Rapport de soutenance 2

Projet EPITA 2019

by

-Team SHEP-

BRIEU Emma
GROLIER Paul
LAVENU Stevie
RUIZ Hugo

TABLE DES MATIÈRES

1. Introduction

Notre projet EPITALE représente un Role Playing Game, inspiré d'un jeu vidéo réputé. Au départ la volonté du groupe était de concevoir quelque chose qui nous soit accessible à tous, due à notre manque d'expérience en programmation avant notre entrée à l'EPITA. Très apprécié par chaque membre et se trouvant être un jeu en deux dimensions, *Undertale* a de ce fait été un choix unanime.

Dans le jeu que nous développons, tout comme dans celui dont nous nous inspirons, le joueur incarne un personnage perdu dans un monde malfaisant. Celui-ci évoluera dans un environnement sombre et mystérieux. Il tentera d'en réchapper. Sur son chemin, il aura à affronter divers monstres et/ou boss qui tenteront de lui barrer la route. Le joueur devra alors faire un choix : les combattre, ou bien faire preuve de clémence en les épargnant.

C'est en général durant les vacances pré-soutenance que les projets peuvent avancer le plus rapidement. Cette semaine de vacances durant le mois de février tombe à point pour nous retrouver plongés entièrement dans le projet.

Ce second rapport de soutenance contient l'ensemble du travail effectué durant ces quelques semaines de développement de notre projet informatique d'Info SUP. Nous allons donc détailler l'avancée de notre projet depuis notre première soutenance. Nous ferons par la suite un point sur les attentes que nous pouvons avoir en ce qui concerne la prochaine qui sera la dernière.

2. Rappel du projet et de l'équipe

2.1. Les membres

2.1.1. BRIEU Emma

L'informatique a toujours été un domaine qui avait le don d'attiser ma curiosité. Pourtant, je n'avais jamais vraiment osé m'y mettre ni essayé d'apprendre un quelconque langage. Lors de mon année de terminale, j'avais hésité longuement entre la spécialité mathématiques et ISN. Au final, j'avais opté pour la première…
Intriguée par ce monde qui m'était encore inconnu, j'ai décidé d'intégrer l'EPITA. Mon entrée dans cette école a rendu possible une certaine immersion dans l'informatique, comme je le souhaitais. De plus, la réalisation de ce projet permettra d'acquérir de nouvelles connaissances dans différents domaines, notamment en ce qui concerne un langage informatique et ses possibles utilisations. Cette création permet de faire quelque chose de concret grâce aux connaissances que nous avons acquises depuis le début de l'année, et que nous acquerrons grâce à nos recherches personnelles. Se lancer là-dedans dès notre première année est, selon moi, une excellente expérience ; nous faisant ainsi rentrer dans le vif du sujet.

2.1.2. GROLIER Paul

En plus du secteur qui m'a principalement attiré vers une école comme EPITA, c'est à dire la sécurité informatique, j'aime à penser que certaines des histoires que j'invente pourraient prendre vie sous la forme la plus accessible et intéressante qui soit pour tous, le jeu. Aussi, créer des jeux m'a toujours intéressé. Les jeux vidéo apportent une plus grande palette de possibilités dans la création d'un jeu par l'apport de sons, de la formation d'univers distincts et totalement différents du nôtre et par l'interaction que nous pouvons avoir dans ceux-ci. Concevoir un jeu vidéo représente dès lors une des meilleures manières de s'exprimer en donnant forme à notre imagination. Cela reste pourtant difficile à concrétiser à cause de la complexité des codes et langages à maîtriser. Composer notre propre jeu vidéo est donc une excellente opportunité pour enrichir nos connaissances dans les langages que nous étudions et aussi pour nous exprimer.

2.1.3. LAVENU Stevie

Etant passionné par les jeux vidéo, ce projet est le moment idéal pour exercer mes passions pour le jeu vidéo et l'informatique. J'ai déjà programmé un jeu en groupe lors de la terminale, dans le cadre de la spécialité ISN. Par ailleurs, le modèle type de ce jeu (*Undertale*) est mon jeu favori. Ayant déjà codé un jeu, cela facilitera la création des collisions et des interactions entre les différents objets. J'ai aussi quelques connaissances en musiques et effets sonores, ce qui permettra au jeu d'être plus interactif. Les membres du groupe sont tous impliqués dans le projet. De plus nous sommes tous assez proches ce qui permettra une bonne collaboration ainsi qu'une écoute fine de l'autre.

2.1.4. RUIZ Hugo

Je suis depuis petit un grand fan de jeux vidéo, j'ai joué sur énormément de plateformes et de consoles différentes. Depuis l'âge de quatorze ans, mon oncle m'a offert mon premier ordinateur sur lequel j'ai découvert la plupart de mes loisirs actuels. J'aimerais profiter de pouvoir développer un jeu vidéo afin d'approfondir mes connaissances et comprendre comment fonctionne réellement un jeu vidéo. Je compte aussi profiter de la découverte de Unity pour découvrir un outil qui pourra m'être utile dans mon futur autant professionnel comme d'étudiant. Celui-ci sera mon premier "vrai" projet de programmation, j'espère donc pouvoir en tirer un maximum de connaissances et d'expérience.

