MATHEO GRANGE BTS SIIO

COMPTE RENDU PPE

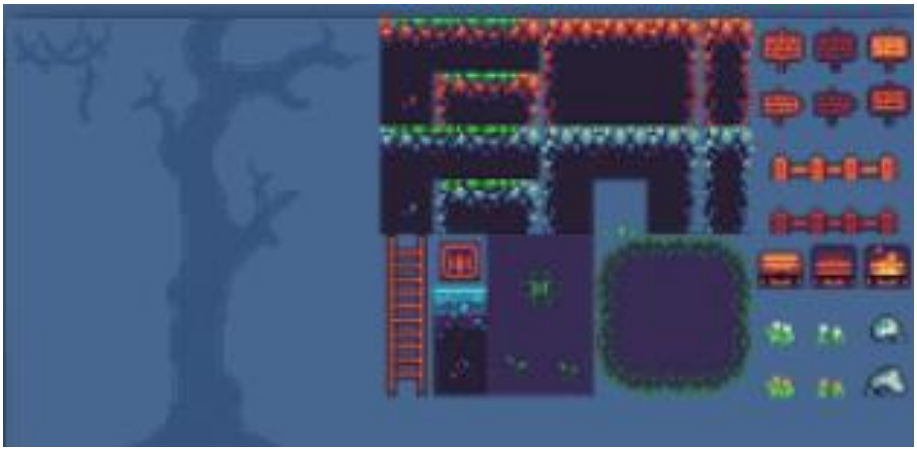
Table des matières

Introduction 1
Installation 2
Préparation 3
Suivi du projet 3
1/Le décor 3
2/Le personnage 4
a.Les déplacements 4
B. Le saut 5
3. Les animations 7

Introduction

Ce projet est un jeu vidéo créé à partir du logiciel Unity. L'objectif de départ était de créer un jeu en deux dimensions, avec plusieurs fonctionnalités telles que le saut, les déplacements et les animations. Pour créer ces fonctionnalités, je ne me suis pas seulement servi de ce que proposait le logiciel Unity, mais j'ai également dû créer des programmes en C# ainsi que récupérer sur Internet des "Tiles map", ce sont des gros blocs de texture créés par la communauté qu'il faut découper en carrés de 16x16, 32x32, 64x64... selon la taille prévue par le créateur. Une fois découpés, les blocs de textures peuvent être placés grâce aux différents outils de Unity.

(Voici la tiles map utilisée pour ce projet)



