EPITA

Rapport de Soutenance n°1 - projet S2

THE OTHER SIDE

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

Groupe : Hanabi

Table des matières

Introduction

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 sortien 2018. Dans celui-ci, chaque ensemble d'actions doit être réfléchi et organisé pour achever untableau. 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

Objectif

Le but de ce premier rendu est de mettre au clair toutes les tâches et leur repartition parmi tous les membres de l'équipe. Il permet également de rendre compte de ce qui a été fait lors des premières semaines et de partagés les ressentis personels des membres vis à vis de l'organisation du travail, des difficultés rencontrées et de ce qui a été accompli.

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 feuxd'artifices. Le personnage principal est un homme qui manque de confiance en soi et qui cherchesa place dans le monde, un but à son existence. Le personnage souffrant du dédoublement de lapersonnalité, cherche à gravir une montagne afin de se réunir avec son double et d'accomplir unexploit 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, celapermettra au double de percevoir ce qui est invisible pour le commun des mortels. Le personnagedoute de sa personne et se veut visible et invisible à la fois, créant ainsi un double qui existeet n'existe pas en même temps. L'entité du double se base sur le théorème de superposition quantique émis par Schrödinger.

Tableau de repartition des tâches : Première soutenance

Petite piqure de rappel de ce que chaque membre devait accomplir pour la première soutenance d'après le cahier des charges.

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 coeur 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'urchance				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

Hugo

Début du projet

Pour cette première soutenance, j'aurais pu m'atteler à de nombreux domaines, tant bien de l'animation, que du code et du design. 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. C'est ainsi que, confiant, j'ai pu commencer à travailler sur notre projet.

Création d'assets pour le jeu

J'ai été chargé pour cette soutenance de dessiner les assets de notre personnage principal, d'un PNJ qui apparaîtra dans notre jeu, dans les salles boutiques, ainsi que l'asset des Raspberries. Tous ces assets ont été réalisés en utilisant le logiciel Piskel, logiciel gratuit permettant de faire du pixel art ainsi que des tilesheets qui pourront plus tard être directement utilisées dans notre jeu, que ce soit pour des animations ou encore une simple image fixe pour un item par exemple.

FIGURE 1 – Interface graphique de Piskel

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

Après avoir chosi cette color palet, j'ai tout de suite commencé à travailler sur les assets de notre personnage, soit ses animations :

— Idle
— De marche
— De début de saut
— De chute (fin de saut)

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. Celles-ci sont également disponibles en trois variantes différentes, une pour chaque type d'équipement de notre personnage, soit rouge pour le classic, vert pour le bouncy et violet pour le light.

FIGURE 2 – Différentes couleurs des équipements

FIGURE 3 – Tilesheets des animations idle, de marche, de saut et de chute

Comme dit précédemment j'ai également été amené dessiner une idle animation pour notre PNJ. Pour cela j'ai réutilisé l'animation de notre personnage principal et l'ai modifiée de façon à ce que notre PNJ puisse avoir sa propre personnalité sans avoir à refaire une animation depuis le début.

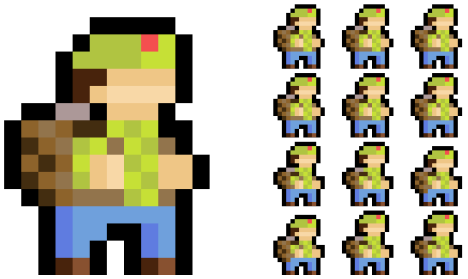

FIGURE 4 – Sprite et tilesheet de notre marchand

Enfin je me suis occupé de faire l'asset d'une de nos monnaies, la raspberry.

FIGURE 5 – Sprite d'une raspberry

C'est seulement une fois tous ces assets faits que j'ai changé de domaine, pour cette fois m'attaquer au site web.

Site web

Seul le squelette du site a été réalisé pour l'instant, celui-ci a été fait en html et en css et ne contient pour l'instant que le nom de notre projet ainsi qu'un menu permettant l'accès à tous les onglets du site. Ces principaux onglets étant :

— Accueil
— Notre projet
— Principe
— Comment Jouer
— Contact

Toutes ces sections seront remplies de façon à pouvoir décrire de façon claire notre projet, son contenu, comment nous avons fait pour coder celui-ci et bien d'autres informations.

Le menu lui est intéractif, lorsque nous passons la souris sur l'une des sections, celle-ci change de couleur, et si cette section contient des sous sections, ces dernières apparaissent en dessous, comme montré dans la capture d'écran en dessous.

