Rapport de soutenance 1

Noé Cornu ; Théo Gille ; Clément Grégoire ; Leandro Tolaini

Projet S2 EPITA

Table des matières

Chapitre 1

Reprise du cahier des charges

Pour revenir sur le cahier des charges, nous avons très bien avancé par rapport objectifs que nous nous étions fixés. Dans un premier temps le multijoueur nous a permis de comprendre et de bien avancé sur l'un des point majeur de ce projet. Dans un second temps le design de la map et implémentassions des joueur nous à permis de mettre en relation le multi avec le gameplay.

Pour cette première soutenance nous avions prévu de bien avancé sur le multijoueur, d'écrire les dialogue, de design les maps, de créer les bases d'un site web pour y intégrer les éléments principaux, d'intégrer un menu basique afin de pouvoir naviguer à travers le multijoueur et finalement commencer l'implémentassions des mouvements basiques des joueurs. Pour mieux comprendre le tout voici le tableau récapitulatif de ce que nous avons fais pendant la soutenance ainsi que leurs avancées.

1ere soutenance	Pas commencé	En cours	Fini
Design et interface	X		
UI & Menus		X	
IA	X		
Personnages		X	
Multijoueurs		X	
Map design		X	
Site Web		X	
Son effet	X		
Mecanique de niveau	X		
Dialogues		X	

Chapitre 2

Noé Cornu - Dialogues

2.1 Histoire

Lors de l'écriture des dialogues, notre priorité a été de respecter le cahier des charges établi pour le jeu. Nous nous sommes inspirés de l'histoire que nous avions déjà écrite précédemment, en veillant à ce que les dialogues soient cohérents avec l'univers du jeu. Nous avons également respecté le synopsis de l'histoire, en mettant en scène deux personnages principaux, un ange et un démon, dans un univers mythologique. Nous avons travaillé avec soin pour assurer que les dialogues soient fidèles à l'essence du jeu et qu'ils offrent une expérience immersive aux joueurs. En fin de compte, nous sommes convaincus que les dialogues que nous avons écrits ajouteront une dimension supplémentaire à l'histoire et permettront aux joueurs de s'immerger dans l'univers du jeu de manière encore plus profonde.

2.2 Intégration

Afin d'intégrer les dialogues au jeu, j'ai travaillé sur la conception d'une chatbox à l'aide de Photoshop. Cette boîte de dialogue sera utilisée pour faire défiler les conversations entre les personnages au fur et à mesure que l'histoire se déroule. Nous avons choisi cette approche car elle permet de maintenir l'immersion du joueur dans l'univers du jeu, tout en offrant une présentation claire et facile à suivre des dialogues. La conception de la chatbox a été soigneusement étudiée pour s'assurer qu'elle soit en harmonie avec le reste de l'interface du jeu. Nous avons également travaillé sur le choix des couleurs et des polices de caractères pour garantir une expérience utilisateur agréable. Grâce à la chatbox, les dialogues seront facilement accessibles aux joueurs, offrant ainsi une expérience de jeu plus riche et immersive.

2.3 Dialogues

Les dialogues jouent un rôle crucial dans l'introduction de l'univers du jeu, en particulier en ce qui concerne les raisons pour lesquelles un ange se retrouve en enfer. Les conversations entre les personnages permettent également de mettre en lumière les différentes capacités spéciales des deux héros, l'ange disposant d'un double saut et le démon de capacités de combat plus avancées. Enfin, les dialogues soulignent l'esprit de coopération nécessaire pour s'échapper de l'enfer. En effet, pour réussir dans ce jeu, les joueurs doivent travailler ensemble et exploiter les compétences complémentaires de chaque personnage. Les dialogues offrent ainsi une immersion complète dans l'univers du jeu, en fournissant non seulement des informations sur les personnages et leur histoire, mais aussi en aidant les joueurs à comprendre les mécanismes de jeu et l'approche à adopter pour triompher des défis qui les attendent.

2.4 Problème rencontré

Au cours de la préparation de cette soutenance, nous avons rencontré un problème lié à l'utilisation de GIT, ce qui a entraîné un retard dans l'avancement du projet. Après plusieurs jours de recherche, j'ai finalement pu identifier l'origine du problème. Nous avions placé un dossier GIT à l'intérieur d'un autre dossier GIT, ce qui a considérablement perturbé le fonctionnement de l'outil. Cette erreur a été corrigée en retirant le dossier GIT interne et en réorganisant les fichiers dans un seul dossier GIT. Cette expérience a mis en évidence l'importance de bien comprendre les outils utilisés et de faire preuve de vigilance lors de l'organisation de nos projets.

2.5 Travail collaboratif

Pendant la préparation de cette soutenance, j'ai été responsable de tâches qui n'étaient pas techniquement très compliquées à effectuer. Cela m'a permis de libérer du temps pour apporter mon aide à mes camarades. En particulier, j'ai travaillé en étroite collaboration avec Clément sur la conception du site web en utilisant les langages HTML et CSS. J'ai également aidé Théo à mettre en place la fonctionnalité multijoueur avec Photon sur Unity. En plus de cela, j'ai été chargé de la rédaction de tous les documents PDF en utilisant LaTeX, ainsi que de la gestion de Git pour le projet. Grâce à ces efforts concertés, nous avons été en mesure de progresser efficacement dans le développement de notre jeu, et nous avons pu présenter une soutenance complète et réussie.

