lorsque l'on parvient à retomber sous les 200 points de stress le monstre disparaît et notre personnage se calme, représenté par l'écran qui perd son aspect troublé et la musique redevenant calme.

Système économique

Il y a deux types de monnaies : Tout d'abord la monnaie courante, représentée par les pièces d'or, les pièces peuvent être gagnées via la casse de jars présentes dans les niveaux rapportant chacun entre 0 et 5 pièces. Il y a environ 2 ou 3 pots par salle. Ces pièces peuvent être dépensées dans les boutiques rencontrées durant la partie et seront perdues en cas de mort des personnages. Les boutiques sont des types de salle permettant d'acheter de nouveaux équipements permettant de faciliter l'ascension. Elles apparaîtront aléatoirement une fois par niveau et le joueur aura accès à trois catégories d'objet : brisé, usé et neuf.

La seconde monnaie est le *raspberry* représentés par une icône en forme de framboise. C'est une monnaie récupérable directement dans les salles. Pour l'obtenir, le joueur devra se mettre en danger en s'éloignant du chemin le plus simple pour compléter une salle. À la fin d'une partie, les raspberries ne sont pas perdus mais stockés et le joueur a accès à une boutique d'amélioration qu'il rencontre à chaque nouvelle game ou à chaque mort.

Zone et niveaux

Le jeu est constitué de trois niveaux différents. Le premier niveau n'aura qu'une zone intitulée "Contrefort". Le second niveau offrira deux zones possibles, le joueur devra choisir l'une des deux

zones et devra recommencer une autre partie s'il veut explorer l'autre zone qu'il n'aura pas sélectionnée. Pour encourager le joueur à recommencer une partie il faudra explorer la totalité des zones pour connaître tout le scénario. Le joueur aura donc le choix entre la seconde zone, en extérieur sur le flanc de la montagne dans "Les sentiers enneigés", ou la troisième dans "Cœur de la montagne". Le dernier niveau n'aura qu'une zone, "le Sommet". Au total il y aura 3 niveaux et 4 zones. Les zones seront générées de façon aléatoire.

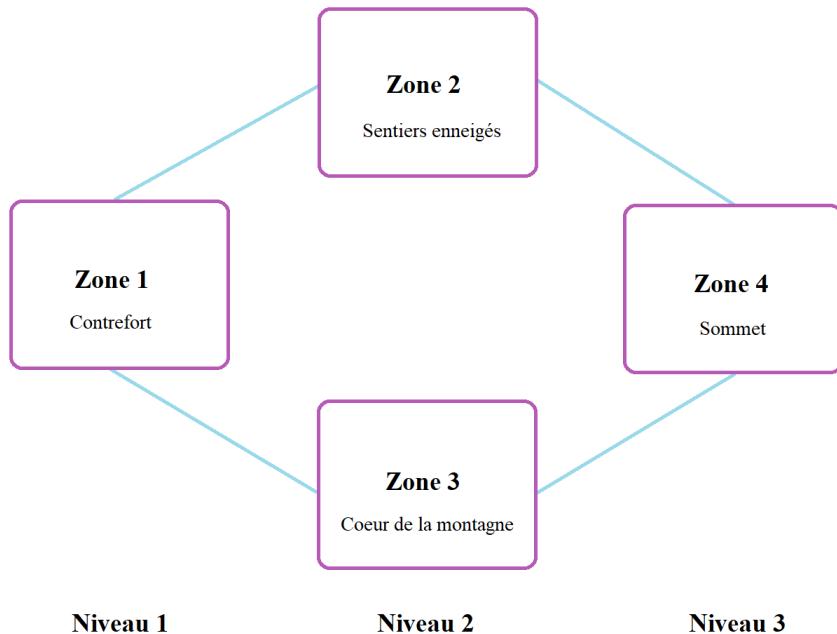


FIGURE 1 – Schéma des zones et niveaux

Dans ces zones il sera possible de rencontrer quatre types de salles différentes :

- Les salles feu de camp, une salle simple avec juste un feu de camp au centre, intéragir avec ce feu permettra au joueur de faire baisser son stress.
- Les salles histoire, une salle avec principalement du dialogue qui permettra de faire avancer le scénario. Ces salles se différencieront à leur décoration (présence de tableaux entre autres).
- Les salles boutique, qui permettent au joueur de dépenser les pièces d'or accumulées pour acheter des items. Pour accéder à la boutique il faudra intéragir avec le PNJ présent dans la salle
- Les salles épreuve, salles puzzle où le joueur passe la majorité de son temps. (Jump, Key, Captor, Torch, Button)
- Les deux joueurs seront sur la même scene et leur deux salle seront une au dessus de l'autre. Ils évoluerons dans le même environnement bien qu'ils ne se voient pas.

Classes

Chacun des deux joueurs pourra choisir un équipement de départ pour son personnage, lui procurant un type de saut différent et des capacités passives de départ. Chaque équipement aura un tee-shirt de couleur différent. Les équipements disponibles sont avec en référence le bloc ayant pour largeur 16 pixel :

- Le *Classic* avec une vitesse de déplacement normale, un saut de hauteur normale (2 blocs), une distance de dash normale (3 blocs) et la possibilité de dash dans huit directions. Le tee-shirt est rouge.

- Le *Bouncy* avec une vitesse de déplacement lente, un saut de hauteur plus élevé que le *Classic* (3 blocs), une distance de dash plus élevé que le *Classic* (4 bloc) et la possibilité de dash dans deux directions, droite/gauche. Si lors d'un dash il rencontre un obstacle (ex : mur) le joueur sera propulsé vers le haut d'une hauteur supérieure au saut(3 bloc) et aura de nouveau la possibilité de dash. Le tee-shirt *Bouncy* est bleu.

- Le *Light* avec une vitesse de déplacement rapide et un saut de hauteur supérieure(5 blocs). Le dash est remplacé par une téléportation, c'est à dire qu'à la place d'avoir le mouvement du déplacement d'un point A à un point B, le personnage passera du point A au point B sans déplacement. Vu qu'il n'y aura plus de déplacement, il y aura de nouvelles possibilités de gameplay comme se téléporter de l'autre côté d'un obstacle. La distance du TP est plus importante que celle du dash. Il sera possible de se téléporter dans deux directions vers la droite et la gauche sur distance courte(2 bloc). Le tee-shirt *Light* est violet.

Ennemi

Si le taux de stress de notre personnage dépasse les 200 (la valeur maximale étant de 220), l'écran se trouble et la musique devient distordue. Un ennemi apparaît alors et suit notre personnage en permanence. Une ombre (fantôme) va alors suivre notre personnage avec un intervalle de 0,5 seconde, de plus, toutes les 5 secondes, si notre stress est toujours au dessus de la barre des 200, une nouvelle ombre va apparaître avec encore une fois un intervalle de 5s par rapport à la précédente ombre et ainsi de suite tant que notre barre de stress reste au dessus des 200. Si nous ne parvenons pas à réduire notre barre de stress et que le joueur entre en contact avec une ombre, c'est Game Over. Lorsque l'on parvient à retomber sous les 200 points de stress les ombres disparaissent et notre personnage se calme, représenté par l'écran qui perd son aspect troublé et la musique redevenant calme.

Tableaux des répartitions des tâches

Petite pique de rappel de ce que chaque membre devait accomplir pour chacune des soutenances.

Soutenance 1	Hugo	Matheo	Angelina	Antoine
Trouver tous les assets libre de droits à utiliser			x	
Dessiner l'idle animation	x			
Dessiner l'animation de saut	x			
Dessiner l'animation de marche	x			
Dessiner un PNJ et son animations	x			
Implémenter le déplacement du personnage		x		
Implémenter le dash classique		x		
Implémenter le dash Bouncy		x		
Implémenter le dash Light		x		
Implémenter l'idle animation du personnage		x		
Implémenter l'animation de marche du personnage		x		
Implémenter l'animation de saut du personnage		x		
Implémenter un PNJ avec lequel interagir			x	
Trouver toutes les musiques libre de droits à utiliser			x	
Implémenter la musique de la première zone			x	
Implémenter le cœur représentant le stress du personnage sur l'UI	x			
Rédaction du scénario de la première zone		x		
Permettre de rejoindre un lobby à l'aide de photon		x		
Implémenter une salle test			x	
Implémenter une salle jump			x	
Implémenter une salle button			x	
Implémenter une salle key			x	
Implémenter une salle histoire			x	
Implémenter un changement de salle			x	
Créer le squelette du site web	x			
Implémenter un Main menu (accès au jeu et options)	x			
Implémenter une boutique vendant 3 items aléatoire au PNJ				x
Implémenter l'item passif : Potion de vitesse				x
Implémenter l'item passif : Potion de saut	x			
Implémenter l'item passif : Guide de l'épicier				x
Implémenter l'item passif : Pretzel d'ur chance				x
Implémenter l'item actif : Plutôt deux fois Khune	x			
Implémenter l'item actif : Updraft	x			
Implémenter l'item actif : Tarte				x
Implémenter le système de recharge des items actifs (Items se rechargent avec le temps)	x			
Implémenter le ramassage d'item et de Coins				x
Implémenter l'inventaire passif avec les items passifs possédés				x
Implémenter l'inventaire actif avec l'item actif en main				x
Implémenter le compteur de Coins				x

Soutenance 2	Hugo	Matheo	Angelina	Antoine
Dessiner l'animation du dash	x			
Implémenter l'animation du dash		x		
Implémenter les variations du stress du personnage, augmentation continue du stress	x			
Implémenter les variations visuelles du stress, blur et musique saturée				x
Implémenter la génération aléatoire			x	
Implémenter une salle capteur			x	
Implémenter une salle torche		x		
Implémenter les salles feu de camp	x			
Implémenter la bibliothèque de la zone 1			x	
Implémenter la bibliothèque de la zone 2			x	
Implémenter une IA capable de reproduire les mouvements du joueur				x
Permettre au joueur 1 de stocker localement toutes les données de la partie (Améliorations achetées, Strawberries et Starfruits en stock)		x		
Implémenter la stabilisation du stress dans les salles boutiques et feu de camp	x			
Implémenter un menu pause	x			
Implémenter tous les items achetables de la boutique				x
Implémenter l'item passif : Long-fall boots	x			
Implémenter l'item passif : Feu de camp portatif		x		
Implémenter l'item actif : The World	x			
Avoir rempli le site web dans son intégralité à l'exception du lien de téléchargement	x			

Soutenance 3	Hugo	Matheo	Angelina	Antoine
Implémenter les salles défis			x	
Implémenter la bibliothèque de la zone 3			x	
Implémenter la bibliothèque de la zone 4			x	
Finir le site web	x			
Terminer l'IA				x
Implémenter le changement entre les deux personnage en mode solo.		x		
Gérer la synchronisation des inventaire pour le J1 et le J2.	x			
Implémenter le PNJ amélioration du personnage				x
Implémenter les améliorations du personnage	x	x	x	x
Chasse aux bugs	x	x	x	x

Hugo

Présentation

Bonjour je m'appelle Antoine Hugo du groupe HANABI et j'aurai pour notre projet, *The Other Side*, été très polyvalent, touchant à presque tous les aspects de celui-ci, mais travaillant cependant majoritairement, sur les objets et le stress, système de vie de notre jeu ainsi que l'aspect graphique et *marketing* de celui-ci. Ce projet n'aura pas été le premier auquel j'ai été confronté, ayant déjà travaillé sur mon TPE en Première et mon projet de programmation en ISN en Terminale, mais il reste cependant le plus poussé et celui auquel je me serai le plus dédié. Travailler sur *The Other Side* aura été un réel plaisir pour moi, en plus de pouvoir renforcer mes capacités en Unity et C# ainsi que mes capacités générales en code, j'ai aimé pouvoir travailler avec mes camarades en pleine liberté sur ce projet qui, je le crois, nous aura fait passer d'excellents moments.

Première soutenance

Pour la première soutenance, j'aurais été assez polyvalent en ce qui concerne mes tâches. J'aurais travaillé sur les animations de notre personnage, sur les bases de notre site ainsi que directement sur Unity. Tout comme mes camarades j'ai regardé bon nombre de tutoriels sur Unity, et plus précisément Unity2D, venant tous des chaînes youtube de [tuto Unity FR](#) et [Brackeys](#).

Création d'assets pour le jeu

La première chose que j'aurais faite, et celle qui m'aura permis de rentrer pleinement dans notre projet est la conception d'animations pour notre jeu ainsi que de certains autres assets. C'est à l'aide du logiciel gratuit Piskel que j'ai dessiné nos assets. Le logiciel étant relativement bien fait, sa prise en main est plutôt rapide et j'ai, après très peu d'entraînement pu commencer à travailler.

J'ai également pris le temps de créer une palette de couleurs qui m'aura servi lors de la création de tous mes assets, afin de rester cohérent dans mes couleurs, celle-ci se retrouvant en bas à droite de l'interface.

J'ai commencé par les animations dites d'idle du personnage, ces animations sont celles qui interviennent lorsque le personnage est immobile. Ne demandant pas de vastes mouvements, ces animations sont assez simples à réaliser. J'en ai également profité pour faire l'animation d'idle de notre marchand, très similaire à celle des personnages jouables, elle aura été assez agréable à faire.

Je suis ensuite passé aux animations de marche. Bien plus difficiles que celles d'idle, elles auront pris un peu plus de temps, mais, maîtrisant déjà un peu plus le logiciel qu'au début, j'aurais tout de même réussi à arriver à un résultat plutôt satisfaisant.

Sont ensuite venues les animations de saut et de chute. Je suis passé par bon nombre de brouillons avant d'arriver à un résultat satisfaisant mais comme pour toutes les autres animations je suis au final arrivé à un résultat qui me convenait.

À noter que comme notre jeu permet de jouer avec trois classes différentes, toutes ces animations existent en trois variantes différentes qui se distinguent par la couleur du T-shirt de notre personnage, bleu, rouge ou violet. Ces variations n'ont en aucun cas été difficiles à faire, juste assez time-consuming, Piskel ne proposant pas d'outil pour changer tous les pixels d'une couleur en une autre.

Toutes ces animations sont constituées de plusieurs dessins mis les uns à la suite des autres et sont supposées tourner en 12 images par seconde, d'où le fait que celles-ci sont toutes constituées d'un nombre pair de dessins.

Mis à part ces animations j'ai fait l'asset de la raspberry, un collectable trouvable un peu partout.



FIGURE 2 – Les trois classes : Heavy, light et classic

Bases du site Web

Je m'étais également occupé pour la première soutenance de coder une base solide en html/css pour notre site. Ayant déjà eu l'occasion d'en faire un lors de ma dernière année au lycée j'aurais pu me lancer assez rapidement dans la création de celui-ci, sans avoir à passer par des tutoriels ou autres vidéos explicatives.

Je me suis donc occupé de coder un menu déroulant fonctionnel, qui aura été la chose la plus ardue à mettre en place. Je me suis ensuite occupé de mettre en place tous les fichiers dont j'aurais besoin à l'avenir afin de faire en sorte que le menu soit fonctionnel sur chacun d'entre eux. Je me suis ensuite occupé de remplir les premières pages du site, soit la page Accueil ainsi que les pages relatant les origines du projet.

Avec une base solide, j'étais prêt à le remplir à chaque soutenance afin de relater notre avancée ainsi que de rendre celui-ci plus agréable à parcourir.

Items

Pour cette première soutenance, j'avais pour mission de créer les items suivants :

- L'item passif Potion de saut

- L'item actif Plutôt deux fois Khune
- L'item actif Updraft

Je vais dans cette partie décrire le fonctionnement de ces items un à un, ceux-ci ayant tous des effets grandement différents.

Potion de saut : Cet item étant un item passif, une fois acheté, il restera en permanence dans l'inventaire passif et ces effets seront constamment appliqués au joueur. La potion de saut est comme tous les autres items passifs disponible en 3 variantes : brisée, usée et neuve, ayant respectivement pour prix 40, 70 et 100 pièces. Chacune augmente la *JumpVelocity* de notre personnage, variable qui permet d'ajuster la hauteur de son saut, de 0.33, 0.66 et 1, soit l'item neuf doublant la hauteur du saut.



FIGURE 3 – visuels de la potion de saut

Plutôt deux fois Khune : Contrairement à la potion de saut, cet item est un item actif, c'est à dire qu'il ne peut être stocké que dans l'unique case de l'inventaire actif (un seul item actif peut être tenu à la fois). Comme tous les autres items actifs, les effets de celui-ci ne sont pas appliqués automatiquement, le joueur doit appuyer sur la touche assignée à l'utilisation d'un item actif afin de déclencher l'effet de ce dernier et devra attendre un certain temps avant de pouvoir l'utiliser à nouveau. Une fois utilisé, l'item Plutôt deux fois Khune permet au joueur d'effectuer un second dash en l'air, en effet, une fois utilisé, celui-ci met la variable *hasDashed* du joueur à false, lui permettant alors d'en effectuer un second. Le temps de recharge de cet item est de 15 secondes.