2.2. Le principe du jeu

Nous avons choisi de prendre un adolescent comme personnage principal. Intéressé par l'EPITA, il décide de se rendre à une journée portes ouvertes de l'école sur le campus de Toulouse. Celui-ci suit donc le parcours qui a été prévu pour l'évènement. La visite prend fin, mais il n'arrive pourtant pas à s'en contenter : il désire en voir plus. Sa volonté ne fait que grandir, captivé par le reste de cet établissement qui lui est encore inconnu... Le lycéen commence à s'aventurer discrètement vers la zone qui était interdite : le *Donjon*.

C'est, la boule au ventre, que le futur étudiant s'engage dans l'escalier descendant de cette fameuse tour. En bas ne se trouve qu'une vieille porte dont il se rapproche instinctivement. Il prend la décision de l'ouvrir, se sentant comme attiré par elle, et s'enfonce

dans la pièce plongée dans une obscurité des plus totales. Il y avait si peu de lumière qu'il n'avait pas remarqué le fossé qui se trouvait juste à ses pieds… L'atmosphère ne le rassurait pas. C'est en rebroussant chemin qu'il perd l'équilibre et tombe dans le monde d'Epitale ...

Le jeu se basera sur la manière dont le personnage principal arrivera à rentrer chez lui, dans le monde réel. Sur son chemin, il rencontrera certains personnages qui lui feront découvrir une partie de l'univers dans lequel il se trouve. D'autres préféreront lui mettre des bâtons dans les roues : il aura à choisir entre les épargner, ou les tuer.

2.3. La situation du projet

Pour notre deuxième soutenance, nous avions prédit un certain avancement et nous avions réussi à respecter nos engagements.

	Emma	Hugo	Paul	Stevie
Graphismes			30%	
Gameplay	30%			
Effets sonores				
Site Web		90%		
Interface				40%

Fig. 1 : Avancement des tâches lors de la première soutenance.

	Emma	Hugo	Paul	Stevie
Graphismes			70%	
Gameplay	70%			
Effets sonores	50%	50%	50%	50%
Site Web		90%		
Interface				80%

Fig. 2 : Avancement prévu lors de la deuxième soutenance

Voici ici la répartition des tâches selon les membres du groupe.

	Emma	Hugo	Paul	Stevie
Graphismes	50%		50%	
Gameplay				60%
Effets sonores	50%		50%	50%
Site Web		90%		
Interface		40%		40%

Fig. 3 : Avancement des tâches lors de la deuxième soutenance

3. Avancement du projet

3.1. Les effets sonores

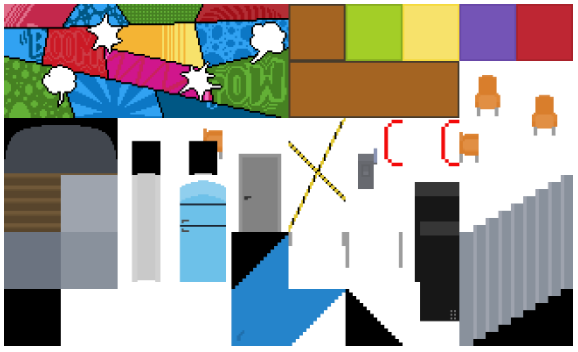
Nous avons choisi d'utiliser un logiciel gratuit afin de créer et modifier les effets sonores de notre projet. Lorsque nous parlons d'effets sonores nous faisons référence aux musiques de fond et aux différents effets de bruitages de notre jeu, que ce soit pour les divers menus ou encore pour l'aventure.

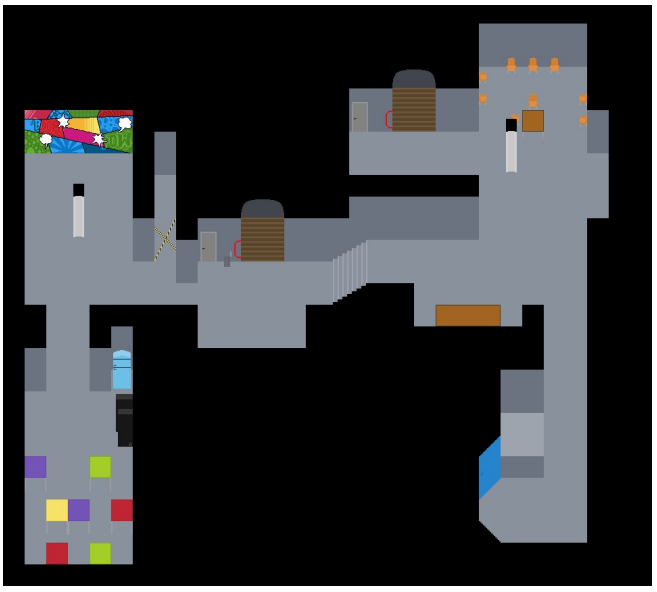
Nous nous sommes renseignés sur la manière d'intégrer des musiques sur les maps et y avons ajouté certaines d'entre elles sur les maps déjà disponibles. Nous avons choisi de mettre des musiques libres de droits, à défaut de les avoir fait nous-même. Lorsque nous serons plus avancés dans le jeu, et donc dans les cartes, nous pourrons ainsi joindre d'autres bandes sons.

