

Cahier de rendu

Menacing Duck Studios

Mai 2025



Table des matières

1. Reprise du Cahier des charges:
 - 1.1. Cahier des charge fonctionnel
 - 1.2. Cahier des charges technique
2. Chronologie du groupe des avancées:
 - 2.1. 1ère soutenance
 - 2.1.1. Unity
 - 2.1.2. Assets
 - 2.2. 2ème soutenance
 - 2.2.1. Unity
 - 2.2.2. Assets
 - 2.3. dernière soutenance
 - 2.3.1. Unity
 - 2.3.2. Assets
 - 2.3.3. communication
3. Récit des réalisations

1 - Reprise du cahier des charges

1.1 - Cahier des charges fonctionnel

L'objectif principal du studio était de créer un univers cohérent et riche, où les joueurs pourraient s'immerger pleinement. Cet univers devait être pensé de manière à offrir une expérience narrative engageante et originale.

Parallèlement, nous souhaitions concevoir un gameplay innovant, qui combine la nostalgie des jeux d'arcade classiques tout en apportant une touche moderne avec des mécaniques de jeu rafraîchissantes et dynamiques.

L'estimation du prix était d'environ 26990€ au total pour réaliser l'ensemble du projet en comptant le matériel et les logiciels.

1.2 - Cahier des charges technique

Le cahier des charges technique à promis que le jeu serait un “Beat them all”, donc un jeu opposant un groupe de joueurs à un nombre important d'ennemis.

L'IA du jeu était prévue de faire de l'attaque, donc que l'IA intégrée dans les monstres puissent être en mesure d'attaquer les joueurs à proximité.

Le jeu était prévu en coopératif, donc plusieurs joueurs se regroupent sous une même bannière pour combattre, et un battle, donc les joueurs doivent combattre, dans notre cas, pour éliminer les monstres.

Le studio a prévu de produire un jeu accessible en lan pour permettre une meilleure expérience locale pour les joueurs.

Le cahier des charges indique que la dimension du jeu serait en 2D, avec des assets fait par le studio.

La musique serait à la charge du studio Menacing Duck.

Les effets spéciaux du jeu seraient aussi à la charge du studio

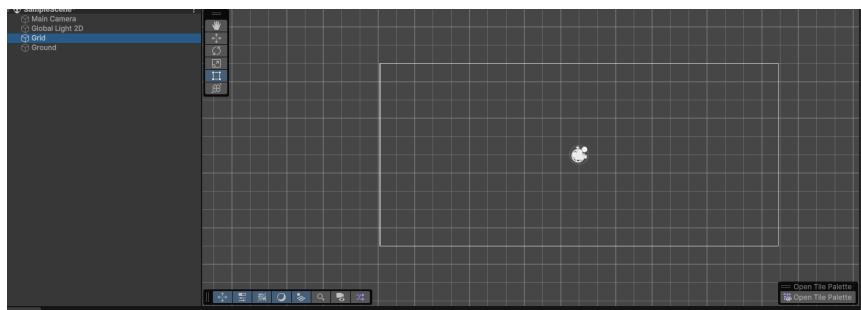
Un site web serait fait sur mesure par l'équipe pour rendre le meilleur produit possible au client

2 - Chronologie du groupe des avancées :

2.1 - 1ère soutenance

2.1.1 - Unity

Le premier objectif que l'on s'est fixé pour le projet était de faire les différents layers de la map tutorielle, pour se faire, on a utilisé un grid, une fonctionnalité de Unity, qui permet de texturer des “tiles”, l’équivalent des cases, pour simplifier la partie design de la map. Le grid simplifie aussi le layering, permettant de faire des maps propres et de poser des objets par dessus le sol sans qu'il n'y ait du vide sous l'objet, rendant la scène plus immersive. Ce système de layers permet aussi de faire une meilleure gestion des collisions avec le joueur, ainsi que de merge les zones de collisions plus facilement, ce qui permet d'obtenir un jeu mieux optimisé.



On a créé deux layers différents au début, le sol et les surfaces collisionnelles, permettant de limiter la zone dans laquelle le joueur a accès, et donc évitant que le joueur ne voit du vide sur les bordures de la map.



Ensuite, il fallait remplir les “tiles” vides des différents layers, pour se faire, Lucie a fourni différents assets d'herbe (cc 2.1.1) pour le sol. Il restait juste deux étapes cruciales pour chaque asset, désactiver la compression automatique des pixel art évitant ainsi des textures floues, et faire le “slicing”, donc découper les zones du pixel art pour sélectionner les parties qui sont intéressantes et le séparer, pour ainsi pouvoir donner l'illusion du naturel, de l'aléatoire pour l'herbe.

Puis on a ajouté le personnage de test dès que la map du tutoriel était dans sa première version



Map tutorielle V1

Le design du personnage, ainsi que la première animation de personnage ont été faits par Lucie (cc 2.1.1).

Puis, pour pouvoir faire les collisions, on a ajouté un rigidbody au personnage. Puis on a fait un script pour gérer les déplacements du personnage, c'est donc sa première version, c'est une version assez rudimentaire avec des déplacements avec des flèches directionnelles, la base du script étant que si l'on relâche la touche, alors on donne au joueur une vitesse de 0 dans la direction de la touche, et si la touche est détectée comme utilisée, alors on donne de la vitesse dans la direction représentée par la touche.

Maintenant il faut que la caméra suive les déplacements du joueur, pour se faire, on a fait un script sur mesure pour rendre les déplacements de la caméra moins rigides (V1) , étant donné que Unity propose une solution à ce problème, cependant les déplacements sont trop directs à notre goût. Ainsi, le script est structuré de telle manière, tout d'abord, on renseigne un joueur que la caméra vise, puis un “follow speed”, donc une vitesse de suivi du joueur, cela permet de ne pas se déplacer instantanément, et enfin, le déplacement de la caméra, pour se faire, on crée un vecteur qui, appliquée à la caméra, donne la position du joueur, et enfin, un “slerp” ou circular linear interpolation, qui permet de rendre les mouvements de la caméra moins directs, rendant les mouvements de la caméra plus fluides.

```
0 references
public class CameraFollow : MonoBehaviour
{
    1 reference
    public float FollowSpeed = 10f;
    2 references
    public Transform target;
    // Start is called once before the first execution of Update after the MonoBehaviour is created
    0 references
    void Start()
    {

    }

    // Update is called once per frame
    0 references
    void Update()
    {
        Vector3 newpos = new Vector3(target.position.x, target.position.y, -10f);
        transform.position = Vector3.Slerp(transform.position, newpos, FollowSpeed*Time.deltaTime);
    }
}
```

Camera Follow V1

Puis on a rajouté notre logo dans le sol, pour se faire, nous avons créé un layer nommé sobrement “Menacing Duck”, qui est par-dessus le sol mais en dessous du joueur. Ainsi, on a obtenu la map V1.

On a aussi créé un site pour le projet, sur celui-ci, on peut télécharger les différents exe des rendus, ainsi que les carnets de rendu, et aussi trouver un peu de lore et même un easter egg, à vous de le trouver!

2.1.2 - Assets

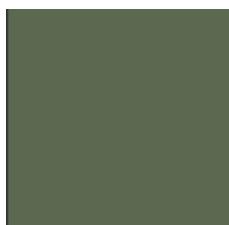
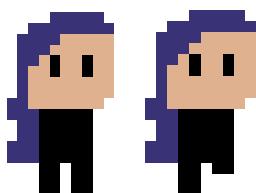
Le tout premier défi pour les assets a été de déterminer l'échelle du jeu. Cela allait également influencer le nombre de détails des sprites.



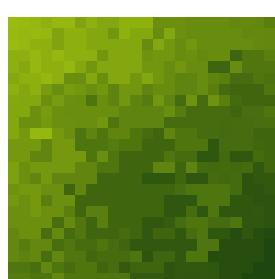
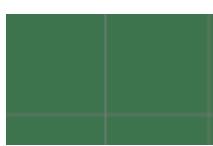
Allions nous faire dans le style des pokémons sur gameboy, (une des premières référence pour nous en terme de design sur les jeux en vu du dessus) ?

Finalement nous avons décidé de voir plus grand, Seulement n'ayant que très peu d'expérience dans le pixel art, les premières idées n'ont été que très peu concluantes...

Mais cela nous a permis d'avoir un sprite test et de commencer à essayer de comprendre comment marchait les animations de marche.



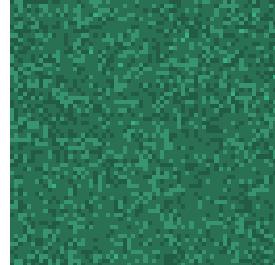
Pour ce qui est du décors, nous avons dû choisir l'aspect du jeu et donc l'intensité des couleurs, la saturation etc... Cela paraît peu pertinent à premier abord mais c'est ce qui allait donner le ton au jeu. Il ne fallait pas que ce soit trop enfantin, joyeux, sans pour autant que ce soit trop sombre.



Nous avons également appris que le détail n'est pas toujours le meilleur choix dans le design d'un jeu, plus un

élément sera détaillé, et plus ce dernier accaparera l'oeil du joueur. Nous voulions donc éviter de surcharger le jeu d'informations inutiles pour qu'il soit le plus agréable à regarder.

Cela peut s'observer sur les jeux plus ou moins connus, comme les jeux nintendo qui ont conservé un style simpliste, ce qui permet d'attirer les joueurs par ce qui est réellement important.



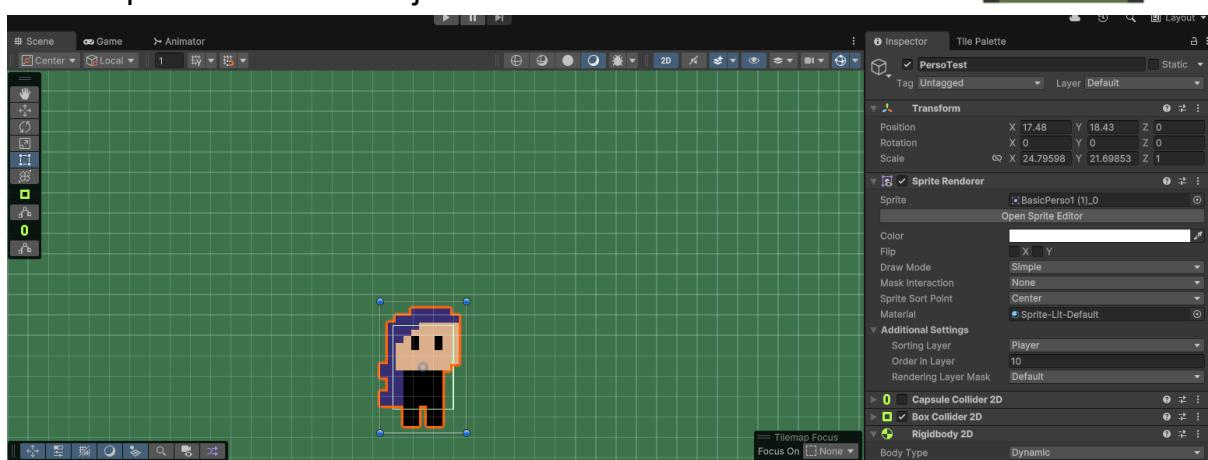
À ce moment, les seuls détails qu'on avait étaient les éléments naturels qui jonchaient le sol.



