



DEV&CO

RAPPORT DE SOUTENANCE 2

Objectif Lune !

Objectif Lune !



melody.huang
chloe.eap

hengrui.ye
nael.hulice-mencle

13/03/2025

Table des matières

1	Introduction	2
2	Rappels de la soutenance 1	3
2.1	Rappel des tâches réalisées	3
2.1.1	Hengrui : Code du personnage, Multijoueur et Réseau	3
2.1.2	Mélody : Site internet et Niveaux	4
2.1.3	Chloé : Graphisme et Bunker	4
2.1.4	Nael : Son et IA	4
2.2	Rappel des modifications apportées au jeu	5
3	Avancement du projet	5
3.1	Réalisation des tâches	5
3.1.1	Hengrui	5
3.1.2	Mélody	9
3.1.3	Chloé	15
3.1.4	Nael	19
3.2	Nouvelles modifications apportées au jeu	20
4	Problèmes rencontrés	21
5	Conclusion	21
6	Source	22

1 Introduction

Pour rappel, “Objectif Lune!” est un jeu de plateforme et d'aventure en 2D en C#, développé sur Unity. Il est réalisé par l'entreprise Dev&Co, qui est composée de Mélody HUANG, Chloé EAP, Hengrui YE et Nael HULIC-MENCLE en E2.

Le but de ce rapport de soutenance est de montrer notre avancement sur le projet, nos difficultés et la répartition des tâches.

L'histoire d’”Objectif Lune!” se déroule en 2550. Notre héros reçoit alors un message de son père seulement composé d'une position géographique. Curieux, le personnage principal décide de s'y rendre. Son père lui aurait-il préparé une surprise après toutes ces années d'absence ? Arrivé sur place, notre héros n'y trouva rien d'autre qu'une mystérieuse trappe. “Où peut-elle bien mener ?” N'ayant à peine le temps de se poser cette question qu'une énorme explosion retentit. Pris de peur et de surprise, notre héros se précipite à l'intérieur de la trappe. La nouvelle pièce était plongée dans le noir complet jusqu'à ce qu'une vidéo se lance. C'est une vidéo de la part de son père lui indiquant que, durant ces dernières années, ce dernier était membre d'une équipe de chercheurs dans le nucléaire dont l'expérience ratée a provoqué l'explosion. L'enregistrement continue son cours en expliquant que les survivants, transformés en mutants, errent à l'extérieur et que notre héros doit maintenant construire une fusée pour s'échapper vers la Lune. Heureusement pour notre personnage principal, le bunker contient tout le nécessaire pour survivre et se préparer : de la nourriture, de quoi se défendre, et bien d'autres choses à découvrir.

Pour rappel, chaque tâche dispose d'un responsable et d'un suppléant qui aidera et complètera la tâche. Voici le tableau des tâches :

Noms/Tâches	Mélody HUANG	Chloé EAP	Hengrui YE	Nael HULICE-MENCLE
Graphisme et Jaquette	Suppléant	Responsable		
Base du jeu	Suppléant		Responsable	
Son			Suppléant	Responsable
Code du personnage			Responsable	Suppléant
Code des niveaux	Responsable	Suppléant		
Code du bunker		Responsable		Suppléant
Site	Responsable			Suppléant
Intelligence Artificielle			Suppléant	Responsable
Multijoueur		Suppléant	Responsable	
Réseau	Responsable		Suppléant	

Pour rappel, voici le tableau détaillé de l'avancement qui a été convenu. A chaque soutenance,

un objectif est fixé afin de terminer le projet à temps.

Noms/Tâches	Soutenance méthodologie	Soutenance 1	Soutenance 2	Soutenance 3
Graphisme et jaquette	20%	40%	60%	100%
Base du jeu	20%	50%	80%	100%
Son	0%	40%	100%	100%
Code du personnage	20%	50%	80%	100%
Code des niveaux	5%	20%	70%	100%
Code du bunker	10%	40%	70%	100%
Site	0%	30%	50%	100%
Intelligence Artificielle	0%	20%	50%	100%
Multijoueur	0%	20%	100%	100%
Réseau	0%	20%	70%	100%

2 Rappels de la soutenance 1

2.1 Rappel des tâches réalisées

À la première soutenance, nous avons réalisé différents premiers prototypes pour la réalisation de notre jeu.

2.1.1 Hengrui : Code du personnage, Multijoueur et Réseau

Pour la première soutenance, je me suis occupé des éléments basiques du code du personnage, comme le saut et les déplacements de gauche à droite. Quelques soucis ont été rencontrés lors de la réalisation du saut, mais après avoir consulté de nombreux tutoriels, le problème a été résolu avec succès.

J'ai ensuite implémenté ces mouvements afin qu'ils puissent fonctionner en multijoueur, en utilisant un outil d'Unity : NetCode for GameObjects. Grâce à différents éléments de cet outil, comme le Network Object, le Network Rigidbody 2D ou encore le Client Network Transform, les mouvements basiques du personnage en solo ont pu être adaptés au multijoueur.

Au niveau du réseau, j'avais uniquement mis en place un réseau local, permettant de faire fonctionner le multijoueur sur un même ordinateur. Cela a été réalisé grâce à l'implémentation d'un NetworkManager dans la scène. De plus, un Canvas contenant des boutons reliés au client, au host et au serveur a été ajouté, permettant ainsi de jouer et de tester le mode multijoueur.

2.1.2 Mélody : Site internet et Niveaux

De mon côté, je me suis occupée de la partie des réalisations des niveaux ainsi que du site internet. Le jeu est composé de 12 niveaux. Un niveau d'introduction, 10 niveaux d'exploration et d'aventure et un niveau consacré au boss final. Ces niveaux n'étaient pas réalisés graphiquement puisque nous ne disposions pas encore de tous les éléments graphiques mais les fonctionnalités étaient présentes. C'est-à-dire, la présence de l'interface et le changement de niveau. De plus, j'ai implémenté l'interface lorsque le joueur meurt. Elle permet au joueur de décider de recommencer le niveau ou de retourner au bunker. Cette interface était une autre scène donc le joueur était ramené à une autre scène. Le menu, l'écran d'accueil était également implémenté de manière à pouvoir lancer le jeu et avoir une interface qui permet de choisir ce que l'on souhaite faire une fois arrivé sur le jeu.

Concernant le site internet, tout le visuel du site était prêt. Pour rappel, le site n'a pas été préfabriqué, il a été réalisé de toute pièce en HTML et en CSS. Faute de temps, je n'ai pas pu implémenter les informations sur le site. De plus, le site n'est pas hébergé.