2.6 Ressentis

u cours de la préparation de cette soutenance, j'ai été confronté à plusieurs défis. Tout d'abord, j'ai trouvé la rédaction des dialogues plus difficile que prévu, car il fallait constamment trouver l'inspiration pour créer des conversations intéressantes et cohérentes avec l'univers du jeu. Ensuite, bien que la rédaction des PDF en LaTeX n'était pas techniquement compliquée, elle était extrêmement longue et fastidieuse. Nous avons également rencontré des difficultés avec le CSS pour le site web, que nous n'avons toujours pas réussi à résoudre. Enfin, la mise en place du multijoueur a pris du temps pour assimiler les notions principales, mais une fois ces dernières bien comprises, cela n'était pas si compliqué. Malgré ces difficultés, je suis convaincu que notre travail d'équipe et notre persévérance nous permettront de surmonter tous les obstacles et de réussir dans la réalisation de notre projet.

Chapitre 3

Leandro Tolaini - Game Design

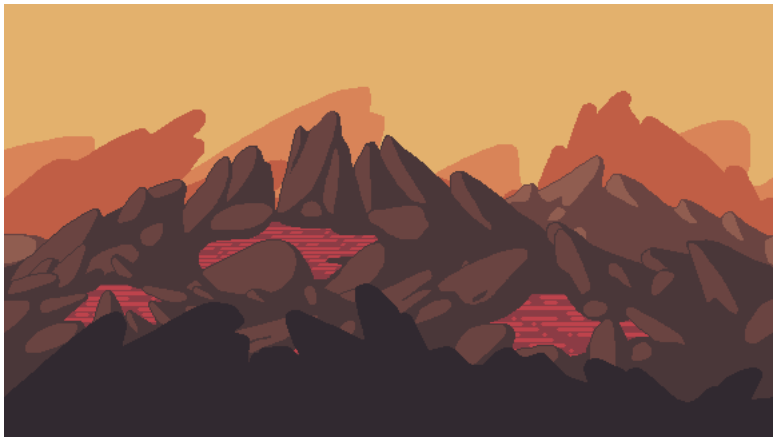
3.1 Assets

J'ai cherché des assets qui pouvaient correspondre à nos attentes : être cohérents avec l'histoire, être en pixel art pour rappeler le côté retro des platformers, et disposer d'animations pour les personnages.

Suivant ces critères, les assets retenus sont les suivants :

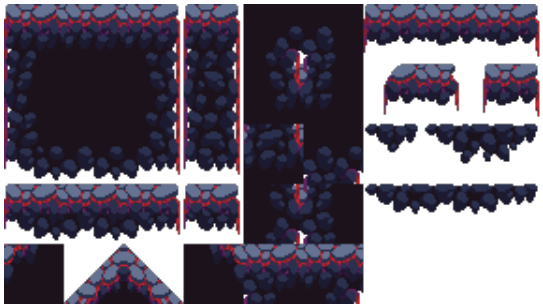
1. **Le background**

   Cette image correspond parfaitement à l'ambiance recherchée pour notre jeu. Un paysage désolé, presque cauchemardesque, qui correspond parfaitement à une interprétation de l'enfer qui colle à notre jeu. Les tons sombres et les couleurs chaudes sont en adéquation avec l'ambiance que nous voulons transmettre dans Hellscape.

2. **Les fondations**

   Ce seront les briques de tous nos niveaux, comme nous le verrons dans les premiers exemples. Elles ont été fournies avec le background, donc séléctionnées pour les mêmes raisons.



3. **Le Démon (Personnage 1)**

   C'est un des héros de l'aventure, et étant un habitant de l'enfer, il se devait d'avoir un design similaire à l'environnement, et également aux autres habitants de l'enfer, qui seront les ennemis des deux héros cherchant à s'enfuir. Il devait posséder une arme pour illustrer sa capacité : attaquer et tuer les ennemis.

4. **L'Ange (Personnage 2)**

   Ce personnage se devait d'avoir un gros contraste avec l'environnement : l'ange est lui un habitant du paradis qui se retrouve en enfer par erreur. C'est pour cela qu'il est très lumineux et gracieux. Il devait posséder des ailes pour illustrer sa capacité : être très agile (double saut).

5. **Les ennemis**

   Nous disposons d'une certaine variété d'ennemis grâce à cet asset : il ne seront surement pas tous implémenter dans notre jeu mais nous laisse une bonne possibilité d'évolution.

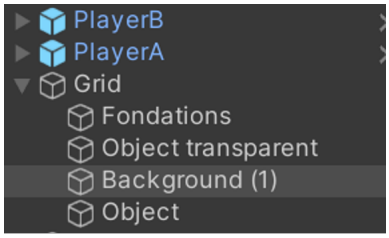
6. **Les items**

   Différents objets sont également à notre disposition à but décoratif, ou même afin de développer des mécaniques pour nos niveaux.

3.2 Conception

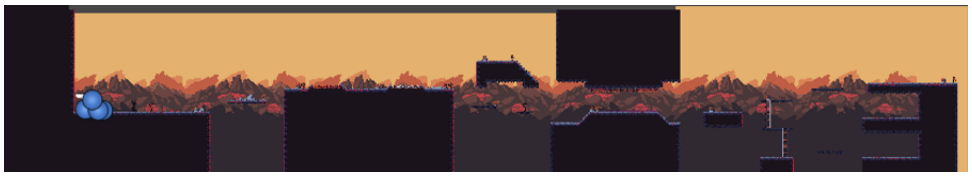
Le premier niveau devait répondre à quelques attentes, il devait notamment faire office de didacticiel. Il devait donc être assez simpliste, tout en guidant les joueurs pour les faire prendre en main les capacités des deux héros et comment elles seront utilisées dans les prochains niveaux, qui seront plus compliqués et plus créatifs.

Le choix des assets et le souhait d'avoir un jeu rétro imposait l'utilisation de tilemap pour le développement des niveaux. Pour le premier, j'ai donc du concevoir les différentes couches à l'aide de tiles palettes qui seront réutilisés pour tous les niveaux.