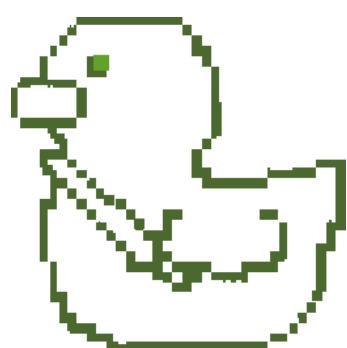
De plus nous avions commencé à réfléchir à l'aspect des personnages jouables, et surtout de la taille



Voici à quoi ressemblait le jeu à ses débuts :



Autre que notre projet, nous avons dû à l'époque décider de l'identité de notre studio. C'est ainsi qu'à été créé le canard de Menacing duck studios



Menacing Duck V1

Et enfin, le logo du projet Rito au lancement, le voici



ProjetRito V1

2.2 - 2ème soutenance

2.2.1 - Unity

On s'est principalement occupé de la scène multijoueur durant cette phase, pour ce faire, on a utilisé Netcode For GameObjects, qui est un package multijoueur de Unity.

Ensuite, dans notre scène principale on a ajouté un empty auquel on y attache un NetworkManager, le cerveau de notre multijoueur, dans le NetworkManager on attache le UnityTransport, c'est comme une autoroute pour les paquets de notre jeu, on y entre l'ip et le port, la vitesse et la taille des paquets etc etc

On a fait une console pour pouvoir faire des tests sur le multijoueur, et plus tard sur les interactions joueur / joueur et monstre / joueur.

Les commandes sont les suivantes:

- HOST, permet de Host une partie sans passer par les menus, existe pour des raisons de test.
- DISCONNECT, permet de ne plus être en mode host, et de fermer la partie.
- CLIENT, permet de rejoindre une partie en localhost

Une map multi de test a été faite pour l'occasion, c'est une map du tutoriel un peu modifiée.

Pour le tutoriel, la map à été mise à jour durant cette période, passant du carré de terre à une vraie map avec des “Props”, des objets avec des collisions ou non qui ajoutent de la vie à la map, utilisant les assets faits (cf 2.2.2), ce qui nous donne cette map :



Scènes/Tutoriel V1

On a créé un nouveau Layer nommé sobrement “Trees”, qui a été fait pour les arbres, étant donné que c'est un layer en dessous de surface mais au-dessus du sol, ça permet de faire des jeux de profondeur avec les arbres et d'avoir des collisions.

Le voici :



Map avec la sélection tree:



Layer Tree Tutoriel

On a aussi ajouté un layer intermédiaire, c'est à dire qu'il se trouve entre le joueur et le sol, permettant de faire des textures supplémentaires par dessus le sol tout en permettant au joueur de se déplacer sur la zone, le voici :



Map avec la sélection Between :



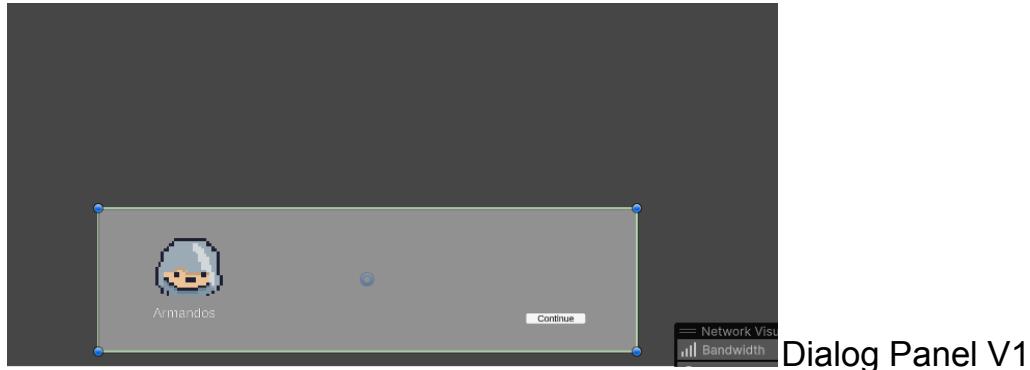
Layer Between Tutoriel

On a aussi implémenté un pnj de test pour le tutoriel, son nom est Armandos, et son rôle est de guider les joueurs dans la suite du tutoriel.

Pour se faire, un Canvas a été ajouté à la scène, nous permettant de design une interface pour l'utilisateur pour qu'il puisse interagir avec le PNJ pour passer au prochain dialogue et aussi pouvoir lire le dialogue, pour se faire,

l'interface est composée d'un panel nous permettant de mettre un fond sur la zone d'échange, une image du PNJ nous permettant de voir à qui on parle, une zone pour le nom, pour savoir de nouveau à qui on parle, un bouton "Continuer" qui nous permet de passer à la prochaine ligne de dialogue, et la zone de dialogue, dans lequel le PNJ parle au joueur et donc communique les informations.

Voici l'UI quand tout est activé :



En jeu, voici le rendu :



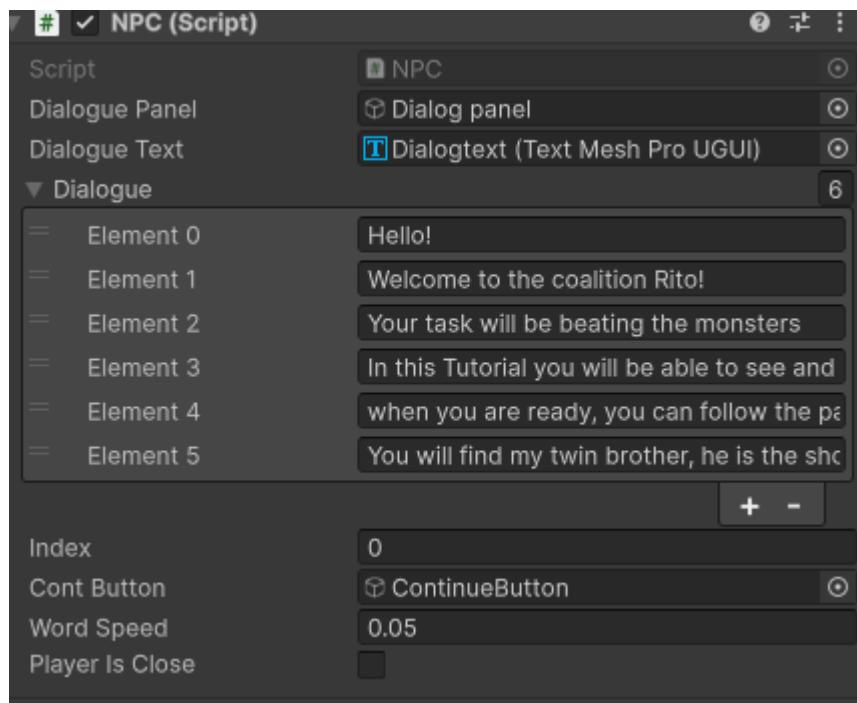
Pour activer cette UI, il faut déjà avoir un PNJ, pour se faire, on a posé un GameObject dans la scène, avec la texture d'Armandos (cf 2.2.2).

Il a été équipé de 2 box colliders 2D, et d'un rigidBody.

Le rigidbody et le premier box collider permet d'avoir une hitbox pour empêcher le joueur de marcher sur le PNJ.

Le deuxième permet de savoir si le joueur est dans la zone de dialogue du PNJ.

Ensuite viens le script du PNJ, nous l'avons intitulé sobrement NPC, il pends en entrée le Panel de dialogue, la zone de texte du dialogue, un array de string qui représente les différentes lignes de dialogue que le NPC donnera au joueur, l'Index du dialogue, il permet de savoir à quelle ligne on est, très pratique pour le débug. Et la vitesse de dialogue.



NPC V1

La vitesse de dialogue est une capacité que nous avons choisi pour notre jeu, en effet, quand on parle on ne dit pas tout d'un coup, mais on parle peu à peu, ainsi, on a décidé que notre PNJ parle peu à peu comme un humain, et donc le texte apparaît lettre par lettre pour améliorer l'immersion de l'utilisateur.

Parlons du script pour les PNJ, tout d'abord, si on rentre dans son collisionneur 2D, le script l'enregistre dans player is close, un booléen qui sera mis à true et si l'on ressort de la zone, alors le script l'enregistre dans le même booléen mais le met à false. De plus, si l'on quitte la zone pendant le dialogue, l'interface disparaît et le dialogue est remis à zéro

voici le code :

```
0 references
private void OnTriggerEnter2D(Collider2D other)
{
    Debug.Log("OnTriggerEnter2D called with: " + other.gameObject.name);

    if (other.CompareTag("Player"))
    {
        ZeroText();
        Debug.Log("Player entered the trigger!");
        PlayerIsClose = true;
    }
}

0 references
private void OnTriggerExit2D(Collider2D other)
{
    Debug.Log("OnTriggerExit2D called with: " + other.gameObject.name);

    if (other.CompareTag("Player"))
    {
        Debug.Log("Player exited the trigger!");
        PlayerIsClose = false;
        StopAllCoroutines();
        ZeroText();
        DialoguePanel.SetActive(false);
        Index = 0;
    }
}
```

Les debug.log servent à debug

Ensute, si le dialogue se joue, le bouton continuer disparaît et quand le dialogue est terminé il apparaît, si on interagit avec le bouton continuer, la zone de texte se remet à zéro et affiche le prochain dialogue jusqu'au bout.

Voici le code pour passer au prochain dialogue:

```
public void NextLine(){
    ContButton.SetActive(false);

    if (Index < Dialogue.Length - 1)
    {
        Index++;
        DialogueText.text = "";
        ZeroText();
        StartCoroutine(Typing());
    }
    else
    {
        ZeroText();
        DialoguePanel.SetActive(false);
        Index = 0;
    }
}
```

Si tous les dialogues ont été joués, alors l'interface disparaît et tout est remis à zéro.

voici le code pour remettre à zéro le dialogue:

```
5 references
public void ZeroText(){
    DialogueText.text = "";
}
```

Il faut que le script vérifie constamment que le joueur essaye d'accéder au PNJ.