2.1.3 Chloé : Graphisme et Bunker

Lors de la première soutenance, nous avions tous les brouillons et les idées des graphiques du jeu sans pour autant avoir l'intégralité des dessins officiels. Voici une vue d'ensemble des graphiques que nous avions à ce moment.



Au niveau du bunker, seuls le point de sauvegarde et l'inventaire étaient partiellement disponibles.

2.1.4 Nael : Son et IA

Dans le cadre de notre soutenance, nous avons eu le plaisir de constater que nous avions achevé presque l'intégralité des bandes sonores prévues pour notre projet. Il ne restait plus qu'à procéder à leur validation officielle et à les intégrer dans le jeu. En ce qui concerne la composante liée à l'intelligence artificielle du jeu, nous étions en train de peaufiner les codes,

qui étaient déjà dans une phase avancée de développement. Pour la plupart, ces éléments étaient prêts à être intégrés dans le système.

2.2 Rappel des modifications apportées au jeu

En ce qui concerne les modifications apportées au jeu lors de la dernière soutenance, nous avons pris la décision de supprimer les systèmes d'intelligence et de force. Cette suppression a, de fait, entraîné l'élimination des livres ainsi que des mécaniques de synthèse associées. Par conséquent, les taux d'obtention ont dû être ajustés afin de correspondre à nos objectifs préétablis. La suppression du système de force à quant à elle entraîné l'utilisation d'une unique arme, étant la batte.

3 Avancement du projet

3.1 Réalisation des tâches

Toutes les tâches globalement ont beaucoup avancé.

3.1.1 Hengrui

Pour cette deuxième soutenance, j'ai tout d'abord essayé de terminer le code du personnage, en implémentant la barre de vie du personnage, les animations des mouvements et la caméra du personnage. Durant la réalisation de ces différentes tâches, je n'ai fait face à aucune réelle difficulté.

Pour la barre de vie, j'ai seulement eu besoin d'implémenter plusieurs canvas et images dans le projet, ainsi qu'un code C# permettant de remplir la barre de vie et de gérer la couleur en fonction du nombre de points de vie restants.

```

1 reference
public class HealthBart : MonoBehaviour
{
    4 references
    public Slider slider;
    2 references
    public Gradient gradient;
    2 references
    public Image fill;
    1 reference
    public void SetMaxHealth(int health)
    {
        slider.MaxValue=health;
        slider.value=health;

        fill.color= gradient.Evaluate(1f);
    }
    1 reference
    public void SetHealth(int health)
    {
        slider.value=health;

        fill.color = gradient.Evaluate(slider.normalizedValue);
    }
}

```

Ci-dessus est le code utilisé pour l'affichage de la barre de vie. Ici j'ai utilisé 3 variables, le Slider qui représente la barre de vie, le gradient un dégradé de couleur qui varie en fonction de la vie, et le fill qui est l'image qui remplit la barre de vie. Ici la première fonction permet de remplir la barre de vie lorsque le nombre de points de vie est au maximum, et la deuxième fonction permet donc de faire varier la barre de vie, lorsque le personnage principal prend des dégâts.

L'animation du personnage a pu être réalisée grâce à l'ajout d'un composant nommé Animator dans le script du personnage principal. Pour la réalisation de cette animation, j'ai simplement superposé les séquences d'images fournies par Coline sur un intervalle de temps. Afin que cette animation ne fonctionne qu'uniquement lorsque le personnage est en mouvement, j'ai créé deux animations : la première représente le personnage à l'arrêt et la deuxième s'active lorsqu'il se déplace. Pour relier ces animations, j'ai ajouté dans le code C# du déplacement un script affiché ci-dessous, permettant d'obtenir le composant Animator, puis de l'appeler pour déclencher les animations appropriées.

```
Flip(rb.linearVelocityX);
float character_velocity = Mathf.Abs(rb.linearVelocityX);
anim.SetFloat("Speed",character_velocity);
```

La première ligne permet au personnage principal de se tourner dans la bonne direction lorsqu'il doit se déplacer vers la gauche ou vers la droite. Le code suivant montre la fonction Flip écrite par mes soins utilisée au-dessus.

```
reference
void Flip(float _velocity)
{
    if(_velocity==0){
        spriteRenderer.flipX=false;
    }
    else if(_velocity>0.1f)
    {
        spriteRenderer.flipX=false;
    }
    else if(_velocity<0.1f)
    {
        spriteRenderer.flipX=true;
    }
}
```

La seule difficulté que j'ai rencontré pour l'animation de la marche a été le fait que, lorsque l'animation s'active, le personnage devient plus grand qu'avant. Ce problème était causé par le fait que les images utilisées pour le personnage et l'animation de la marche n'étaient pas de la même taille. Ce problème a été résolu assez facilement en redimensionnant l'image d'origine.

Pour la caméra du personnage, j'avais au départ choisi d'utiliser un système proposé par Unity, le *Cinemachine Caméra*, qui fonctionne parfaitement en solo. Cependant, lorsqu'on passe au mode multijoueur, *Cinemachine Caméra* ne permet plus d'assurer le suivi de la caméra. Avec l'utilisation de *Cinemachine Caméra*, ce suivi n'était effectué que sur le dernier personnage apparu dans la scène du jeu, ce qui ne correspondait pas à notre objectif : nous voulions que la caméra suive chaque personnage individuellement. Aucune

solution n'a été trouvée car la plupart des tutoriels disponibles dataient d'anciennes versions d'Unity et ne fonctionnaient plus avec notre version. C'est pourquoi j'ai décidé d'adopter une autre approche en réalisant ce suivi de caméra avec un script en C#. Grâce à ce code, le suivi de caméra est désormais opérationnel en mode multijoueur, avec la possibilité de définir des limites de position empêchant la caméra de se déplacer au-delà de certaines zones. La mise en place du mode multijoueur n'a donc pas été une grande difficulté puisque comme mentionné précédemment, le seul problème rencontré a concerné le suivi de la caméra.

```
[Header("Cible et Décalage")]
3 references
public Transform target;
1 reference
public Vector3 offset = new Vector3(0, 0, -10);

[Header("Fluidité du Suivi")]
1 reference
public float smoothTime = 0.3f;
1 reference
private Vector3 velocity = Vector3.zero;

[Header("Optionnel : Limites de la caméra")]
2 references
public Vector2 minPosition;
2 references
public Vector2 maxPosition;
1 reference
public bool useBoundaries = false;
0 references
private void LateUpdate()
{
    if (!IsOwner) return;

    if (target == null)
        target = transform;

    Vector3 desiredPosition = target.position + offset;

    if (useBoundaries)
    {
        desiredPosition.x = Mathf.Clamp(desiredPosition.x, minPosition.x, maxPosition.x);
        desiredPosition.y = Mathf.Clamp(desiredPosition.y, minPosition.y, maxPosition.y);
    }
    Camera.main.transform.position = Vector3.SmoothDamp(Camera.main.transform.position, ref velocity, smoothTime);
}
```