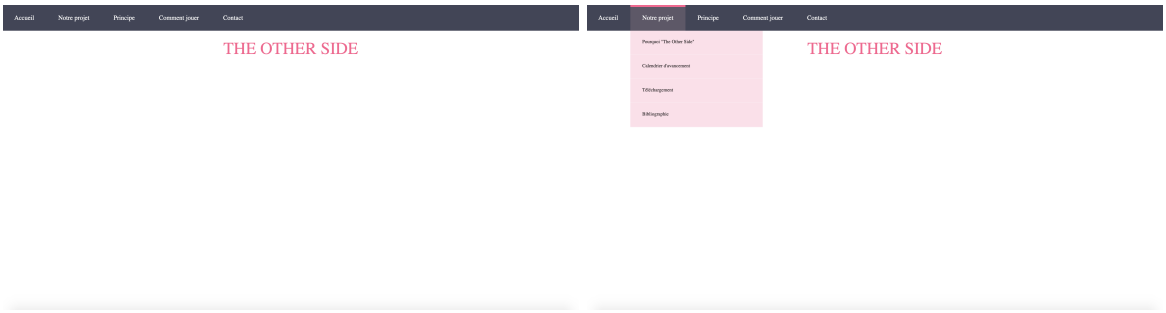

FIGURE 6 – Captures d'écran de notre site

Comme dit, tous les fichiers du site ont été créés, soit le travail le plus dur aura été fait, il ne nous reste maintenant plus qu'à remplir ce dernier dont voici l'architecture des fichiers :

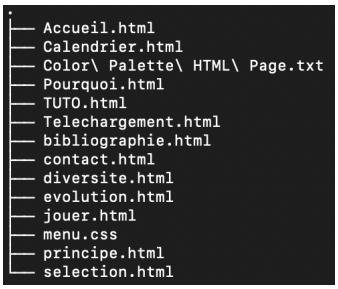

FIGURE 7 – Architecture du site web

Une fois le squelette du site terminé et les tilesheets faites, il était temps pour moi de passer au sérieux et de commencer à travailler sur Unity, en commençant par créer des items pour notre jeu.

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 augemente 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.

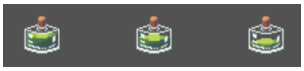

FIGURE 8 – visuels de la potion de saut

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 9 – 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 10 – 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 apparait 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.

```
switch (currentItem.id)
{
    //if the item is "updraft"
    case 8:
        PlayerMovement.instance.itemJump = true;
        cooldown = Time.time + 15f;
        timeStart = 15f;
        break;
    //if the item is "plutôt deux fois Khune"
    case 15:
        PlayerMovement.instance.hasDashed = false;
        cooldown = Time.time + 15f;
        timeStart = 15f;
        break;
    //if the item is "sandwich triangle"
    case 7:
        PlayerStress.instance.HealStressplayer( amount: 20);
        cooldown = Time.time + 40f;
        timeStart = 40f;
        break;
}
```

FIGURE 11 – switch qui associe à l'id de litem utilisé son effet et son cooldown

```
private void Update() //updates the countdown for the current active item
{
    if (contenu.Count > 0)
    {
        if (timeStart > 0)
        {
            timeStart -= Time.deltaTime;
            displayTime.text = Mathf.Round(timeStart).ToString();
        }
        else
        {
            displayTime.text = "ready";
        }
    }

    else
    {
        displayTime.text = "";
    }
}
```

FIGURE 12 – fonction décrémentant le cooldown affiché, s'appelle a chaque frame

FIGURE 13 – *Plutôt deux fois khune avant et immédiatemment après utilisation*

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'ai 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 rétissant à l'idée de m'attaquer au menu d'options, l'idée de travailler directement avec les paramètres de la machine me parraissant assez ardue. Mais heureusement, Unity aura rendu la tache 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.

FIGURE 14 – Main et Options Menu actuels de notre jeu

Animation du coeur (système de vie)

Comme dit précédemment, la vie de notre personnage sera représentée sous forme de stress, plus notre personnage est stressé, plus son coeur bât 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 coeur en haut à gauche de notre écran jouera l'animation *HeartBeat slow*, si son stress est supérieur à ce seuil de 100 points, le coeur joue l'animation *HeartBeat fast* et si le stress est supérieur à 150, le coeur jouera l'animation *HeartBeat crazy*.