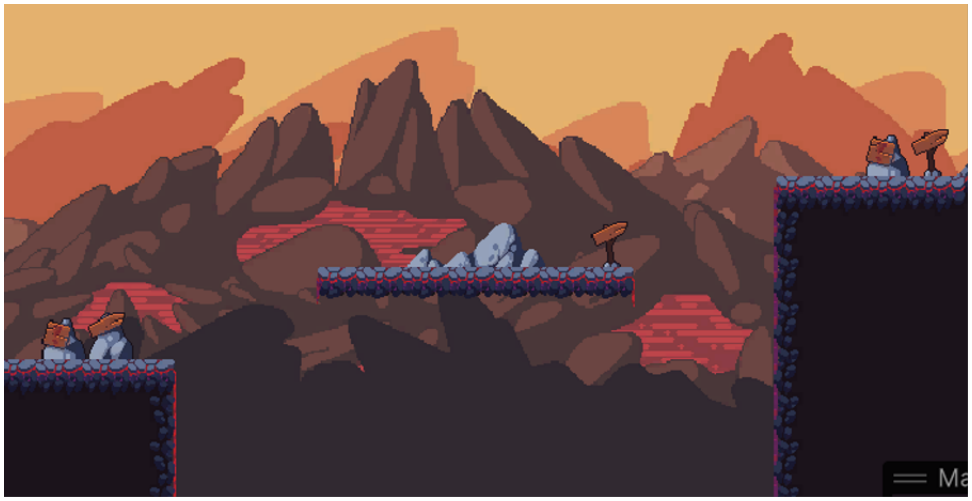
La première couche est logiquement le background, la deuxième, les fondations, c'est a dire toutes les structures des niveaux ; ensuite les objets transparents, qui seront des éléments du décor comme des arbres ou des rochers, qui seront en arrière plan par rapport aux joueurs. Enfin la dernière couche sera celle des joueurs et des objets qui ne seront pas traversable par les joueurs sans les éviter.

Finalement, voici l'aspect du premier niveau :

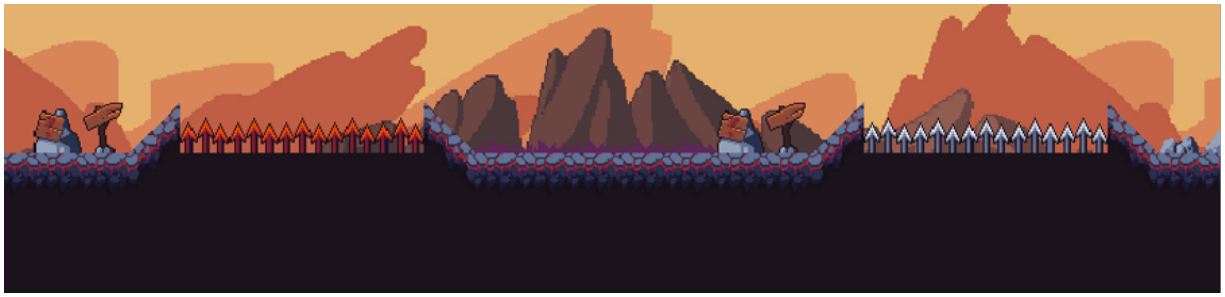


3.3 Mécanique

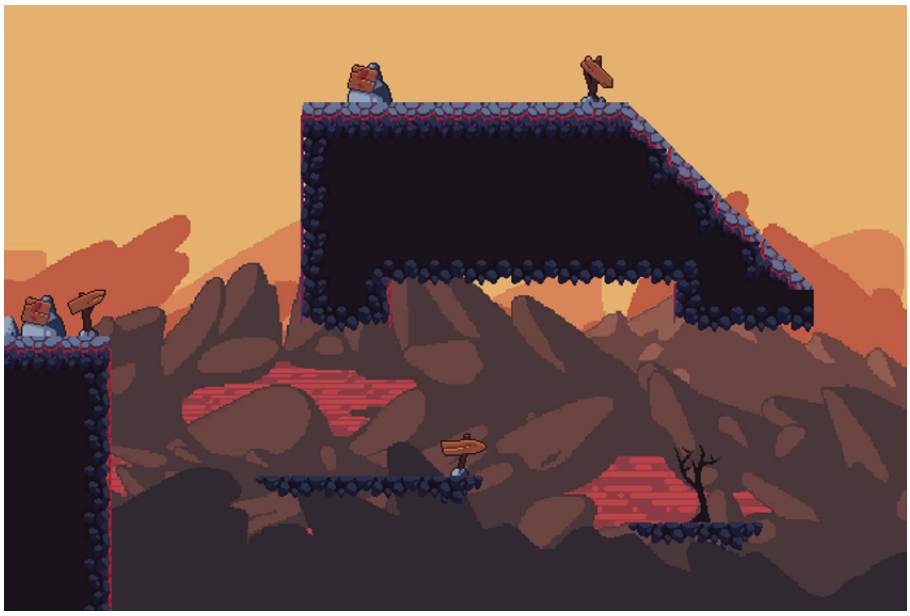
Maintenant nous allons voir les différentes mécaniques du jeu, à travers la construction du premier niveau, malgré qu'elles ne soient pas encore implémentées.