Ceci est le code réalisant le suivi de caméra. Il est séparé en deux grandes parties : la première est l'initialisation des différentes variables qui vont être utilisées, avec l'ajout d'un *Header* qui permet une meilleure compréhension des différentes variables lorsqu'on voudra les modifier. La deuxième partie est donc le code qui permet de réaliser le suivi de caméra. L'ajout du booléen *IsOwner* est donc l'élément qui permet au suivi de caméra d'être réalisé pour le multijoueur. Le déplacement de la caméra est réalisable grâce à la dernière ligne de la fonction, qui permet de transformer la position de la caméra à la position désirée calculée précédemment. L'ajout de la condition *useBoundaries* permet de définir les limites de position pour la caméra : si nous ne voulons aucune limite, on laisse à l'état faux, sinon on le met à vrai.

Enfin, pour la gestion des animations en mode multijoueur, aucun souci n'a été rencontré. Grâce à un composant fourni par *NetCode for GameObjects d'Unity*, le *Network Animator*, il a suffi d'ajouter ce composant au script de notre personnage pour que l'animation fonctionne parfaitement en mode multijoueur.

Pour conclure, concernant le code du personnage et du multijoueur, l'une des choses restantes à réaliser est l'implémentation du système d'attaque pour éliminer les monstres.

Même si un code a été réalisé pour le système d'attaque, cependant avant que je puisse implémenter l'animation de l'attaque, cette tâche ne peut être considérée comme terminée.

3.1.2 Méloidy

Pour cette seconde soutenance, j'ai pu avancer sur le site internet. De nombreuses informations ont pu être ajoutées sur le site tels que le téléchargement des rapports précédents, le synopsis du jeu, les sources, les descriptions des membres du groupe. De plus, pour ajouter plus de dynamisme sur le site, j'ai ajouté du *JavaScript* qui a permis, par exemple, d'avoir un calendrier où au clic sur un mois permet d'afficher ce qui a été réalisé sur le projet.

Voici comment se présente l'avancement du projet :

Avancement du Projet

Octobre Novembre Décembre Janvier Février Mars Avril Mai

Détails du mois
Cliquez sur un mois pour voir les détails.

Avancement du Projet

Octobre Novembre Décembre Janvier Février Mars Avril Mai

Octobre 2024

Le projet a début. Les premières idées d'“Objectif Lune !” sont survenues lors d'une discussion autour d'un repas. Au départ, nous n'avions qu'un nom, repris pour sa sonorité d'une série de vidéos Minecraft créée par Frigiel, un Youtuber français reconnu pour ses vidéos Minecraft, accompagné d'autres créateurs de contenu. Cette série de vidéos avait pour but d'atteindre la Lune, en ne partant de rien, en équipe ou seul, seulement équipé de sa capacité à réfléchir à un moyen de survivre pour atteindre son objectif. Nous avons donc construit les débuts d'“Objectif Lune !” à partir de nos propres idées ainsi qu'en reprenant le concept de la série. Pour écrire le fil directeur de l'introduction au jeu, nous nous sommes inspirés du début de la série Netflix, “The Rain” en reprenant l'environnement post-apocalyptique dû à un incident : un virus pour la série et un accident nucléaire pour notre jeu. “Objectif Lune !” offrira donc une expérience captivante alliant exploration et survie.

Ensuite, il est donc possible de télécharger les derniers rapports de soutenance :

Rapports à télécharger

- Télécharger le cahier des charges
- Télécharger le rapport 1
- Télécharger le rapport 2
- Télécharger le rapport 3

Diaporama à télécharger

- Télécharger le diaporama méthodologie

Par la suite, j'ai pu ajouter une fonctionnalité avec *JavaScript* qui permet d'afficher nos contacts une fois arrivé en bas de page :

Bienvenue sur Objectif Lune !

Découvrez notre passionnant projet d'exploration lunaire. De la conquête spatiale aux futures missions, nous vous invitons à plonger dans l'histoire et les enjeux de cette grande aventure.

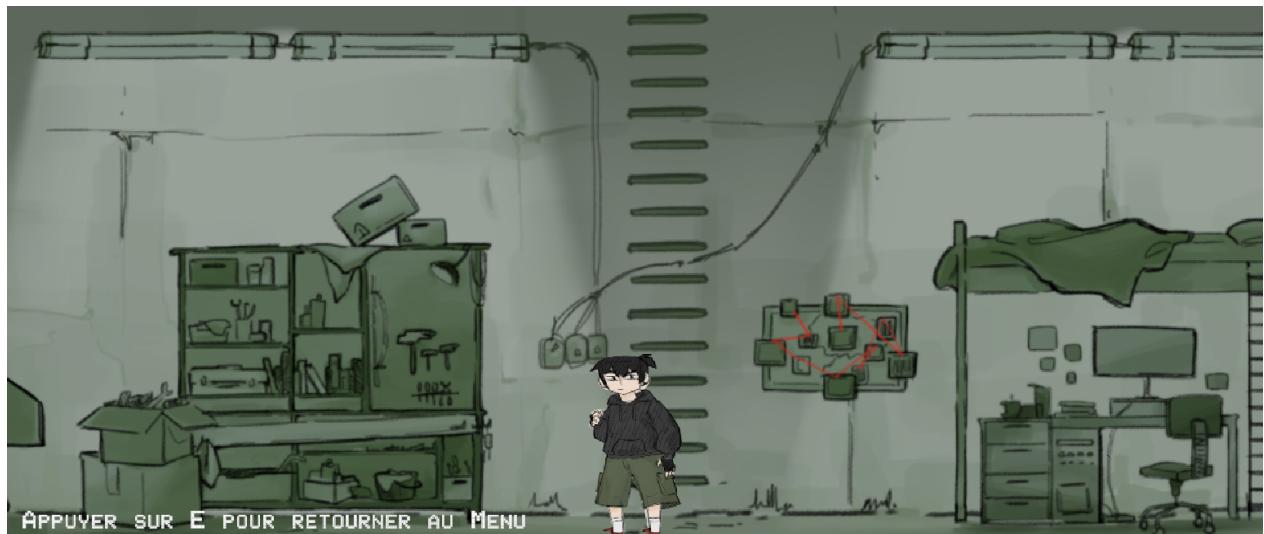
L'histoire d'"Objectif Lune !" se déroule en 2550. Notre héros reçoit alors un message de son père seulement composé d'une position géographique. Curieux, le personnage principal décide de s'y rendre. Son père lui aurait-il préparé une surprise après toutes ces années d'absence ? Arrivé sur place, notre héros n'y trouva rien d'autre qu'une mystérieuse trappe. "Où peut-elle bien mener ?" N'ayant à peine le temps de se poser cette question qu'une énorme explosion retentit. Pris de peur et de surprise, notre héros se précipite à l'intérieur de la trappe. La nouvelle pièce était plongée dans le noir complet jusqu'à ce qu'une vidéo se lance. C'est une vidéo de la part de son père lui indiquant que, durant ces dernières années, ce dernier était membre d'une équipe de chercheurs dans le nucléaire dont l'expérience ratée a provoqué l'explosion. L'enregistrement continue son cours en expliquant que les survivants, transformés en zombies, errent à l'extérieur et que notre héros doit maintenant construire une fusée pour s'échapper vers la Lune. Heureusement pour notre personnage principal, le bunker contient tout le nécessaire pour survivre et se préparer : de la nourriture, de quoi se défendre, des livres, et bien d'autres choses à découvrir.