FIGURE 15 – UI in game située en haut à gauche de notre écran montrant le coeur et les compteurs de pièces

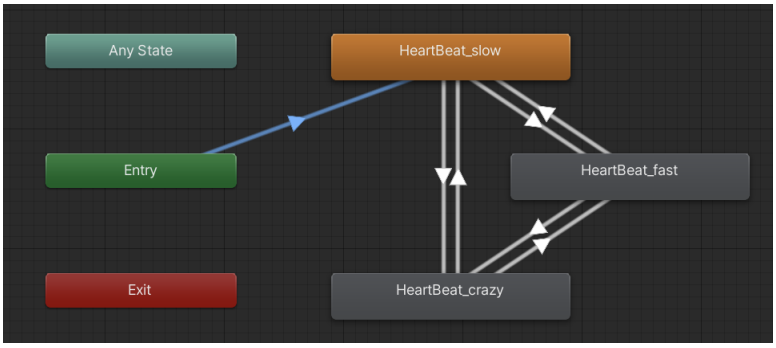

FIGURE 16 – liaisons des animations du coeur de notre personnage

Plan pour la prochaine soutenance

Pour la prochaine soutenance, je serai chargé de terminer toutes les animations de nos personnages, soit toutes les animations des dashs correspondants aux trois équipements.

Une grande partie de mon travail pour cette seconde soutenance sera centrée sur la mécanique du stress, je serai chargé d'implémenter l'augmentation progressive du stress ainsi que la perte de stress proportionnellement au temps pris pour compléter la dernière salle. Je vais également devoir implémenter les salles *feu de camps* qui, lorsque le joueur en découvre une, restaurera une certaine quantité de stress en fonction du niveau d'amélioration de la salle. Toujours en rapport avec la gestion du stress au sein d'une salle, il me faudra faire en sorte que le stress se stabilise dans les salles *feu de camps* et *shop*, qui permettront alors au joueur de se reposer tout en prenant compte de ses items acquis jusqu'ici.

Une fois mon travail sur le stress terminé, il me faudra implémenter un menu pause, qui sera accessible en pleine partie et qui permettra au joueur de quitter sa partie ou bien d'accéder au menu des options.

Quant aux items, il me faudra en implémenter deux : un actif, *the world* qui permet au joueur de se téléporter en avant au moment de l'utilisation, et un passif, *les long-fall boots*, qui quand elles sont dans votre inventaire, vous font gagner moins de stress au moment où le joueur tombe dans un trou.

Pour finir je remplirai le site web et le peaufinerai de telle façon à ce que celui-ci soit intégralement rempli à l'exception du lien de téléchargement pour le jeu.

Mathéo Romé

Début du projet

Pour cette soutenance, je me suis surtout occupé de la partie personnage (ses déplacements et ses animations) et de la partie multijoueur.

Mon premier reflex au début du projet a tout d'abord été de me documenter sur Unity et plus précisement sur unity2D au travers de nombreux tutos et de page de documentation du site Unity et Photon. Cela a été très instructif est m'a permis de commencer mon projet plus sereinement.

Déplacement du personnage

Mon premier objectif dans ce projet a été d'avoir un personnage pouvant se déplacer le plus rapidement possible afin que tous les autres membres de mon groupe puissent faire plus facilement leurs tests et de manière plus pratique grâce à l'utilisation d'un personnage pouvant se déplacer en avant, en arrière et sauter. J'ai donc écrit un script assez et créé un personnage rapidement dans ce but.

Une fois ce prototype terminé, je me suis occupé de l'améliorer en ajoutant une variation de gravité au fur et à mesure du temps passé en l'air afin de rendre le saut plus réaliste et en ajustant les mécaniques de déplacement.

La prochaine étape a été d'implémenter la possibilité de dash pour les personnages. Il y a trois types de dash possible :