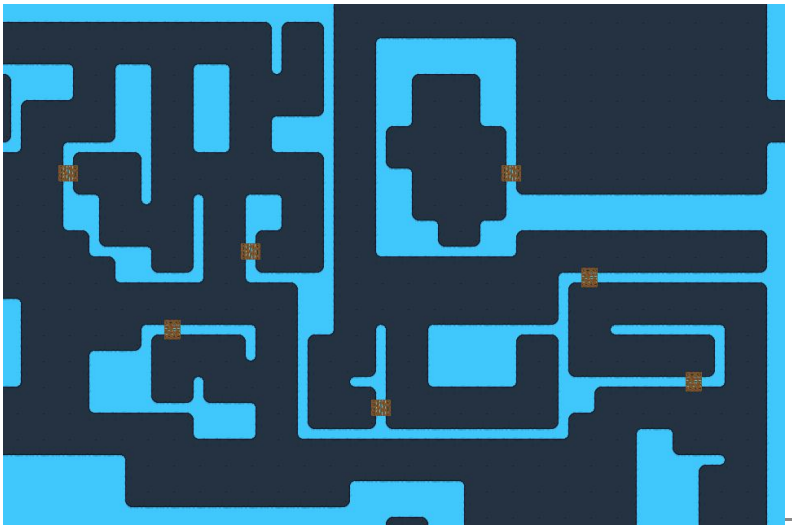
En ce qui concerne tout ce qui est bruitage, nous avons déjà commencé à enregistrer quelques sons qui nous serons utiles afin d'améliorer la fluidité du jeu ainsi que toutes les animations. Nous préférons les y adhérer quand nos plans seront terminés, car nous voulons en premier lieu nous concentrer sur le plus important.

3.2. Les graphismes

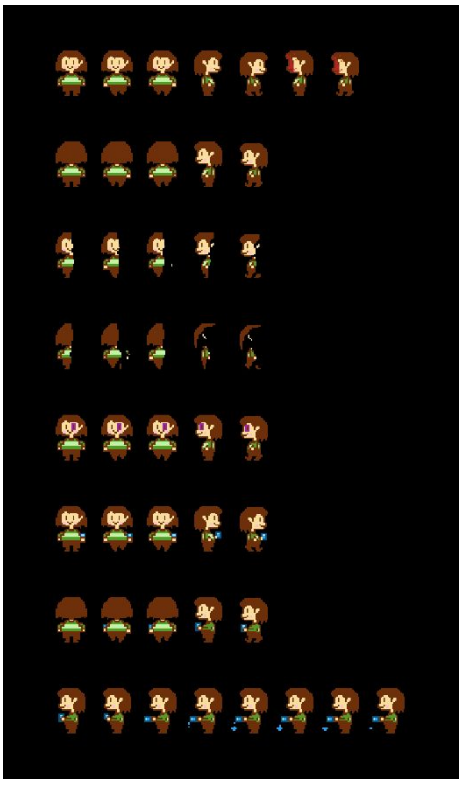
En ce qui concerne les graphismes, les décors du jeu sont inspirés directement d'*Undertale* et sont donc réalisés en pixel art. Nous avons donc pris des cours d'infographie via le site Gamecodeur.fr qui propose entre autres pléthore de cours, ateliers et méthodes d'apprentissage sur le codage de jeux vidéo. Aussi, jusqu'à la première soutenance, nous avions utilisé le logiciel GraphicsGale pour la réalisation de nos cartes et autres éléments de décor, mais les limites de ce logiciel nous ont poussé à opter pour un autre logiciel bien plus pratique, PyxelEdit. Grâce à celui-ci, nous avons déjà pu réaliser un ensemble de « tilemaps », entendez par là, des grilles de tuiles. Le principe est simple à comprendre. A la manière de carreleurs, nous collons bout-à-bout des tuiles de la grille pour réaliser un décor, une carte, etc. Voici quelques exemples de tilemaps et le résultat obtenu grâce à elles.





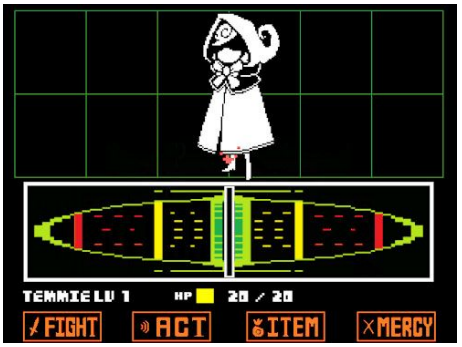


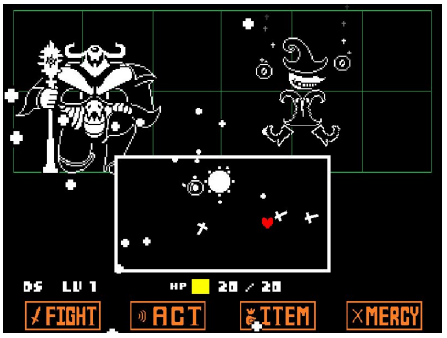
Nous avons également pu dessiner des feuilles de sprites. Nous appelons « sprite », de l'anglais « lutin », un élément graphique capable de se déplacer dans un décor. Un sprite est donc très souvent un personnage de jeu. Une feuille de sprite représente alors les diverses poses que notre personnage peut prendre lorsqu'il marche, cours, s'assoit ou tient un objet en particulier. Certaine images mises en relation peuvent d'ailleurs ainsi former des animations de notre personnage. Voici un exemple de feuille de sprite.