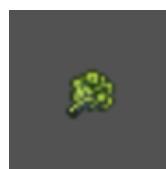


FIGURE 4 – visuel de Plutôt deux fois Khune

Updraft : Tout comme Plutôt deux fois Khune, l'updraft est un item actif et fonctionne donc de la même manière. L'updraft, quand utilisé, envoie au script contrôlant le joueur l'information comme quoi celui-ci serait sur le sol et aurait appuyé sur le bouton saut, le faisant alors sauter même si celui-ci est en l'air, lui permettant alors d'effectuer un double saut. Cet item a un temps de recharge de 15 secondes.



FIGURE 5 – visuel de Updraft

Cooldown des items actifs

Comme dit dans la section précédente, les items actifs, avant de pouvoir être réutilisés, doivent d'abord se recharger. Une fois un item actif utilisé, son temps de recharge restant apparaît en dessous de son icône et baisse de seconde en seconde jusqu'à afficher *Ready* une fois le cooldown terminé, laissant alors savoir au joueur qu'il peut à nouveau utiliser l'item. Et je fus donc celui chargé d'implémenter ce système.

Lorsqu'un Item est utilisé, la variable *cooldown* est set à la valeur de cooldown de l'item en question plus le temps actuel passé in game (Unity gardant toujours trace de celui-ci). Ainsi il est impossible pour le joueur d'utiliser une nouvelle fois l'item tant que ce temps passé in game n'est pas supérieur à la variable *cooldown*. La variable *cooldown* est à chaque frame décrémentée du temps (dit *deltaTime*) qui s'est écoulé depuis la dernière frame, afin que le cooldown soit toujours décrémenté de la même façon même avec la présence de lags.

J'ai rencontré quelques difficultés à implémenter ce programme, bien qu'en apparence simple, ayant eu quelques problèmes à afficher correctement le compte à rebours et à le décrémenter de façon constante, mais après avoir visualisé de nombreux tutoriels j'aurais réussi à faire quelque chose dont je suis grandement satisfait.

Main Menu / Options Menu

Je fus également celui chargé de mettre au point un main menu ainsi qu'un menu d'options. J'étais assez réticent à l'idée de m'attaquer au menu d'options, l'idée de travailler directement avec les paramètres de la machine me paraissant assez ardue. Mais heureusement, Unity aura rendu la tâche bien plus facile que je ne le pensais !

J'ai donc commencé par faire le main menu, composé de trois boutons : **PLAY**, **OPTIONS** et **QUIT** qui permettent respectivement de lancer le jeu, d'accéder à l'Options menu et de quitter le jeu. Le menu sera sûrement amené à changer à l'avenir, que ce soit de part l'ajout d'un logo ou encore d'un changement d'arrière plan.

Le menu d'options lui est composé d'un bouton **BACK**, qui permet de retourner au main menu, un slider volume qui permet d'ajuster le volume in game, une checkbox qui permet de passer le jeu en plein écran ainsi qu'un bouton permettant de sélectionner une définition pour les graphismes du jeu. Tout ça grâce à des boutons et options proposées par Unity directement.

Animation du cœur (système de vie)

Comme dit précédemment, la vie de notre personnage est représentée sous forme de stress, plus notre personnage est stressé, plus son cœur bat vite à l'écran, afin de permettre au joueur de prendre connaissance de la vie de son personnage de façon plus subtile qu'avec de simples chiffres.

J'ai donc créé trois animations différentes, chacune associée à un état de stress du personnage (rappelons que notre personnage commence avec 0 points de stress et avec un maximum de 220). Lorsque le stress est inférieur à 100, notre cœur en haut à gauche de notre écran jouera l'animation *HeartBeat slow*, si son stress est supérieur à ce seuil de 100 points, le cœur joue l'animation *HeartBeat fast* et si le stress est supérieur à 150, le cœur jouera l'animation *HeartBeat crazy*.

Seconde soutenance

Pour la seconde soutenance, j'aurais encore eu la chance de travailler sur de nombreux aspects de notre jeu. J'aurais premièrement continué à travailler sur les assets de notre jeu avant de m'occuper de gérer l'évolution du stress de notre personnage ainsi que l'implémentation des salles "feu de camp". La suite aura pour moi été de travailler sur le menu pause puis de terminer de créer tous les items restants et enfin remplir et peaufiner le site web. Travailler sur cette soutenance, difficultés mises à part, aura été plus fluide que pour la première, ayant déjà acquis les base de Unity et n'ayant pas à faire des allers-retours entre mon code et divers tutoriels.

Création d'assets pour le jeu

J'ai été chargé pour cette seconde soutenance de finir de dessiner les assets de notre personnage principal, soit les animations de dash ainsi que ceux de notre IA, à l'allure fantomatique et ressemblant à notre personnage. Tout comme pour la première soutenance, j'ai utilisé Piskel pour dessiner les quelques assets restants, assets dont je suis plutôt satisfait, maîtrisant bien mieux le logiciel.



FIGURE 6 – Tile set de notre fantôme

J'aurais commencé par faire l'animation de notre IA. Elle aura été faite pour ressembler à notre personnage tout en ayant un air plus menaçant. Celle-ci n'est visible que quand notre stress passe un certain seuil, après lequel elle retrace tous nos mouvements, la moindre erreur ou doute pouvant nous être fatal (cf. le rapport d'Antoine sur l'IA).

J'aurais ensuite travaillé sur les animations des dashes. Simples en apparence, j'ai du en réalité refaire celui-ci plusieurs fois pour atteindre un certain niveau de fluidité. J'ai également du faire cette animation dans 8 directions différentes, le dash classiques pouvant dasher dans 8 directions. L'animation est de couleur en fonction de la classe respectant le code couleur précédemment énoncé. Il existe donc également deux tile sets pour le dash boucny (celui-ci ne pouvant dasher que à droite

ou à gauche). La classe light elle n'avait pas besoin d'animations de dash, le sien consistant en une téléportation.

Gestion de l'évolution du stress

Ma seconde tâche pour cette soutenance était d'implémenter l'évolution continue du stress, les joueurs gagnent du stress à raison d'un point toutes les deux secondes. Celui-ci augmente en continu avec un maximum de 200 points de stress, palier qui une fois atteint résulte en l'apparition de l'IA. Il existe cependant quelques exceptions à cette règle, j'ai fait en sorte que le stress n'augmente pas dans les salles histoire et les salles feu de camp, qui sont faites pour que le joueur puisse prendre son temps et se reposer, et ce, en vérifiant au moment d'y entrer l'index de la salle afin d'en déduire son type, n'augmentant le stress que si l'index ne correspond pas à l'une des salles histoire ou feu de camp.

Je me suis justement aussi occupé d'implémenter ces salles feu de camp. Comme mentionné juste avant, le stress ne monte pas dans ces salles, une seule apparaîtra par étage de la montagne et il est impossible de chuter ou mourir dans celles-ci. Au milieu de ces salles se trouvera toujours un feu de camp et si le joueur s'approche du feu, soit qu'il touche son collider, son stress sera diminué de soixante pour cent. J'ai délibérément fait en sorte que le stress ne soit retiré au joueur que s'il s'approche du feu de camp de telle façon à ce que, s'il le souhaite, il puisse rendre sa run plus difficile en décidant de ne pas faire usage de ce bonus.



FIGURE 7 – Salle feu de camp de la zone 1

Menu pause

Tout de suite après avoir terminé de travailler sur les salles feu de camp, je me suis occupé de faire le menu pause de notre jeu. Il peut être ouvert à n'importe quel moment en appuyant sur la touche *echap*, freezant le jeu, le temps arrêtant de s'écouler et assombrissant le fond d'écran. Sur ce menu sont visibles trois boutons, *RESUME*, *MENU*, *OPTIONS* et *BACK*, qui permettent respectivement de reprendre le cours du jeu, de retourner au main menu, d'accéder au sous menu options du menu pause et de quitter le jeu. Pour nous ramener au main menu, le bouton *MENU* nous ramène à la salle d'index 0, qui correspond toujours à l'index du main menu. À noter qu'il est aussi possible de sortir du menu pause à tout moment en pressant la touche *echap*.

Le sous menu options est presque identique à celui présent dans le main menu, on y retrouve les même options de *Fullscreen*, *graphics* et *Volume*. Le bouton *BACK* lui nous ramène au premier

menu pause. Il est également possible de reprendre le cours du jeu à partir de ce menu en pressant la touche echap.

Implémentation des derniers Items

Pour ce qui est des items, je me suis occupé de l'item passif *long-fall boots* et de l'item actif *The world*.

Comme tous les items actifs, les *Long-fall boots* peuvent être trouvées dans des boutiques dans trois variétés différentes, *broken*, *used* et *new* qui, une fois achetées, resteront de façon permanente dans l'inventaire passif. Cet item à pour effet de réduire le nombre de points de stress gagnés au moment de tomber dans un trou. S'il est dans l'état *broken*, l'item nous sauve 1 point de stress en tombant, s'il est dans l'état *used* il nous en sauve 2 et s'il est dans l'état *new* il nous en sauve 3. Ces valeurs peuvent sembler petites, mais il faut rappeler qu'il est possible de posséder plusieurs fois le même item passif afin d'additionner ses effets, il est donc même possible si l'on a la chance de trouver plusieurs fois cet item de ne plus du tout gagner de stress en tombant.

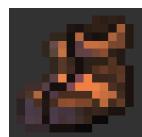


FIGURE 8 – Long-fall boots

Pour ce qui est de *The world*, étant un item actif, un fois possédé, il est stocké dans le slot des items actifs et il doit être activé afin d'utiliser son effet. Une fois utilisé, cet item permet d'effectuer un dash identique à celui de la classe light, soit une téléportation vers la gauche, le haut ou la droite. Le cooldown avant la prochaine utilisation de l'item est de 15 secondes, comme tous les autres items actifs de mouvement.

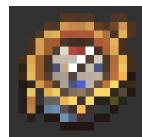


FIGURE 9 – The world

Site web

Pour ce qui est du site web, je me suis occupé de le remplir dans sa quasi-intégralité, laissant juste de côté les sections *Téléchargement* et *Bibliographie* que je ne serai en mesure de compléter qu'une fois le jeu terminé, comme nous ne cessons de chercher de nouvelles sources d'inspiration et que notre jeu n'est pour l'instant pas complet. Je me suis aussi occupé de changer les noms et de revoir certaines sections afin d'au mieux pouvoir présenter notre jeu. La section *Accueil* est une simple présentation de notre groupe ainsi que de ce que notre jeu se veut être. La section *Pourquoi The other side* elle met en avant les origines de notre projet ainsi qu'un bref résumé du scénario.

La section *Avancement et objectifs* contient les fichiers.pdf de nos rapports de soutenance et de notre cahier des charges, pour les plus curieux qui voudront en apprendre plus sur comment nous avons fait pour venir à bout de notre projet. La section *Comment jouer* est la plus longue d'entre toutes, elle contient tout ce que vous devez connaître de jouer à *The other side*, toutes les mécaniques des salles, des différentes classes, les contrôles et autre y sont expliqués. Enfin, la section *Contact* contient les mails de tous les membres du groupe ainsi que l'adresse et le nom de notre établissement.

Dernière soutenance

Pour cette dernière soutenance, j'aurais surtout travaillé à préparer la soutenance, ainsi que sur le site web. J'aurais également aidé à implémenter une des améliorations permanentes ainsi qu'à la fusion des inventaires des deux joueurs.

Création d'un manuel de jeu

En m'aidant de ce que j'avais déjà écrit sur mon site, j'ai réalisé sur latex un manuel de jeu complet, comprenant un guide d'installation pour le jeu ainsi que plusieurs conseils et contrôles essentiels afin de pouvoir jouer à *The Other Side*. Celui-ci est disponible en téléchargement sur notre site.

Création d'une jaquette pour le jeu

J'aurais également créé la jaquette de notre jeu sur le site canva.com, celle-ci reprend tous les éléments phares du jeu et a été faite pour rester le plus simpliste tout en résumant au mieux celui-ci. Elle sera imprimée afin d'être apposée sur la boîte qui contiendra le jeu lors de la dernière soutenance et est également retrouvable sur le site sur la page de téléchargement.



FIGURE 10 – La jaquette de notre jeu

Site web

J'ai aussi pour cette soutenance terminé le site web dans son intégralité. Je me suis occupé de faire en sorte qu'il n'y ai plus de bugs visuels comme il pouvait y en avoir par le passé et j'aurais rempli toutes les pages restantes. La page *Comment jouer* aura été approfondie pour comprendre les contrôles rajoutés entre temps, la page *Bibliographie* est maintenant complète et comprend des liens vers tous les assets et musiques utilisées dans notre projet. La page de téléchargement est elle aussi fonctionnelle et on y retrouve un jeu en lui même ainsi que le manuel évoqué précédemment, soit tout pour pouvoir vous mettre à jouer à The Other Side.

Histoire

Comme tout bon jeu, il nous fallait une histoire qui vous donne envie d'en savoir plus, envie de rejouer. C'est pour cette raison que celle-ci vous est contée petit à petit dans The Other Side. Au fur et à mesure de votre partie, vous arriverez dans des salles histoires qui sont générées aléatoirement parmi toutes celles disponibles, ce qui signifie que vous ne tomberez pas sur la même à chaque partie. Dans ces salles se trouvent des tableaux qui vous content l'histoire du jeu, et j'ai donc pour cette troisième soutenance pris le soin de rédiger cette dite histoire, soit un total de 11 textes au total. On y retrouve principalement deux types de textes, ceux qui nous narrent de façon abstraite l'histoire de la montagne, pourquoi nous la montons ou encore des réponses à certains mystères qui l'entourent ainsi que des textes, tous reliés cette fois-ci, qui nous content l'histoire de quatres aventuriers. Ces derniers avaient tout comme notre personnage décidé de grimper la montagne et ont laissé à plusieurs endroits clefs des notes relatant leurs mésaventures, ce sont ces notes que nous allons retrouver au fur et à mesure, afin d'en apprendre plus sur le sort de ces quatre aventuriers.

Vous trouvez un mot accroché au tableau, il est décoré d'un beau ruban doré, peut-être sera-t-il important pour la fin de votre périple :

« Bonjour je m'appelle Antoine... Bonjour je m'appelle Antoine... Bonjour je m'appelle Antoine... »

Vous avez comme une impression de déjà-vu... Dans tous les cas, vous ne comprenez pas l'intérêt de le répéter 3 fois dans un même texte...

FIGURE 11 – Un exemple non exhaustif des textes d'histoire de notre jeu

Améliorations permanentes

J'ai également aidé à implémenter les améliorations permanentes. Celles-ci sont achetables en fin de partie avec les raspberries, seconde monnaie du jeu. Elles sont sauvegardées et chacune possède un total de cinq niveaux, afin de rendre vos futurs runs plus facile, but de tout rogue-like afin de pouvoir plus tard venir à bout du jeu. Le marchand fur codé par Antoine et designé par moi et se trouve dans une salle, faite par Angelina. Comme dit, on apparaît dans la salle dès que l'on commence une run ou que l'on meurt, c'est une sorte de HUB pour se préparer à votre

prochaine Ascension. En plus d'avoir participé à la mise en place générale de ce système, j'aurais également participé à la création d'une des améliorations, et sûrement l'une des plus embêtantes à implémenter par ailleurs, *Random.org*. Celle-ci, en fonction de son niveau d'amélioration permet au joueur de commencer sa run déjà équipé de certains items actifs choisis au hazard, soit avec 1 item casé au niveau 1, 1 item usé au niveau 2, 1 item neuf au niveau 3, 1 item casé et un item neuf au niveau 4 ou encore 1 item usé et un item neuf au niveau 5. Tout ça fut assez embêtant à implémenter mais l'amélioration est bien fonctionnelle dorénavant, tant bien en solo qu'en multi, grâce à l'aide de mathéo.