✉ dev_and_co@outlook.fr ⚡ 66 Rue Guy Môquet, 94800 Villejuif

De plus, j'ai pu contribué à la réalisation du bunker, le lieu qui regroupe toutes les informations nécessaires pour jouer telles que l'inventaire ou encore la carte du monde. Pour la réalisation de l'inventaire et de la carte j'ai utilisé des panels : il s'agit d'interfaces qui apparaissent à l'écran. Ces interfaces s'affichent uniquement lorsque le joueur est proche et qu'il appuie sur E pour l'inventaire et T pour la carte. Un texte apparaît également en bas à gauche pour prévenir le joueur qu'il est possible d'interagir avec un objet. Cette interaction a pu être réalisée à l'aide des *box collider 2D* que propose Unity. En effet, avec la fonctionnalité *is trigger* permet au joueur de se superpositionner à l'objet tandis que le code permet de vérifier si le joueur actuel est proche à l'objet grâce aux tags qui ont été attribués. Les scripts pour les affichages ont été réalisés sans difficulté. En effet, la première fut un peu compliquée notamment pour l'inventaire puisque j'avais encore du mal à me familiariser. J'ai donc suivi de nombreux et divers tutoriels ce qui m'a permis de le réaliser avec succès. J'ai par la suite pu l'implémenter pour la carte sans souci majeur puisque les étapes sont exactement identiques.