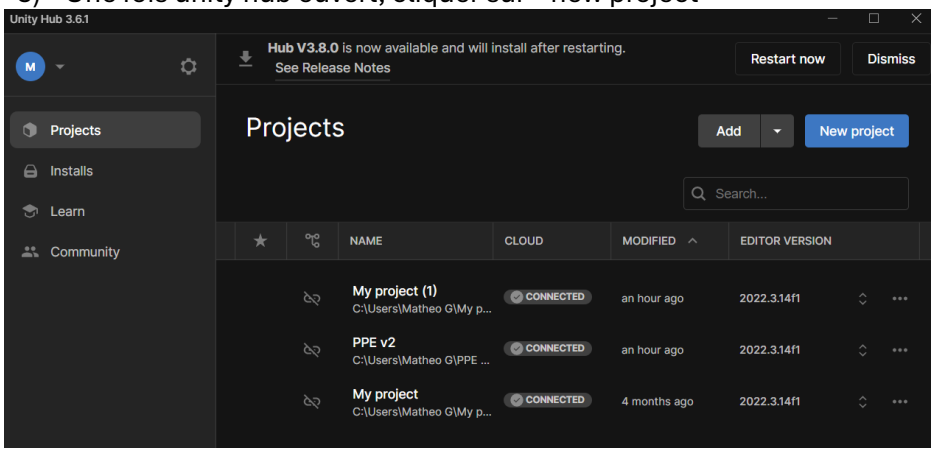
Installation

Etapes pour installer le projet

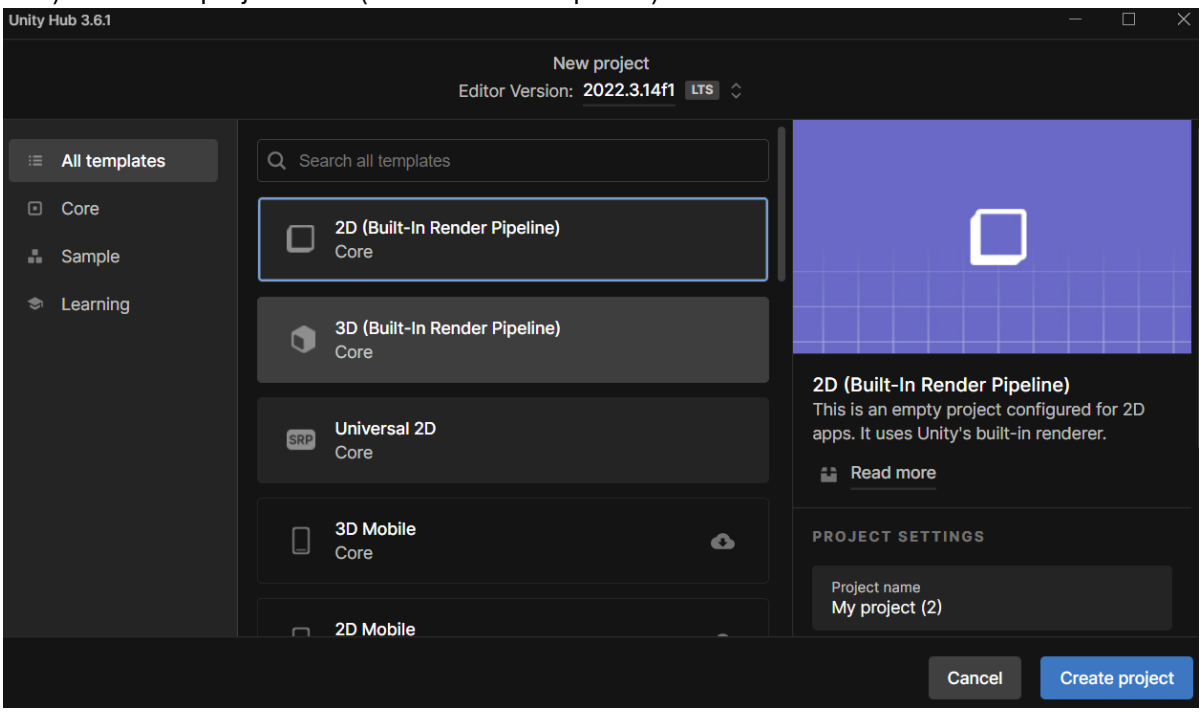
1) Installer Unity via votre navigateur (https://unity.com/download#how-get-started)

2) Créer un compte

3) Une fois unity hub ouvert, cliquer sur « new project »



4) Créer un projet en 2D (Built in render Pipeline)



5) Une fois dans le logiciel de création, quitter le et revenez dans le hub

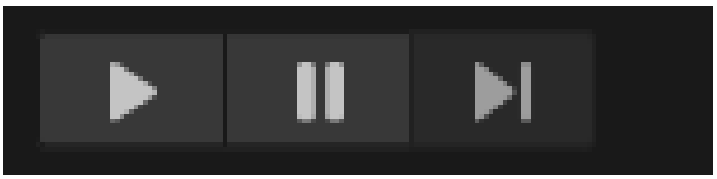
6) Une fois dans le hub, cliquer sur les 3 points a droite de votre projet puis sur « show in explorer »

7) Récupérer les fichiers du jeu que je vous ai préalablement envoyer via teams/Outlook et remplacer par ceux la les fichiers qui était déjà dans le dossier de votre jeu.(relancer ensuite le projet)