Inventaires

Je devais pour cette soutenance, dans la continuité de mes précédents travaux sur les inventaires, m'occuper de synchroniser les inventaires passifs des joueurs 1 et 2, travail qui demandait que je plonge pleinement dans l'aspect multijoueur de notre jeu. J'ai eu beaucoup de mal pour trouver comment venir à bout de cette tâche, ayant très peu de connaissances sur photon. Cependant, Mathéo, qui lui est responsable multijoueur de notre projet, lui était bien plus à l'aise sur le sujet et aura au final résolu le problème en un rien de temps. Je ne peux donc pas dire que je suis celui qui a résolu le problème mais j'aurais tout de même, en essayant, acquis de nouvelles connaissances quant à photon et certains fonctionnements du multijoueur. Je ne regrette donc pas d'avoir été assigné cette tâche malgré les difficultés. Maintenant, lorsque le joueur 1 achète un item passif, celui-ci se retrouvera aussi dans l'inventaire du joueur 2, permettant aux deux joueurs de partager les améliorations.

Spawn des items actifs

Pour ce qui est des items actifs, il n'y avait pas de façon concrète d'en obtenir, c'est pourquoi je me suis occupé de les faire apparaître de façon aléatoire dans certaines salles prédéfinies. Un item aléatoire parmi les 4 existants a donc une chance sur 8 de spawner dans les salles torches, campfire et la première salle de chaque zone. Le spawn d'un item du côté du joueur 1 ne garantit pas que le joueur 2 en verra aussi un apparaître chez lui, les chances sont indépendantes. Aussi, ramasser un nouvel item actif remplacera celui actuellement possédé, sans pouvoir le récupérer, à vous de choisir ou non de changer.

Correction d'éventuels bugs et autres

Comme les autres membres du groupe, j'aurais aussi aidé à fixer certains bugs. Notamment dans les menus et au niveau de certains items, la plupart d'entre eux étant des bugs mineurs en rapport avec mes parties (j'aurais également découverts certains bugs mais n'étant pas en capacité de corriger ceux-ci, ils seront détaillés dans les parties des autres membres). J'aurais également pris du temps pour revoir les menus et les boutons, que ceux-ci apparaissent un tant soit peu plus esthétiques.

Mathéo Romé

Présentation

Je me présente, je m'appelle Henry Mathéo Romé et je me suis principalement occupé de la partie programmation du projet. Pour ce dernier, j'ai été en charge entre autres de l'implémentation du solo, du multijoueur ainsi que d'aider mes camarades sur certains points de leurs parties. Ce n'est pas le premier projets que j'ai dû mener à bien lors de ma scolarité (cf le TPE en première et le projet interdisciplinaire en terminal) mais c'est de loin le plus structuré et complet que j'ai plus accomplir. J'ai toujours apprécié accomplir ce type de projet, car ils proposent un challenge sur le long terme et un format complètement libre pour réfléchir à ses propres solutions.

Première soutenance

Pour la première soutenance, je m'étais surtout occupé de la partie personnage (ses déplacements et ses animations). Ainsi que de la partie multijoueur.

J'avais également passé un certain temps à me familiariser avec les différentes mécaniques de base d'Unity et de Photon (pour le multijoueur).

Déplacement et Dash

Je m'étais tout d'abord occupé des déplacements des personnages afin que l'on puisse avoir le plus vite possible de quoi faire des tests. J'avais donc commencé par créer un prototype qui s'est avéré après de nombreuses évolutions être le script final. Notre personnage a ainsi pu rapidement être en mesure de sauter, marcher en avant, en arrière. Néanmoins à ce stade, le saut ne me satisfaisait pas, j'avais donc mis en place un petit algorithme permettant de rendre la chute du personnage plus réaliste et plus agréable au niveau du gameplay.

Une fois les déplacements terminés, je m'étais occupé d'une des parties clés d'un jeu de type platformer : les dashes. Le dash que possédera notre personnage dépendra de l'équipement sélectionné en début de partie.

Notre jeu en possède trois :

- Tout d'abord le dash de base, le plus connu, le dash "Classique" qui permet de se projeter dans huit directions : vers l'avant, l'arrière, en haut, en bas et dans les diagonales.

C'est un dash simple mais pratique permettant de s'en sortir dans quasiment toutes les situations.

Son fonctionnement est simple : lorsque le joueur appuie sur le bouton "Dash", le jeu va acquérir les touches sur lesquelles le joueur appuie à cet instant (une touche pour les dashes en ligne droite et deux pour ceux en diagonale) puis propulser le joueur dans la bonne direction sur un certain intervalle de temps.

Ce dash m'a servi de base pour les deux autres qui reprennent ce principe en y ajoutant chacun leur subtilité.

L'équipement symbolisant ce dash est un T-shirt rouge.

- Ensuite un dash plus original, le dash "Bouncy" qui ne permet que de se projeter vers l'avant ou vers l'arrière mais qui possède un avantage indéniable pour tout grimpeur en herbe : en cas de contact avec un mur à la fin du dash, le personnage sera projeté vers le haut, lui permettant d'atteindre des plates-formes difficilement accessibles et créant ainsi de nouveaux chemins empruntables seulement par lui.

Comme dit précédemment il a le même fonctionnement que le dash Classique mais en rajoutant une condition à la fin du dash qui vérifie si le personnage est contre un mur et le projeter vers le haut si c'est le cas.

Lors de la création de ce dash, j'avais eu quelques soucis notamment avec le type de collider, le polygone collider permettant d'arrondir la partie inférieure du personnage et ainsi éviter qu'il ne se bloque dans les légères différences de hauteur parfois présentes entre deux blocs. Lors de la mise en place de ce polygone collider, j'avais en effet créé les côtés gauche et droit légèrement en diagonale ce qui a eu pour effet de bloquer le personnage dans le mur lors de sa propulsion vers le haut lors du dash et plus généralement lors du saut de celui-ci contre un mur.

L'équipement symbolisant ce dash est un T-shirt vert.

- Enfin un dash plus compliqué à utiliser, le dash "Light" qui permet de phaser au travers de certains obstacles. Il permet de se projeter en avant dans trois directions : en haut, en arrière et en avant. Ce dash pourrait faire penser à une téléportation mais son fonctionnement est celui des dashes précédents : on acquiert la direction puis on propulse le personnage dans cette direction. La différence majeure était que le SpriteRenderer du joueur ainsi que son Collider ait été désactivé au début du dash puis seront réactivés à la fin, rendant ainsi le personnage à la fois invisible et intangible.

Pour ne pas ce téléporter dans un décor, j'avais utilisé la fonction `OnTriggerEnter2D` qui se déclenche à chaque collision avec un collider. À chaque collision, j'incrémentai une variable, initialement à 0 si à la fin du dash cette variable est impaire, le personnage est re-téléporté à l'emplacement initial, stocké dans une variable au début du dash.

Ce dash a été pour moi le plus compliqué à implémenter et j'ai dû essayer plusieurs solutions avant d'arriver à une qui me convienne.

J'avais notamment tout d'abord pensé à utiliser la fonction `RayCastAll` du module `Physics2D` qui testait sur une longueur donnée le nombre de collider rencontré et les places dans un tableau. Je pensais ainsi pouvoir compter les deux côtés du terrain. Malheureusement ces deux côtés ne comptaient que pour un car ils appartenaient au même collider et je me suis donc tourné vers une autre solution.

L'équipement symbolisant ce dash est un T-shirt violet.

Animation

La seconde partie de mon travail avait été d'animer les personnages. Comme nous avons décidé de faire nous-mêmes toutes les animations des personnages, nous avions séparé l'implémentation en deux parties. Pour la première soutenance j'ai ainsi utilisé les Tilesets d'Hugo pour implémenter l'animation de course du personnage, l'animation "Idle" du personnage (quand personnage ne le déplace pas) ainsi que l'animation de saut.

- L'animation de course se répète toute les 0.5 secondes avec 4 Sprites différents sur cette durée,

placés à distance variable afin d'obtenir un résultat satisfaisant.

- L'animation Idle se répète toutes les 1 secondes avec 6 Sprite différents sur cette durée, placé à distance variable
- L'animation de saut qui se divise en deux parties : une phase de montée avec 6 Sprite différent répartie sur 1 seconde puis de la phase de descente également composée de 6 Sprite différent répartie sur 1 seconde

L'animation "Idle" est celle qu'aura notre personnage par défaut : tant que la valeur absolue de la vitesse du personnage sur l'axe x est inférieure à 0.3 (mettre la valeur à 0 provoque un bug) et qu'il sera au sol, le personnage effectuera en boucle cette animation.

Si le personnage possède une vitesse supérieure à 0.3 et qu'il est au sol, alors c'est l'animation de course qui va se jouer en boucle.

Enfin, si le personnage n'est ni au sol ni contre un mur, il jouera l'animation de saut

Multijoueur

Pour la première soutenance, je m'étais également occupé d'implémenter le multijoueur. Pour cela, j'avais utilisé le Photon Engine permettant de créer des salons de jeu rejoignable par deux joueurs sur des machines différentes.

J'avais ainsi créé un GameObject vide qui va, lors de la connexion à la partie d'un joueur, lui assigner son point d'apparition : s'il est le premier à se connecter, il sera considéré comme le joueur un sinon, ce sera le joueur 2.

Nous avons décidé de placer les deux joueurs sur la même scène et sur deux fois la même salle, placée l'une au-dessus de l'autre. Mais bien que les deux joueurs soient sur la même scène, ils ne se verront jamais et ne pourront donc pas savoir qu'ils sont proches l'un de l'autre.

J'avais rencontré beaucoup de problèmes pendant l'implémentation notamment sur la gestion des deux caméras, une pour chacun des joueurs. En effet, j'ai tout d'abord créé deux caméras dans la scène et essayé d'en attribuer une à chacun des joueurs, mais cette solution ne marchait pas et les deux joueurs partageaient la même caméra ou alors les deux caméras étaient toujours inversées. Je me suis donc tourné vers une seconde solution : attacher la caméra directement au joueur en tant que "sous gameObject" cette solution a été beaucoup plus satisfaisante, mais il y avait néanmoins un problème, la caméra suivait le joueur dans tous ses déplacements. Or, nous voulions une caméra fixe, "posé" au centre du niveau.

Pour pallier ce problème, j'ai tout simplement ajouté un component "Rigidbody" à la caméra et je lui ai bloqué tous ses déplacements sur les axes x et y (z n'étant pas important ici, car nous travaillons en 2d)

Scénario

Enfin, je m'étais occupé de la partie scénario, qui était à ce moment assez léger, mais qui sera plus développé lors des soutenances suivantes.

Cette partie a été une des moins chronophages, mais reste intéressante, car est très différente des autres parties dont je me suis occupé qui étaient surtout centrées sur le code.

J'ai donc écrits une première introduction de l'histoire de notre jeu qui pourra être découverte au fur et à mesure de l'ascension du joueur dans la montagne.

L'histoire pourra ainsi être découverte dans des salles dédiées en interagissant avec des objets spécifiques (par exemple un tableau dans notre première salle histoire).

Seconde Soutenance

Pour la seconde soutenance, je devais m'occuper d'implémenter les animations de la classe *Classique* et de la classe *Bouncy*, la classe *Light* étant une téléportation n'avait pas besoin d'animation. J'avais également implémenté un item passif, le feu de camp portatif. J'avais créé un type de salle épreuve supplémentaire : la salle torche. Enfin, j'avais créé un système de stockage de données afin de conserver ses données entre les parties.

De plus, j'avais grandement amélioré le système de multijoueur, ajoutant une interface de création de salon ainsi que la synchronisation donnée entre les joueurs.

Les Difficultés et les résolutions de bug de la seconde soutenance

Lors de cette semaine, notre groupe a été confronté à un problème majeur qui était lié à la mise à jour du launcher Unity Hub. Quand j'ai lancé le projet, celui-ci s'est lancé comme projet 3D. Nous avions eu du mal au début à trouver l'origine du problème, après plusieurs heures de recherches et de tentatives à convertir notre projet pour le rendre compatible avec l'ancienne version d'Unity Hub, j'ai réussi à remplacer le projet 3D par une ancienne version de notre projet qui était encore présente sur l'ordinateur d'un de mes coéquipiers qui n'avait pas encore pull la version bug du projet.

Afin d'améliorer la qualité de jeu, j'avais dû apporter des modifications à la classe *Light* qui avait de nombreux problèmes lors de sa téléportation : en effet, il lui arrivait assez régulièrement de pouvoir traverser les murs extérieurs de la scène et ainsi tomber dans le vide, mais aussi d'atterrir à l'intérieur d'un obstacle dans la scène. Ces bugs ne se manifestaient que dans certains cas si bien que nous nous n'en sommes pas immédiatement rendu compte : il fallait être contre directement contre le mur pour provoquer ce bug. J'avais donc modifié le comportement de ce dash en ajoutant un nouveau colliderr, plus petit, qui ne s'active que durant le dash et en désactivant le collider principal. Cela avait ainsi permis de régler le de téléportation lorsque le joueur se tenait contre un mur.

Feu de camp portatif

Mon premier objectif pour la seconde soutenance a été d'implémenter l'item passif feu de camp portatif qui permet d'augmenter l'intervalle de temps auquel le stress augmente. L'intervalle de stress augmente de 0.25/0.5/0.75 selon la rareté de l'objet. Grâce au modèle des autres items, cette partie a été assez rapide à faire.

Dispositif anti-chute infinie (DACI)

Tout jeu de plateforme 2D à un système respawn digne de nom lorsque son personnage meurt, par exemple lorsque celui-ci tombe dans le vide. Si rien n'est fait, le malheureux chutera vers l'infini et au-delà. Pour empêcher cette tragédie, si le joueur a la malchance de tomber un gameobject que j'ai intitulé void sera là pour le rattraper. "Void" est une plateforme transparente qui recouvre toute la salle en longueur en dessous de celle-ci. Si le joueur entre en contact avec lui, il sera téléporté à son point de spawn positionné au début de la salle et le stress du joueur augmentera de 10.

Sauvegarde

Je m'étais ensuite occupé de créer un système de sauvegarde qui permet aux joueurs de sauvegarder leurs données telles que les pièces, les raspberries ainsi que les améliorations achetables à la fin d'une partie (ces dernières n'avaient pas encore été implémentées.) puis de charger ces données au début de la prochaine partie. Il est à noter que c'est la sauvegarde du joueur 1 qui sera prise en compte lors de la sauvegarde et de la récupération de données. De plus, le système ne sauvegardera les données qu'en fin de partie, à la mort du personnage ou à son arrivée à la fin du jeu : quitter la partie en cours de route laissera la sauvegarde tel qu'elle était au début de partie.

Salle torche

Par la suite, je m'étais occupé d'implémenter le système des salles torches, pour ce faire, je m'étais basé sur le système des capteurs. Pour que le joueur déverrouille sa porte, il doit activer 5 torches dans un ordre bien précis. Si une torche est allumée trop tôt, toutes les torches déjà allumées vont s'éteindre et les joueurs devront recommencer depuis le début. Pour résoudre ce puzzle, les deux joueurs devront donc coopérer, en effet, le premier joueur peut activer les torches tandis que le second à connaissance de l'ordre dans lequel il faut les allumer.

Le premier joueur ne peut pas activer une torche si ce n'est pas son tour et s'il se trompe toutes les torches vont s'éteindre et le joueur doit recommencer depuis le début. Pour allumer une torche, le joueur doit se tenir à côté d'une d'entre elle c'est-à-dire que le collider du joueur est en contact avec le collider de la torche qui reconnaît que c'est un joueur et il doit appuyer sur E. Lorsqu'une torche est activée le sprite de la torche éteinte change pour l'animation de la torche allumée.

Pour l'animation de la torche, j'ai utilisé le pack gratuit de [Torch Sprite Pack by Asymmetric](#)

Don't Destroy on load

De nombreux objets de notre jeu doivent être présents à tout moment quelle que soit la scène. C'est bien sur le cas du joueur et de tout ce qui l'entoure (sa caméra, son canvas) mais aussi pour l'audioManager gérant la musique, le canvas servant au menu pause ainsi qu'un gameManager. Pour cela, j'ai utilisé une fonction de Unity, *DontDestroyOnLoad* qui permet d'indiquer à Unity de ne pas détruire les objets affectés par cette fonction lors d'un changement de scène.

Amélioration du multijoueur

La plus grosse partie de travail pour la seconde soutenance avait été l'amélioration du multijoueur. Certes le multijoueur était déjà fonctionnel, mais il était améliorable et c'était dans ce but que je ne m'étais pas fixé énormément de travail pour cette soutenance.