Interaction avec la carte



Interaction pour retourner au menu

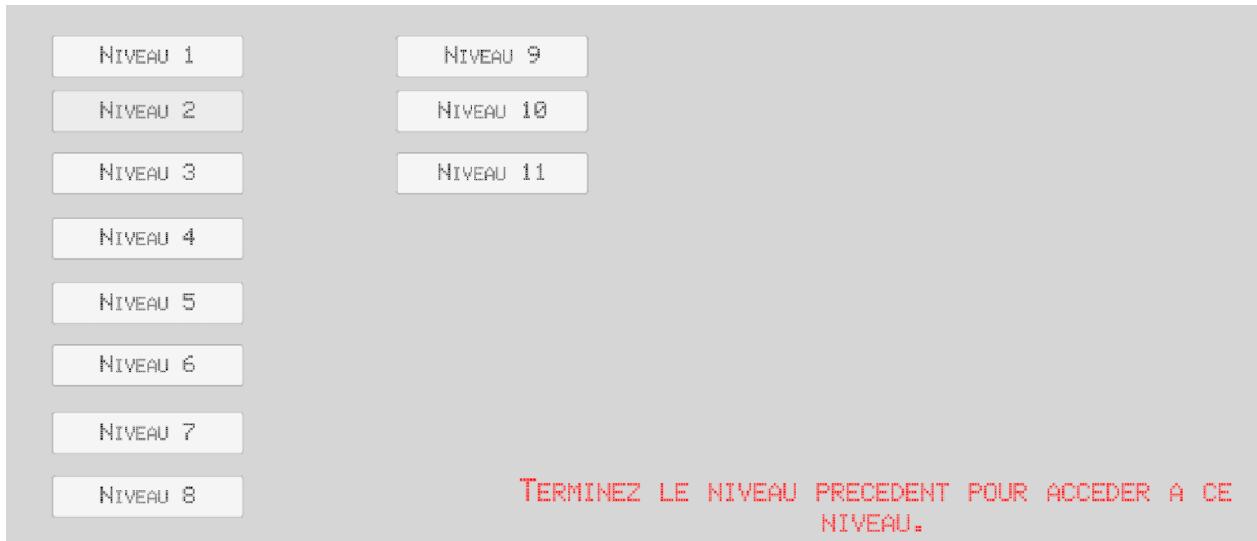


Interaction avec l'inventaire



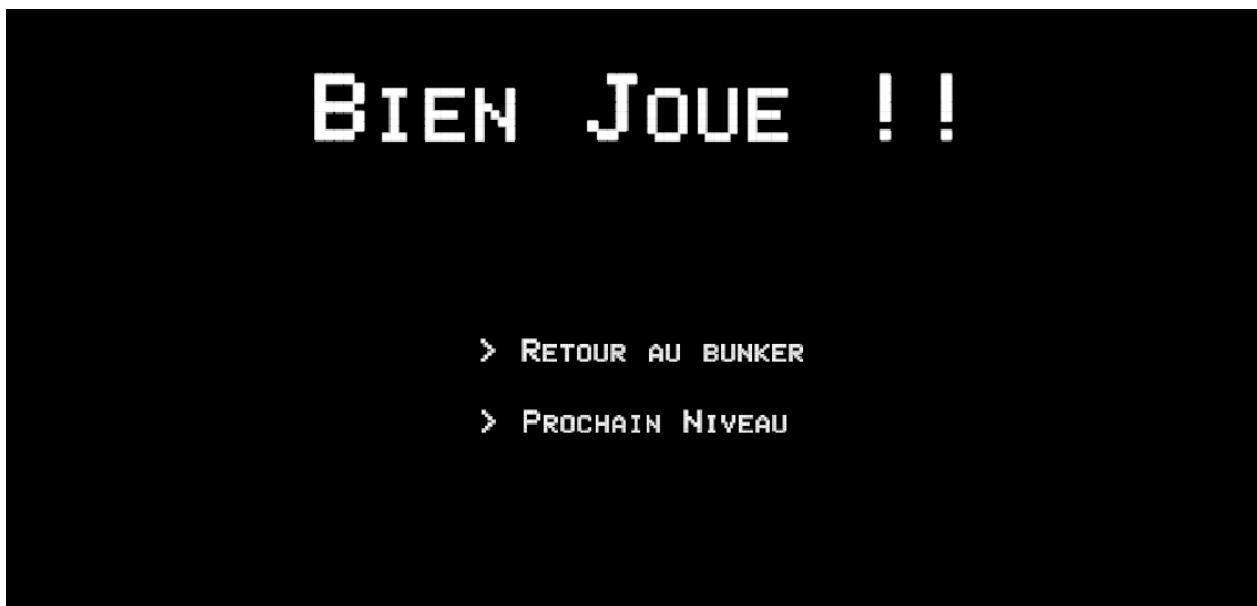
Ouverture de l'inventaire

Le système de carte a été réalisé en faisant en sorte que le niveau suivant ne soit pas réalisable si le niveau précédent n'a pas été réussi. Cette fonctionnalité n'est pas encore très au point puisque le système de sauvegarde n'a pas été réussi. J'ai tenté d'implémenter un premier essai mais sans succès.



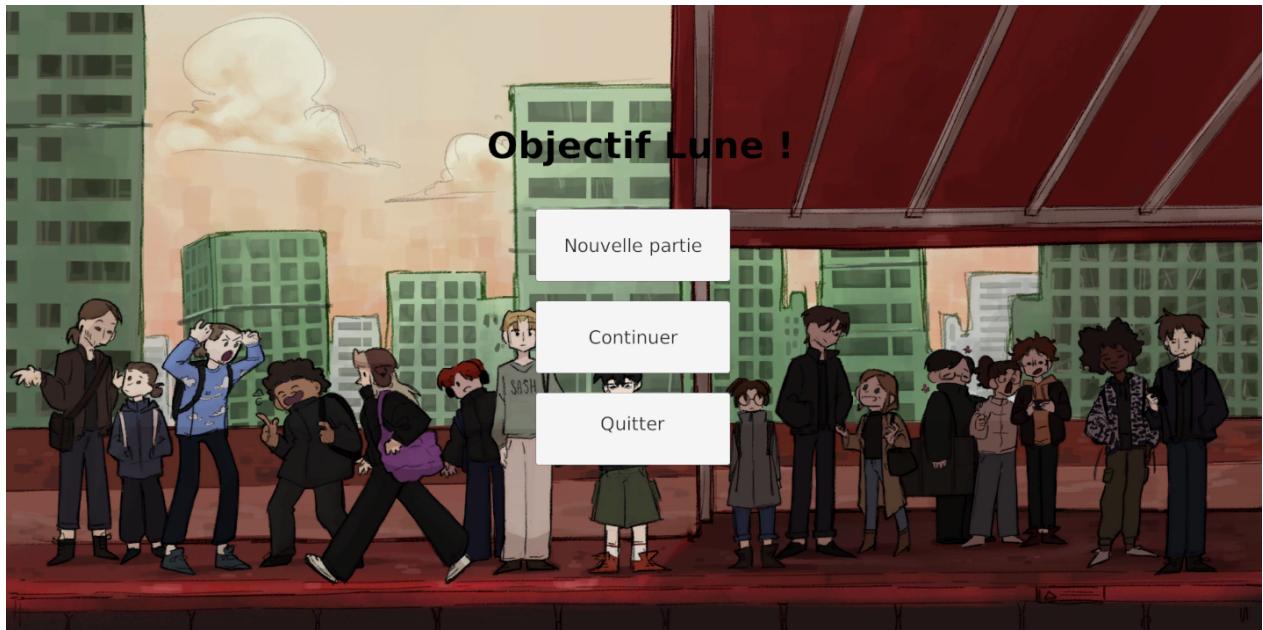
Affichage de la carte provisoire lorsque nous cliquons sur le niveau 2

J'ai pu également améliorer le système de changement de niveau. A la soutenance 1, le changement de niveau était instantané lorsque l'on arrivait à un certain point. Maintenant, il faut interagir avec une statue pour terminer le niveau. Une interface s'affiche demandant au joueur s'il souhaite passer au niveau suivant ou retourner au bunker.



Apparition de l'interface de fin de niveau après une interaction avec la statue

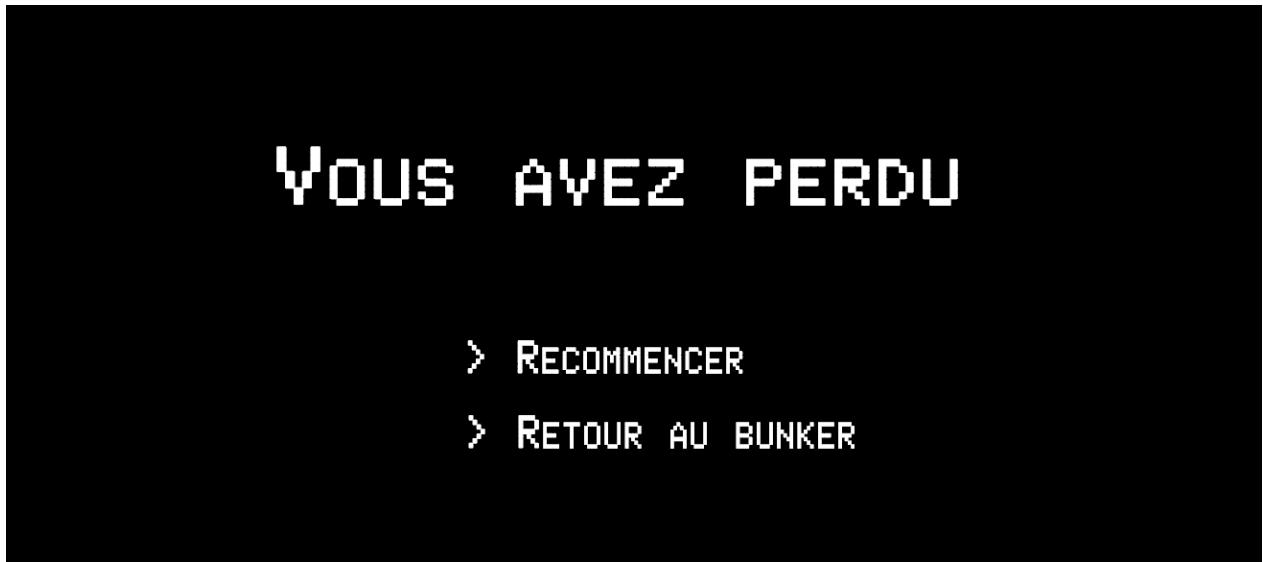
Par ailleurs, n'étant pas très talentueuse au niveau des réalisations graphiques et n'ayant pas d'imagination, Chloé s'est occupée du design du menu du jeu, des niveaux, des interfaces de fin de niveau ou encore lorsque le personnage meurt. Je me suis donc chargée seulement du script. Cette cohésion permet un réel travail de groupe et une bonne communication. Sans elle, le jeu n'attirait pas les futurs joueurs.