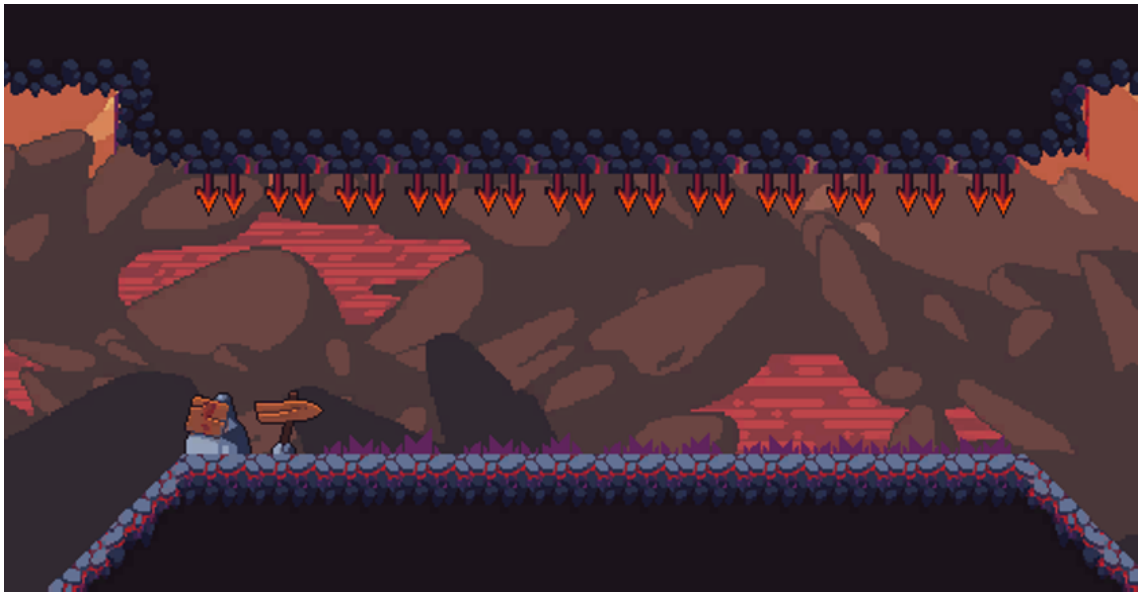
Le premier obstacle sert à prendre en main la physique du jeu et la mécanique de saut.



Ensuite, il y a deux types de piques : les rouges sont inoffensifs pour le démon et mortels pour l'ange et inversement pour les piques blancs.



Ensuite, l'ange sera le seul à pouvoir atteindre une clé qui se trouvera en haut de la plateforme (clé nécessaire pour réussir le niveau) grâce à sa capacité de double sauts tandis que le démon devra passer par en dessous.



Tandis que au niveau de ce dernier obstacle, le démon devra libérer le passage à l'ange en tuant un ennemi qui sera au bout du couloir grâce à sa capacité à se battre, afin d'atteindre la fin du niveau.

3.4 Déplacements joueurs

Pour de commencer à attribuer les premières animations aux personnages, j'ai du manipuler les premiers scripts de déplacements de ces derniers, de manière assez basique. J'ai commencé

par appliquer les collisions de ma tilemap et de mes personnages, ainsi que la gravité pour avoir la première physique du niveau.

Ensuite j'ai pu écrire mon premier script de déplacement.

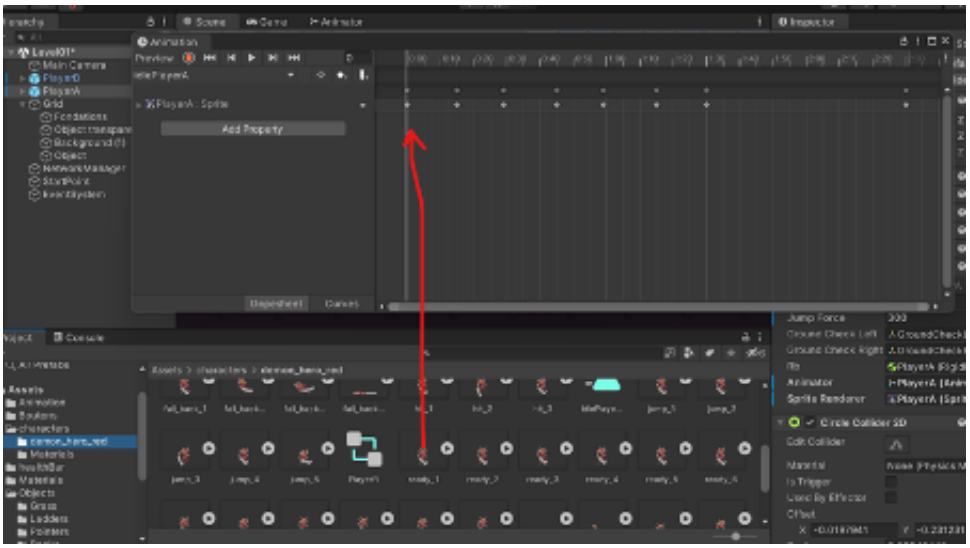
```
public float moveSpeed;
    public float jumpForce;

    private bool isJumping;
    private bool isGrounded;

    public Transform groundCheckLeft;
    public Transform groundCheckRight;
```

Les deux premières variables sont des constantes arbitraires pour déterminer respectivement la vitesse et la puissance de saut du personnages ; ensuite les deux booléens suivant sont déterminés grâce aux objets groundCheckLeft et groundCheckRight.

Ils sont représentés par ces deux cercles juste en dessous du personnage, et servent à détecter si celui-ci est en contact avec une structure ou non.

```
if (Input.GetButtonDown("Jump") && isGrounded)
        {
            isJumping = true;
        }
```

3.5 Animations



Cette partie est assez redondante : pour chaque personnage, il faut découper chacune de ses frames de la tilesheet, pour ensuite les glisser dans l'outil Animation de Unity à des intervalles de temps que j'ai choisi afin qu'elles soient jouées dans le bon contexte.

Pour se faire, l'ajustement des scripts de déplacement était nécessaire. Prenons l'exemple de l'animation de marche d'un personnage : pour avoir un bon rendu, il fallait assurer un renversement sur l'axe x de l'image du personnage si sa vitesse est négative (si l'utilisateur veut le faire avancer de droite à gauche).