Je m'étais suis tout d'abord occupé d'améliorer le menu précédemment créé par Hugo afin de pouvoir choisir le mode de jeu entre solo et multijoueur (le mode solo n'est pas encore fonctionnel.). Cliquer sur le bouton multijoueur dirige le joueur vers une interface où il pourra choisir son pseudonyme, créer un salon avec un nom spécifique et enfin voir la liste des salons déjà créer et les rejoindre. Une fois que le joueur a rejoint une salle, il aura accès sur la gauche à une interface permettant de quitter le salon et ainsi nous ramener à l'écran de sélection de salons. Il y a ensuite la liste des noms des joueurs présent dans la salle situé au centre de l'écran. Sur la droite de l'écran, se situent trois boutons permettant de choisir la classe voulue pendant la partie. Cliquer sur un des boutons désactivera automatiquement les deux autres ainsi que le bouton correspondant à cette classe chez l'autre joueur. En effet au pied de la montagne il n'y a qu'un seul équipement par type et les joueurs devrons donc se mettre d'accord pour choisir leur équipement.

De plus, le joueur 2 devra cliquer sur le bouton "N" pour signifier qu'il est prêt à lancer la partie. Une fois que chacun a choisi une classe différente et que le joueur 2 à cliqué sur le bouton prêt, le joueur 1 peut lancer la partie en cliquant sur "start game", instanciant un joueur pour chaque personne.

Une fois la partie lancée, il fallait encore gérer la synchronisation de certaines informations entre les deux joueurs, notamment le stress qui est partagé par les deux joueurs. Pour cela, j'ai utilisé la procédure RPC de photon permettant d'envoyer une série d'instruction aux autres joueurs présent sur la salle à exécuter. Cela permet notamment à ce que tous les joueurs du salon voient leur stress augmenter en cas de chute de l'un des joueurs par exemple.

Cette partie avait été, pour moi, une des plus compliquées à faire depuis le début du projet, car elle demandait des connaissances dans un sujet que je ne maîtrisais pas du tout et qui nécessitait des connaissances que je n'avais pas encore. J'ai donc du implémenter mes fonctions à l'aide de différents tutos sur Internet et en lisant la documentation présente sur le site de photon pour avancer pas à pas dans la compréhension de Photon.

Animation de dash

Enfin, je m'étais occupé d'ajouter aux classes *Classique* et *Bouncy* les animations manquantes, c'est-à-dire celles des dashes. Encore une fois, c'est Hugo qui s'est occupé de créer les tileset. La classe *Classique* possède ainsi sept animations de plus : une pour le dash avant/arrière, une pour les dash diagonaux vers le haut, une pour les dash diagonaux vers le bas, une pour le dash vers le

haut et une pour celui vers le bas. La classe *Bouncy* quant à elle possède une animation de plus : celle animant le dash vers l'avant/arrière.

Dernière Soutenance

Pour cette dernière soutenance, je me suis occupé d'implémenter le solo puis de résoudre tous les problèmes en découlant

Solo

La plus grosse partie de mon travail pour cette soutenance à implémenter le mode solo dans notre jeu qui n'était jusque-là que jouable en multi.

Pour cela, j'ai tout d'abord amélioré le menu principal, en ajoutant la possibilité de cliquer sur le bouton "Solo" lors de la sélection du mode de jeu qui emmène le joueur vers une nouvelle fenêtre possédant un bouton permettant de quitter cette fenêtre, un bouton pour lancer la partie et deux colonnes de trois boutons colorés (de haut en bas) en rouge, vert et violet permettant de choisir respectivement la classe *Classique*, la classe *Bouncy* ou la classe *Light*.

La première colonne permet de choisir la classe du premier personnage tandis que la seconde colonne permet de choisir la classe du second personnage. Les deux personnages ne peuvent bien évidemment pas avoir la même classe, il n'y a qu'un seul équipement de chaque disponible au pied de la montagne. Tant que les deux équipements n'ont pas été choisis appuyé sur le bouton "Start Game" n'aura aucun effet.

Une fois dans une partie, le but est le même qu'en mode multijoueur : il faut traverser les salles et avancer dans le jeu et aller le plus loin possible avant d'être trop stressé et de se faire toucher par un fantôme.

La différence étant que cette fois si on contrôle les deux personnages. Pour alterner entre les deux personnage, il suffit d'appuyer sur le bouton 'G'.

Jarre à pièces

Pour cette soutenance, j'ai également implémenté des jarres à pièces qui sont la seule source de revenue en pièces de notre jeu.

Ces jarres disposé à différents endroits de chaque salle donnerons aux joueurs un nombre de pièces aléatoire compris entre 0 et une valeur maximum qui sera de 3 au début mais qui pourra être augmentée grâce aux améliorations achetables en fin de partie. Une jarre pourrait donc donner entre 0 et 7 pièces quand l'amélioration est au maximum.

Son fonctionnement est simple : la jarre est équipée d'un BoxCollider2D en mode "IsTrigger" qui permet de détecter les collisions au lieu de les empêcher. Lors d'une collision, le script attaché à la jarre va lancer une animation qui va casser l'urne puis détruire le GameObject représentant le jar à la fin de l'animation. En même temps, les pièces vont être ajoutées sur le compte de tous les joueurs.

Dans le cas du multijoueur, j'ai du également faire en sorte que les cassages d'urnes ne soit pas compté quand c'est "l'instance" de l'autre joueur qui la touche, c'est-à-dire le personnage présent

sur la scène d'un joueur, mais contrôlé par l'autre joueur.



FIGURE 12 – Sprite de la jarre

Raspberries

Je me suis ensuite occupé d'améliorer le système de récupération de Raspberry précédemment fait par Antoine afin de le rendre fonctionnel en multijoueur et en solo.

Pour cela, j'ai utilisé exactement la même méthode que pour les jarres en changeant les noms de variables.

Resolution de bug

L'implémentation du mode solo a nécessité de nombreuses modifications :

- J'ai tout d'abord décidé de créer une version "solo" du scripte "PlayerMovement" qui gère tout ce qui touche au déplacement des personnages ainsi que du scripte "PlayerStress" qui gère l'augmentation du stress sous toutes les formes. Ces scriptes sont des versions dépourvus de fonction ayant trait au multijoueur.
- Tout les gameobject interagissant avec les joueurs ont ainsi eu besoin d'être réviser afin d'être utilisable à la fois en multijoueur, mais aussi en solo. Pour cela avant chaque action ayant besoin d'être différencier selon le mode de jeu, j'ai utilisé un booléen, le PhotonNetwork.isConnected qui vérifie si on est connecté en multijoueur.
- J'ai également utilisé de nombreux tags pour savoir différencier facilement les différents cas. Ainsi, en solo, le personnage 1 aura le tag Player1, le personnage 2 aura le tag Player2 tandis qu'en multijoueur le deux personnage auront le tag Player, de même les différents Gameobjet Inventaire possèdent des tag les différenciant selon leur appartenance et le mode de jeu. J'ai pour cela utilisé la fonction GameObject.FindGameObjectWithTag("") qui permet de récupérer tous les GameObjects de la scène ayant un tag spécifique.

J'ai ensuite résolu un certain nombre de bugs sur l'ensemble du projet. La plupart étaient causés par une méthode que nous utilisions pour récupérer certains scripts en créant une instance statique de ces derniers. Le problème étant que dans la plupart des cas ces scriptes était présent en plusieurs fois sur la même scène si bien que lorsque l'on y voulait y accéder, on tombait sur un de ces scriptes de façon aléatoire. Bien entendu dans la plupart des cas, cela causait un problème.

Pour contourner ce problème, j'ai ainsi utilisé les méthodes mentionnés ci-dessus (différencier les cas et utiliser le tag pour accéder aux différents gameObject).

J'ai ainsi modifié une grande partie des scriptes du projet afin de les rendre fonctionnels pour le multijoueur et le solo.

J'ai également réglé un problème qui permettait au joueur 2 de pouvoir accéder aux boutiques bien que le marchand ne soit pas présent sur la scène. Il y avait également un problème similaire pour ramasser des items actifs, des Raspberries, et casser les jarres. Ces problèmes étaient liés en majorité au multijoueur et était causé par le personnage du joueur 1 se déplaçant également dans le client du joueur 2 et activant le shop pour ce dernier. Pour résoudre ce bug, je vérifie à chaque collision si la photonView du joueur (variable qui indique à qui appartient le gamObject) appartient bien au joueur. Si c'est le cas on exécute les instructions (on ouvre le shop, on casse l'urne, on récupère la Raspberry, on ouvre le tableau). Il y avait également des problèmes sur

certaines salles, celles de type capteur et celles de type clé. En effet, ces salles ont besoins de savoir exactement quel joueur elle contiennent (pour pouvoir réinitialiser le dash en cas de collision avec un capteur et les capteurs en cas de chute pour la salle capteur et savoir qui la clé doit suivre et quel joueur doit toucher la porte pour l'ouvrir pour la salle clé)

Angelina

Présentation

Je m'appelle Angelina, seule représentation féminine de l'équipe et pendant la réalisation du cahier des charges, j'ai été assigné à plusieurs éléments à développer. À savoir l'immersion, comprenait le visuel, l'audio et l'intrigue du jeu ; l'espace de progression comprenait les salles puzzles avec tous les mécanismes pour les résoudre, les salles boutiques et les salles histoires où le joueur devra progresser ainsi que la génération des zones ; l'interface pour afficher des informations sur l'UI.

C'est le premier projet aussi organisé et aussi sérieux que j'ai fait avec également des coéquipiers investis qui répondent toujours à l'appel quand l'un de nous a des difficultés.

Première soutenance

Début du projet

Pendant la période où je devais coder, je me suis aidée des chaînes de tutoriels sur Unity de [tuto Unity FR](#) et de [Brackeys](#). Je me suis également documenté sur des forums lorsque je rencontrais des bugs.

Assets du projet

Ma première ambition était de trouver tous les assets que nous allions utiliser dans le jeu. La recherche était une des parties les plus longues au cours du début de ce projet. En effet, il fallait que je trouve des assets gratuit libre de droit qui correspondait à l'esprit du jeu et qui convenaient à toute l'équipe. J'ai débuté par une grande sélection d'assets qui connurent plusieurs tries pour, au final, être présentés aux teammates. Parfois, il fallait refaire des recherches après le début de la construction du jeu, car ils ne convenaient plus à certain teammates ou on se rendait compte qu'ils rendaient moins bien in game. Nous voulions à tout prix avoir un asset de paysage enneigé, cependant aucun des assets gratuit ne nous convenaient. Nous avons donc acheté un pack à 2 euros d'un asset de neige qui rentrait dans l'esprit du jeu. Les assets sélectionnés sont :

Zone 1, [Kauzz Forest by Kauzz](#)

Zone 2, [Frozen Forest by Quintino Pixels](#)

Zone 3, [Cave Tileset by GrafxKid](#)

Zone 4, [StringStar by Trixie](#)

Items [Simple Potions by Armisius](#) , [RPG Assets by Ssugmi](#) , [Assorted Icons by Quintino Pixels](#) , [DailyDoodles Variations by RavenTale](#) , [Free Pixel Food by Henry Software](#) , [Tarot Major Arcana by Alex Vágandr](#) , [Kyrise's free RPG Icon Pack by Kyrise](#) , [Free Pixel Art Skill Icons by Quintino Pixels](#)

Décors [Tree by Barf Vader](#) , [Overgrowth by Ruuskii](#) , [Space by VectorPixelStar](#) , [Semi-Realistic Cloud by LateNighCoffe](#) , [Free Pixel Plants by Danaida](#)

Animation [2D Pixel Heart by gpway](#) , [Firework Light by NYKNCK](#) , [Animated by Stealthix](#) , [Magical Animation Effects by pimen](#) , [Battle Effects by pimen](#) , [Pixelated Attack/Hit Animations by pimen](#) , [Flame Animated by Lelex Game](#) , [Campfire Pixel Art by LadySachment](#) , [Fire Column pixel art effect by sanctumpixel](#) , [Free Tileset Objects - Breakable Pots by Seliel the Shaper](#) , [Free Pixel Effects art effect by XYEzawr](#)

Musique

Pour le choix des musiques, j'ai procédé comme pour la recherche des assets. J'ai débuté par des recherches sur différents sites proposant des musiques libre de droit, j'ai fait une sélection d'une trentaine de musiques qui auraient pu convenir au jeu, puis j'ai fait un tri parmi toutes ces musiques. Le jeu comportera au total 7 musiques différentes, une pour chaque zone, une pour les salles histoires, une pour les salles défis et une pour le menu.

Pour la zone 1, nous utilisons. [Inspiration by BoxCat Games](#)

Pour la zone 2 nous utilisons [Arpanauts by Eric Skiff](#)

Pour la zone 3, [8 Bits Surf by David Renda](#)

Pour la zone 4, [The Final Road by Visager](#)

Pour les salles défis, [CPU Talk by BoxCat Games](#)

Pour les salles histoires, [Passing Times by BoxCat Games](#)

Pour le menu, [Ending by Komiku](#)

Actuellement, les salles test, key, button, jump, shop auront la musique de la zone 1 qui s'activera dès le chargement de la salle et se jouera en boucle. La salle histoire aura la musique des salles histoires et la menu aura la musique menu.

Implémentation de six salles différentes et changement de salle

A partir de l'asset de la zone 1, j'ai pu construire les visuels des différentes salles. Chaque salle à plusieurs tilemap : fondation qui est une tilemap qui a rigidbody, ce calque contient toutes les plateformes sur lesquelles le joueur peut marcher. Props pour tous les éléments décoratifs comme les plantes, les petits cailloux ou les murs d'une maison. Si besoin un calque feuillage pour que s'il y a un arbre avec une plateforme dans ses branches, le joueur passe derrière le feuillage. Si il y a des maisons, il faut un calque supplémentaire pour ajouter des portes, des fenêtres et des cheminées par-dessus les murs. Le background de la zone 1 est une superposition de 3 images de forêt de différentes couleurs pour donner un effet de profondeur. J'ai également anticipé et ajouté des plateformes pour les futurs monnaies ramassables pour quand elles seront implémentées. J'ai du recommencer plusieurs fois à refaire les salles pour diverses raisons, les premières versions étaient trop petites et il n'y avait pas suffisamment d'espace pour ajouter un nombre satisfaisant de plateformes. La particularité de l'asset de la zone 1 est qu'il y a un plafond. J'ai du refaire les murs qui rendent visible la délimitation de la salle et les plateformes car au départ elles avaient l'air trop "cubique" comme des legos superposés l'une sur l'autre.

Toutes les salles ont été pensées pour être réussies sans l'aide d'aucun items. Cependant, certaines salles sont plus difficiles que d'autre dépendant du type de dash du joueur. Le dash classique est le plus adapté pour réussir aisément toutes les salles contrairement aux dash light qui demandera plus de technique et de temps pour recommencer les jumps. Le bouncy dash est à mon avis le plus intéressant et amusant à jouer car la dynamique de rebond permet d'emprunter pleins de chemins différents et enrichis les possibilités de gameplay.

Par la suite, j'ai pu écrire les différents scripts reliés à chaque type de défi de chaque salle (le script de la key, du button et de leur porte respective). J'ai implémenté par la même occasion un cheatcode pour chaque salle où il y a une porte, nous pouvons l'ouvrir en appuyant sur F.

- La salle jump 1.1. C'est la première salle (après le shop amélioration) dans laquelle on apparaît lorsque on appuie sur "Play" dans le menu. C'est également la première salle où la plupart des mécanismes des puzzles ont été testés en premier. Celle-ci est une salle jump en plus facile du à une présence plus importante de plateforme et la distance réduite entre elles. C'est le début du jeu, il faut laisser le joueur découvrir son personnage et les différentes caractéristiques de celui-ci. Comme dit précédemment le vide est en réalité un sol invisible, j'ai ajouté une plate-forme en dehors de la camera pour empêcher le joueur de tomber. Les trous seront bouchés jusqu'à l'implémentation du spawnpoint au début du niveau.



FIGURE 13 – Salle Jump 1.1

— La salle jump 1.2.

C'est la salle la plus basique que j'ai fais, celle-ci comporte des plateformes et des éléments décoratifs.

— La salle key

Pour passer à la prochaine salle, le joueur doit récupérer une key et la ramener jusqu'à la porte. Dans la première zone qui est le contrefort de la montagne, les portes seront représentées par des branches. Lorsque la porte est ouverte, il reste des traces des branches de la même hauteur que l'herbe comme ci, ils avaient été coupés. La porte a deux collider, un qui a un rigidbody pour empêcher le joueur de la traverser et le deuxième qui est "IsTrigger" pour détecter la présence du joueur à proximité. Au chargement de la salle, la key sera positionnée à un point fixe dans la salle. Lorsque le joueur entrera en contact avec la key, celle-ci aura pour nouvelle target le PointFollowKey qui sera légèrement à l'arrière du joueur et le suivra partout avec une animation de flottaison allant du haut vers le bas. Lorsque le joueur aura la key en sa possession, un booléen indiquant qu'il a la key passera sur true et permettra d'ouvrir la porte. Lorsque le joueur s'approchera de la porte avec la key en sa possession en entrera dans le collider de "IsTrigger" de la porte, la key se déplacera au-dessus de la porte pour la faire disparaître. Le joueur peut ainsi aller à la prochaine salle. Autre particularité de la salle, l'ajout de nouveau type de plateformes, à savoir le champignon et les planches dans les feuillages de l'arbre pour sortir de la monotonie des plateformes cubiques.