Si le joueur est proche et que le joueur utilise la touche interaction, alors le Panel du PNJ est affiché et le bouton continuer est caché, on lance aussi la coroutine typing() qui permet d'écrire caractère par caractère dans la zone de texte.

```
0 references
void Update()
{
    KeyCode interactKey = (KeyCode)System.Enum.Parse(typeof(KeyCode), PlayerPrefs.GetString("interact", "Q"));
    if (Input.GetKeyDown(interactKey) && PlayerIsClose)
    {
        if (DialoguePanel.activeInHierarchy)
        {
            ZeroText();
        }
        else
        {
            DialoguePanel.SetActive(true);
            ContButton.SetActive(false);
            StartCoroutine(Typing());
        }
    }

    if (DialogueText.text == Dialogue[Index])
    {
        ContButton.SetActive(true);
    }
}
```

Voici le code pour typing() :

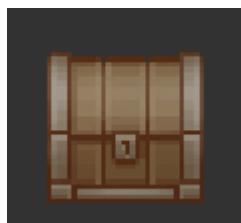
```
2 references
IEnumerator Typing(){
    foreach(char letter in Dialogue[Index].ToCharArray()){
        DialogueText.text += letter;
        yield return new WaitForSeconds(WordSpeed);
    }
}
```

Pour chaque lettre, on l'ajoute à la zone de texte et on continue après “Word Speed” secondes, ce qui donne ce délai dans l'apparition des lettres dans la zone de texte.

2.2.2 - Assets

Après avoir gagné beaucoup d'expérience en pixel art grâce aux nombreuses vidéos et à force d'en faire, nous avons décidé d'améliorer nos assets, et d'en ajouter beaucoup afin de donner plus de vie au jeu. Pour plus de qualité et de réalisme nous avons choisi cette fois de fortement nous inspirer d'items d'autres jeux.

On a produit des design de différents props pour la projet, un coffre fermé et ouvert par exemple, le voici :



Chest1 V6



Chest2 V2

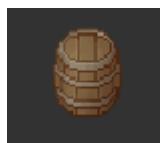
Il nous fallait aussi des boites / des tonneaux pour ajouter à l'esthétique médiévale, les voici :



Storage1 V2



Storage1 V3



Storage2 V3

Les design de coffres ont tous été retenus, les design Storage V2 / Storage V3 ont été retenus pour la soutenance et pour la suite

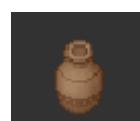
Pour s'entrainer au pixel art, des vases en céramique ont été fait, et ils ont finalement été réutilisés pour le projet



Vase1V2



Vase2V1



Vase3V1

Encore une fois nous avons décidé de changer le style de notre jeu et nous avons refait encore une nouvelle fois de l'herbe, avec cette fois du détail intégrée à l'intérieur des tuiles, nous évitant de mettre les pousses d'herbe à la main.

4 Tuiles différentes ont été fait pour éviter que la redondance ne soit trop facilement visible.



Bien que nous avions déjà des rochers et des buissons, nous savions qu'il manquait encore quelque chose. Quelque chose de plus grand qui rajouteraï encore de la variété végétale : des arbres.



Arbre1V3



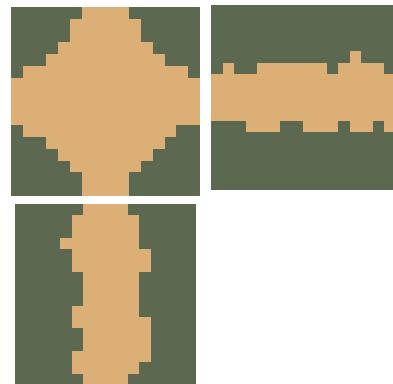
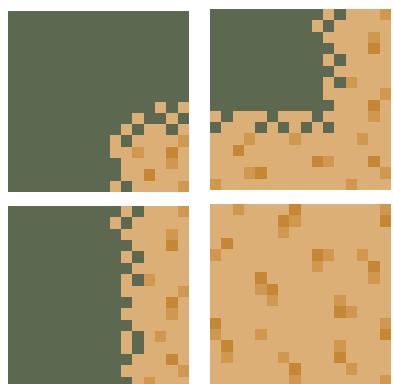
Arbre1V4



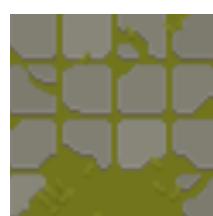
Arbre2V2

Pour une des premières fois nous avons décidé d'ajouter également une version ultérieure d'un sprite qui finalement agit comme un arbre différent.

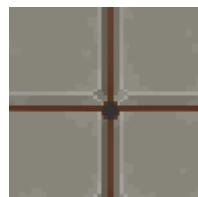
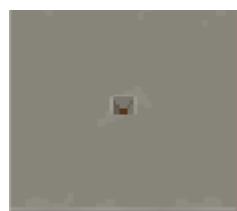
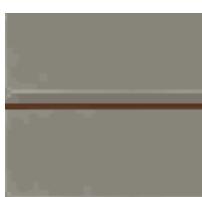
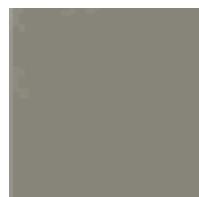
Revenons à l'histoire des détails qui sont là pour attirer l'oeil du joueur.
Nous avions besoin de chemins pour indiquer la direction où il faut aller.
Premièrement nous avions des chemins très basiques



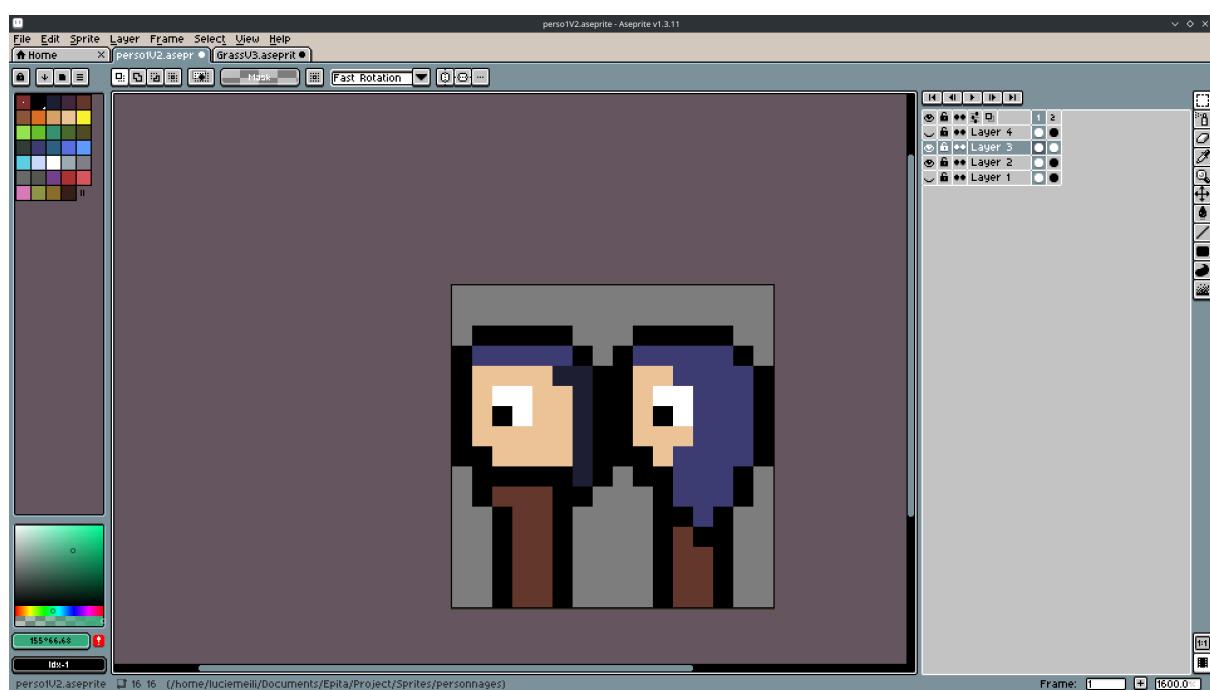
Puis finalement nous avons opté pour un chemin en pierre plus élaboré :



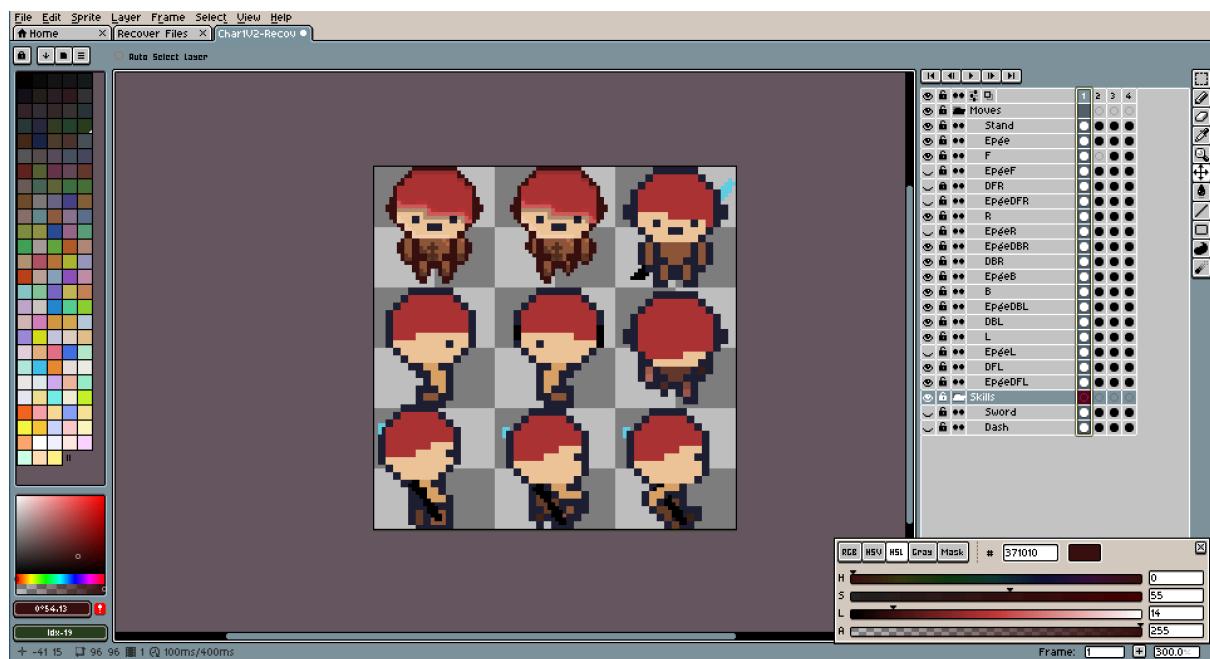
Appréciant le rendu nous avons continué dans le même sens mais avec cette fois de vrais dalles de pierre.