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

La méthode Flip est donc venue s'ajouter à mon script. l'argument _velocity représente la vitesse actuelle du personnage.

3.6 Ressentis

La prise en main de Unity n'a pas été facile au début notamment à cause de la richesse de ses outils, les premiers paramètrages ont donc été fastidieux et maladroits, m'obligeant par exemple à recommencer plusieurs fois mon niveau à cause de problèmes de couches ; ou encore de bugs au niveau des différents scripts.

Les nombreux tutoriels et forums trouvables sur YouTube ou internet m'ont permis de prendre

en main Unity et corriger mes erreurs pour finalement avoir un premier niveau jouable, bien que encore assez simpliste. Cet apprentissage assez long était nécessaire, car quand ce premier niveau sera complètement opérationnel, le développement du jeu sera bien plus rapide et redondant.

Chapitre 4

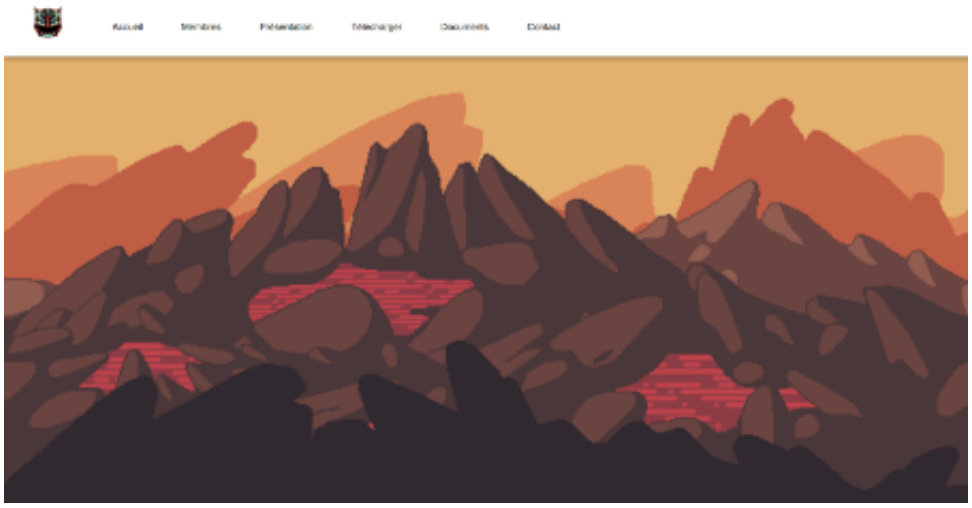
Clément Grégoire - Site Web

4.1 Conception

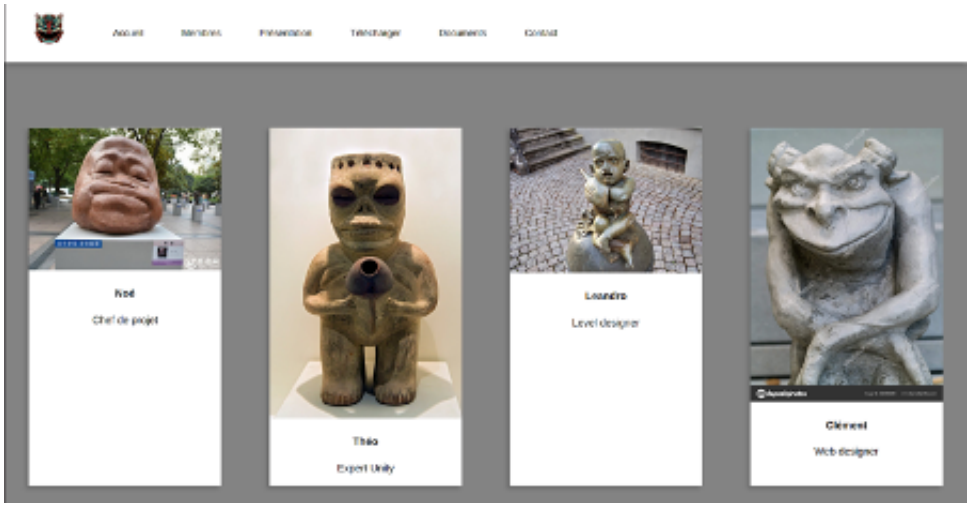
Le site web à été entièrement conçu en html et en css à l'aide de Visual Studio Code. Nous avons repris plusieurs fois la structure et l'identité graphique du site avant de trouver une version qui satisfaisait tous les membres du groupe.