— La salle button



FIGURE 14 – Salle key mécanisme de la key

C'est également une salle où il y a un mécanisme pour ouvrir la porte. Cette fois-ci, l'ouverture se fait à l'aide d'un button, qui, désactivé est noir et activé rouge. Lorsque le joueur entre en contact avec le bouton noir, c'est à dire lorsque le collider du joueur entre dans le collider "IsTrigger" du bouton, celui-ci change de sprite et devient rouge pour montrer qu'il est activé. À cet instant un chronomètre de 7 secondes se lance pendant lequel la porte du niveau est déverrouillée. À la fin de ce temps, le bouton redevient noir et la porte se referme. Le joueur peut donc recommencer à activer le bouton pour réouvrir la porte. Il n'y a pas de limite d'activation du bouton.

— La salle boutique.

Présence d'un marchand interactif, explicité dans la prochaine subsection. Petite particularité de la salle, pour sortir des salles monotones qui manquent d'originalité avec uniquement des plateformes cubiques de terre, j'ai ajouté les toits des maisons en tant que plateforme. Ainsi, le joueur passe devant les maisons, car ce sont des décors, mais il peut marcher sur les toits lui permettant d'atteindre les plateformes supérieures.



FIGURE 15 – Salle button1.1 et Salle boutique

— La salle histoire. C'est une salle avec une musique différente des précédentes, plus calme et mystérieuse. Il y a dans cette salle présence du premier tableau avec les premières lignes du scénario. Le tableau et le système interactif sont explicités dans la prochaine subsection.



FIGURE 16 – Salle Histoire

Toutes les salles ont un index défini, devant la sortie de chaque salle, il y a présence d'une fleur jaune. Lorsque le joueur entre en contact avec celle-ci, cela signifiera que la salle est terminée et il y aura le chargement de la salle suivante, c'est-à-dire l'index de la salle actuelle +1. Lorsque c'est une salle défi avec une porte, la fleur est située à l'arrière de 1 bloc de la porte pour empêcher le changement de salle. La salle qui sera chargée sera la salle ayant l'index suivant de celui de la salle actuelle. Pour parcourir toutes les salles du jeu, nous pouvons lancer le main menu qui a pour index 0 et appuyer sur game qui lancera la salle d'index 1 qui est la salle test. J'ai choisi cette fleur jaune, car elle correspondait au code couleur de l'asset de la première zone, il y aura donc une fleur différente à chaque zone. Actuellement, les salles se suivent, mais je prévois de faire un système de chargement de salle random avec les index des salles qui n'ont pas encore été rencontrés.

Implémentation d'un PNJ et d'un tableau interactif

Dans la salle histoire, j'ai dû implémenter un premier PNJ, ou plutôt un objet interactif se comportant comme un PNJ, qui a une apparence d'une peinture. Lorsque le joueur se trouve à proximité de celui-ci, c'est-à-dire lorsque le collider du joueur entre dans le collider de la peinture, un message apparaît sur l'UI du joueur disant "PRESS E TO INTERACT". Si le joueur appuie sur E, une fenêtre de dialogue apparaît avec une animation de glissade du bas vers le haut s'arrêtant au milieu de l'écran contenant trois informations : le nom du PNJ, le dialogue et en bas un bouton "...>" pour passer au dialogue suivant. Le dialogue s'affiche de manière progressive, pour avoir cet effet visuel, chaque lettre est écrite avec un écart de 0.05 secondes. À la fin de la dernière phrase la fenêtre de dialogue repart vers le bas. Si le joueur le souhaite, il peut à nouveau relancer le dialogue. Le premier tableau du scénario est une nymphe accrochée à un arbre, je lui ai ajouté le dialogue redigé par Mathéo. Le tableau permet d'annoncer au joueur le choix qu'il devra faire ensuite, car il devra choisir où continuer son aventure entre la zone 2 et la zone 3 qui sont respectivement "sentiers enneigés" et "cœur de la montagne".



FIGURE 17 – UI du joueur affichant "press E to interact" et dialogue du tableau

Le PNJ marchand a le même fonctionnement que le tableau pour l'interaction et l'apparition de la fenêtre, lorsque le joueur est à proximité il y a également le message sur l'UI "PRESS E TO INTERACT". Contrairement au tableau, le marchand n'est pas statique. Je me suis également occupée de son animation. Pour implémenter l'animation de l'idle j'ai utilisé la tileset dessiné par Hugo, animation qui se répète toutes les secondes. Je me suis occupée de faire les bases du marchand pour le donner ensuite à Antoine pour qu'il implémente la boutique.

Seconde soutenance

Début du deuxième rendu

Pour cette soutenance, je devais m'occuper d'implémenter la génération aléatoire de salles, le système des salles puzzle capteurs. Enfin, finir entièrement les bibliothèques des deux premières zones. Je souhaitais débuter par la création du système de capteur pour par la suite le donner suffisamment tôt à Mathéo pour que cela lui serve de base pour la création du système de torche.

Correction de bugs

Avant d'attaquer ce travail, j'ai commencé par améliorer certaines mécaniques déjà réalisées pour la première soutenance. À présent, lorsque le joueur aura la clé en sa possession, celle-ci

Menu démarrage	ZONE 1	ZONE 2	ZONE 3	ZONE 4
0	total 12	total 12	total 12	total 12
	1 jump1.1	15 jump2.1	30 jump3.1	45 jump4.1
	2 jump1.2	16 jump2.2	31 jump3.2	46 jump4.2
	3 key1.1	17 key2.1	32 key3.1	47 key4.1
	4 key1.2	18 key2.2	33 key3.2	48 key4.2
	5 captor1.1	19 captor2.1	34 captor3.1	49 captor4.1
	6 captor1.2	20 captor2.2	35 captor3.2	50 captor4.2
	7 torch1.1	21 torch2.1	36 torch3.1	51 torch4.1
	8 torch1.2	22 torch2.2	37 torch3.2	52 torch4.2
	9 button1.1	23 button2.1	38 button3.1	53 button4.1
	10 button1.2	24 button2.2	39 button3.2	54 button4.2
	11 shop1	25 shop2	40 shop3	55 shop4
	12 campfire1	26 campfire2	41 campfire3	56 campfire4
		27 history2	42 history5	57 history8
	13 history1	28 history3	43 history6	58 history9
	14 room_choice	29 history4	44 history7	59 history10
				60 last_scene
				61 defi1
				62 defi2
				63 defi3
				64 defi4
				65 crédit

FIGURE 18 – Tableau récapitulatif des index des scènes du jeu

verra son box collider détruit pour éviter d'avoir des bugs tel que le joueur pouvant marcher sur la clé. Pour ce qui est des salles boutons, le bouton aura dorénavant une seconde supplémentaire passant de 7 secondes à 8 car certaines salles étaient vraiment impossible à faire et il aurait fallu rendre la salle très facile à finir ce qui enlèverait du challenge au joueur. Pour ce qui est du dialogue, il est désormais possible de parler qu'une seule et unique fois au marchand. Auparavant, si les items vendus ne convenaient pas, le joueur pouvant ouvrir et fermer la boutique sans limite jusqu'à ce que la génération aléatoire d'item lui donne l'item qu'il souhaitait. Correction également d'un bug avec le dialogue des tableaux des salles histoires, avant, une fois le dialogue lancé, il restait sur l'écran du joueur tant qu'il ne finissait pas la lecture de tout le dialogue. Maintenant, si le joueur s'éloigne du tableau, le dialogue se fermera automatiquement et le joueur devra reprendre la lecture du début s'il reparle au tableau.

Système capteur

Pour le système capteur, pour que le joueur déverrouille la porte, il doit activer 6 capteurs. Les capteurs fonctionnent ainsi : de base, ils apparaissent bleu clair, lorsque le joueur touche un capteur, c'est-à-dire lorsque le collider du joueur entre en collision avec le collider du capteur, le capteur devient violet et le dash du joueur est réinitialisé. Lorsque tous les capteurs sont activés, la porte est déverrouillée. Je prévois pour la prochaine soutenance pour rendre ce système jouable en coopération, les capteurs du premier joueur ouvriront la porte du deuxième joueur et vice-versa. (Illustration des capteurs activés et désactivés plus bas.)

Génération aléatoire

Pour la génération aléatoire, j'ai réfléchi à un système utilisant des listes comportant tous les index de scènes s'actualisant tout au long du jeu. Toutes les scènes auront un index bien défini dans les build settings qui ne changeront pas.

Actuellement, il y aurait 65 scènes de prévu. Lorsque le joueur lancera le jeu, il sera sur la scène 0, c'est-à-dire la main menu et chaque fois que le joueur ira dans une nouvelle zone il sera téléporté sur la scène "jump x.1". Donc, lorsque le joueur lancera une partie tout au début, il commencera son aventure sur la scène "jump1.1" soit l'index 1 dans les builds settings. Le système de changement de salle est comme précédemment une fleur avec laquelle il faut rentrer en contact. Chaque fleur aura le script "Don't destroy on load", cela implique que toutes les fleurs seront initialement présentes sur les scènes "jump x.1" et seront téléporté sur chaque scène suivante générée aléatoirement. Pour ce qui est du multijoueur, la fleur du premier joueur comporte un booléen qui est sur false de base et devient true lorsque le deuxième joueur a atteint sa fleur. Lorsque ce booléen est vrai et que le premier joueur atteint sa propre fleur, le changement de salle aura lieu et évidemment le booléen repassera à false. Chaque zone aura sa propre fleur qui correspondra au paysage de la zone (fleur de forêt, fleur de neige, fleur de grotte) et chaque fleur aura un script avec deux listes, une liste avec les index des salles puzzles et une autre avec les index des salles histoires.



FIGURE 19 – Fleurs des zones 1 2 3 et 4.

Pour chaque zone, le joueur pourra parcourir entre 10 et le total des salles, c'est un changement par rapport à ce que j'avais dit dans le rendu de la première soutenance, car 8 salles faisaient trop peu à mon goût. Chaque fois qu'il est possible, les salles histoires, feu de camp et le shop auront 1 chance sur 5 de sortir. Pour les trois premières salles, uniquement des salles histoires ou puzzles peuvent sortir. Au-delà de ces trois-là, il sera possible d'avoir feu de camp ou shop. Si le joueur n'a pas eu de feu de camp au bout de 7 salles, celle-ci sortira automatiquement et si le joueur n'a pas eu de shop au bout de 8 salles elle sortira obligatoirement. Il n'y a pas d'obligation sur les sorties des salles histoires, chaque aventure sera unique et si le joueur souhaiterait découvrir l'histoire dans son intégralité, il devra recommencer le jeu. Il n'y a pas d'obligation sur les sorties des salles histoires, chaque aventure sera unique et si le joueur souhaiterait découvrir l'histoire dans son intégralité, il devra recommencer le jeu. Pour la zone suivante, la scène chargée en première sera toujours la "jump x.1". Chaque fois qu'une salle est chargée, son index est retiré de la liste et donc le joueur ne pourra jamais avoir deux fois la même salle.

Petite particularité de la zone 1, celle-ci ne comporte qu'une seule salle histoire, dans cette salle le tableau annonce au joueur que pour la suite de son aventure, il devra choisir entre la zone 2 et la zone 3, c'est-à-dire qu'au total le joueur ne va parcourir que 3 zones et s'il voudrait toutes les faire, il devra recommencer le jeu. La salle qui sera après la première salle histoire sera la "room choice" et celle-ci n'aura pas une sortie mais bien deux, une allant dans la zone 2 et l'autre allant dans la zone 3. (visuel de la salle plus bas.)

Musique

Modification de l'audio manager, auparavant, il y avait un audio manager sur chaque scène avec une seule musique. Maintenant il n'y a qu'un seul audio manager qui sera généré sur la toute première scène du jeu c'est-à-dire notre bon main menu avec le script "Don't Destroy On Load" qui comportera une playlist de toutes les musiques du jeu. Avant, chaque fois que le joueur changeait

de salle la musique recommençait. A présent, la musique sera continue et le script de l'audio manager testera juste l'index de la salle. Si l'index correspond à une salle histoire la musique joué sera celle des salles histoires, si c'est une salle défi ce sera la musique des salles défis et sinon il testera si l'index est compris dans les intervalles [1,12] ou [15,26] ou [30,41] ou [45,56]. Selon l'intervalle dans lequel l'index est compris, la musique de la zone correspondante sera jouée.

Création de toutes les salles des zones 1 et 2

Mon objectif majeur de cette semaine était de créer toutes les salles inférieures à l'index 30 si on se réfère au tableau des index des salles. Les salles qui étaient déjà créées étaient : "Main menu", "jump 1.1", "jump 1.2", "key 1.1", "button 1.1", "shop1" et "history 1".

Ici la salle "captor 1.1" avec des capteurs désactivés en bleu et des capteurs activés en violet.



FIGURE 20 – Salle Captor1.1

Ici la "room choice" qui est après la salle "history1" qui mène soit à la zone 2 "sentiers enneigés" la sortie du haut, soit la zone 3 "cœur de la montagne" la sortie du bas.



FIGURE 21 – Room Choice

Viennent ensuite toutes les autres salles que j'ai du faire. Pour la deuxième zone, j'ai utilisé un asset gratuit supplémentaire [Kauzz Winter Forest by Kauzz](#) qui vient du même créateur que l'asset de la zone 1. J'ai utilisé certaine décos telle que la maison ou bien encore le bonhomme de neige pour compléter avec l'asset de la zone 2. J'ai également réalisé un mélange entre les deux backgrounds de l'asset qui était prévu de base et de celui-ci. J'ai utilisé trois layers de sapin du nouvel asset et je l'ai rajouté sur le background originel du ciel violet.

Pour créer des salles jouables en multijoueur, une fois que j'avais fini de faire la salle du joueur 1 positionnée en coordonnées $x = 0$ $y = 0$, je créais une deuxième salle identique à la première sur la même scène, mais de coordonnées $x = 0$ et $y = 40$.

Toutes les salles de la zone 1 et 2 sont en annexes.

Montage vidéo

Pour réaliser la vidéo de la soutenance, nous avons dû recourir à de nouveaux logiciels. Pour enregistrer l'écran et le gameplay, j'ai utilisé le logiciel gratuit OBS Studio. Son utilisation était assez instinctive évitant des soucis et des difficultés incongrues.

Pour le tournage, nous étions en équipe en vocal. Je filmais tout en partageant mon écran en jouant avec un coéquipier tandis le reste du groupe étaient devenu des directeurs de production. Suite à cela, j'ai pu m'attaquer aux montages, armée de toutes les runs qu'on a pu filmer et d'un script que nous avons rédigé tous ensemble. Pour faciliter le montage, nous avons délimité chaque partie présentée en un nombre précis de seconde. Pour le montage, j'ai pu découvrir le logiciel de ShotCut qui est également un logiciel disponible gratuitement. J'ai avancé dans cette aventure soutenue par des vidéos tutos sur youtube. Enfin, j'ai pu transmettre la vidéo montée à mon groupe pour qu'il puisse s'enregistrer avec les bonnes images sous les yeux pour avoir le rendu le plus propre possible. Pour les introductions personnelles, nous nous sommes mis d'accord pour que chacun filme la sienne de son côté et qu'elle soit ajouté à la fin au montage au début de la vidéo.

Dernière soutenance

Petit rappel du cahier des charges pour cette soutenance, je devais implémenter les salles défis (salles tic-tac pour les intimes) qui devraient être les salles les plus dures du jeu, il doit y en avoir 4, une par zone. Ensuite, finir les bibliothèques des zones 3 et 4, corriger les éventuels bugs rencontrés. Développer la partie histoire du jeu en ajoutant tous les tableaux et en créant la toute dernière scène du jeu. Pour finir, je me mettrai au travail qui concerne l'amélioration du personnage.