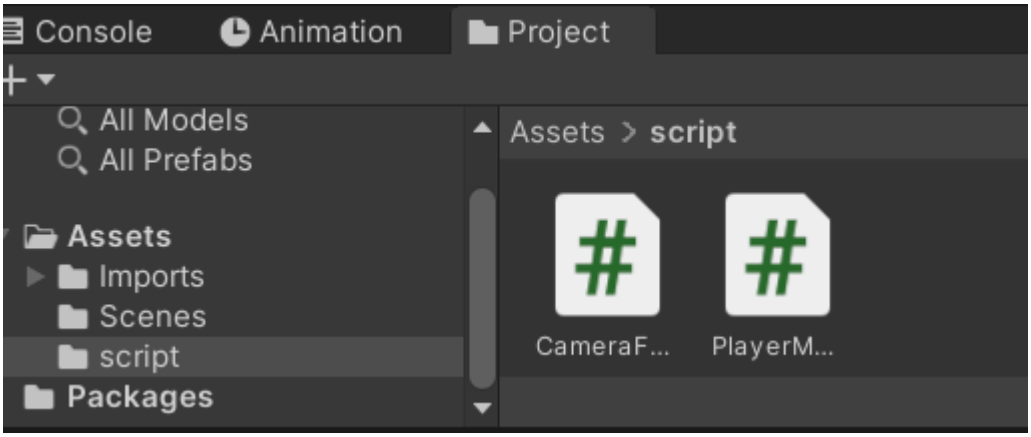
Préparation

Différentes fonctionnalités pour utiliser Unity :

- Pour lancer le jeu, cliquer sur le bouton start en haut au milieu de l'écran



- Pour accéder aux différents programmes en C#, cliquez sur Project > Assets > Scripts. Vous aurez accès aux deux programmes C# : celui qui permet de contrôler le joueur ainsi que celui qui contrôle la caméra.