4.2 Structure

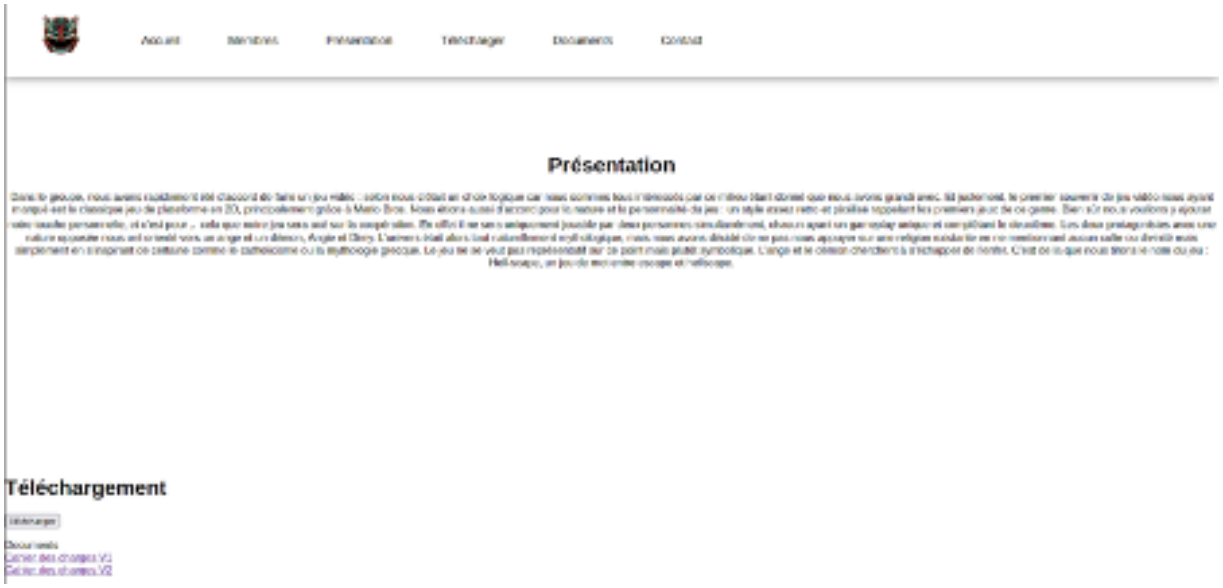
Lors de l'ouverture de la page web, on retrouve l'image de fond des niveaux de notre jeu. Cela permet de montrer immédiatement au joueur l'ambiance générale du jeu. Nous avons décidé de faire une barre de menu qui reste en permanence affichée à l'écran. On peut y retrouver notre logo ainsi que des boutons qui emmènent vers différents endroits de la page.



Nous avons mis différentes informations comme les nom et prénoms des membres du groupe, nos contacts ainsi que les différentes versions de notre cahier des charges. Comme nous n'avions pas de photos de nous à mettre dans la page, nous avons pris des images d'internet pour combler le vide. Ces photos sont provisoires et seront modifiées avant la prochaine soutenance.

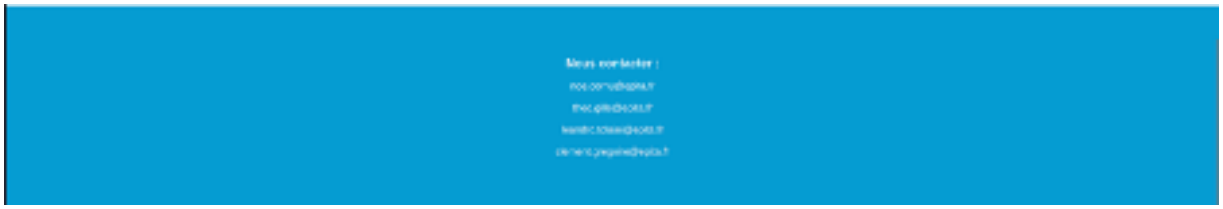


Ensuite, nous avons mis des informations sur le jeu, nous avons repris la présentation du jeu que nous avions écrite dans le cahier des charges.



Le bouton de téléchargement du jeu n'est pas fonctionnel car le jeu n'est pas terminé. En revanche, les différentes versions du cahier des charges sont disponibles au téléchargement.

Le css de ces deux dernières parties n'est pas encore implémenté.



Enfin, nous avons ajouté nos contacts tout en bas de la page.

4.3 Difficultés

La principale difficulté a été la mise en forme du site grâce au css. Le html et le css n'étant pas des langages qui se compilent, les erreurs ne sont pas clairement indiquées. A force de rajouter des lignes, le code devient rapidement surchargé ce qui diminue sa lisibilité. Réussir à trouver l'origine d'un problème ainsi que le résoudre deviennent donc des tâches ardues à réaliser.

4.4 Soutenance

Pour cette première soutenance, le site est disponible uniquement en local, la plupart des informations sont déjà présentes. La mise en page est faite sur une partie du site. Le bouton de téléchargement du jeu n'est pas fonctionnel car le jeu n'est pas terminé.

4.5 Ressentis

J'ai sous-estimé la conception de ce site web, je pensais que j'arriverais à le faire plus rapidement. J'ai donc été pris par le temps, et je n'ai pas beaucoup dormi. Je commencerai plus tôt pour les prochaines soutenance.

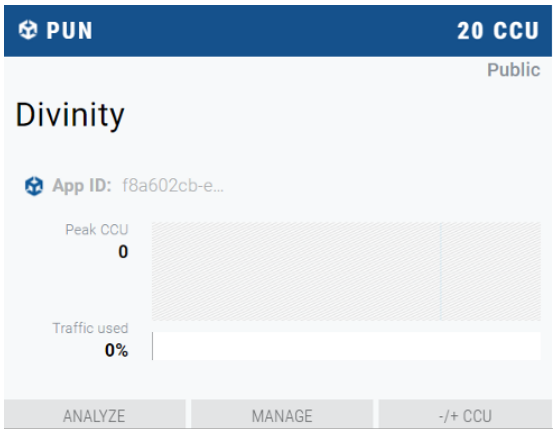
Chapitre 5

Théo Gille - Multijoueur

5.1 Multijoueurs

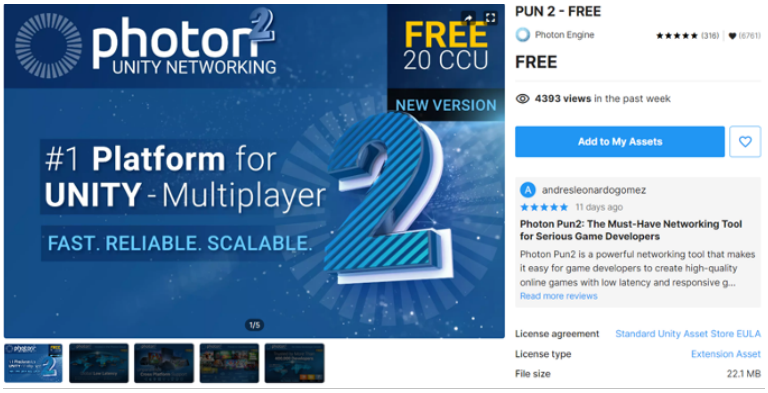
Avant de commencer la programmation des personnages, la mise en place et le bon fonctionnement du multijoueur est nécessaire.