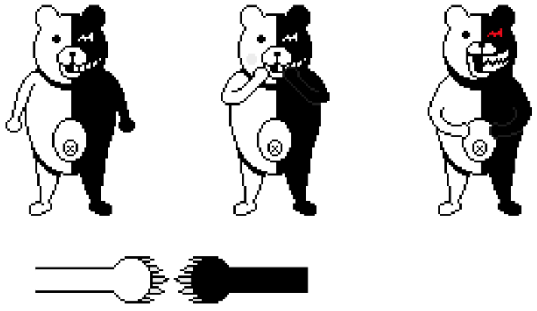
On peut voir différentes vues de notre personnage dans des situations distinctes que ce soit s'il marche, s'il renverse de l'eau, s'il est éclairé d'une lumière vive ou plongé dans le noir.

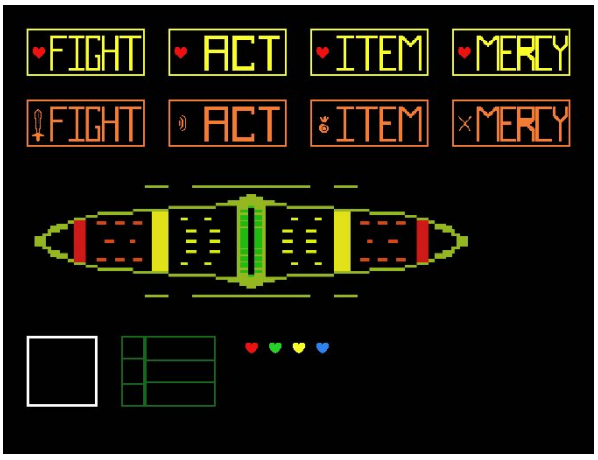
De plus, notre jeu est marqué du fait que le personnage peut parfois rencontrer des monstres, ennemis, qui lui barreront la route et qu'il devra combattre par les mots ou par les armes. Sachant que l'interface change lors d'un passage vers un combat, nous devions donc aussi en dessiner les graphismes. En principe, toujours en s'inspirant du jeu *Undertale*, l'interface de combat se doit de ressembler à cela.





Lors d'un combat, nous sommes représentés par un cœur dans *Undertale*, une gemme dans notre jeu, et ledit objet se trouve dans une « cage » telle une arène dans laquelle il devra esquiver les attaques adverses comme montré par l'image de droite ci-dessus. Pour ce qui est de notre attaque, nous disposons d'une jauge oblongue et une barre verticale s'y déplaçant horizontalement. Nous devons alors cliquer pour stopper la barre le plus au centre possible de la jauge pour infliger un maximum de dégâts à l'adversaire. Voici des exemples de graphismes réalisés pour l'interface de combat.