— Je me suis tout d'abord occupé du premier, un dash pouvant aller dans huit directions différentes (vers le haut, le bas, gauche, droite et les quatre diagonales).
Ce dash m'a servi de calque pour les deux autres dashs que j'ai modifié à partir de celui là. Ce dash est donc divisé en deux sous-fonctions : tout d'abord avec la première sous-fonction, on cherche la direction dans laquelle le joueur souhaite aller. Suivant les touches directionnelles appuyées, la fonction va stocker une des huit possibilités de direction.
La seconde sous-fonction ne s'active que quand la direction stockée est différente de 0. Cette fonction va "pousser" le RigidBody du joueur dans la direction demandé pendant 0.1 secondes et une fois ce temps dépassé remettre la direction à 0 afin de permettre le prochain dash.
Ce type de dash étant un classique dans beaucoup de jeux de plateforme il nous tenait à cœur de pouvoir l'implémenter dans notre jeu.

```
private void DashClassique()
{
    if (dashTime <= 0)
    {
        direction = 0;
        dashTime = startDashTime;
        rb.velocity /= 3;
        isDashing = false;
    }
    else
    {
        dashTime -= Time.deltaTime;
        hasDashed = true;
        NextDash = Time.time + dashCD;
        switch (direction)
        {
            case 1 :
                rb.velocity = (Vector2.up + Vector2.left).normalized * dashSpeed; //diagonal haute gauche
                break;
            case 2 :
                rb.velocity = (Vector2.up + Vector2.right).normalized * dashSpeed;  //diagonal haute droite
                break;
            case 3 :
                rb.velocity = (Vector2.down + Vector2.left).normalized * dashSpeed; // diagonal basse gauche
                break;
            case 4:
                rb.velocity = (Vector2.down + Vector2.right).normalized * dashSpeed; //diagonal basse droite
                break;
            case 5 :
                rb.velocity = Vector2.up * dashSpeed; //haut
                break;
            case 6 :
                rb.velocity = Vector2.down * dashSpeed;  //bas
                break;
            case 7 :
                rb.velocity = Vector2.left * dashSpeed; //gauche
                break;
            case 8 :
                rb.velocity = Vector2.right * dashSpeed; //droite
                break;
        }
    }
}
```

FIGURE 17 – Fonction faisant dash le personnage sur un temps donné suivant la direction demandé

— Le second type de dash, le dash de la classe "Bouncy" est un dash plus original qui n'a que deux directions possible mais possède la capacité unique de pouvoir, à la collision avec un mur,

de propulser le joueur vers le haut.
Comme dit précédemment, ce dash s'appuie sur le même fonctionnement que le dash de la classe "Classique" avec pour modification la réduction du nombre de directions possible à deux et un test à la fin du dash de collision avec un mur qui permet de savoir si le joueur doit être propulsé vers le haut.
Lors de la création de ce dash, j'ai 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 côté 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.

— Le dernier type de dash, le dash de la classe "Light" est une téléportation permettant de passer au travers des obstacles. Pour implémenter ce dash je me suis basé sur le même type dash que le dash de la classe "Classique" en ne conservant que la possibilité de dash vers la droite, la gauche et le haut. La différence principale étant qu'au début du dash, le "SpriteRender" et le "Polygone2D Collider" sont désactivé permettant au personnage de passer au travers des obstacles et devenant invisible pendant la duré du dash, donnant un effet de téléportation. Pour ne pas se téléporter dans un décor, j'ai utilisé la fonction OnTriggerEnter2D qui se déclenche à chaque collisions avec un collider. A chaque collision, j'incrémente une variable integrer, initialement à 0 si à la fin du dash cette variable est impaire, le personnage est re-téléporter 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 du essayer plusieur solution avant d'arriver à une qui me convienne. J'ai notament tout d'abord pensé à utiliser la fonction RayCastAll du module Physics2D qui teste sur une longueur donné le nombre de collider rencontré et les places dans un tableau. Je pensais ainsi pourvoir compter les deux cotés du terrain. Malheursement ces deux cotés ne comptait que pour un car ils appartenaient au même collider et je me suis donc tourné vers une autre solution.

```
private bool collTest(Vector2 dir)
{
    var tab :RaycastHit2D[] = Physics2D.RaycastAll( origin: (Vector2) player.transform.position, direction: dir, distance: 2f);
    if (tab.Length % 2 == 1)
        return false;
    else
        return true;
}
```

FIGURE 18 – Première fonction testant si l'on arrive dans un terrain

```
private void OnTriggerEnter2D(Collider2D other)
{
    isInside++;
}
```

FIGURE 19 – Seconde fonction testant si l'on arrive dans un terrain

Animation

Je me suis ensuite occupé d'animer les personnages. Comme nous avons décidé de faire nous-même toutes les animations des personnages, nous avons séparé l'implémentation en deux parties, car la création de l'animation prend un certain temps. Ainsi, j'ai utilisé les Tilesets d'Hugo pour implémenter l'animation de course du personnage, l'animation "Idle" du personnage (quand personne ne le déplace) ainsi que l'animation de saut.

— L'animation Idle se répète toutes les 1 secondes avec 4 Sprite différents sur cette durée, placés à distance variable.

— L'animation de course se répète toute les 0.5 secondes avec 6 Sprite différent sur cette durée, placé à équidistance.