Avant - Menu du jeu



Après - Nouveau menu du jeu



Interface lorsque le joueur meurt

Le graphisme est assez simple mais qui reste dans le thème de notre jeu, la 2D et la lune.

3.1.3 Chloé

Ma tâche principale a été de synchroniser toutes les parties du jeu que l'on possédait en un seul projet Unity. C'est une mission plus longue que ce qu'on pourrait croire car il a fallu refaire tous les aspects visuels de chaque parties récupérées : refaire l'animation du personnage principal lors de son importation, reconnecter toutes les fonctions à leurs panels et boutons respectifs sans oublier tous les problèmes liés à régler et les fichiers à télécharger pour le multijoueur.

En plus de cette mission, je suis toujours chargée des graphismes en nous mettant en lien avec notre illustratrice Coline. Pour ce côté là, les graphismes liés aux niveaux et au bunker sont tous finis. Vous pouvez voir ci-dessous les différents éléments graphiques qu'elle nous a fournis. Ceux liés aux niveaux se déroulant le jour :

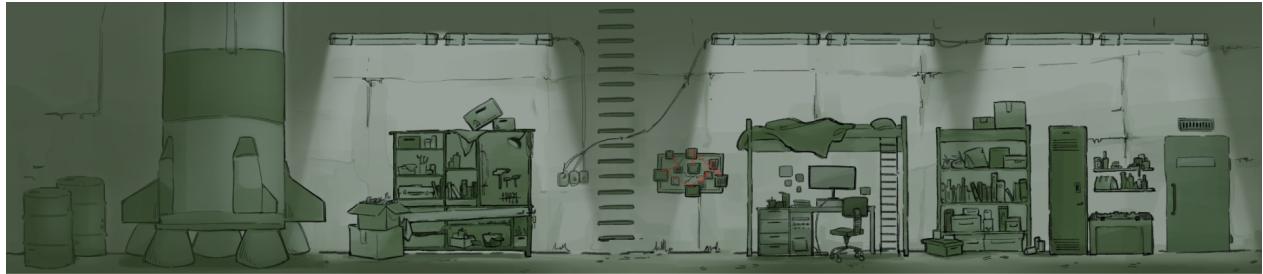


Ceux liés aux niveaux se déroulant de nuit :



Les ennemis que l'on pourra trouver au sein des niveaux : à gauche ceux de jour et à droite ceux de nuit. La taille de ces ennemis importe aussi : les plus grands infligent plus de dégâts que les plus petits.

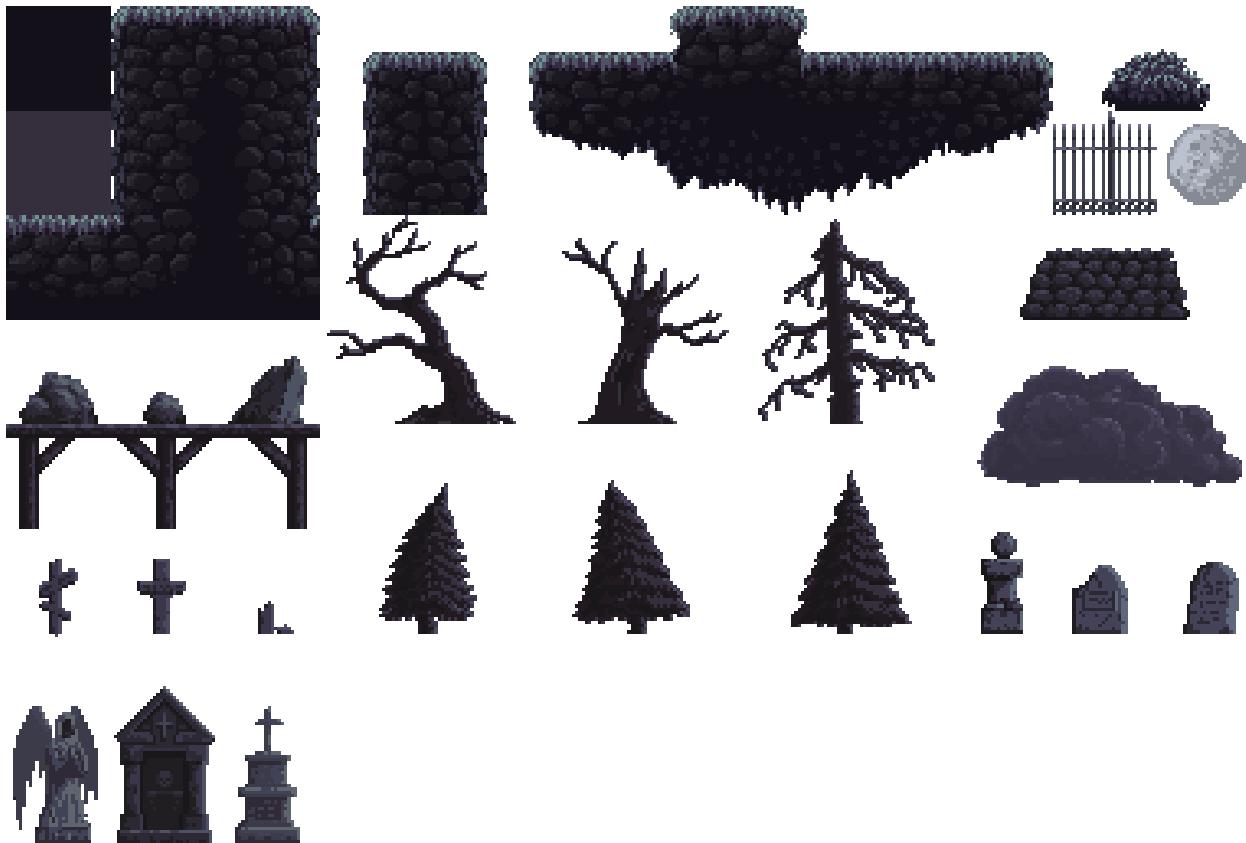
Le bunker officiel :



Nous aurons la jaquette, l'animation du lancement du jeu et celle de la fin du jeu pour la prochaine soutenance.