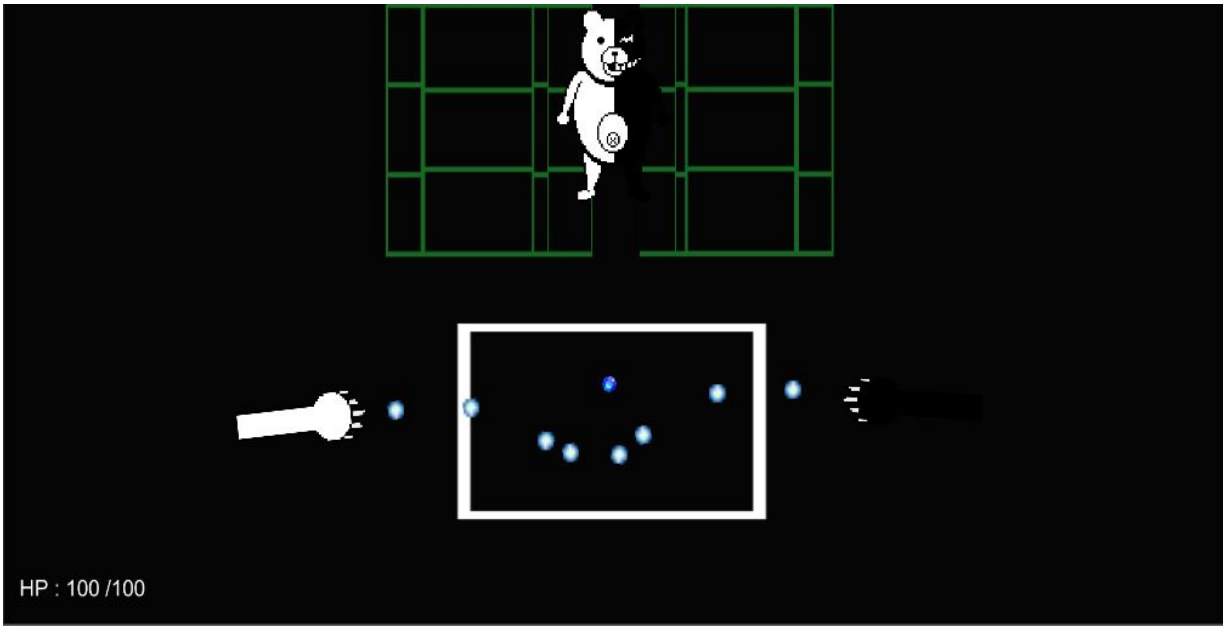




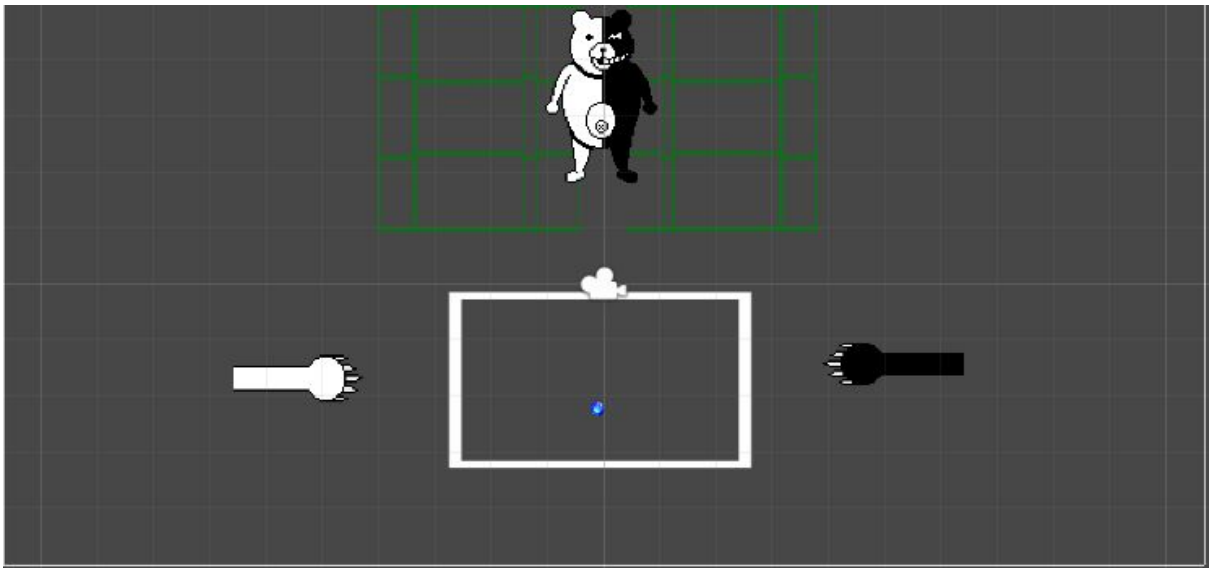
Notre jeu se base sur une palette de références diverses et variées qui nous inspirent et que nous souhaitons retranscrire dans celui-ci avec par exemple la présence de monokuma, un ours au corps mi-sympathique mi-démoniaque, en tant qu'adversaire que notre personnage pourra combattre dans le jeu. Nous pourrons aussi les utiliser à titre d'adjuvants pour guider le personnage principal, afin qu'il ne soit pas perdu durant sa quête.

3.3. Les combats

Les combats dans *Undertale* sont particuliers. En effet ils mélangent le space shooter et le système de JDR classique. On a donc essayé de reproduire ce système de gameplay dans notre jeu. En voici une image :



On peut voir le personnage, l'ennemi et enfin les projectiles ainsi que la vie du joueur.