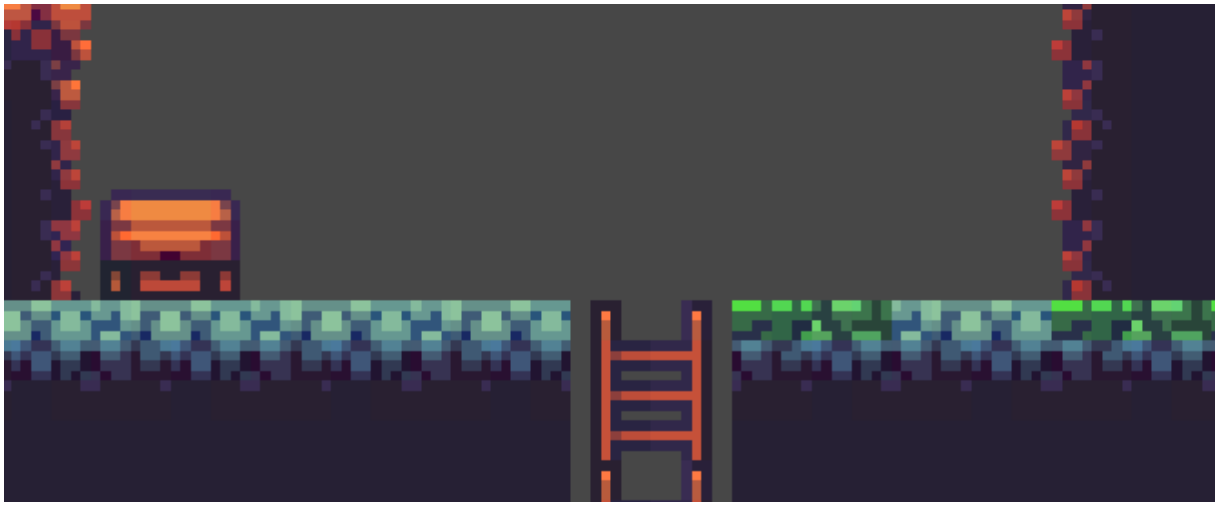


Suivi du projet

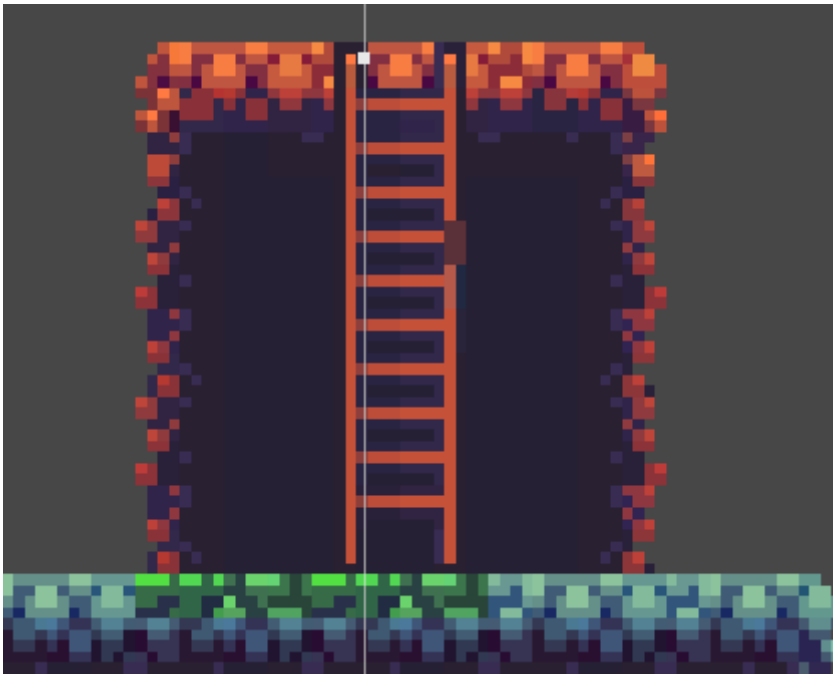
1/Le décor

Pour créer un jeu vidéo à partir de Unity, j'ai d'abord dû choisir parmi plusieurs tiles maps disponibles sur différents sites où la communauté partage ses créations. Mon choix s'est arrêté sur la tile map ci-dessus, qui incluait des textures de fond, des éléments de décor ainsi que différents personnages.

Une fois les tiles découpés en 16x16, j'ai pu commencer à construire ma plateforme et à mettre en place le décor. Cependant, un problème est rapidement survenu : les tiles ne se superposaient pas correctement, ce qui compromettait l'aspect visuel de mon jeu



Pour remédier a ce problème, j'ai alors créer une autre « couche » qui passe derrière la première se qui permet de les faire se superposer.



(J'ai également créé une couche pour l'eau ainsi que les éléments de décors)

2/Le personnage

a. Les déplacements

Une fois le personnage intégré au jeu, il était essentiel de lui appliquer la gravité et de gérer les collisions. Pour ce faire, j'ai ajouté à mon personnage un composant "Rigidbody" pour simuler les effets de la gravité, ainsi qu'un "Collider" qui définit la zone de collision, agissant comme une sorte de "hit box" autour du personnage. Cela garantit que le personnage ne traverse pas les murs et les obstacles, assurant ainsi une expérience de jeu réaliste et immersive.

Dans mon code C# j'ai créé les 2 variables qui me permettrons de gérer le déplacement ainsi que les collisions.

```
public float moveSpeed; //vitesse de déplacement du joueur
public Rigidbody2D rb; //Fait référence a l'élément Rigidbody de mon personnage
```

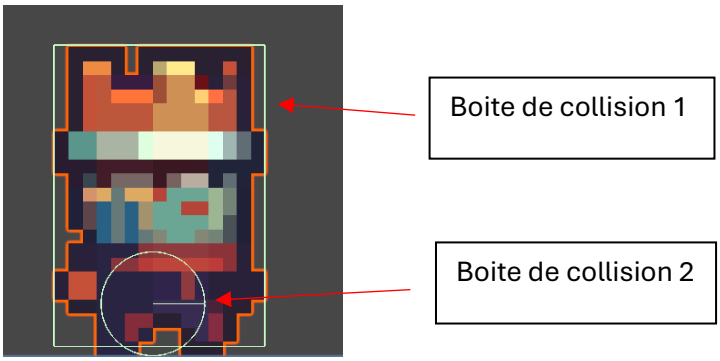
Ensuite pour faire déplacer le personnage il a fallut calculer son mouvement.

```
float horizontalMovement = Input.GetAxis("Horizontal") * moveSpeed * Time.deltaTime; //Permet de calculer le mouvement
```

Puis l'effectuer grâce à la méthode MovePlayer.

```
void MovePlayer(float _horizontalMovement)
{
    Vector3 targetVelocity = new Vector2(_horizontalMovement, rb.velocity.y);
    rb.velocity = Vector3.SmoothDamp(rb.velocity, targetVelocity, ref velocity, .05f);
}
```

Par la suite, j'ai également été confronté à un autre problème dans Unity : la boîte de collision de mon joueur se bloquait entre les tiles, ce qui l'empêchait de se déplacer librement. Pour remédier à ce problème, j'ai dû créer une deuxième boîte de collision, cette fois-ci de forme ronde au niveau des jambes de mon personnage. Cette approche a permis de garantir que le personnage puisse se déplacer sans rencontrer d'obstacles indésirables