En ce qui concerne les niveaux, en tant que suppléante et comme dit dans la partie de Mélody, je me suis occupée de dessiner les niveaux à partir des dessins montrés plus tôt ainsi que de plateformes prises d'internet, en libre service. Voici le PNG utilisé :



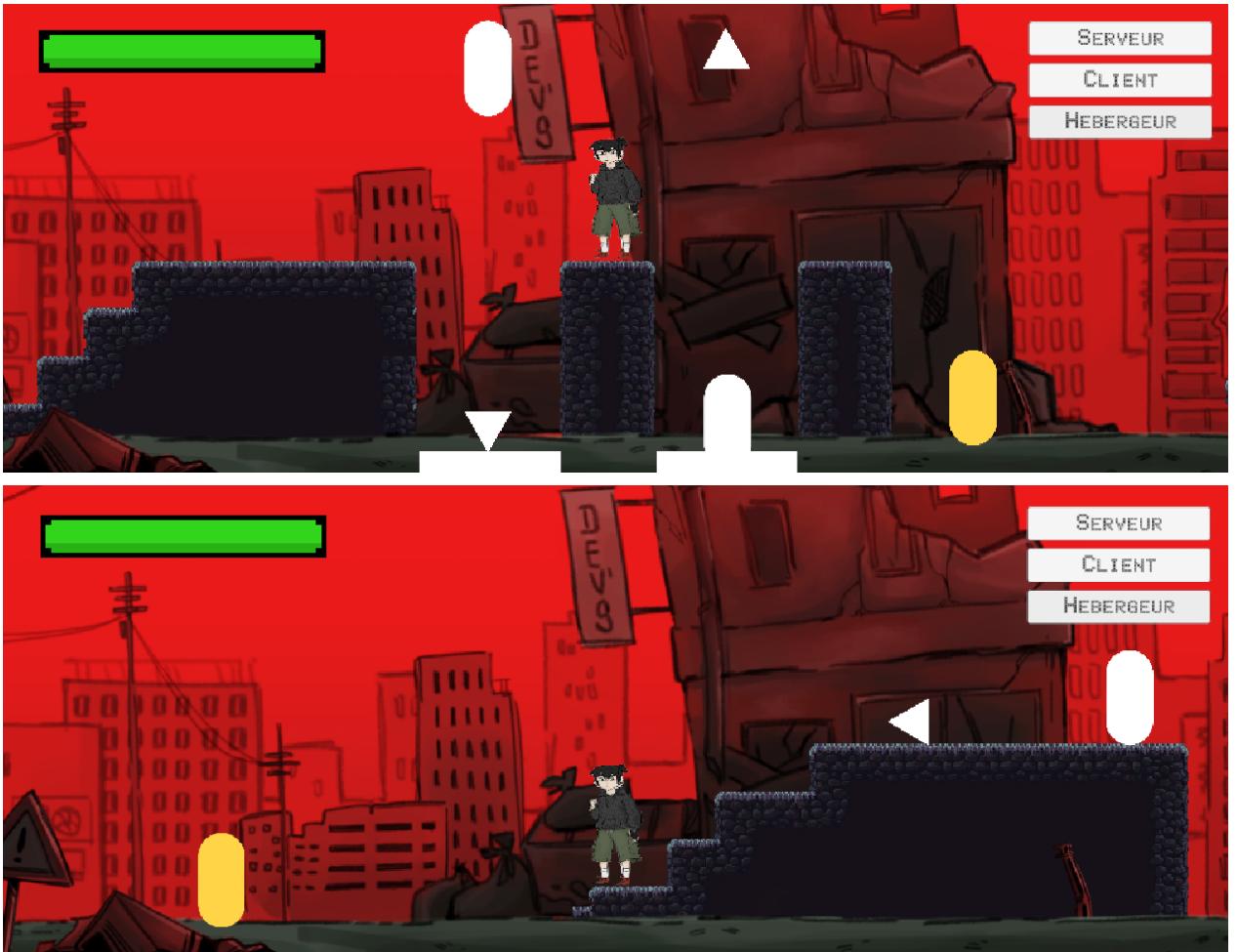
Globalement, je me suis occupée du visuel des niveaux et, de la même manière que pour la fusion de l'ensemble du jeu en un projet, de la coordination des codes de Mélody avec ma partie visuelle.

Tous les niveaux seront dessinés à partir de ces ressources en dehors des niveaux 3 et 5 puisqu'ils ne sont pas des niveaux d'exploration. Ce sont des niveaux "énigme", des niveaux dans lesquels le joueur devra utiliser ses connaissances pour les passer. Ils se déroulent sous forme de QCM. Voici un rapide aperçu du début du niveau 3 :



Tandis que les niveaux 1, 2, et 10 sont des niveaux de jeux de plateformes classiques :





Les capsules des aperçus des niveaux 2 et 10 représentent en réalité des ennemis que je n'ai pas eu le temps de mettre, les jaunes suivant le personnage et les blancs faisant des allers-retours.

3.1.4 Nael

Durant l'intervalle de temps séparant les deux soutenances de notre projet, de nombreux changements et améliorations ont été apportés, notamment en ce qui concerne les aspects sonores et l'implémentation de l'intelligence artificielle.

Tout d'abord concernant la partie sonore, un important travail d'écoute et de sélection minutieuse des sons et bandes sonores appropriées a été nécessaire afin de les adapter aux différents moments et ambiances du jeu. Cette phase de sélection et de peaufinage de l'environnement sonore a représenté la majeure partie des efforts consacrés à cet aspect du projet. En effet, il a fallu passer au crible de nombreuses pistes audio, en testant leurs adéquations avec les différentes scènes et situations du jeu, avant de retenir celles qui seraient finalement implémentées.