Il existe plusieurs méthodes pour implémenter du multijoueur sur Unity, mais j'ai personnellement choisi d'utiliser Photon Pun. Tout d'abord j'ai dû créer un Serveur sur le Site de Pun en configurant divers paramètres comme notamment la région de celui-ci ou bien le nombre de joueurs

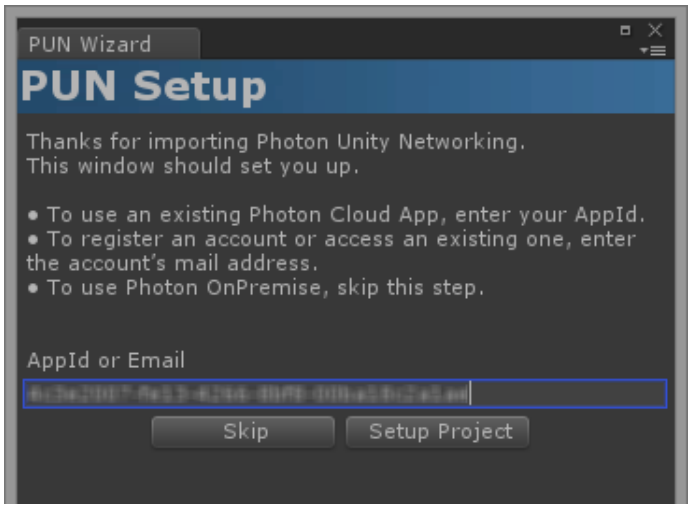


Après avoir fait ceci, il faut donc ajouter un asset Photon Pun à notre Projet pour mettre de lier le serveur au projet.

https ://assetstore.unity.com/packages/tools/network/pun-2-free-119922



Enfin Unity va nous demander l'adresse du Serveur Photon auquel nous voulons accéder



5.2 Implémentation

Pour implémenter le multijoueur dans le jeu Unity, il est important de suivre les bonnes pratiques de développement. Par exemple, il est important de minimiser la quantité de données synchronisées entre les clients et le serveur, afin de réduire la latence et d'améliorer la jouabilité. Il est également important de tester le jeu avec plusieurs joueurs pour s'assurer que la communication entre les clients et le serveur fonctionne correctement.

La première chose à faire va être de connecter notre première scène au serveur et donc aux paramètres associés à celui-ci.

```
using Photon.Pun;
using UnityEngine.SceneManagement;

public class ConnectToServer : MonoBehaviourPunCallbacks
{
   // Start is called before the first frame update
```

```
    void Start()
    {
        PhotonNetwork.ConnectUsingSettings();
    }
    public override void OnConnectedToMaster()
    {
        PhotonNetwork.JoinLobby();
    }

    public override void OnJoinedLobby()
    {
        SceneManager.LoadScene("Menu");
    }
}
```

Tout d'abord pour les Scripts de connexion au Serveur et où on utilise photon, on doit écrire MonoBehaviourPunCallbacks ce qui permet d'utiliser les Callbacks du serveur.

Ce script va ainsi être nécessaire pour la première partie de l'implémentation avec notamment le :

```
PhotonNetwork.ConnectUsingSettings();
```

et

```
public override void OnConnectedToMaster()
{
    PhotonNetwork.JoinLobby();
}
```

Ces deux codes vont permettre d'abord de charger et de le connecter au paramètre exigé et dans un deuxième cas de connecter l'hôte au lobby.

```
public override void OnJoinedLobby()
{
    SceneManager.LoadScene("Menu");
}
```

Tout d'abord pour les Scripts de connexion au Serveur et où on utilise photon, on doit écrire MonoBehaviourPunCallbacks ce qui permet d'utiliser les Callbacks du serveur.

Ce script va ainsi être nécessaire pour la première partie de l'implémentation avec notamment le :

```
PhotonNetwork.ConnectUsingSettings();
```

et

```
public override void OnConnectedToMaster()
{
    PhotonNetwork.JoinLobby();
}
```

Ces deux codes vont permettre d'abord de charger et de le connecter au paramètre exigé et dans un deuxième cas de connecter l'hôte au lobby.

```
public override void OnJoinedLobby()
{
   SceneManager.LoadScene("Menu");
}
```

Ici le lobby va être la scène nommé "Menu" et on y accède grâce a SceneManager qui permet de gérer les scènes existentes dans notre projet.

Une fois que le serveur est en ligne, il est temps de commencer à implémenter le multijoueur dans le jeu Unity. Les scripts et les composants nécessaires pour le développement de jeux multijoueurs en Unity avec Photon sont les suivants :

PhotonView : le composant PhotonView est utilisé pour synchroniser les objets du jeu entre les clients et le serveur. Il permet aux joueurs de voir les actions des autres joueurs dans le jeu.

PhotonTransformView : ce composant est utilisé pour synchroniser les transformations des objets du jeu. Il permet aux joueurs de voir les mouvements des autres joueurs dans le jeu.

PhotonAnimatorView : ce composant est utilisé pour synchroniser les animations des personnages dans le jeu. Il permet aux joueurs de voir les animations des autres joueurs dans le jeu.

PhotonNetwork : cette classe est utilisée pour gérer la connexion des clients au serveur de jeu. Elle permet aux joueurs de se connecter au serveur de jeu et de communiquer entre eux.

Ainsi selon nos besoin, nous devons les ajouter ou non.

Dans un second temps, il est nécessaire et préférable que chaque joueur puisse rejoindre divers lobbys, pour ça nous allons utiliser des boutons(voir Partie2) et leurs scripts qui leur sont associés(voir Partie2).