Nous en avons également profité pour changer drastiquement les personnages jouables de Rito Project.

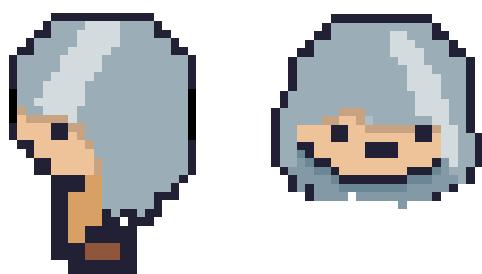


Comme vous pouvez le voir ci dessus, le tout premier personnage a été amélioré, et pourtant il a vite été abandonné au profit du premier personnage officiel de notre jeu : le rogue.



Avec lui ont été testés les 8 directions possibles sur 2 frames chacun. Que ce soit en marche ou immobile, ce qui fait un total de 32 frames pour un même personnage.

Nous avons aussi accueilli notre tout premier pnj, et probablement le dernier, pour l'instant...

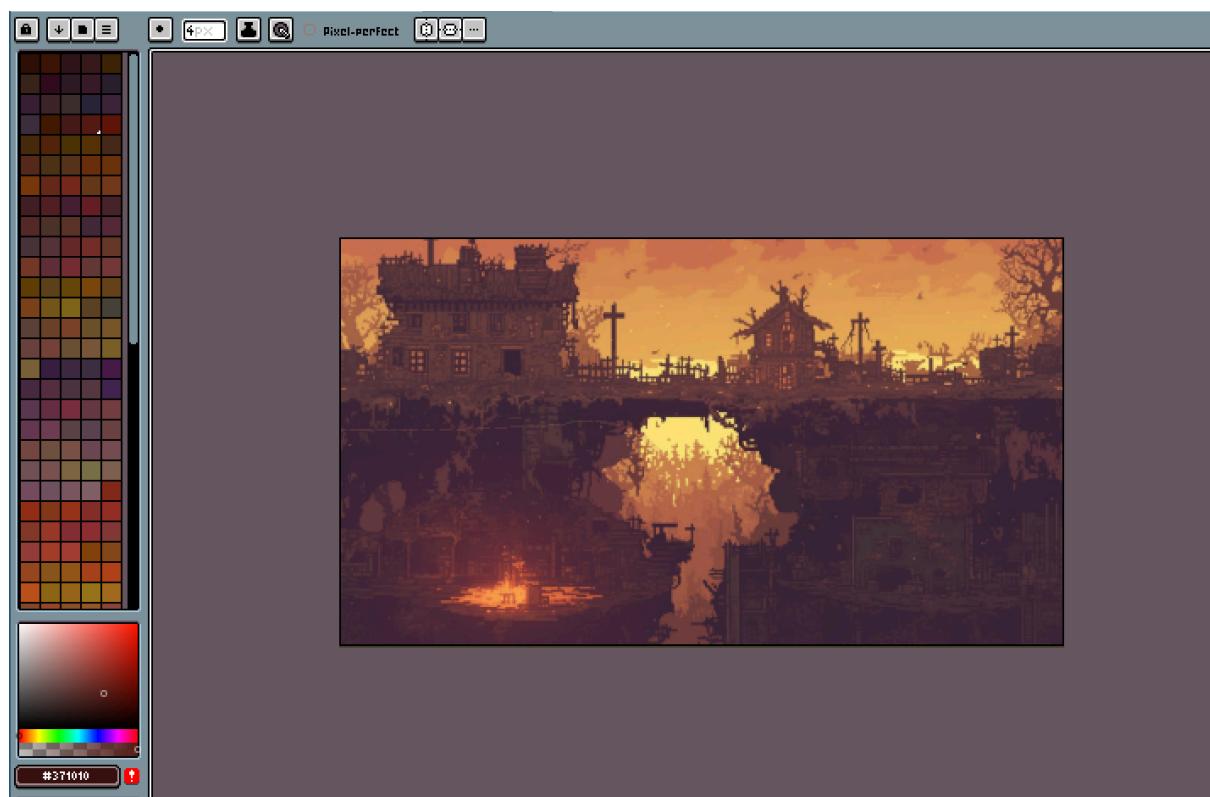


Son nom est Armando

Le logo du jeu a été changé également ;
passant de ça : à ça :



Pour finir sur la partie assets de cette deuxième soutenance, il faut savoir également que nous nous sommes penché sur le menu. Et pour la toute première fois lors du design, nous nous sommes aidé d'une IA générative pour faire l'écran du menu. Bien sûr nous l'avons modifié après coup pour enlever les imperfections, ajouter certaines choses etc....



Et par la même occasion nous avons fait les boutons, le premier menu est fait dans le style médiéval fantasy, tandis que le second est dans un style un peu plus moderne notamment avec les câbles.

Certains de ces boutons ont été quand même modifié après pour mieux correspondre à ce qu'on voulait, mais voici la toute base :



2.3 - Dernière soutenance

2.3.1 - Unity

Pour la dernière soutenance, une zone a été ajoutée au tutoriel, la zone shop, dans celle-ci, on peut retrouver le shopkeeper Armandos, le frère Jumeau de Armandos le premier

PNJ. Voici la zone :

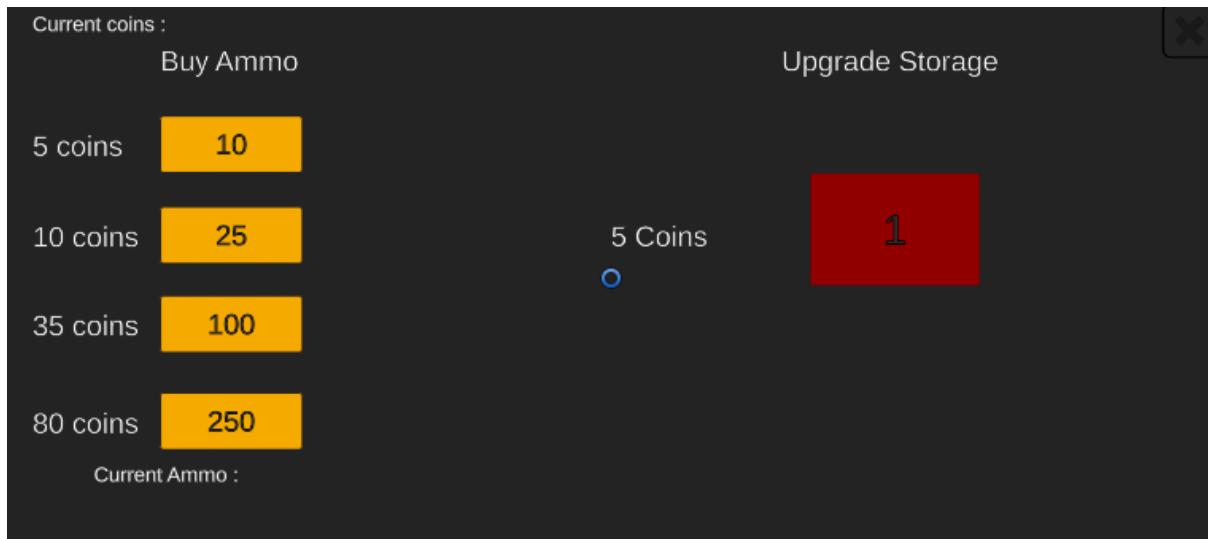


Lui, il s'occupe du magasin. Dans ce magasin, on peut retrouver des améliorations cruciales tout comme des consommables vitaux pour les joueurs, ainsi, pour se familiariser avec, un magasin avec 1000 pièces est à disposition du joueur pour faire des tests économiques pour les stratégies. Les objets que le magasin vends sont du mana en différentes quantités, plus la quantité est grande, plus le prix à l'unité est intéressant, mais le stockage

de mana est limité, pour l'augmenter, il faut payer des améliorations de stockage, et donc augmenter la capacité d'emport de mana / munitions qui sont clés pour utiliser les abilités des personnages.

Armandos le vendeur a un script sur mesure pour la vente, pour se faire, le joueur doit se trouver à portée du vendeur, et interagir avec la touche d'interaction (espace pour le shop), alors l'interface apparaît en fonction de la faction du joueur, si l'on fait partie de la faction médiévale, on obtient le magasin médiéval vendant du mana, cependant, si l'on fait partie de la faction des modernes, on a accès au magasin moderne vendant des munitions.

Voici le shop pour la faction moderne qui utilise des munitions :



Deux zones de textes dynamiques représentent l'argent que le joueur détient et la quantité de munitions. Une autre zone de texte dynamique est celle de l'amélioration de la capacité de stockage, qui change en fonction de niveau, plus le niveau est haut, plus le prix est élevé.

Le texte sur le bouton rouge indique le niveau actuel de stockage.

Le PNJ s'occupe seulement de rediriger les utilisateurs vers leurs shop quand la map charge, et si l'on interagit avec le PNJ tout en étant proche, l'UI shop qui contient MedShop et ModShop sera activée, et étant donné que l'on a activé l'un des deux au début, il nous reste juste à activer / désactiver l'UI shop. Voici la structure



Voici le script derrière le PNJ :

```
0 references
void Start()
{
    if (Player.IsMed)
    {
        MedShop.SetActive(true);
    }
    else
    {
        ModShop.SetActive(true);
    }
}
```

Ce script sert donc à activer le shop de la bonne faction au lancement de la scène.

```
0 references
void Update()
{
    KeyCode interactKey = KeyCode.Space;

    if (Input.GetKeyDown(interactKey) && PlayerIsClose)
    {
        if (Shop.activeInHierarchy)
        {
            Shop.SetActive(false);
        }
        else
        {
            Shop.SetActive(true);
        }
    }
}
```

Ce script sert à savoir si le joueur essaie d'interagir avec le PNJ, et si oui, active l'UI shop.

```
0 references
private void OnTriggerEnter2D(Collider2D other)
{
    if (other.CompareTag("Player"))
    {
        PlayerIsClose = true;
    }
}

0 references
private void OnTriggerExit2D(Collider2D other)
{
```

Cette partie du code sert à savoir si le joueur est rentrée dans la zone de collision et est donc à proximité du vendeur

Maintenant, Il faut savoir que chaque joueur détient un script dénommé Player Eco, permettant de traquer l'argent du joueur en solo, le mana, le niveau de la capacité de stockage et le stockage max de munitions / mana, voici le code :

```
using UnityEngine.Collections;
using UnityEngine;

4 references
public class PlayerEco : MonoBehaviour
{
    31 references
    public int Money;

    13 references
    public int CurLvl;
    1 reference
    public int Hp;

    35 references
    public int Mana;

    31 references
    public int MaxMana;

    2 references
    public bool IsMed;

    0 references
    public PlayerEco(bool ismed){
        Money = 10;
        Hp = 50;
        Mana = 10;
        MaxMana = 20;
        IsMed = ismed;
        CurLvl = 1;
    }
}
```

Ce script sert seulement à stocker les informations, elles seront traitées par d'autres scripts par la suite.