Implémentation des salles défis

Pour les salles défis, j'ai commencé par juste faire des tests sur deux blocs qui étaient sur deux grid "fondation" différents. Lorsqu'un bloc était actif, l'autre devait être désactif. Après un certain nombre de tests j'en suis arrivé à la conclusion : créer un gameobject "fondation controler" qui a pour objectif d'activer et désactiver les deux grid de fondation chacun leur tour. Lorsque qu'un grid fondation est actif, il est visible et à un collider, ce qui veut dire que le joueur peut marcher dessus. Lorsqu'il est désactif le grid devient invisible et son collider est désactivé. Pour synchroniser le temps d'apparition des plateformes avec la musique, le fondation controler à un script qui utilise un delta time et laisse un grid actif pendant 3 secondes avant de le désactiver et d'activer le second pendant 3 secondes et ainsi de suite.

Pour donner un rendu visuel plus esthétique et plus simple à retenir sur court terme les plateformes des deux grids qui s'activent ont des couleurs différentes, toujours en accord avec la zone dans lequel le joueur évolue.

Les salles défis sont donc plus difficiles à réaliser, car le joueur est obligé de se déplacer en permanence, car les plateformes en dessous finissent par disparaître et il est obligé de retenir

la position des plateformes de couleur opposée pour savoir vers où se diriger pour quand elles réapparaissent.

Au vu de leur difficulté, les salles défis apparaîtront à la fin de chaque zone, juste avant de changer de zone. Particularité de la première zone, puisque le joueur doit faire un choix entre la zone 2 et la zone 3, la zone défi de la première zone apparaît juste après la première salle histoire avant la "choice room".



FIGURE 22 – Salle défi de la zone 2

Création de toutes les salles des zones 3 et 4

Pour la complétion des deux dernières zones référencés dans le tableau des index de toutes les salles (voir soutenance 2) j'ai utilisé les deux assets que j'avais sélectionnés pour la première soutenance en ayant augmentés la difficulté générale des salles. Ainsi, les deux zones ont chacune deux salles key, deux salles button, deux salles torch, deux salles captor, une salle shop, une salle feu de camp, trois salles histoires et deux salles jumps. Les salles jumps "x.1" sont comme pour les précédentes zones les premières que le joueur rencontre dans son aventure. Pour ce qui est de la partie musique de chaque zone je n'ai pas eu à m'en occuper car je l'avais déjà fait pour la seconde soutenance, dès lors où j'ajoutais les salles dans le build tout se faisait automatiquement à partir de leur index. Les musiques jouées pour chacune des zones sont celles qui avaient été sélectionnées pour la première soutenance.

La totalité des salles est trouvable en annexe.

Approfondissement du lore

Aspect négligé jusqu'à présent, l'histoire du jeu. Pour ce travail, je me suis alliée à Hugo. Je devais m'occuper de la réalisation de toutes les salles histoires et de l'implémentation de tous les PNJ représentés par des tableaux dans chacune de celle-ci. Pour rendre la chose plus réaliste, le joueur lira des notes accrochées au tableau. Hugo, quant à lui, devait s'occuper de toute la partie rédaction de tous les dialogues.

À la fin de jeu, peu importe le chemin parcouru par le joueur, malgré la neige ou l'humidité des cavernes, le joueur arrivera sur le sommet de la montagne. Rappelons le nom du groupe, Hanabi, feu d'artifice en japonais. Dès le début du projet, nous voulions à la fin du jeu un spectacle de feu d'artifice pour célébrer la fin du parcours tumultueux du joueur. Pour ce faire, j'ai animé 9 feux d'artifices allant du très pixelisé, au coloré ou à la petite explosion au presque réaliste pour avoir

une diversité importante dans le ciel. Tout en haut de la montagne se trouvera accroché sur un arbre le dernier tableau du jeu qui mettra fin à l'histoire et à l'aventure du jeu.



FIGURE 23 – Last Scene

Amélioration du personnage

Pour ma partie amélioration du personnage, je me suis occupé de plusieurs choses. J'ai débuté par la réalisation de la salle du PNJ marchand des items permanents. C'est la salle qui doit apparaître lorsque le joueur perd et doit recommencer sa partie, dès lors, il faut comprendre que le joueur est redescendu de la montagne. Pour la réalisation de la salle, j'ai cherché à représenter une fin de ville début de forêt avec moins de végétations et plus de civilisations. La sortie de la salle ne se trouverait pas en hauteur, car le joueur ne commencerait pas encore à gravir la montagne. Une fois faite, j'ai légué ce petit village à Antoine qui devait s'occuper de l'implémentation du PNJ.

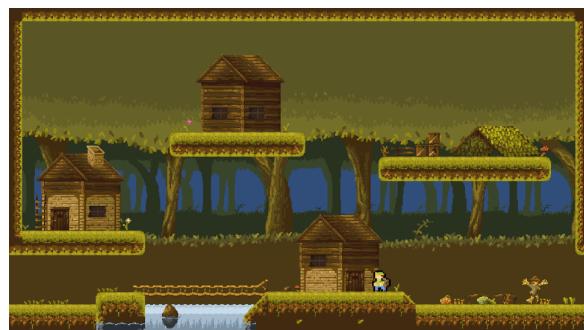


FIGURE 24 – Salle shop avec le marchand des améliorations

Suite à cela j'ai dû m'occuper d'ajouter dans la totalité du jeu les raspberries ramassables et les jars cassables. Pour les jars, chaque salle en contient entre 2 et 3, de plus, les jars ne sont pas disposés de la même façon entre la salle du joueur 1 et la salle du joueur 2. Pour ce qui est des raspberries, petite baie beaucoup plus rare que des jars, il n'y a que la salle du joueur 1 ou du joueur 2 qui peut avoir la chance d'en avoir une. De plus, il n'y a qu'une raspberry par type de salle dans chaque zone. C'est-à-dire par exemple, il n'y aura qu'une raspberry entre la salle "key 1.1" et "key 1.2". Les salles histoires et feu de camp ne contiennent aucun des deux items ramassables, les salles shop uniquement quelques jars et les salles défis une petite raspberry.

Scène de crédit

Tout bon jeu qui se respecte se doit d'avoir une scène de crédit, pour se faire, j'ai créé une salle qui se lancera lorsque le joueur descendra du sommet de la montagne de la "last scene". Pour faire les crédits défiler, j'ai fait une animation d'un texte qui contient tous les assets et les musiques que j'ai utilisé avec leurs auteurs ainsi que les chaînes youtube qui ont beaucoup aidé notre groupe. L'animation fait défiler le texte vers le haut jusqu'à la fin de celui-ci. À la fin des "special thanks" le joueur sera redirigé vers le main menu mettant fin à cette belle histoire.



FIGURE 25 – Credit

La scène de mort a un fonctionnement semblable à la scène de crédit, un message indiquant "Votre stress a eu raison de vous" apparaît lorsque le joueur est touché par le fantôme puis celui-ci est redirigé vers le main menu.

Cheatcode

Ajout de nouveau cheatcode pour le bien de tous sur le pavé numérique lorsque le joueur aura chargé une partie. La touche 2 pour se téléporté à la zone 2, la touche 3 pour se téléporté à la zone 3, la touche 4 pour se téléporté à la zone 4, la touche 5 pour se téléporté à la salle défi de la zone 4 et la touche 6 pour se téléporté à la dernière scène du jeu. Chaque téléportation peut être faite uniquement une seule fois.

Antoine

Présentation

Je me présente, je m'appelle Antoine RIQUET et dans ce projet, je me suis, tout comme Mathéo, principalement occupé de la partie programmation de celui-ci. J'ai eu pour tâches principales d'implémenter le système monétaire du jeu, les différentes boutiques (d'améliorations ou d'objets), le système d'inventaire, et enfin, l'intelligence artificielle de notre jeu. Ce fut ma première expérience de projet informatique, mais également de travail conséquent en groupe sur plusieurs mois, et je vais donc vous présenter le travail que j'ai pris plaisir, et qui était très intéressant à réaliser durant cette période.

Première soutenance

Début du projet

Mon rôle dans cette première soutenance avait été d'implémenter une grande majorité du système économique, les objets et le stress du jeu (la majorité de ces notions avaient été réalisées avec la collaboration d'Hugo.) Pour cela, je m'étais appuyé sur des tutoriels et certaines documentations internet pour avoir des notions sur le travail que je devais faire. Ce début de projet fut assez difficile pour ma part, car n'ayant jamais fait de jeu vidéo (comme la plupart de la promotion, j'imagine), j'avais découvert une « programmation nouvelle » avec des notions différentes à connaître et à maîtriser. De plus, Unity étant également tout nouveau pour moi, apprendre ce logiciel avait été laborieux au début, mais peu à peu, j'avais appris à maîtrisé les notions basiques du logiciel.



FIGURE 26 – Différentes Monnaies

Mon premier travail dans la soutenance avait été de créer un système monétaire dans notre jeu. Nous avions donc décidé de créer deux monnaies différentes, le "coin" et la "raspeberries". Pour ces deux monnaies, il avait également fallu implémenter un système de porte-monnaie, qui se rapporte tout simplement au nombre d'éléments que nous possédons parmi ces deux catégories. Concernant l'obtention de ces monnaies, elles peuvent être ramassées dans le jeu grâce à leurs deux instances qui leur sont associées.



FIGURE 27 – Interface du joueur pour les monnaies

Concernant la dépense de ces deux monnaies, je m'étais occupé de la création d'un menu marchand pour un PNJ. Pour cela, j'avais récupéré le PNJ créé par Angelina et je lui avais ajouté un menu permettant de vendre des items (la monnaie utilisée pour la vente de ces items étant le coin).



FIGURE 28 – Marchand

Pour cela, à chaque fois que nous allons interagir avec le PNJ, celui-ci va piocher dans sa liste d'items qu'il a à sa disposition puis va créer des instances de boutons cliquables afin que l'on puisse acheter ces différents items. En tout et pour tout, chaque marchand ne pourra vendre que 3 items qu'il choisit de manière aléatoire.



FIGURE 29 – Interface de la vente d'item (Exemple 1)



FIGURE 30 – Interface de la vente d'item (Exemple 2)

Concernant ces trois items, j'avais créé un système très basique, qui permet d'avoir plus de chance d'avoir des objets cassés que des objets usés ou neufs. Pour cela, j'avais simplement mis 3 items breaks dans la boutique, 2 items usés, et seulement 1 item neuf. Il nous sera donc possible d'acheter plusieurs fois le même objet, et si nous n'avons pas de chance, d'avoir deux fois le même objet dans la boutique du marchand. Il sera donc plus compliqué pour le joueur, d'acquérir des objets neufs qui seront plus rares, mais également plus chers.

En effet, à ces 3 catégories d'objets, correspondent 3 prix différents, concernant l'objet cassé, le prix correspondant est 40 coins, pour l'objet usé, il est de 70 coins, et pour l'objet neuf, il est de 100 coins. Cependant, les prix de ces objets ne sont pas fixes. Il est possible, pour le joueur, d'acheter un « guide de l'épicier » que j'avais implémenté, et qui réduira, en fonction de son usure, le prix des objets, de 5, 10 ou 15 coins de réduction. Ces réductions sont également cumulables, et le joueur pourra acheter 2 « guides de l'épicier cassé » s'il le souhaite, ainsi, il pourra bénéficier de 10 coins de réduction sur les objets. Ces réductions ne seront applicables qu'au prochain marchand que le joueur va rencontrer (il faut lire le livre avant d'acquérir les compétences de marketing d'un épicier.)

Les objets et le système d'inventaire

Concernant les objets que j'avais implémentés pour cette première soutenance, ils étaient au nombre de 4 (3 objets passifs et 1 objet actif). Le premier objet que j'avais implémenté est la potion de vitesse. Lors de son achat, celle-ci confère au joueur un certain bonus de speed en fonction de l'usure de l'objet. Sois 5, 10 ou 15 % de la vitesse de base du joueur. L'ensemble des objets passifs ne sont trouvables que chez le marchand.



FIGURE 31 – Potion de Vitesse

Le second objet passif que j'avais implémenté était le guide de l'épicier dont j'ai parlé précédemment.



FIGURE 32 – Guide de l'épicier

Le dernier objet passif que j'avais implémenté est le Pretzel d'ur chance. Cet objet est une sorte d'ange gardien pour le joueur. En effet, si le stress du joueur atteint les 200, et que celui-ci possède un Pretzel d'ur chance, son stress va diminuer instantanément du montant du « compteur de Pretzel » qu'il possède. Ce compteur est situé dans l'inventaire Passif du joueur, et est incrémenté lorsque celui-ci achète un Pretzel en fonction de l'usage de celui-ci. S'il est cassé, le stress réduit est de 50, s'il est usé, il est de 75 et s'il est neuf, de 100.



FIGURE 33 – Pretzel d'ur chance

L'ensemble des objets passifs est stocké dans un inventaire Passif qui apparaît lorsque nous appuyons sur la touche « i » et qui disparaît de la même manière. Dès que nous achetons un objet chez le marchand, celui-ci est placé dans l'inventaire afin que nous puissions savoir quels objets nous possédons actuellement.



FIGURE 34 – Inventaire vide et avec objets

L'objet actif que j'avais implémenté pour cette première soutenance était la « Tarte ». En effet, celle-ci permet au joueur, lorsqu'il l'active, de se régénérer instantanément de 10 points de stress. La Tarte ne pourra être utilisée que toutes les 40 secondes car l'ensemble des objets actifs possède un cool down avant de pouvoir être réutilisé par le joueur. Pour la gestion du cool down, je vous renvoie au rapport de soutenance de Hugo.



FIGURE 35 – La Tarte

Le joueur ne peut posséder qu'un seul objet actif à la fois. Ces derniers sont ramassables par terre dans les niveaux du jeu. Lorsque le joueur va ramasser l'un de ces objets, l'objet qu'il tient dans sa main va disparaître et être remplacé par l'objet ramassé. L'objet en possession du joueur est stocké dans un inventaire actif. L'objet en question ne peut être utilisé qu'en appuyant sur une touche ou bien en cliquant sur la case où est stocké l'objet.



FIGURE 36 – Inventaire Actif

Système de Stress

Pour finir, la dernière chose que j'avais faite pour cette première soutenance était un système de stress en collaboration avec Hugo. Ce système est assimilable à un système de points de vie inversé, car 0 est la valeur où notre personnage est en pleine forme et 200, où il est en arrêt cardiaque. Pour le visuel de ce système, je vous renvoie à la soutenance d'Hugo. Concernant mon travail pour ce système, j'avais simplement effectué le corps du système avec la création des différentes variables ainsi que la liaison avec les différents items du jeu qui sont reliés au stress.

Seconde soutenance

Mon rôle dans cette seconde soutenance avait été d'implémenter les effets visuels du stress, finir les objets du jeu (en collaboration avec Hugo) et enfin, faire l'intelligence artificielle. Pour l'ensemble de mes tâches, je m'étais appuyé sur des tutoriels et certaines documentations internet pour avoir des notions sur le travail que je devais faire. Ce milieu de projet fut assez compliqué pour moi, car je n'avais jamais abordé la notion d'intelligence artificielle, et il a fallu me documenter afin de trouver des pistes pour débuter sa conception.

Effets visuels du stress

Dans le but de créer des effets visuels pour le stress, nous avions décidé au début avec l'ensemble du groupe, de faire du flou et de saturer la musique au fur et à mesure que le stress augmentait. Cependant, par souci d'esthétisme, nous avions préféré abandonner l'idée du blur, car, lorsque nous appliquions celui-ci sur l'écran du joueur, cela dénaturait beaucoup le jeu. Pour remplacer celui-ci, nous avions donc choisi d'utiliser des filtres qui obscurcissent l'écran du joueur au fur et à mesure. Nous avions choisi de mettre 3 stades pour ces filtres, le premier est transparent, nous voyons le jeu parfaitement bien (c'est dans le cas où notre stress est inférieur ou égal à 100.) Le second filtre obscurcit un peu plus l'écran et concerne la plage de stress comprise entre 100 et 150. Le dernier filtre obscurcit l'écran et ajoute un effet rouge, il se met lorsque le stress est supérieur à 150. Parallèlement aux filtres, le son de la musique du jeu sera de plus en plus distordu. Sa distorsion commence à 100, tout comme le second filtre, et augmente à 150 avec l'arrivée du troisième filtre.



FIGURE 37 – Filtre numéro 1



FIGURE 38 – Filtre numéro 2

Récapitulatifs de l'ensemble des objets

Concernant les objets pour cette soutenance, je m'étais occupé, en collaboration avec Hugo, de la finition de tous les objets. Me concernant, je vais vous lister l'ensemble des objets que j'ai faits dans ce projet (pour connaître leurs effets, je vous invite à relire la catégorie première soutenance du rapport de projet.)