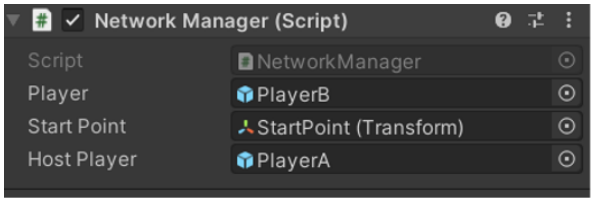
```
using Photon.Pun;
using UnityEngine.UI;

public class MainMenu : MonoBehaviourPunCallbacks
{
   public InputField createinput;
   public InputField joinInput;

   public void CreateRoom()
   {
```

```
        PhotonNetwork.CreateRoom(createinput.text);
    }

    public void JoinRoom()
    {
        PhotonNetwork.JoinRoom(joinInput.text);
    }

    public override void OnJoinedRoom()
    {
        PhotonNetwork.LoadLevel("Level01");
    }
}
```

Et pour finir il y a aussi l'appartion des joueurs après avoir rejoins la partie qui celui-ci est gère par un script qui se nomme NetworkManager dans notre cas :

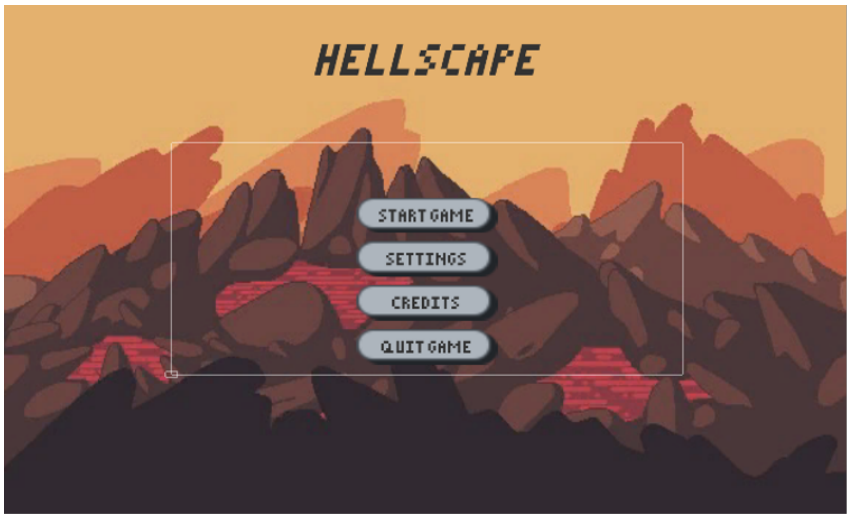
```
using UnityEngine;
using Photon.Pun;

public class NetworkManager : MonoBehaviourPunCallbacks
{
    public GameObject player;
    public Transform startPoint;
    public GameObject hostPlayer; // GameObject pour l'hôte
    // public GameObject myPrefab;

    private void Start()
    {
        if (PhotonNetwork.IsMasterClient) // si le joueur est l'hôte
        {
            PhotonNetwork.Instantiate(hostPlayer.name, startPoint.position, Quaternion.i
        }
        else // si le joueur est un client
        {
            PhotonNetwork.Instantiate(player.name, startPoint.position, Quaternion.ident
        }
    }
}
```

Ce script va permettre d'instancier les joueurs qui rejoignent à une position (startpoint) et à un Render définit dans Unity(player et hostPlayer)



5.3 UI & Menu

Lorsque nous démarrons notre jeu, on commence par afficher un menu principal composé de boutons permettant de nous emmener dans différentes scènes. Les boutons sont formés grâce à un Canvas et des boutons disponibles de base dans Unity que nous modifions pour ajouter certaines fonctionnalités.




```
using UnityEngine;
using UnityEngine.SceneManagement;

public class Menu : MonoBehaviour
{
   public string levelToLoad;

   public GameObject settingsWindow;

   public void StartGame()
   {
       SceneManager.LoadScene(levelToLoad);
   }

   public void SettingsButton()
   {
       settingsWindow.SetActive(true);
   }

   public void CloseSettingsWindow()
   {
       settingsWindow.SetActive(false);
   }
```

```
    public void LoadCreditsScene()
    {
        SceneManager.LoadScene("Credits");
    }

    public void QuitGame()
    {
        Application.Quit();
    }
}
```

L'ensemble de ses scripts va permettre d'interagir avec les boutons en plus des différents paramètres mis sur Unity.

Une nouvelle fois, on utilise le système de SceneManager pour gérer et passer d'une scène à une autre. De plus Application.Quit() va permettre d'arrêter le programme en cours.

Les Menu en règle générale pour cette première soutenance sont assez basiques et auront des modifications au fur et à mesure de l'avancement. Cette partie d'un projet peut être vue en second plan, alors que les menues d'un jeu vont être la première chose qu'un utilisateur va voir, c'est ainsi une chose importante dans la conception d'un jeu.

De plus on a pu créer nos propres boutons de différentes couleurs chacun permettant ainsi que grande diversité en cas de besoin.

5.4 Ressentis

Pour cette première soutenance, je pense que la plus grande difficulté à laquelle j'ai fait face à été la compréhension du multijoueur, les scripts sont globalement courts, mais quand utiliser quoi été assez complexe. De plus c'est assez différent de tout le reste je trouve, il faut ainsi beaucoup plus se renseigner que sur certaines notions.

Globalement sur la partie Menu et UI cela s'est déroulé assez rapidement je trouve, le menu pour cette première soutenance possède les caractéristiques de base et ne possède pas les designs de fin de projet, mais cela m'a permis de mieux comprendre comment fonctionnent les UI sur Unity en règle générale.