Maintenant que chaque joueur à son économie, on peut implémenter le shop pour pouvoir utiliser son argent. Pour se faire, un script pour chacune des deux factions a été fait, même s'ils se ressemblent grandement.

MedShop, le script du magasin médiéval, a besoin d'un Player Eco, du GameObject Shop, l'UI shop pour pouvoir quitter le shop quand on a fini de faire ses achats, et les Text Mesh Pro des textes dynamiques pour les modifier lors des achats.

Voici le code pour le script :

```

0 references
void Update()
{
    UpdMana();
    UpdMon();
}

0 references
public void Quit()
{
    Shop.SetActive(false);
}

```

A chaque frame de jeu, on met à jour les textes dynamiques qui représentent la quantité de mana et d'argent.

La fonction Quit() permet de quitter le shop, donc de désactiver l'UI.

```

8 references
public void UpdMon(){
    bourse.text = $"Current Money = {Player.Money} coins";
}

5 references
public void UpdMana(){
    manabourse.text = $"Current Mana = {Player.Mana} mana";
}

```

La fonction UpdMon() permet de mettre à jour le texte dynamique représentant l'argent du joueur

La fonction UpdMana() permet de mettre à jour le texte dynamique représentant la quantité de mana que le joueur possède

```

0 references
public void Buy10()
{
    if (Player.Money >= 5 && Player.Mana != Player.MaxMana)
    {
        if (Player.Mana + 10 >= Player.MaxMana)
        {
            Player.Mana = Player.MaxMana;
        }
        else
        {
            Player.Mana += 10;
        }
        Player.Money -= 5;
        UpdMana();
        UpdMon();
    }
}

0 references
public void Buy25(){
    if(Player.Money >= 10 && Player.Mana != Player.MaxMana){
        if(Player.Mana + 25 >= Player.MaxMana){
            Player.Mana = Player.MaxMana;
        }
        else{
            Player.Mana += 25;
        }
        Player.Money -= 10;
        UpdMon();
        UpdMana();
    }
}

```

Buy10() permet d'acheter le pack a 10 mana pour 5 pièces et Buy25() permet d'acheter 25 mana pour 10 pièces.

```
0 references
public void Buy10(){
    if(Player.Money >= 35 && Player.Mana != Player.MaxMana){
        if(Player.Mana + 100 >= Player.MaxMana){
            Player.Mana = Player.MaxMana;
        }
        else{
            Player.Mana += 100;
        }
        Player.Money -= 35;
        UpdMon();
        UpdMana();
    }
}

0 references
public void buy25(){
    if(Player.Money >= 80 && Player.Mana != Player.MaxMana){
        if(Player.Mana + 250 >= Player.MaxMana){
            Player.Mana = Player.MaxMana;
        }
        else{
            Player.Mana += 250;
        }
        Player.Money -= 80;
        UpdMon();
        UpdMana();
    }
}
```

Comme au dessus, mana100() et mana250() servent à acheter 100 et 250 de mana respectivement, il y a une sécurité pour éviter de perdre de l'argent, si l'on est déjà max niveau mana on ne peut plus acheter de mana, il faut aussi faire très attention à notre niveau maximal de mana quand on achète du mana.

```
0 references
public void upgstor(){
    if(Player.CurLvl == 3){
        if(Player.Money >= 35){
            Player.MaxMana = 250;
            Player.Money -= 35;
            Player.CurLvl = 3;
            xpc.text = "Max level reached!";
            upglvl.text = "4";
            UpdMon();
        }
    }
    if(Player.CurLvl == 2){
        if(Player.Money >= 15){
            Player.MaxMana = 100;
            Player.Money -= 15;
            Player.CurLvl = 3;
            xpc.text = "35 Coins";
            upglvl.text = "3";
            UpdMon();
        }
    }
    if (Player.CurLvl == 1)
    {
        if (Player.Money >= 5)
        {
            Player.MaxMana = 50;
            Player.Money -= 5;
            Player.CurLvl = 2;
            xpc.text = "15 Coins";
            upglvl.text = "2";
            UpdMon();
        }
    }
}
```

Upgstor() permet d'améliorer la capacité d'emport de mana, donc le niveau de capacité, ainsi, si l'on est niveau 1, alors l'amélioration coûte 5 pièces mais

par exemple passer au niveau 4 coûte 35 pièces. Le texte dynamique représentant le prix se met à jour à chaque amélioration et le texte dynamique affichant le niveau actuel du stockage sur le bouton se met aussi à jour.

Le script ModShop est exactement le même que le script MedShop à l'exception que les textes dynamiques pour les consommables changent.

Voici l'UI pour le magasin de la faction médiévale :



Dans le tutoriel, un UI nous permettant de fermer l'application a été ajouté, voici sa structure :



Voici l'interface en jeu :



Le script derrière est très simple, si l'on interagit avec le bouton back, alors l'interface est désactivée, et si l'on interagit avec le bouton exit game, le jeu est fermé.

Une refonte totale de la map multijoueur a été faite. Séparée en trois zones principales, la nouvelle map multijoueur est un gigantesque terrain de jeu pour que les joueurs s'amusent. Les trois zones principales sont, le spawn, une plateforme en béton sur laquelle se trouve 5 différentes zones d'apparition, le shop du multijoueur, ainsi que l'accès aux autres zones.

Voici un screenshot de la zone :



La seconde zone est composée de petits bois, de chemins de terres et de ruisseaux, un endroit plein de nature, parfait pour les combats en zone ouverte.

Voici un screen :



Enfin, la dernière zone, le village. Le village est composé de plusieurs places plus ou moins grandes et de ruelles. Une zone dangereuse car il est facile de se faire encercler dans les ruelles étroites.

Voici une image de la zone :



Les personnages ont été implémentés pendant cette période,
On a tout d'abord le rogue, le tout premier personnage type d'un voleur /
assassin appelé la Dague. La personnage détient une capacité dash qui
permet de s'enfuir rapidement en cas de danger ou de surprendre l'ennemi.

Un script stats a été fait pour gérer les joueurs comme les ennemis.
Son rôle est de gérer les statistiques des GameObject sur la scène
Multijoueur et de suivre leur points de vie, leur mana, leur points de vie
maximum et leur mana maximum. Il s'occupe aussi d'appliquer les buffs sur
les joueurs, il prévoit aussi une régénération passive de mana et des effets de
régénération des points de vie. Il gère aussi les cooldowns de ces buffs et de
la durée d'application.

Un script de Team a été fait pour pouvoir faire le pve, ainsi, deux équipes sont créées et peuvent se mettre des dégâts mutuellement, cependant deux GameObjects de la même équipe ne peuvent pas se mettre de dégâts mutuellement, ainsi on évite le team kill et les comportements toxiques envers les autres joueurs.

Une classe entière de script a été créée, les Spells.

Les spells sont les capacités spéciales des personnages, donc on a créé un script pour chaque type de spell.

Tout d'abord les AOE, ou Area of Effect. Ce sont des capacités qui font des dégâts de zone, donc qui font des dégâts à tous ennemis qui se trouvent dans la portée de l'attaque.

Pour continuer, les Attach Spell. Ce sont des capacités qui appliquent des effets aux ennemis, sur une durée plus ou moins longue.

Ensuite, les Dash, ou des scripts de mouvement, ce sont des capacités appliquées directement sur l'utilisateur, et qui permettent une augmentation de la vitesse de mouvement sur une courte durée.

Il y a enfin les Projectile Spell, ce sont des capacités qui permettent d'envoyer des projectiles appliquant des effets ou des dégâts à l'adversaire touché par le dit projectile.

Maintenant, il faut que l'on puisse sélectionner le personnage de notre choix. Pour ce faire, nous avons fait une scène spéciale pour la sélection des personnages, quand on fait Host, on peut choisir notre personnage et jouer, et si on fait Join, alors, le jeu vérifie si il y a une partie en lan, si oui, on est appelé à entrer le code de la partie. Si le code est bon, on se retrouve dans le salon de sélection des personnages, il suffit plus que de sélectionner un personnage et d'appuyer sur le bouton jouer

Voici la scène de sélection en jeu :



On a produit un script pour faire apparaître des monstres, `Spawn Enemy`. Il faut fournir un Prefab, et le script s'occupe de le faire spawner sur l'objet qui a le script. Ainsi, si on fournit un prefab de monstre, l'objet qui a `Spawn Enemy` fait apparaître des monstres.

Voici le code :

```
using UnityEngine;
public class SpawnEnemy : MonoBehaviour
{
    public GameObject Monster;
    public void spawn()
    {
        Instantiate(Monster, transform.position, transform.rotation);
    }
}
```

On a ajouté encore 3 commandes à l'invite de commande du jeu, `Debug Command SHOW_DEBUG`, permet de montrer les stats des personnages, leur niveau de mana, de munitions, de points de vie etc... `Debug Command SPAWN` permet de spawn avec un personnage sur la scène multijoueur, utile pour debug et tester des features

Debug Command <string, string> EDIT permet de changer les valeurs des statistiques d'un joueur / monstre instantanément.

Nous avons également implémenté un système de monstres qui poursuivent les joueurs grâce à un pathfinding dynamique. Les ennemis analysent leur environnement pour se rapprocher intelligemment du joueur.

L'intégration de cette logique en multijoueur local a été un vrai défi, notamment pour que les positions et déplacements des monstres restent cohérents sur toutes les machines.

À cela s'ajoute un système de phases de jeu où les ennemis apparaissent par vagues successives. Un des enjeux a été de gérer la transition entre ces phases tout en permettant aux joueurs de mourir et de respawn au cours d'une vague, ce qui a nécessité une gestion précise des états des joueurs et des timers individuels.

Un autre aspect crucial du gameplay est le système de capture de drapeau : si trop de monstres restent à proximité de celui-ci, une jauge se remplit progressivement, et si elle atteint 100%, la partie est perdue. Cela force les joueurs à se positionner stratégiquement pour défendre la zone.

L'ensemble de ces mécaniques (pathfinding, phases, respawn et capture) a demandé une coordination rigoureuse entre la logique de gameplay locale et le système réseau, afin d'assurer une expérience fluide et synchronisée.

2.3.2 - Assets

Avec la deuxième soutenance, nous avons freiné sur la conception des décors de la map.