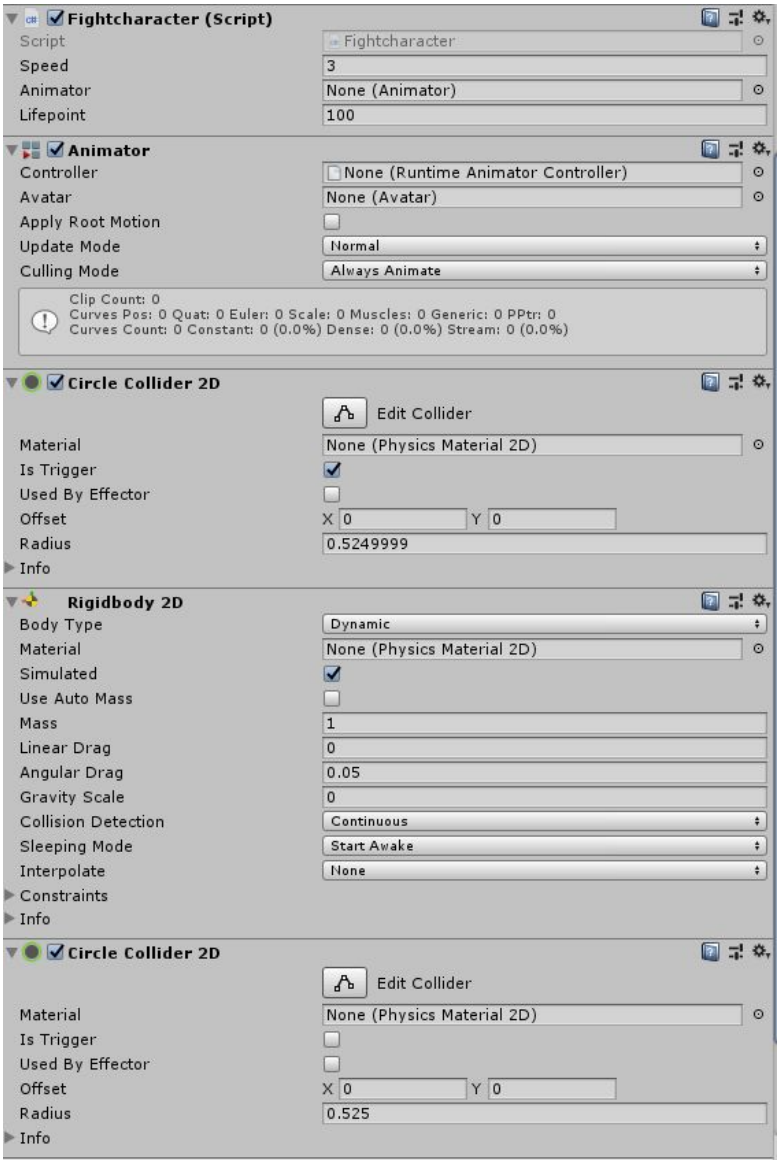


Nous avons d'abord commencé par ajouter les éléments sur la scène. Il y a donc comme éléments : le boss, le décor, les deux bras, la cage blanche, et enfin le personnage.
Le boss et le fond sont seulement des images tandis que les bras, la cage et le personnage vont être travaillés.

Effectivement, les bras ont pour objectif de tirer des projectiles sur le personnage et donc de se tourner vers lui pour projeter le projectile vers lui. Ce projectile aura des propriétés propres à lui-même.

Le personnage lui devra bouger, esquiver, mais il ne pourra pas sortir de la cage. Mais les projectiles devront cependant la traverser.

Tout d'abord étudions le personnage …
Voici son inspecteur :



On peut remarquer que le personnage a 2 circles collider ainsi qu'un script, un rigidbody et un animator.

L'animator et le Rigidbody serviront aux mouvements du personnage ainsi qu'à ses collisions. Ce sera aussi le cas du second Circle Collider qui lui n'a pas le Trigger coché. Il interagira donc avec la cage. Pour ce qui est de l'autre Circle collider, nous détaillerons cela dans la partie sur les projectiles.