B. Le saut

Afin de faire sauter mon personnage, j'ai créé une variable isJumping qui est de base sur faux car quand le joueur apparait il ne saute pas

```
public bool isJumping = false;
```

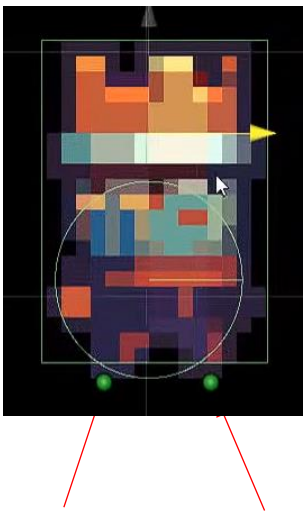
J'ai ensuite créé un « if » qui fait que si le joueur appui sur espace isJumping passe en « true »

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

J'ai également créé une deuxième condition, si isJumping est en true, alors le joueur vas sauter, puis isJumping repassera en false

```
if(isJumping == true)
{
    rb.AddForce(new Vector2(0f, jumpForce));
    isJumping = false;
}
```

Ensuite, il a fallu développer un code permettant au personnage de sauter uniquement s'il se trouve au sol, car sans cette restriction, il était possible d'appuyer continuellement sur la touche espace et de faire sauter le personnage en l'air de façon répétée. Pour résoudre ce problème, j'ai créé deux objets positionnés sous les pieds du personnage, reliés par une liaison. Le saut n'était autorisé que lorsque ces deux objets (droit et gauche) touchaient le sol.



Objets qui permettent ou non le saut

J'ai ensuite dû rajouter dans mon code un programme pour savoir si le personnage est au sol.

```
isGrounded = Physics2D.OverlapArea(groundCheckLeft.position, groundCheckRight.position);
```

Puis j'ai modifié ma condition de saut en incluant isGrounded pour que le personnage puisse sauter

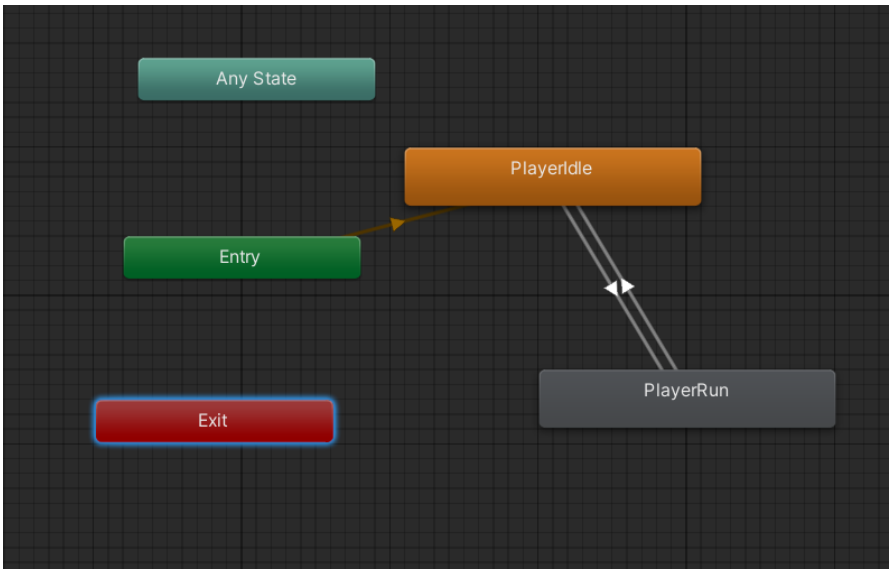
```
if (Input.GetButtonDown("Jump") && isGrounded)
{
    isJumping = true;
}
```

(un autre problème persiste sur 10 saut le personnage ne sautera que une ou deux fois)

3. Les animations

Pour donner plus de vie a mon personnage je lui ai créer des animations, une pour quand le joueur est surplace et une quand il avance.

Les animations sont une succession d'images fait grâce à unity.



L'outil animator dans unity permet de passer d'une animation a l'autre, quand le joueur apparait, l'animation commence sur playeridle, et quand le joueur commence à bouger l'animation change sur Playerrun, l'animator sait quand le joueur bouge car le joueur obtient alors une vitesse