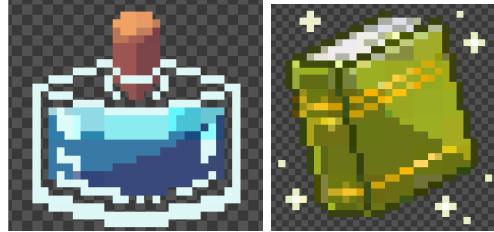


FIGURE 39 – Potion de Vitesse et Guide l'épicier



FIGURE 40 – Pretzel d'urchance et La Tarte

Intelligence artificielle

Concernant l'intelligence artificielle que je devais faire pour cette soutenance, elle m'a paru infaisable au premier abord. En effet, j'avais passé plus de deux journées complètes pour trouver un système fonctionnel qui puisse refaire les mouvements de notre personnage. Tout d'abord, ma première idée fut de stocker dans une liste les positions successives de mon personnage afin que l'intelligence artificielle les suive avec un certain décalage et cela permettrait de faire en sorte que l'intelligence effectue les mêmes mouvements que mon personnage avec un certain décalage. Cependant, cette première idée fut abandonnée, car elle ne marchait pas du tout. J'ai donc pensé que c'était la structure de liste qui était embêtante, car cela me faisait une liste infinie de positions du joueur et donc que la structure de donnée n'était pas adaptée à mes besoins. Je me suis alors tourné vers les files, car cette structure était parfaitement adaptée à mes besoins, mais encore une fois, mon intelligence artificielle ne faisait pas du tout ce que je demandais et allais vers mon joueur par le chemin le plus rapide au lieu de reproduire les mouvements de celui-ci. De ce fait, j'en ai donc conclu que ce n'était pas la structure de données qui était erronée, mais plutôt ma façon de faire l'intelligence artificielle. Par la suite, j'étais retourné faire des recherches sur l'internet afin de trouver des idées qui permettraient d'aboutir à mon intelligence artificielle. Ainsi, j'avais trouvé une méthode qui permettait d'effectuer une update d'une fonction (autre que la fonction update prédefinis) sur un intervalle de temps et ainsi n'effectue l'update que toutes les secondes ou demi-secondes (intervalle de temps que je devais définir). J'avais donc créé une fonction qui permettait de façonner un path(= chemin) jusqu'à mon joueur, et parallèlement à ça, j'avais créé une liste de chemins (jusqu'à mon joueur) et j'update le path actuel toutes les 0.14 secondes avec les paths que je mets dans ma liste. Cette méthode m'avait permis d'aboutir à une intelligence artificielle complète qui reproduit à l'identique les mouvements de notre joueur.

Je vais maintenant vous montrer comment fonctionne l'intelligence artificielle en pratique, grâce à des exemples visuels. Premièrement, j'avais créé un layer appelé "obstacle" qui définit dans chaque salle les éléments qui doivent être considérés par l'intelligence artificielle comme des obstacles qu'il doit prendre en compte dans son path et donc contourner. Voici un exemple de l'une

des salles :



FIGURE 41 – Fonctionnement de l'intelligence artificielle

Sur l'image ci-dessus, nous pouvons remarquer de nombreux éléments graphiques inhabituels sur la scène. Tout d'abord, la zone bleue représentée sur la scène est la zone où l'intelligence artificielle à le droit de circuler et au contraire, les éléments qui n'ont pas de filtre bleu sur eux sont les éléments considérés comme des obstacles par l'intelligence artificielle. Sur l'image ci-dessous, nous pouvons remarquer de nombreux éléments graphiques inhabituels sur la scène. Tout d'abord, la zone bleue représentée sur la scène est la zone où l'intelligence artificielle a le droit de circuler et, au contraire, les éléments qui n'ont pas de filtre bleu sur eux sont les éléments considérés comme des obstacles par l'intelligence artificielle. Cependant, si l'intelligence artificielle touche notre joueur, celle-ci va disparaître et notre joueur va mourir. La seconde manière de faire disparaître l'intelligence artificielle sans mourir, est de faire diminuer son stress en dessous des 190 points de stress, c'est-à-dire que pour que le fantôme disparaîtse, le joueur doit avoir un montant de stress inférieur ou égal à 189.

Dernière soutenance

Dans cette dernière soutenance, mon rôle a tout d'abord été de terminer l'intelligence artificielle de notre jeu. Pour rappel, celle-ci consiste à reproduire les mouvements de notre joueur lorsque celui-ci atteint 200 points de stress. Si le joueur se fait toucher par le fantôme, c'est game over. Le plus gros du travail de l'intelligence artificielle ayant été fait à la soutenance précédente, il ne me restait plus qu'à faire en sorte que toutes les 15 secondes, si le joueur n'a pas été touché par le fantôme, et que ses points de stress sont toujours au-dessus de 189, un nouveau fantôme apparaisse et commence lui aussi à poursuivre notre joueur.

Ensuite, mon second travail pour cette soutenance consistait à réaliser une boutique d'amélioration, qui permettrait au joueur d'acheter des améliorations qu'il garderait jusqu'à la fin du jeu. J'ai également participé à la création de la majorité des améliorations avec Mathéo.

Finition de l'intelligence artificielle

Dans le but de terminer l'intelligence artificielle, mon premier travail pour cette soutenance a été de me pencher sur la question de créer plusieurs instances du fantôme avec, pour chacune, un décalage de 15 secondes entre leurs apparitions. Ce premier travail fut assez simpliste, car il me suffisait d'utiliser le delta time (outil implémenté dans unity) afin de créer un intervalle de 15 secondes. Cependant, j'ai dû me confronter à un problème quant à la disparition des fantômes lorsque l'un d'eux touchait le joueur, ou encore, que notre stress retombait en dessous de la barre des 189. En effet, le souci était que lorsque l'un des fantômes touchait notre joueur, celui-ci disparaissait de façon normale, mais s'il y avait d'autres instances de fantômes, celles-ci restaient quant à elles sur la scène, et donc ne disparaissaient pas. J'ai donc essayé de résoudre ce problème en regardant mon code, et en essayant de changer certaines choses, mais je ne comprenais pas pourquoi les fantômes ne disparaissaient pas, car ils devaient obligatoirement disparaître lorsque le stress de notre joueur était en dessous de 189. J'ai donc pensé que le souci venait de l'instanciation des différents fantômes, et qu'un moyen de le résoudre serait de maîtriser toutes ses instanciations en les mettant tous dans une sorte de "boîte" puis ensuite de détruire cette "boîte". Ainsi, la solution qui finit par fonctionner fut de créer une game object regroupant toutes les instanciations de fantôme, puis de le détruire, ce qui a fortiori, détruisait toutes les instances de fantôme.

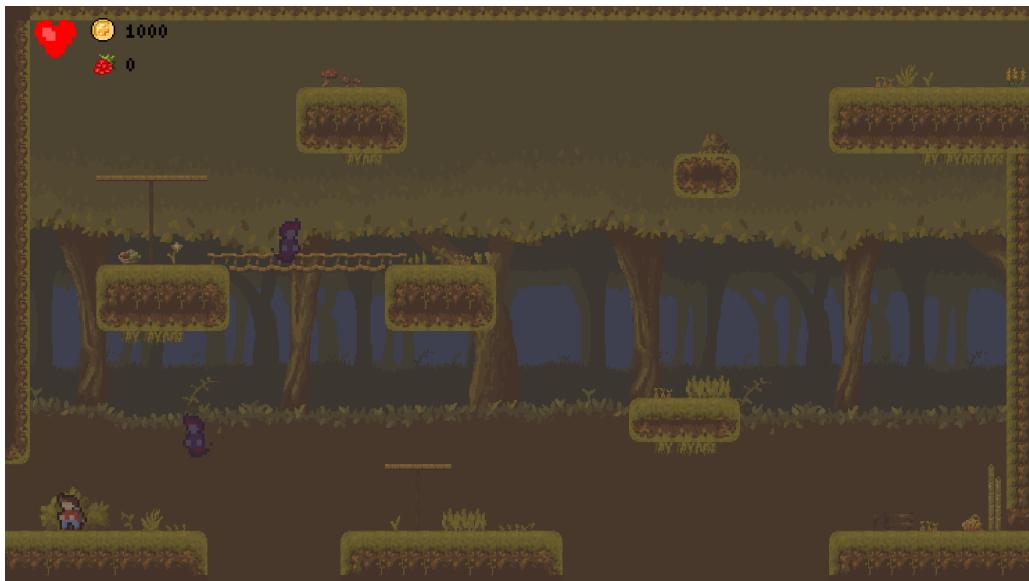


FIGURE 42 – Extrait du jeu avec deux fantômes

Système d'améliorations et création du PNJ d'améliorations

Ensuite, je me suis penché sur notre système d'améliorations pour notre joueur. Le concept de ces améliorations est assez similaire au système d'achat des items, mais en même temps assez différent. Tout d'abord, je me suis occupé de la partie interface, en créant une interface similaire à celle du marchand pour les items. Mais contrairement au marchand, dans cette interface, toutes les améliorations sont visibles. Pour acheter ces améliorations, il faut avoir assez de monnaies raspberries ainsi que de posséder le niveau précédent de chaque amélioration pour les améliorations d'un rang supérieur au niveau 1.

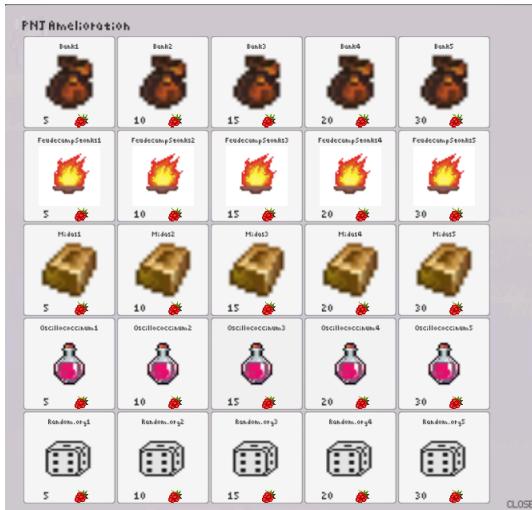


FIGURE 43 – Interface du PNJ des améliorations

Comme vous pouvez le voir sur la capture d'écran ci-dessus, les prix augmentent avec le niveau des améliorations, allant de 5 à 30 raspberries. Dans ce système d'améliorations, j'ai fait en sorte que nous ne puissions pas acheter l'amélioration 2 sans avoir acheté l'amélioration 1. De plus, lorsque nous achetons l'une des améliorations, la prochaine fois que nous retournerons dans la boutique, celle-ci comportera dans son nom "Unavailable" afin qu'on puisse savoir quelles améliorations nous avions acheté au préalable. Lorsque nous achetons ces améliorations, celles-ci sont sauvegardées dans un inventaire d'améliorations, et les buffs liés à celle-ci sont immédiatement appliqués sur notre joueur.

Les améliorations

Après avoir fini le système d'améliorations, il ne manquait plus qu'à les faire. Cependant, ayant retiré le système de flou jugé trop peu esthétique, l'amélioration "optic 2025" n'a pas pu être réalisé. Cependant, toutes les autres améliorations ont pu être faites. Tout d'abord, parmis ces améliorations, nous retrouvons l'amélioration "Bank", qui permet de pouvoir garder une partie de ses pièces à la fin d'une partie. Chaque niveau augmente la quantité maximum de pièces gardées de 25 pour un total de 125 pièces sauvegardées à la fin d'une partie au niveau maximum.

Ensuite, nous retrouvons l'amélioration "feu de camp" qui augmente la quantité de stress rendu de 10% par le feu de camp, pour un total de 90% de stress rendu une fois l'amélioration au niveau maximum.

Il y a également l'amélioration "Midas" qui donne plus de pièces quand un pot est brisé. Chaque niveau augmente le nombre maximum de pièces récupérables par pot de 1. L'intervalle de départ étant de 0-1, l'intervalle final sera donc de 0-6 au niveau maximum.

Pour finir, la dernière amélioration que j'ai faite en collaboration avec Mathéo fut "Oscillococcinum" qui augmente la quantité de stress que peut subir le personnage. Chaque niveau augmente la quantité de stress subi maximum de 20 points de la valeur initiale pour une augmentation de 100 points au niveau maximum.

La dernière amélioration a été réalisée par Hugo, et elle s'appelle "Random.org". Cette amélioration permet au joueur de commencé la partie avec un item permanent (buff) aléatoire. Plus le niveau est haut, meilleure est la rareté. Le niveau 1 donne accès à un item brisé, le niveau 2 donne

accès à un item usé, le niveau 3 donne accès à un item neuf, le niveau donne accès neuf et un item brisé et enfin le niveau 5 donne accès à un item neuf et un item usé.

Ci-dessous les visuels pour chacune des améliorations.



FIGURE 44 – De gauche à droite, "feu de camp" "bank" "midas" "Oscillococcinum" "random.org"

Ressentis sur le projet

Ce projet nous a permis de nous rendre compte de la difficulté de se lancer dans une entreprise à partir de zéro. Cela a été néanmoins très instructif et nous avons trouvé le challenge très intéressant à relever. En effet c'est le premier travail où nous sommes vraiment libres et nous devons nous débrouiller en totale autonomie vis-à-vis de l'équipe pédagogique pour avancer. Nous restons tout de même une équipe qui se soutient et ne sommes donc pas abandonnés et livrés à nous-même. Cela nous a permis de faire l'expérience d'un "vrai" projet de groupe ainsi que la notion de deadline, imposé par l'école ou bien entre nous afin de pouvoir avancer efficacement.

Nous ne nous attendions néanmoins pas à une telle charge de travail pour réaliser un projet de ce genre due à tous les détails à prendre en compte pour obtenir un résultat satisfaisant, mais la nature stimulante du projets à rendu cette charge de travail passionnante.

Nous avons également su se montrer patient face aux nombreuses difficultés auxquelles nous avons du faire face, et il n'y a jamais eu de mauvaise ambiance dans notre groupe même si nous avions presque perdu notre projet à cause d'une mise à jour unity.

L'ensemble des solutions à tous ces problèmes ont été trouvées en groupe grâce à un bon esprit d'équipe.

Le groupe Hanabi.

Conclusion

Et c'est là-dessus que se termine notre aventure. Nous pensons avoir bien réussi et avons accompli tous les objectifs que nous nous étions fixés dans le cahier des charges. Bien sûr nous avons rencontré des difficultés, mais ce projet reste toujours aussi intéressant et permet une approche de travail différente des cours "normaux", plus libre où chacun se fixe ses objectifs. Ce projet aura été riche en expérience et nous en garderons tous un très bon souvenir.

Nous espérons que vous apprécierez autant que nous le fruit de notre travail autant que nous avons apprécié à le faire.