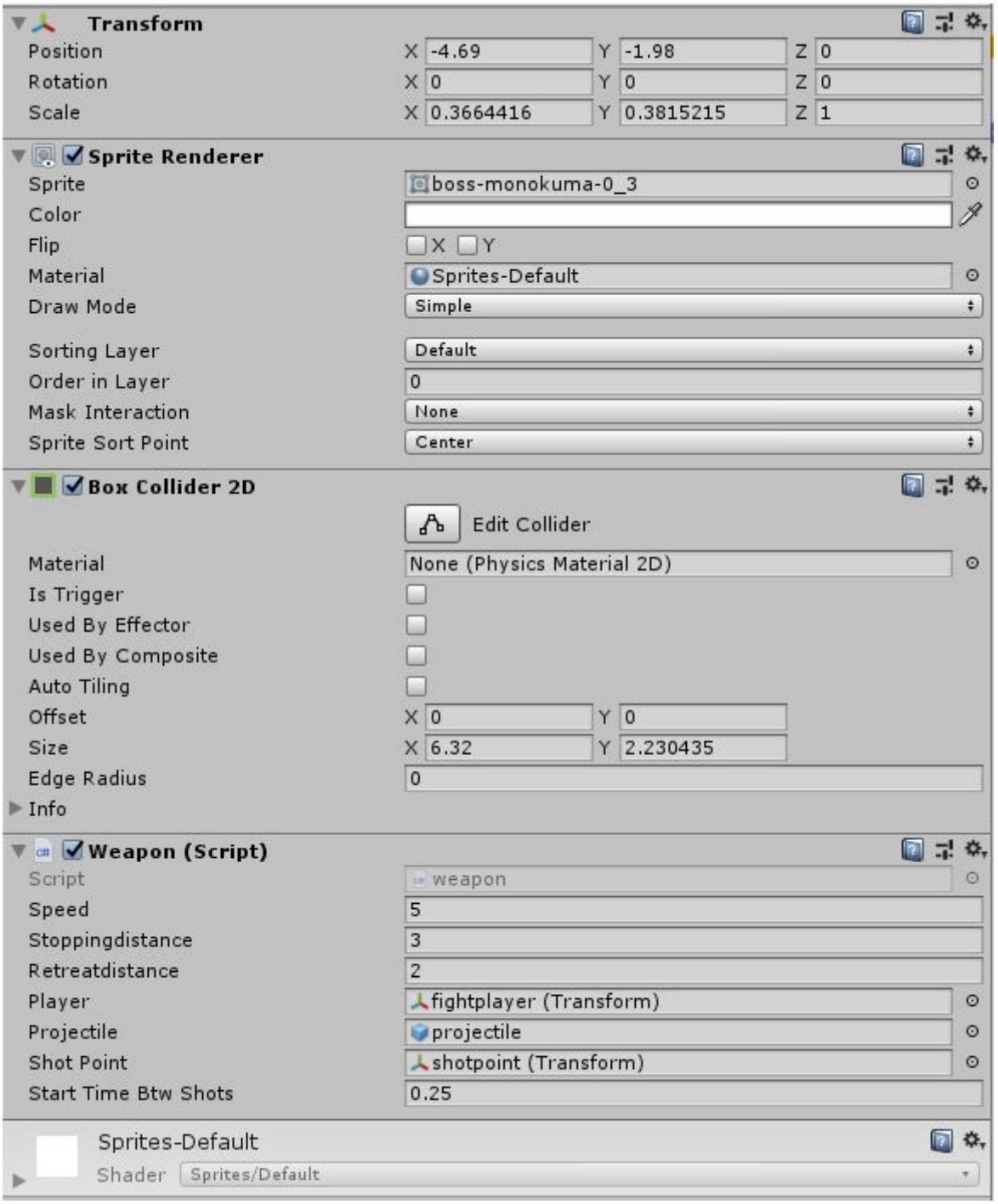
Regardons donc le script du joueur :

```
public class Fightcharacter : basicobjects
{

    public int lifepoint;

    // Start is called before the first frame update
    protected override void Start()
    {
        base.Start();
    }

    // Update is called once per frame
    protected override void Update()
    {
        getInput();

        base.Update();

        if (lifepoint <= 0)
        {
            Destroy(gameObject);
        }
    }

    public void TakeDamage(int damage)
    {
        lifepoint -= damage;
    }
```

On va donc expliquer les ajouts faits depuis le dernier rapport (sachant que base et GetInput() étaient 2 fonctions permettant le déplacement que l'on ne va donc pas réexpliquer). Nous pouvons donc voir l'ajout d'une nouvelle variable lifepoint qui définira les points de vie du personnage. La fonction TakeDamage sert à enlever des points de vie au personnage. Elle sera appelée par les projectiles, ou d'autres objets qui causeront des dégâts.

Nous allons maintenant expliquer le fonctionnement des armes.

Voici leur inspecteur :



Nous allons donc étudier le script des armes :

```
public class weapon : MonoBehaviour
{
    public float speed;
    public float stoppingdistance;
    public float retreatdistance;

    public Transform player;
    public GameObject projectile;
    public Transform shotPoint;

    private float timeBtwShots;
    public float startTimeBtwShots;

    // Start is called before the first frame update
    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player").transform;

        timeBtwShots = startTimeBtwShots;
    }

    // Update is called once per frame
    void Update()
    {
        Vector3 difference = player.position - transform.position;

        float rotZ = Mathf.Atan2(difference.y, difference.x) * Mathf.Rad2Deg;

        transform.rotation = Quaternion.Euler(0f, 0f, rotZ);

        /*if (Vector2.Distance(transform.position, player.position) > stoppingdistance)
        {
            transform.position = Vector2.MoveTowards(transform.position, player.position, speed * Time.deltaTime);
        }
        else if((Vector2.Distance(transform.position, player.position) < stoppingdistance &&
        Vector2.Distance(transform.position, player.position) > retreatdistance)
        || transform.position.x >= player.position.x -5)
        {
            transform.position = this.transform.position;
        }
        else if (Vector2.Distance(transform.position, player.position) < retreatdistance)
        {
            transform.position = Vector2.MoveTowards(transform.position, player.position, -speed * Time.deltaTime);
        }*/


        if(timeBtwShots <= 0)
        {
            Instantiate(projectile, shotPoint.position, transform.rotation);
            timeBtwShots = startTimeBtwShots;
        }
        else
        {
            timeBtwShots -= Time.deltaTime;
        }
    }
```