— L'animation "Idle" est celle qu'aura notre personnage par défaut : tant que la valeur absolue de la vélocité du personnage sur l'axe x est inférieur à 0.3 (mettre la valeur à 0 provoque un bug) et qu'il sera au sol, le personnage effectueras en boucle cette animation.
— Si le personnage possède une vélocité supérieur 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 qui se divise en deux parties : la phase de montée quand le personnage possède une vélocité positive sur l'axe y et la descente quand le personnage possède une vélocité négative sur l'axe y.
Ces deux parties sont composé de 6 sprites placé toutes les 0.1 secondes.

Le graphique liant les différentes animations est ainsi représenté sous cette forme avec toutes les possibilités de passage d'une animation à une autre :

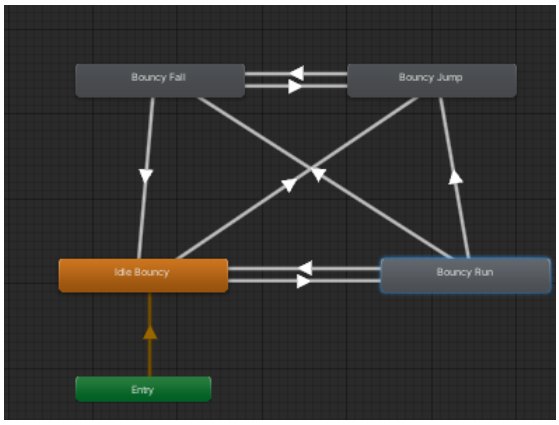

FIGURE 20 – schéma de l'animator unity

Multijoueur

Pour cette première soutenance, je me suis également occupé d'implémenter le multijoueur. Pour cela, j'ai utilisé le Photon Engine permettant de créer des salons de jeu rejoinable par deux joueurs sur des machines différentes.

J'ai 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 1 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 pourrons donc pas savoir qu'ils sont proches l'un de l'autre.

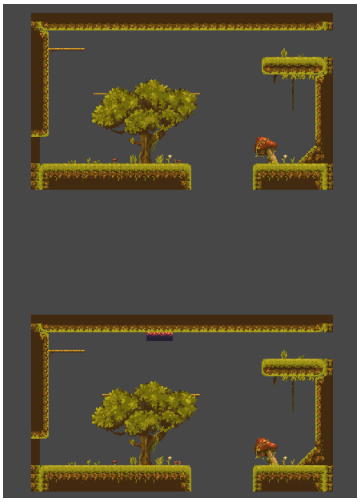

FIGURE 21 – positionnement des salles pour le multijoueur

J'ai 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 était toujours inversé. 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 palier ce problème, j'ai tout simplement ajoute un component "Rigidebody" 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) Pour savoir quelle caméra doit être désactivé pour chaque joueur, j'ai utilisé le code suivant :

```
if (photonView.IsMine)  //Active la caméra du joueur est éteint celle de l'autre joueur
{
    playerCamera.SetActive(true);
}
else
{
    playerCamera.SetActive(false);
}
```

FIGURE 22 – Fonction attribuant chaque caméra à son propriétaire

Scénario

Enfin, je me suis également occupé de la partie scénario, qui est pour l'instant assez légère, mais qui sera plus développé lors des soutenances suivante.

Cette partie a été une des moins chronophage, mais reste intéressante, car est très différentes des autres parties dont je me suis occupé qui étaient surtout centré sur le code. J'ai donc écris 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).

Prochaine Soutenance

Pour la prochaine soutenance, je prévois d'implémenter un système de sauvegarde de données lorsque le joueur quitte le jeu. Ces données seront sauvegardées localement chez le joueur 1 dans un fichier texte qui sera lu au lancement d'une partie afin d'en extraire les informations des parties précédentes.

Je m'occuperais également d'implémenter le type de salle "torche" qui consistera pour le joueur 1 à allumer des torches dans le bon ordre selon les indications du joueur 2 qui aura les informations d'ordre dans sa salle.

Enfin, je vais m'occuper d'implémenter l'item "feu de camp portatif" augmentant l'intervalle d'augmentation du stress ainsi que l'item "The World" permettant d'effectuer une téléportation vers l'avant à la manière du dash de la classe "Light". Je vais également développer le système de multijoueur, déjà fonctionnel afin de répondre au besoin apparaissant au fur et à mesure de notre projet.

Angelina

Début du projet

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. Pendant la période où je devais coder, je me suis aidée des chaînes de tutoriels sur Unity de tuto Unity FR de Brackeys et de Code Monkey. 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 de ces semaines. 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 avions 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 Varitaions by RavenTale , Free Pixel Food by Henry Software , Tarot Major Arcana by Alex Vángandr , 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 trie 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 possibiltés 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 ou bien la refermer en appuyant sur G.