Nous avons notamment ajouté un spawn avec les mêmes la même colorimétrie des pierres utilisés auparavant



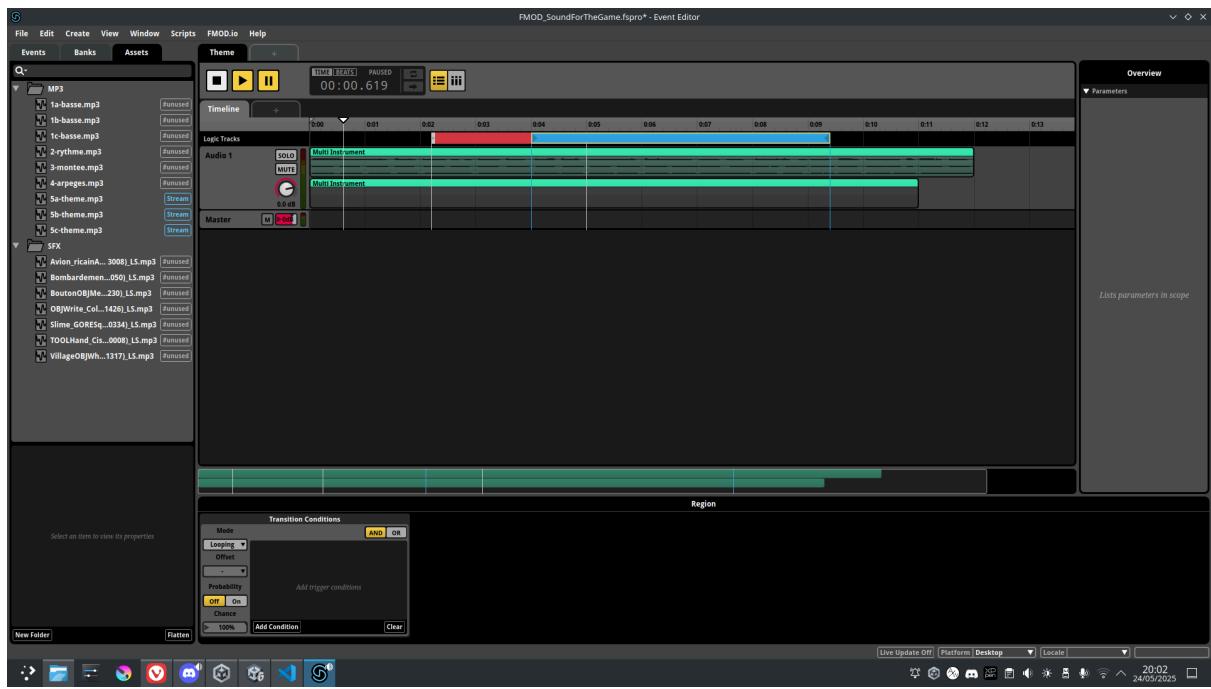


Et après une remarque pertinente nous n'avions pas cette marque de "post apocalyptique" pour faire référence au lore du monde. Nous avons décidé d'ajouter au moins une ruine pensant en faire d'autres si on avait le temps, ce qui finalement ne s'est pas fait

Avec de l'aide extérieure nous avons fait cette statue aussi qui ajoute encore plus du contexte au monde.
C'est, selon nous, le plus bel asset qu'on a pu faire.



Nous avons fait de la musique en 8 bits propre au jeu et nous souhaitions l'implémenter via FMOD un package hautement compatible avec Unity très utile pour faire la musique des jeux, les sons d'ambiance, les bruitages etc..



Seulement le package étant très lourd, il a complètement détruit le git lors du merge entre les branches ce qui nous a valu de devoir faire des reset du git l'abîmant ainsi.

Il a donc été décidé d'abandonner la musique pour l'instant.

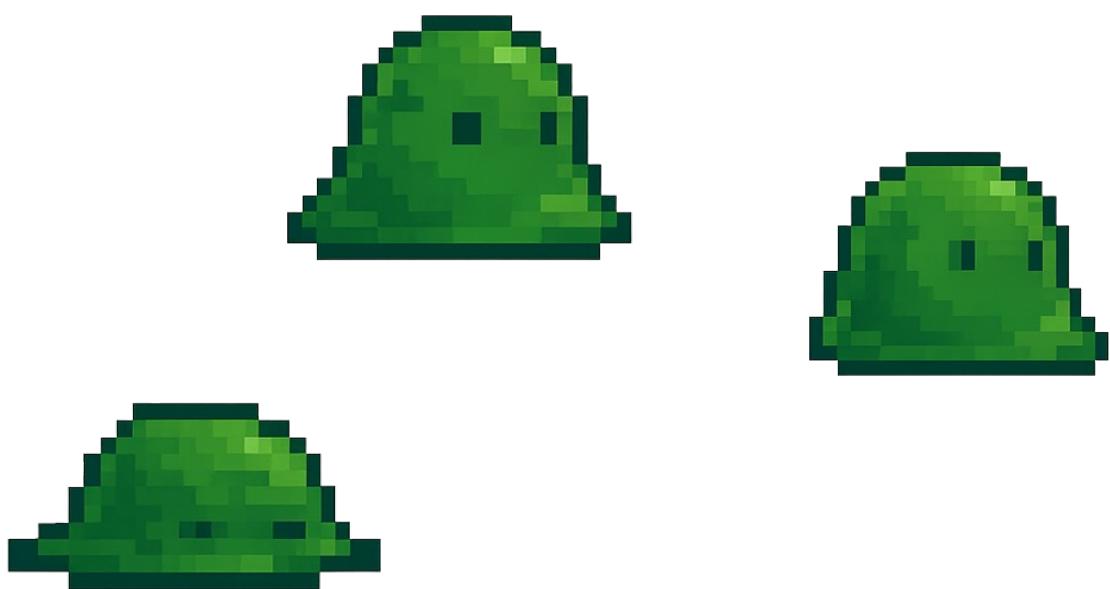
Des bruitages ont été préparés aussi pour chacun des monstres.

Nous avons aussi implémenté le premier monstre :
la chauve sourie, animal mutant qui va vous suivre jusqu'au bout rien que
pour vous sucer le sang. C'est le monstre le plus commun du jeu
Un bruitage de papier était prévu pour accompagner ses battements d'ailes



D'autres monstres sont en cours de développement comme le slime, un amas de radioactivité qui laisse une traînée gluante ralentissant le joueur quand ce dernier marche dessus.

Un bruitage de boyaux était prévu pour le côté gluant de ce monstre.



Le prochain monstre sur la liste était la tortue, un énorme monstre lent mais rempli de point de vie voué à mettre une pression au joueur l'obligeant à sans arrêt sans bouger.

Un bruitage de coquillage avec des pas lourds était prévu pour ce monstre en rapport à sa carapace.



Ensuite la grenouille bombardière était prévu pour mettre de la pression au joueur à distance.

Elle lance de grandes boules de feu du fond de sa gorge bombardant ainsi le personnage joueur.

Un bruitage de feutre sur papier était prévu pour ce monstre, car cela pouvait s'apparenter à un croassement rauque.



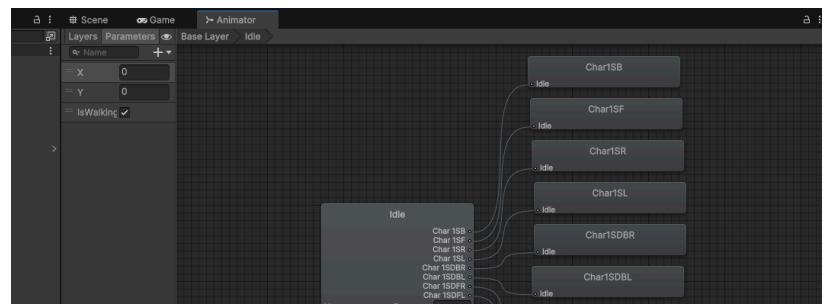
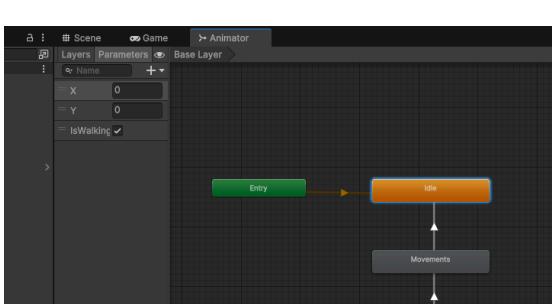
Nous nous devons de rappeler que chaque sprite et son son sur le projet a été fait par nos soins (à part certains bruitages de monstre)

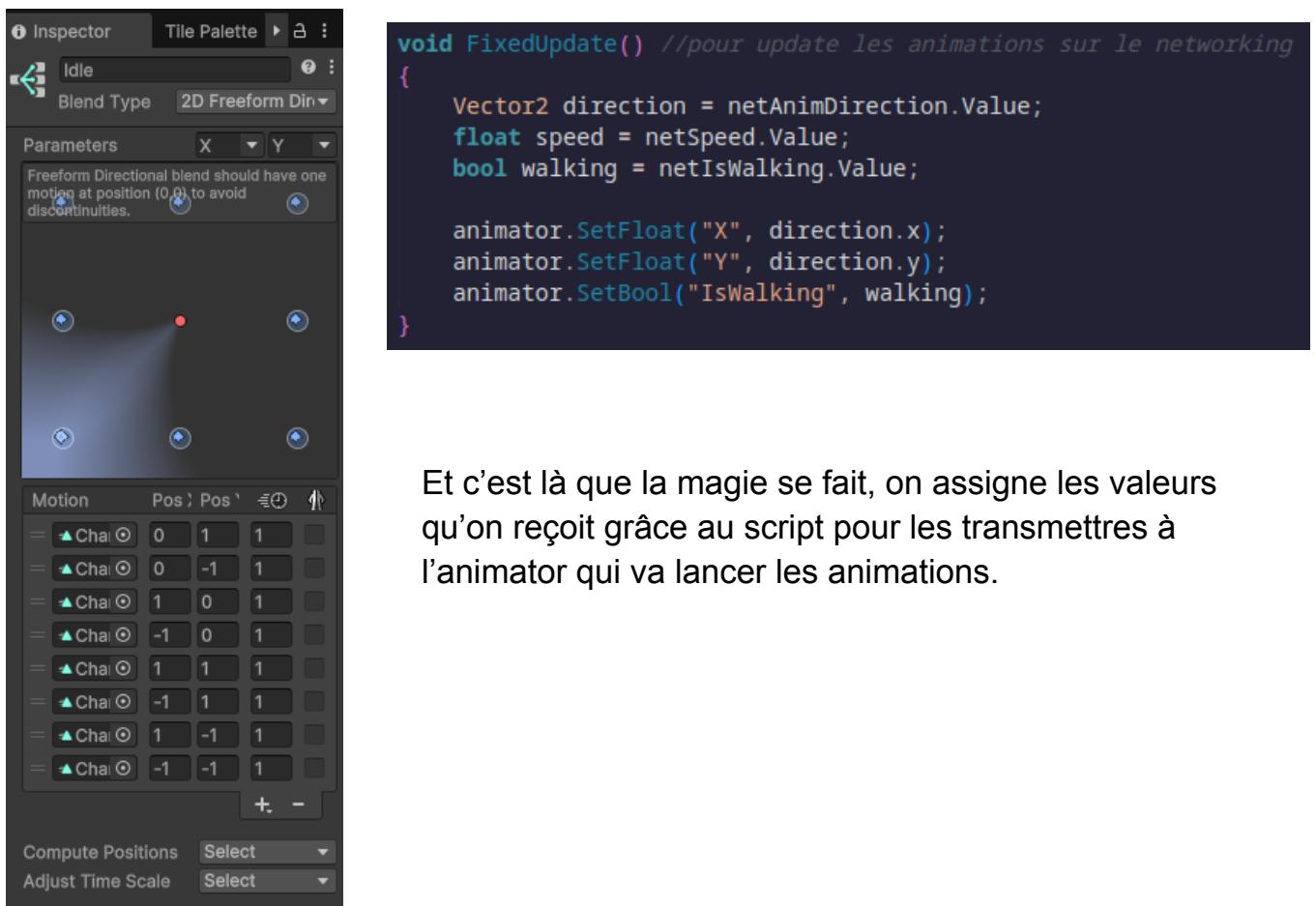
Ainsi, les esquisses de monstres ont bien eux aussi été faites par notre équipe.