Par la suite, l'intégration technique de ces bandes sonores sélectionnées s'est avérée

relativement aisée, grâce à l'utilisation d'un script C# spécifiquement développé pour permettre la lecture des fichiers audio fournis en amont. Afin d'assurer une parfaite cohérence entre l'environnement sonore et les animations visuelles, nous avons opté pour l'option de lecture en boucle (*loop*) déjà implémentée dans le moteur de jeu Unity. Cette fonctionnalité nous a permis de faire jouer les fichiers audio en continu sur l'ensemble des niveaux et scènes du jeu, garantissant ainsi une immersion sonore constante et fluide.

En ce qui concerne les choix d'implémentation des sons, nous avons décidé de les intégrer à différents niveaux : dans les scènes de jeu elles-mêmes, dans la base principale de notre joueur, ainsi que dans l'écran d'accueil du jeu. Cette répartition nous a permis de créer un environnement sonore riche et varié, apportant une dimension supplémentaire à l'expérience de jeu.

Concernant maintenant l'implémentation de l'intelligence artificielle, celle-ci repose sur l'automatisation des mouvements et comportements des ennemis rencontrés par le joueur. Pour ce faire, une multitude de scripts C# ont été créés et testés, représentant plus de 8 versions différentes, avant d'en retenir seulement deux répondant parfaitement aux objectifs fixés. Il s'agit, d'une part, d'un script gérant l'automatisation des déplacements des ennemis entre plusieurs points prédéfinis dans l'environnement de jeu, et d'autre part, d'un script assurant le suivi complet et réactif du personnage principal par les ennemis.

L'implémentation de ces scripts d'intelligence artificielle dans le jeu s'est avérée particulièrement fastidieuse, car les codes initialement créés ne convenaient pas totalement à l'utilisation souhaitée. Un important travail de débogage a donc été nécessaire afin de réaliser les connexions requises entre les différents composants du système, notamment pour permettre la prise de dégâts du personnage principal par les ennemis de manière cohérente. Un important travail de coordination et d'harmonisation des codes a dû être fourni pour faire fonctionner l'ensemble de manière interconnectée et fluide.

Concernant le deuxième script C# relatif à l'intelligence artificielle, celui-ci n'a pas encore été implanté dans le jeu car les niveaux les plus difficiles n'ont pas encore été créés. Néanmoins, ce script a été testé à de nombreuses reprises dans des fichiers de test autonomes et fonctionne parfaitement, étant ainsi prêt à être intégré dès que les conditions le permettront.

3.2 Nouvelles modifications apportées au jeu

En ce qui concerne la récente modification apportée au jeu, celle-ci se rapporte à la composante réseau multijoueur. Plus précisément, cette modification consiste en la transformation d'un réseau initialement conçu pour le mode en ligne en un réseau local.

Nous avons par la suite remarqué que la suppression de la table de synthèse pour la soutenance 1 entraîne une modification du système de taux d'obtention. Avant sa suppression nous avions prévu de faire un système qui générera un certain nombre de matériaux. Dorénavant un niveau donnera un matériel précis alors que certains des niveaux ne donneront rien. Cela évitera au joueur de refaire le même niveau et de ne pas obtenir le matériel nécessaire pour avancer sur le jeu.

4 Problèmes rencontrés

Le problème majeur que nous avons rencontré est l'utilisation de Git. En effet, l'utilisation de Git est essentielle pour mettre en commun notre projet. Cependant, nous n'arrivions pas à manipuler correctement cet outil. Nous avions initialement fait un répertoire, où nous avions chacun notre branche avec notre travail. Cependant, cela ne permettait pas de fusionner notre travail correctement : les scènes Unity reprises de Git ne gardent aucun aspect visuel, tous les imports graphiques et mises en forme se colorent en un rose fushia. Nous avons donc changé de stratégie pour nous faire gagner du temps. Pour ce faire, nous avons rassemblé tous les travaux fait par chacun sur un ordinateur, créé un nouveau répertoire où nous avons mis le projet puis nous avons travaillé en faisant en sorte qu'une personne implémente son travail puis actualise le projet actuel avec les nouvelles fonctionnalités. Et nous récupérons le projet pour ajouter d'autres fonctionnalités. Nous évitons d'actualiser le projet en même temps sur le projet pour éviter tous les conflits. Cette solution nous a permis de gérer une bonne cohésion et une bonne collaboration entre chaque membre du groupe.

5 Conclusion

Pour cette deuxième soutenance, nous avons essayé de terminer un maximum de parties pour notre projet. Au niveau du personnage principal, il ne reste plus qu'à implémenter l'animation de l'attaque puisque nous disposons déjà de son code, pour le mode solo et le mode multijoueur. De même pour la partie IA, les monstres peuvent maintenant se déplacer de manière autonome et infliger des dégâts à notre personnage principal. Il suffit de dessiner les niveaux pour les y poser. En parlant des niveaux, notre jeu possède désormais trois niveaux officiels.

Et donc pour la prochaine et dernière soutenance, nous allons donc commencer par terminer tous les éléments manquants de notre projet : les niveaux restants, l'animation de l'attaque, l'animation du début et celle de la fin. Après avoir terminé ces éléments manquants, nous allons tester le jeu de notre côté et de régler toutes les imperfections qui peuvent encore subsister ainsi que finaliser la jaquette.

Pour conclure, maintenant que nous sommes arrivés à cette deuxième soutenance, cela signifie aussi que la fin du projet arrive à grand pas. Notre jeu commence peu à peu à prendre la forme d'un vrai jeu vidéo que l'on peut retrouver sur le net, avec un graphisme unique et des mécanismes codés par nous-mêmes. Nous sommes vraiment fiers de voir cet accomplissement. "De 0 à 1" est le slogan de notre groupe, et à travers ce projet, nous avons réellement suivi ce slogan, nous sommes parties d'un groupe passionné ne connaissant pas grand chose, à un groupe qui a réalisé le jeu en entier en seulement quelques mois. Et le moins que l'on puisse dire est que ce jeu reste une vraie fierté.

6 Source

Graphisme : Coline EAP, étudiante en école d'animation

Contact :

- Instagram : meimeii.draws
- LinkedIn : Coline EAP

Sons : Alexandra TOURE

Contact :

- toure.alexandra@yahoo.com
- 0652792744

Plateforme : <https://anokolisa.itch.io/moon-graveyard>