— La salle test (Index 1). C'est la première salle dans laquelle on apparait 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 serait comparable à une salle jump en plus facile du à une présence plus importante de plateforme et la distance réduite entres elles. Comme dit précédemment le vide est en réalité un sol invisible, j'ai ajouté une plateforme 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.

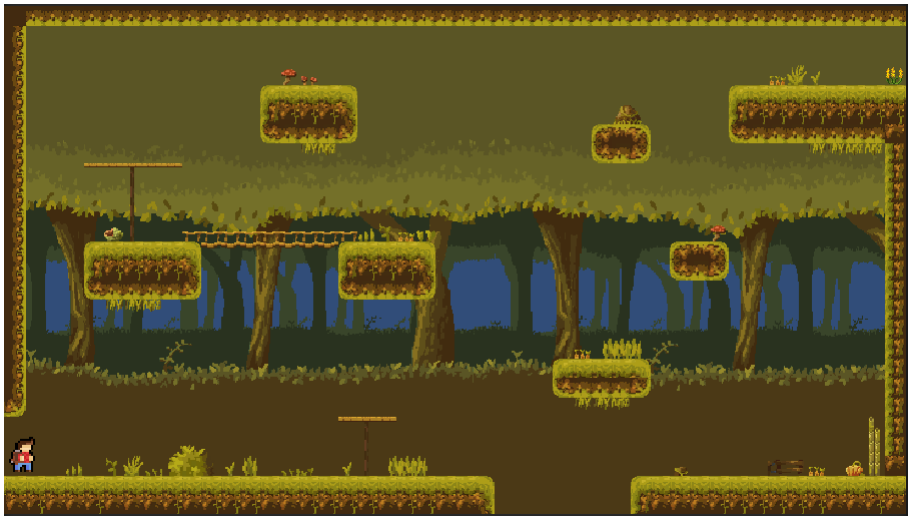

FIGURE 23 – Salle Test

— La salle jump (Index 2). C'est la salle la plus basique que j'ai fais, celle-ci comporte des plateformes et des éléments décoratifs.

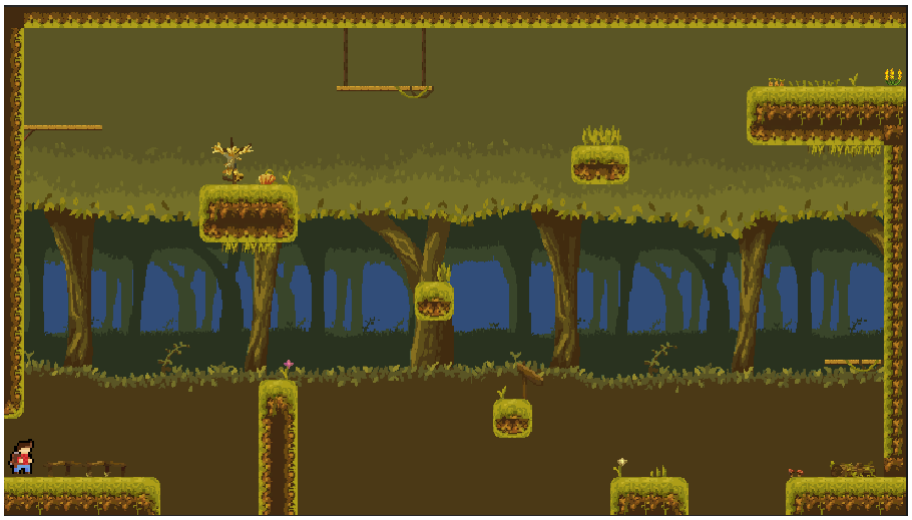

FIGURE 24 – Salle jump

— La salle key (index 3) 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.

FIGURE 25 – Salle key

FIGURE 26 – Mécanisme de la key

— La salle button (index 4). 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.

FIGURE 27 – Salle button

FIGURE 28 – button activé

— La salle boutique (index 5). 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.