Mais la grosse partie est sans nul doute les 5 personnages jouables. Toutes les animations des personnages jouables ont été fait grâce à l'Animator intégré à Unity

Chaque personnage a deux BlendTree principaux, un pour l'animation quand le personnage ne marche pas qui est l'état par défaut, et un pour l'animation de marche qui s'active quand la valeur booléenne "IsWalking" est à true (indiquée grâce au script attaché).

Et chaque BlendTree s'occupe de départager quel animation va être joué selon la direction du personnage dans l'axe X et Y.





Et c'est là que la magie se fait, on assigne les valeurs qu'on reçoit grâce au script pour les transmettre à l'Animator qui va lancer les animations.

Enfin pour finir voici les 5 différents personnages jouables.
Cette partie va permettre de faire une transition au niveau du lore du monde.

tout en montrant deux frames du personnage, son arme et son HUD pour donner une prévisualisation de comment ce sera dans le jeu.

La barre verte qui se teint de rouge selon l'état du joueur est la barre de point de vie,

la barre bleue, est ce qui permet aux humains mutants d'utiliser leur magie, et la barre rouge représente la batterie, l'équivalent du mana pour les humains des bunkers. Cette dernière leur permet de fabriquer toute sorte de chose, et surtout ce qui nécessite des explosions plus ou moins grosses.

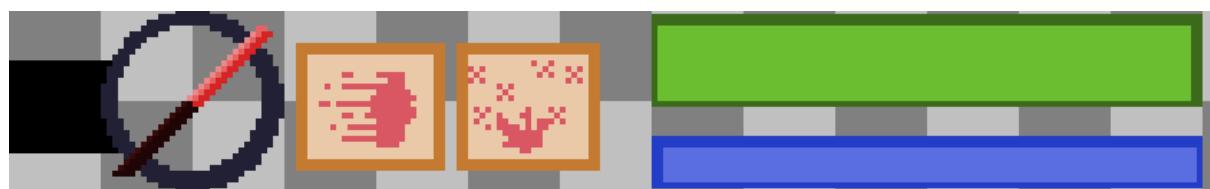
Déjà nous avons fait une nette amélioration du premier personnage - le rogue - aussi surnommé dans le jeu "la dague" qui vient donc du peuple medieval fantasy.

Il est rapide et précis, très bon pour faire des dégâts sur la durée sans mettre de côté la survie.

Il a toujours été très curieux et très farceur malgré ce décor apocalyptique. C'est toujours vers lui qu'on se tourne quand il faut aller chasser des monstres dur à attraper.

Il a une petite soeur qui compte énormément pour lui, et c'est notamment ça qui lui donne la force de se battre chaque jour.

Il utilise principalement sa lame de feu que sa grand mère avait confectionné à l'aide de sa magie de vitesse, il peut dash et assommer ses ennemis pendant un court instant.



Pour le deuxième personnage, le sniper Simo, nous avons tenu à faire référence au tout premier perso qui a été créé, nous l'avons donc encore amélioré une troisième fois en quelque sorte, si ce n'est pas refait.

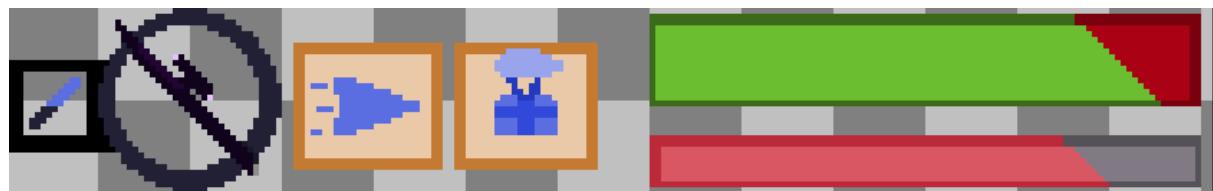
En tant que Sniper, Simo sait tirer de loin des balles dévastatrices, très peu mobiles, mais d'une grande puissance.

D'une nature aussi très silencieuse, c'est une personne qui fait peur autant au premier coup d'œil que pour les personnes les plus proches.

Et pourtant, c'est peut-être le plus humain de tous....

Il utilise principalement son sniper, personne ne sait d'où vient l'arme et beaucoup la convoite à cause de sa puissance. Cependant il utilise également un petit couteau pour les combats plus rapprochés.

Avec les balles qu'il confectionne de lui même, Simo peut charger son tire qui fera plus ou moins de dégât selon le temps qu'il a passé à charger ce dernier, et il peut tirer une balle spécial pour son sniper pour indiquer qu'il est en difficulté. Ses seuls amis ne sont jamais très loin pour lui apporter des munitions n'importe où n'importe quand.



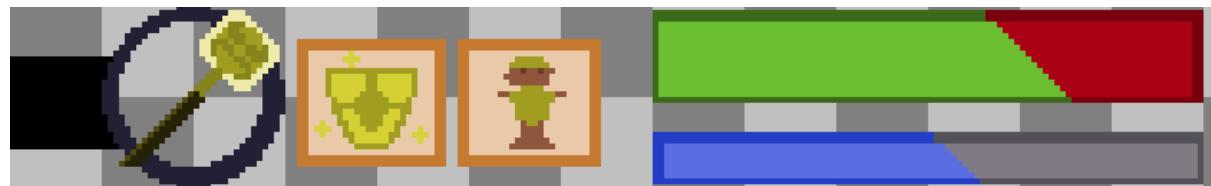
Le troisième personnage c'est Pierre le paladin, il a beaucoup de point de vie et de défense. Jusqu'à ce qu'ils rencontrent les anciens humains qui vivaient dans les bunker durant 100 années, il était persuadé que les vestiges anciens sont arrivés avec des dieux qui habitaient ces terres.

Peu sont religieux dans ce monde, mais lorsqu'il a commencé à prier devant ce qu'il pensait être un objet divin (une montre fossilisée bloquée dans du béton) un dieu extérieur qui passait par là et qui s'ennuyait l'a entendu et l'a pourvu d'un grand pouvoir voué à le protéger.

Il se balade toujours avec un énorme marteau qu'il a modifié depuis la rencontre de ce dieu pour paraître "plus lumineux". Il s'en sert pour intimider les gens qui lui veulent du mal.

Il peut donc augmenter sa défense, aussi étant quelqu'un de très créatif, il a toujours de quoi détourner l'attaque des monstres sur autre chose que lui.

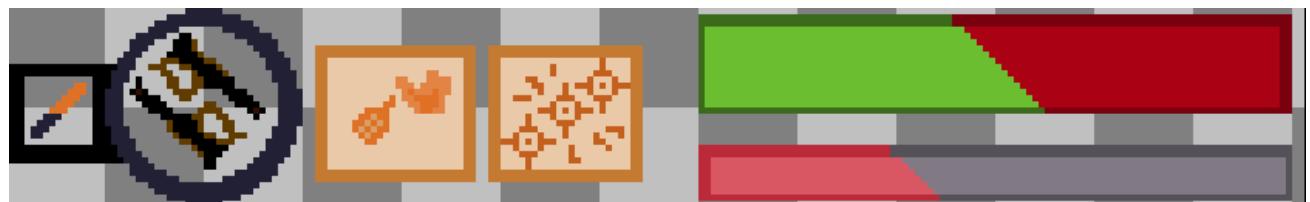
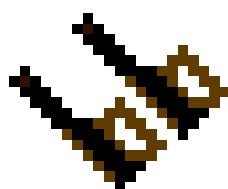
Il peut détourner ainsi l'aggro des monstres.



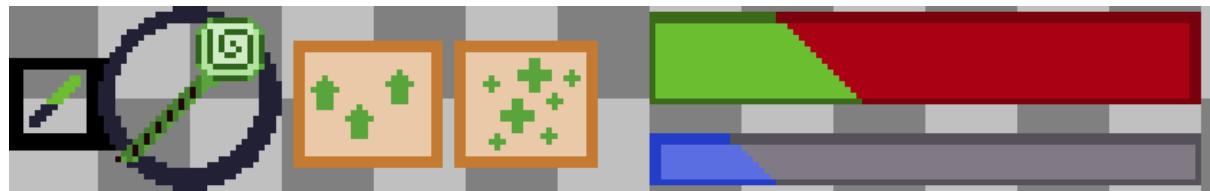
Mason c'est le 4eme personnage, il n'est mauvais en rien, il a passé sa vie dans les bunkers à s'entraîner pour ressembler au maximum à la photo d'un bodybuilder qui posait dans une très ancienne pub de film (avant la guerre nucléaire) qui a toujours été dans sa famille. Il a également toujours cherché à reproduire exactement les mêmes armes que ce dernier tenait.

Même si ce n'est pas parfait les deux petites mitrailleuses marchent, il les utilise donc, les préférant nettement au petit couteau qui l'accompagne pour les combats au corps à corps.

Amateur des explosions, il adore la poudre noir, et en fait des grenades qu'il s'ammuse à lancer. C'est donc sa première compétence, et aussi il peut envoyer une grosse grenade très spéciale qui se fragmente en plusieurs morceaux qui explosent tous. À chaque lancer de cette dernière il aime crier "frappe aérienne" ce qui a toujours déconcerté les ancêtres du bunker chez qui ça leur faisait remonter de mauvais souvenirs.