En vous remerciant de l'attention portée à ce projet, et de nous avoir offert l'opportunité de réaliser cette aventure

Hugo, Mathéo, Angelina, Antoine

Annexe



FIGURE 45 – Jump1.1 et jump1.2



FIGURE 46 – Key1.1 et key1.2



FIGURE 47 – Button1.1 et button1.2



FIGURE 48 – Captor1.1 et captor1.2



FIGURE 49 – Torch1.1 et torch1.2



FIGURE 50 – Shop1 et history1



FIGURE 51 – Campfire1



FIGURE 52 – Room choice et defi1



FIGURE 53 – Jump2.1 et jump2.2

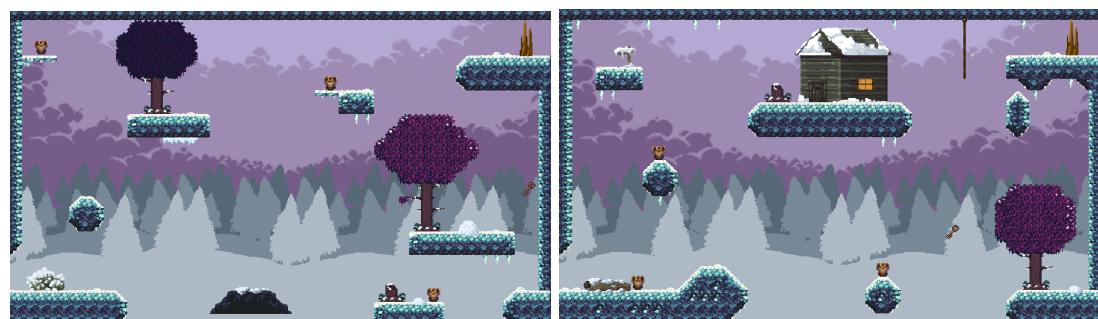


FIGURE 54 – Key2.1 et key2.2



FIGURE 55 – Button2.1 et button2.2

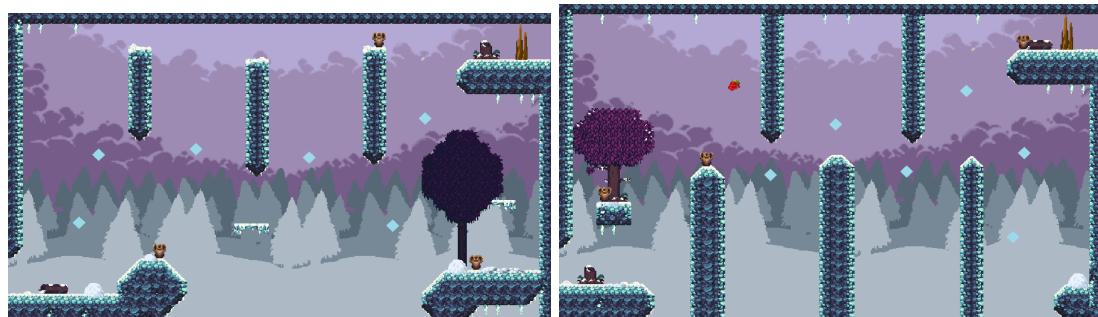


FIGURE 56 – Captor2.1 et captor2.2



FIGURE 57 – Torch2.1 et torch2.2



FIGURE 58 – Shop2 et campfire2



FIGURE 59 – History2 et history3



FIGURE 60 – History4 et defi2

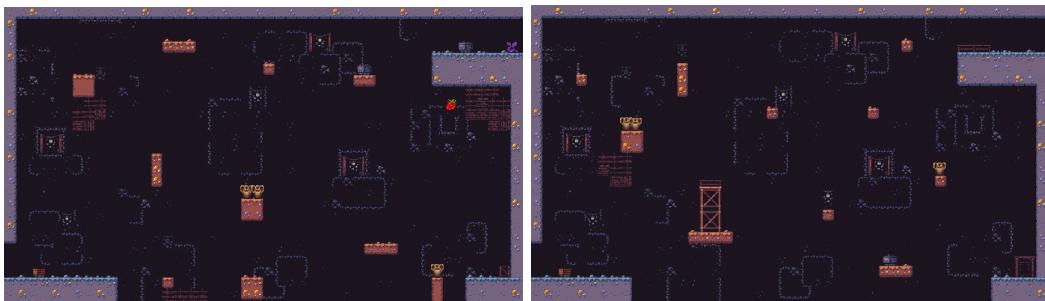


FIGURE 61 – Jump3.1 et jump3.2

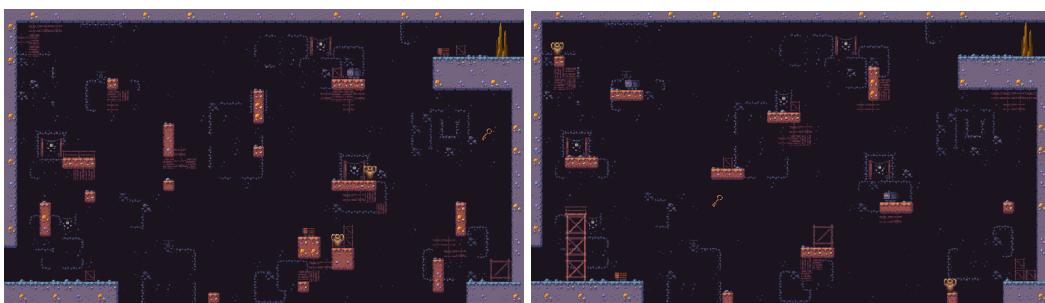


FIGURE 62 – Key3.1 et key3.2

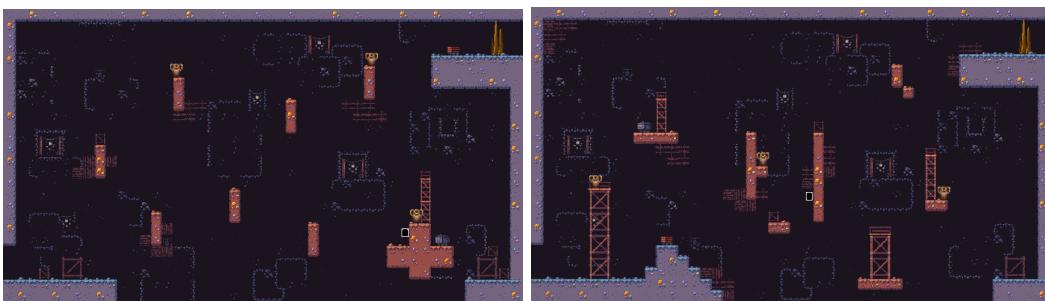


FIGURE 63 – Button3.1 et button3.2

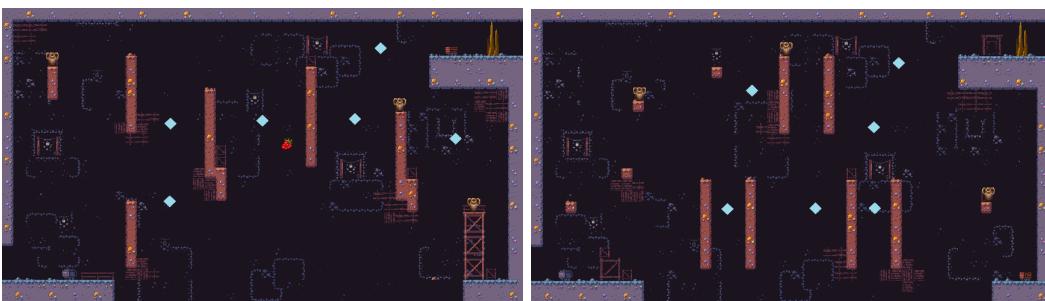


FIGURE 64 – Captor3.1 et captor3.2

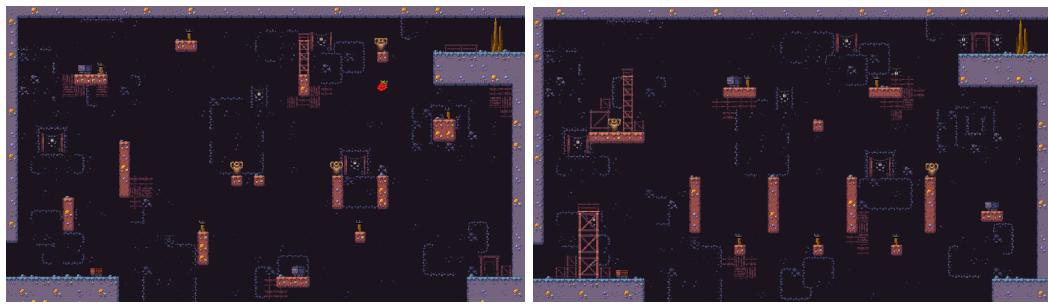


FIGURE 65 – Torch3.1 et torch3.2

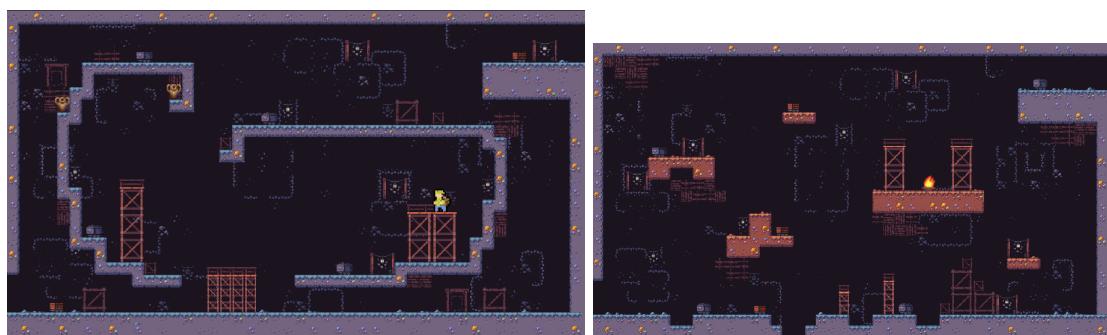


FIGURE 66 – Shop3 et campfire3

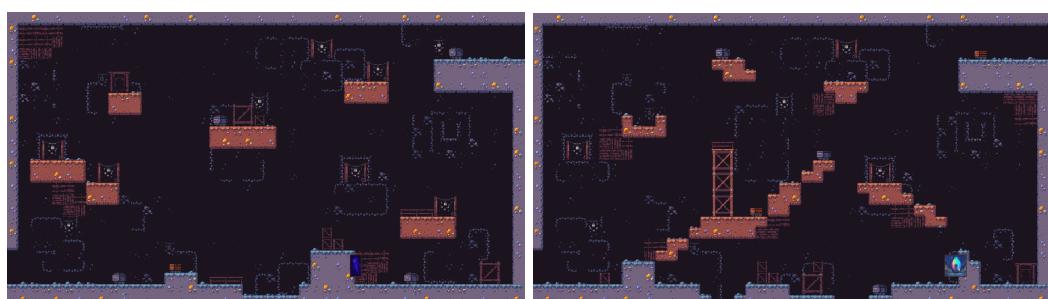


FIGURE 67 – History5 et history6

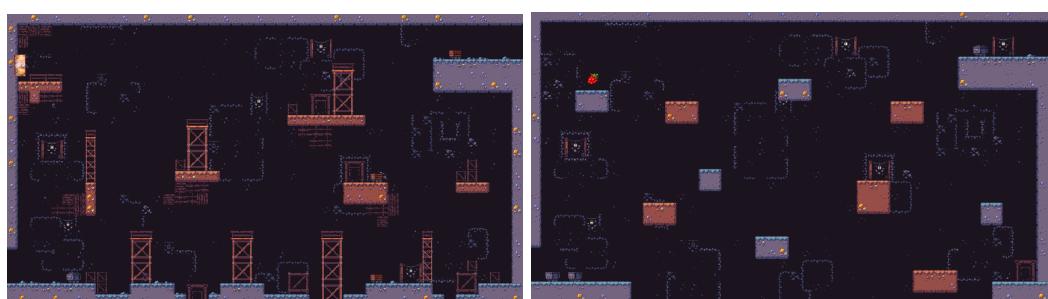


FIGURE 68 – History7 et defi3

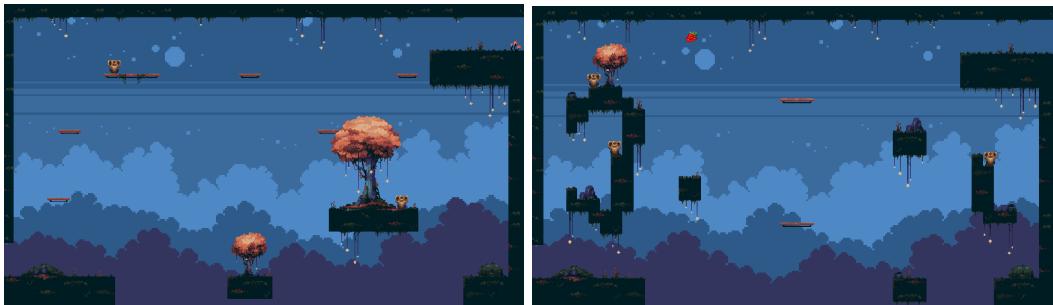


FIGURE 69 – Jump4.1 et jump4.2



FIGURE 70 – Key4.1 et key4.2



FIGURE 71 – Button4.1 et button4.2



FIGURE 72 – Captor4.1 et captor4.2

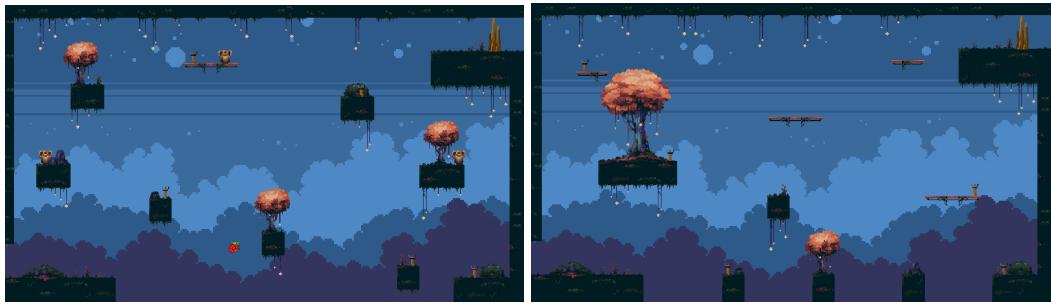


FIGURE 73 – Torch4.1 et torch4.2



FIGURE 74 – Shop4 et campfire4

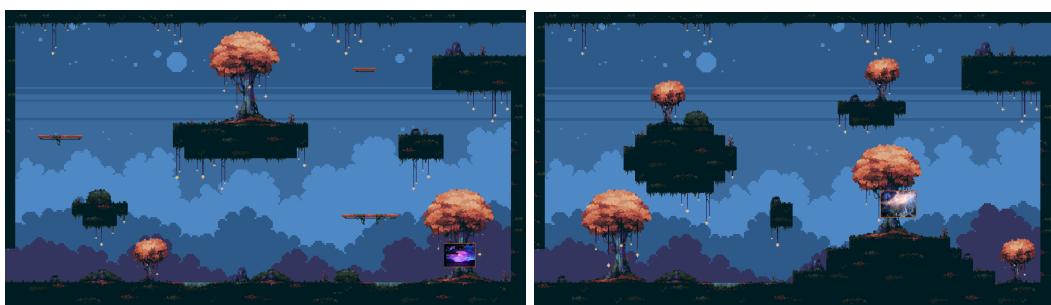


FIGURE 75 – History8 et history9

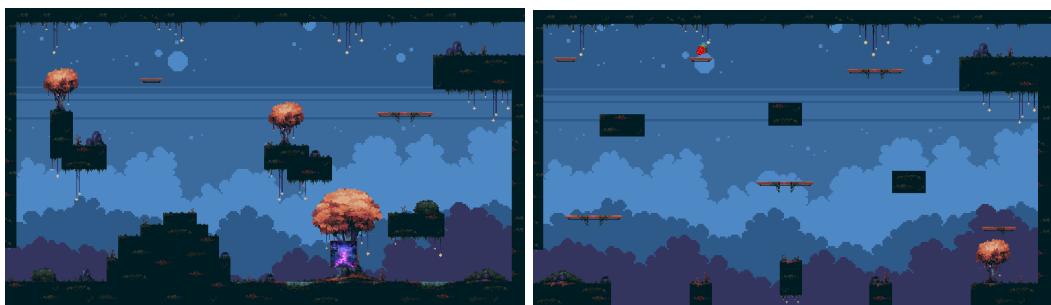


FIGURE 76 – History10 et defi4

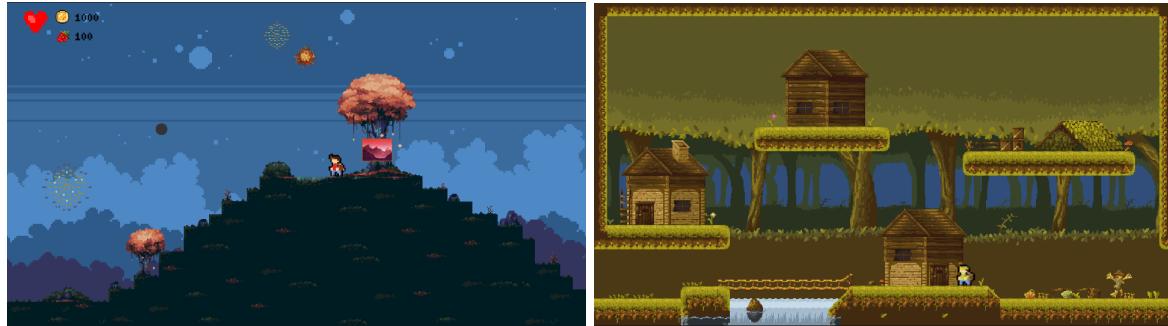


FIGURE 77 – Last scene et marchand amélioration

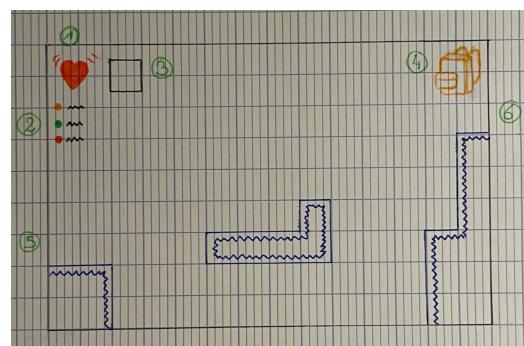


FIGURE 78 – Schéma initial de notre HUD