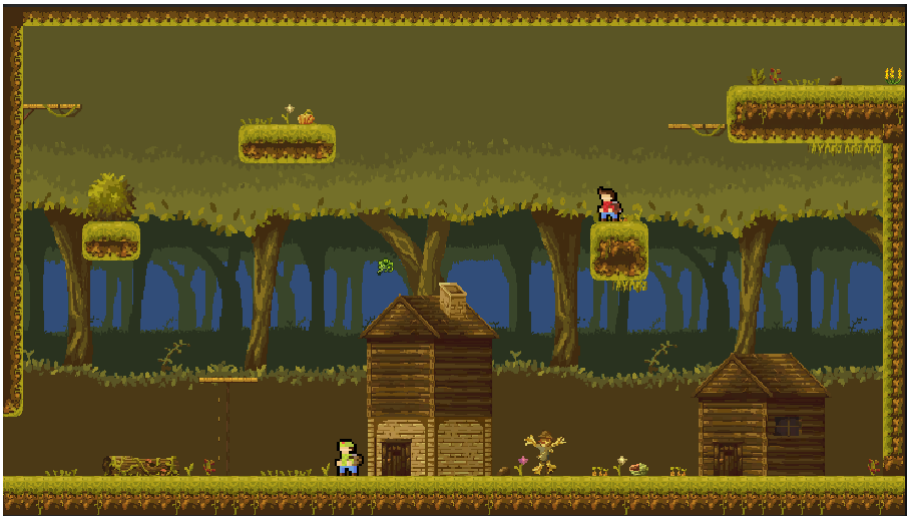

FIGURE 29 – Salle boutique

— La salle histoire (index 6). 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.

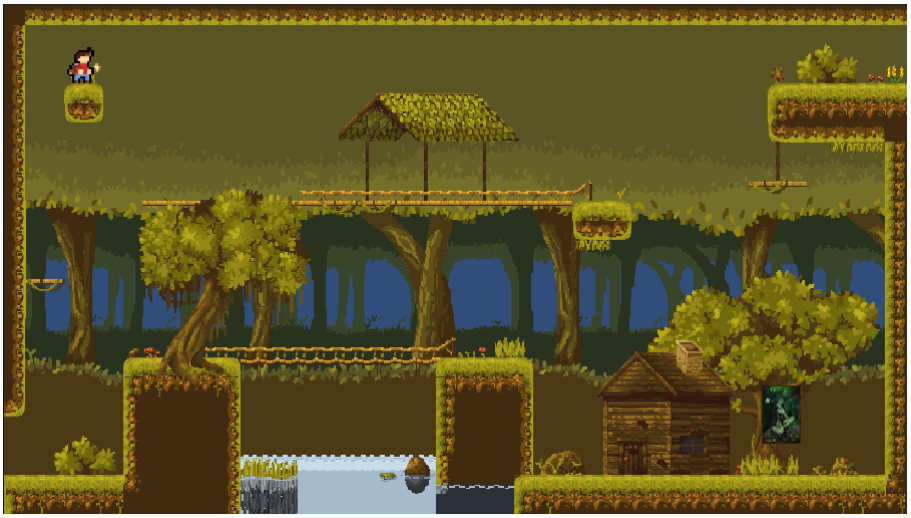

FIGURE 30 – Salle Histoire

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.

FIGURE 31 – Fleur à la fin des salles

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'arrêtantau 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 accroché 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 32 – Tableau de la nymphe

FIGURE 33 – UI du joueur affichant "press E to interact"

FIGURE 34 – 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.

FIGURE 35 – Marchand

Prochaine Soutenance

Pour la prochaine soutenance, je vais devoir m'occuper d'implémenter une salle puzzle supplémentaire, la salle capteur. Pour déverrouiller, la porte le joueur devra activer 6 capteurs en les touchant, l'ordre importera peu. À partir des mécanismes de tous les puzzles déjà développés, je vais m'occuper de créer les bibliothèques des zones 1 et 2. C'est-à-dire 24 salles, 12 pour chaque zone. Quand les bibliothèques seront complètes, je m'attaquerai à la génération aléatoire des salles directement à l'aide de photon pour le mode multijoueur. Avec la génération aléatoire, le joueur va parcourir entre 8 et 12 salles. Les salles générées seront choisies aléatoirement dans la bibliothèque et à partir de 8 salles parcourues, il y aura possibilité de charger la dernière salle de la zone qui mènera à la zone suivante.

Antoine

Début du projet

Mon rôle dans cette première soutenance a été d'implémenter une grande majorité du système économique, les objets et le stress du jeu (la majorité de ces notions ont été réalisées avec la collaboration d'Hugo). Pour cela je me suis 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'ai découvert une « programmation nouvelle » avec de nouvelles notions à connaitre et à maitriser. De plus, Unity étant également tout nouveau pour moi, apprendre ce logiciel a été laborieux au début, mais peu à peu, j'ai maitrisé les notions basiques du logiciel.