Enfin, le dernier personnage est la personne la plus terrifiante du jeu. Mais c'est également celle qui va procurer les meilleurs soins. Elle sourit tellement souvent que les gens se demandent si elle ne cache pas des pensées sombres derrière. Depuis petite, elle ramasse un peu tout par terre, donc notamment des choses qui viennent des vestiges humains. Et avec ça elle s'est toujours amusée à construire des bâtons de toute forme. Un jour, alors qu'elle a construit un bâton, elle s'est aperçue que ce dernier amplifie ses pouvoirs régénératrices, c'était grâce à du cuivre présent dedans. Mais cela elle ne le savait pas. Grâce en partie à ce bâton, ses pouvoirs permettent de revigorir et de donner de la force à ses alliés, mais elle peut naturellement les soigner. Sa famille n'arrête pas de parler d'elle, tellement que ça lui met une énorme pression au quotidien. Ensuite, il n'y avait pas non plus besoin de sa famille pour qu'elle ait cette pression depuis qu'elle a fait repousser le bras de quelqu'un du village.



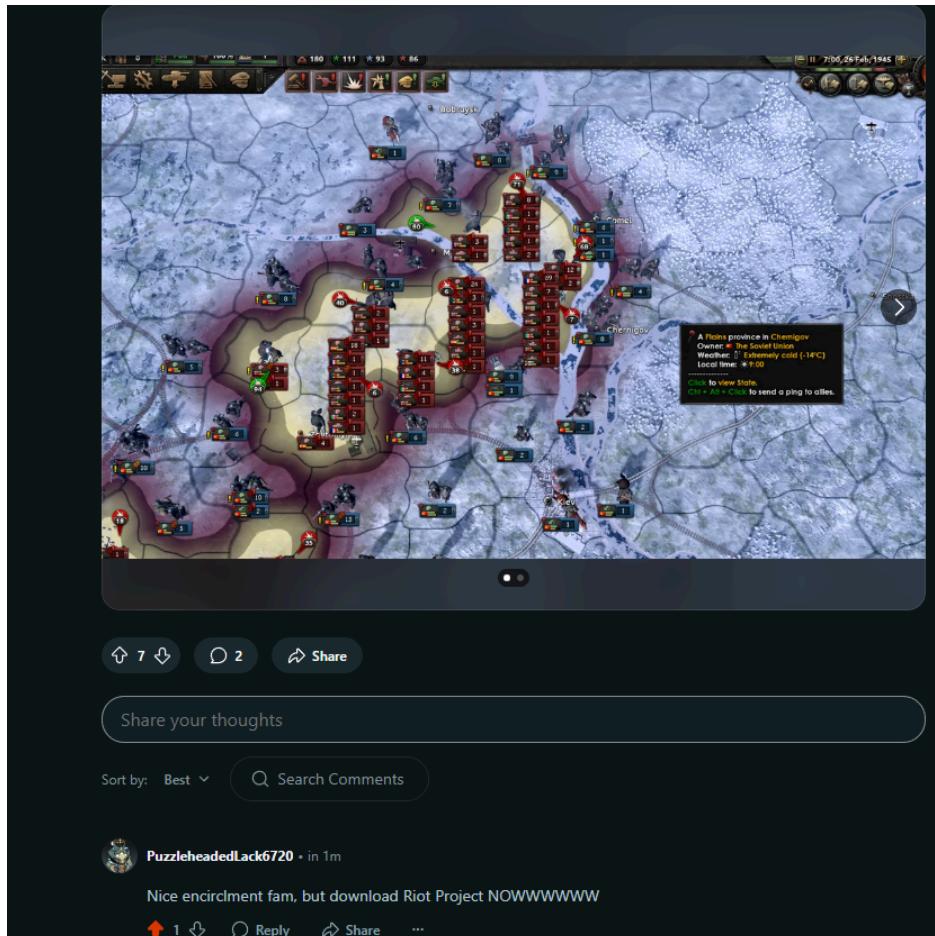
LORE DU JEU :

Selon les anciens, il y a très longtemps, une guerre d'ego entre un groupe de pays a éclaté. Cela a vite évolué en guerre nucléaire, ce qui a décimé la surface de la Terre, la rendant invivable après les atrocités qui s'y sont déroulées. Malheureusement, seule une poignée de personnes a eu le temps et la chance de se réfugier dans un bunker souterrain, les places étant limitées. Selon eux, ce fut la deuxième catastrophe planétaire après l'extinction des dinosaures, mais cette fois causée par la bêtise humaine. Depuis, cent ans et des générations ont passé. Ils ont creusé de grandes galeries, appris à survivre dans les entrailles de la Terre en utilisant les minéraux qu'offrait l'environnement, d'abord pour en faire des armes à feu aux balles perçantes, mais aussi pour fabriquer des outils. Ils ont vite appris à récolter et à filtrer l'eau pour éviter les maladies qui avaient décimé la moitié des rares survivants à l'époque. Ils ont un régime à base de champignons et de viande de rat géant (qui est apparu sans aucun doute avec les mutations dues à la surface) mais aussi à base d'un nouveau type de salade fade, possédant des propriétés médicinales et ne poussant que près des lacs souterrains.

Les accès à la surface sont interdits par les anciens, ainsi, plus personne ne sait rien de ce qu'elle est devenue. Tout ce qu'ils ont, ce sont des vieilles photos conservées par les anciens avant leur mort. Beaucoup d'enfants rêvaient d'aller à la surface, de partir à l'aventure, mais lorsqu'un jour un de ces groupes d'amis a réussi à contourner les interdictions, ils ont à peine sorti leur tête qu'un des membres s'est fait enlever par un monstre sanguinaire. Ce fut un traumatisme pour le peuple souterrain tout entier, et tout le monde comprit qu'au grand jamais on ne devait retourner à la surface. Le problème est que leur population ne cessait d'augmenter à une vitesse alarmante, et les humains de la surface ont dû rapidement créer des groupes d'extermination. Mais plus le temps passait, plus cela devenait difficile. C'est alors qu'un groupe d'éclaireurs envoyé pour espionner l'activité de ces monstres est tombé sur une trappe mystérieuse menant à l'un des bunkers dont parlaient les récits les plus anciens. C'est ainsi qu'ils se sont retrouvés face à face avec des humains vivant sous terre. Les deux peuples, bien que méfiants l'un vis-à-vis de l'autre au début, ont vite marché côte à côte. Les uns étaient contents de pouvoir enfin découvrir la surface, et les autres d'avoir des alliés de taille pour le combat contre les monstres. Peut-être que cette collaboration permettra de regagner la planète et d'y faire prospérer la paix...

2.3.3 - Communication

Nous avons fait de la communication, on a parlé du projet rito sur internet, voici un screen de publicité sur un reddit Hoi4 (un jeu de stratégie donc c'est de la publicité ciblée)



Nous avons fait des affiches pour promouvoir notre jeu, à moitié fait par IA et à moitié modifié par nous avec le QR code qui renvoie les personnes sur notre site.



3 - Récit des Réalisations

Mathias :

En tant que premier projet, j'ai apprécié le faire, même s'il y avait des moments plus compliqués. Après tout, c'est dans les moments comme juste avant les soutenances ou l'on se demande si l'on aurait pas pu faire plus, si le script qui jusque là marchait parfaitement, n'allait pas fonctionner.

Ces moments quand on a un problème et on se demande quelles sont les causes, par exemple, j'ai cherché pendant une matinée entière pourquoi mes boutons d'UI ne fonctionnaient pas sur mon tutoriel, juste pour me rendre compte qu'il y avait un UI invisible par dessus, que j'avais moi-même placé, que j'avais complètement oublié en 15 minutes.

C'est cette satisfaction de résoudre des problèmes compliqués, de réussir après tant d'échecs, qui fait la beauté de coder.

Je me suis personnellement rendu compte durant ce projet que faire des erreurs est le meilleur moyen d'apprendre, que sans mes après midi sans fin de git merge conflict, je n'aurais pas appris comment marche réellement github.

C'est ce projet, cette envie de résoudre l'erreur qui m'a poussé à en apprendre tant sur plein de domaines différents.

Par exemple, je pensais que le pixel art n'était pas fait pour moi, je ne sais pas dessiner et je me suis dit que ça allait être la même chose, mais au contraire, j'ai adoré faire les design pour les maisons alsaciennes sur la map par exemple.

j'ai appris à travailler en équipe, moi qui avait une hantise de travailler en groupe, j'ai adoré travailler.

Mon seul mot de fin étant que même si github a décidé d'imposer dans mes bras à 20h17 alors que le dernier rendu est à 21h, que tant de mon travail a disparu, je ne regrette rien. Je n'ai aucun remord, j'ai tout donné, et ce n'est pas la note qui m'apportera le plus, mais l'expérience que j'ai acquise.

Merci de nous avoir proposé ce projet !

Lucie :

La difficulté pour moi a été de se coordonner suffisamment et de communiquer au maximum.

L'organisation a également été un point central du projet, c'était le premier projet de jeu vidéo de cette ampleur pour moi et pour le groupe donc nous n'avions aucune idée de comment nous organiser exactement et cela a retardé l'avancé du projet au début, mais je sens que durant cette année j'ai tout appris pour être plus efficace à l'avenir.

J'aurai beaucoup aimé avancer encore bien plus sur ce projet, j'admet que nous nous sommes lancé dedans avec plein d'étoiles dans les yeux.

Ce dernier mériterait sans aucun doute qu'on y passe encore énormément d'heures. Ce fut d'ailleurs particulièrement difficile d'allier le projet avec les révisions des autres matières, surtout vers la fin.

Cependant, créer Rito Project m'a donné plus envie que jamais de faire d'autres jeux, d'y ajouter mon style propre et de le voir évoluer avec le temps. L'ambiance dans l'équipe a été globalement assez bonne durant l'année, ce fut un plaisir d'y participer. Un plaisir stressant, certes, mais un plaisir.

Louis :

Durant ce projet, j'ai vécu beaucoup de moments inoubliables. Entre les réunions présentielles, les moments où j'étais coincé sur une erreur, les moments où j'avais l'impression d'avoir codé en une heure un script qui aurait facilement pu prendre plus d'une journée. Ce projet était une première pour moi, c'était une expérience bizarre, mais pas dans le mauvais sens. J'ai appris tant durant ce projet, j'ai même l'impression d'avoir laisser mon impulsivité du début d'année de côté pour devenir plus calme. Ce projet m'a aidé à me définir intérieurement pour devenir une meilleure personne, une personne que j'aspirais à être.

**Merci pour avoir lu et de nous avoir proposé ce projet
Cordialement l'équipe Menacing Duck Studio**