Système économique

Dans le but de créer un système économique, nous avions besoin d'implémenter un système de monnaie qui puisse être ramassée et dépensée. Pour ma part, j'ai créé pour cette première soutenance le Coin et la Raspberry.

FIGURE 36 – Différentes Monnaies

). Pour ces deux monnaies, il a fallu également implémenter un système de portemonnaie, qui se rapporte tout simplement au nombre d'élément que nous possédons parmis ces deux catégories. Concernant l'obtention de ces monnaies, elles peuvent être ramassées dans le jeu grâce à leurs deux instances qui leurs sont associées.

FIGURE 37 – Interface du joueur pour les monnaies

Concernant la dépense de ces deux monnaies, je me suis occupé pour cette première soutenance de la création d'un menu marchand pour un PNJ. Pour cela, j'ai récupéré le PNJ créé par Angelina et je lui ai ajouté un menu permettant de vendre des items (la monnaie utilisé pour la vente de ces items étant le coin).

FIGURE 38 – 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.

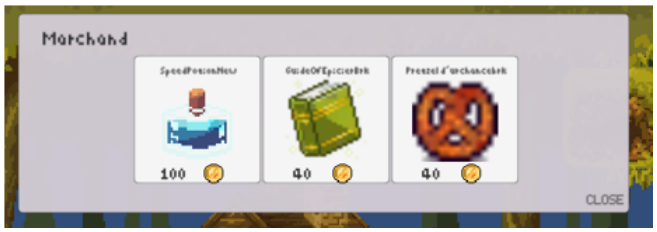

FIGURE 39 – Interface de la vente d'item (Exemple 1)

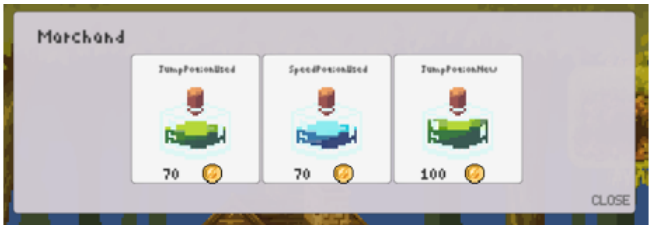

FIGURE 40 – Interface de la vente d'item (Exemple 2)

Concernant ces trois items, j'ai 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'ai 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'ai implémenter, 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 markéting d'un épicier). .

Les objets et le système d'inventaire

Concernant les objets que j'ai implémenté pour cette première soutenance, ils sont au nombre de 4 (3 objets passifs et 1 objet actif). Le premier objet que j'ai 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. Soit 5, 10 ou 15 % de la vitesse de base du joueur. L'ensemble des objets passifs n'est trouvable que chez le marchand. .

FIGURE 41 – Potion de Vitesse

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

FIGURE 42 – Guide de l'épicier

Le dernier objet passif que j'ai implémenté est le Pretzel d'urchance. Cet objet est une sorte d'ange gardien pour le joueur. En effet, si le stress du joueur atteints les 200, et que celui-ci possède un Pretzel d'urchance, sont 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 43 – Pretzel d'urchance

L'ensemble des objets passifs est stocké dans un inventaire Passif qui apparait lorsque nous appuyons sur la touche « i » et qui disparait 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 44 – Inventaire vide et avec objets

L'objet actif que j'ai implémenté pour cette première soutenance est la « Tarte ». En effet, celle-ci permet au joueur, lorsqu'il l'active, de se régénérer instantanément de 10 points de stresse. 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 rendu de soutenance de Hugo.

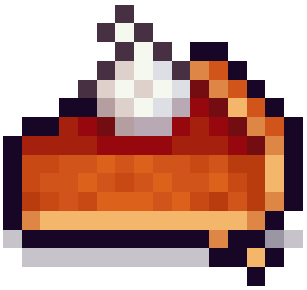

FIGURE 45 – La Tarte

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 disparaitre 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 46 – Inventaire Actif

Système de Stress

Pour finir, la dernière chose que j'ai faite pour cette première soutenance est un système de stress en collaboration avec Hugo. Ce système est assimilable à un système de point 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'ai 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.

Conclusion

Cette première partie du projet nous a permis de nous rendre compte de la difficulté de se lancer dans un projet 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 projet où nous sommes vraiment libres et nous devons nous débrouiller en total 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 deadline, imposé par l'école ou bien entre nous afin de pouvoir avancé efficacement.

Pour conclure, nous pensons être dans les temps par rapport à notre cahier des charges et nous somme plutôt satisfait de notre avancement dans ce projet.