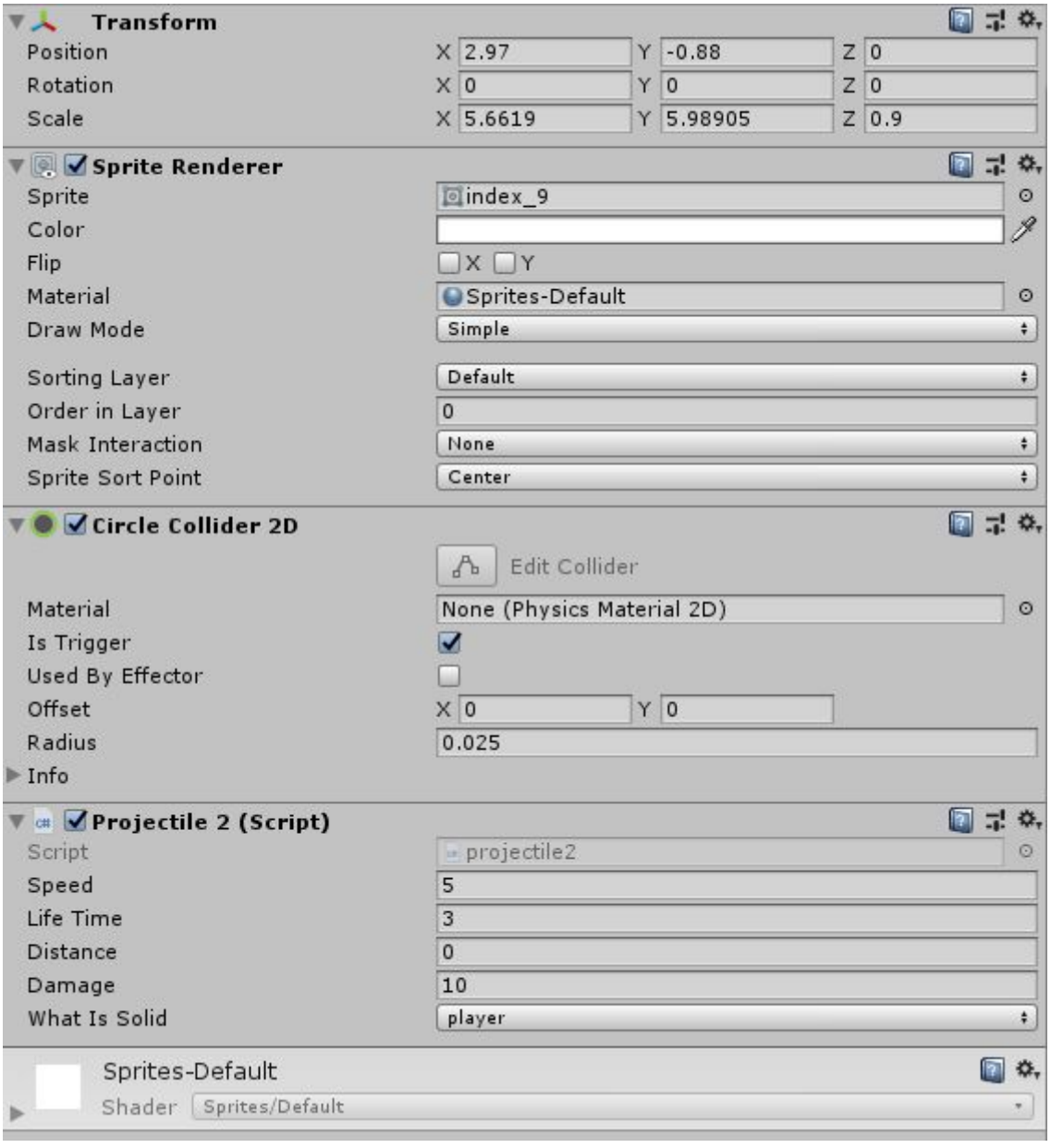
Ici on parle donc du script des “bras” visibles sur la scène. On peut voir un code en commentaire que nous expliquerons dans un second temps.

Tout d’abord nous commençons dans Start où l’on récupère les coordonnées de l’objet ayant pour tag “Player” que l’on aura ajouté nous même. On ajuste ensuite la cadence de tirs avec TimeBtwShots (Time between shots).

Voici maintenant le code pour orienter la “main” vers nous. On relève d’abord la différence de position entre les deux objets. Suite à cela nous calculons l’angle dont l’objet

devra se tourner. Enfin nous appliquons la rotation à la main afin qu'elle se tourne vers le joueur.
Ensuite on crée notre projectile avec la même orientation que nous et on met un temps d'attente entre chaque tir.

Voyons donc maintenant les projectiles :



Ici nous allons parler du script du projectile :

```
public class projectile2 : MonoBehaviour
{

    public float speed;
    public float lifeTime;
    public float distance;
    public int damage;
    public LayerMask whatIsSolid;

    // Start is called before the first frame update
    void Start()
    {
        Invoke("DestroyProjectile", lifeTime);
    }

    // Update is called once per frame
    void Update()
    {
        RaycastHit2D hitInfo = Physics2D.Raycast(transform.position, transform.right, distance, whatIsSolid);
        if (hitInfo.collider != null)
        {
            if (hitInfo.collider.CompareTag("Player"))
            {
                Debug.Log("PLAYER MUST TAKE DAMAGE !");
                hitInfo.collider.GetComponent<Fightcharacter>().TakeDamage(damage);
            }
            DestroyProjectile();
        }

        transform.Translate(Vector2.right * speed * Time.deltaTime);
    }

    void DestroyProjectile()
    {
        Destroy(gameObject);
    }
}
```

Nous avons ici plusieurs variable dont la lifetime , la distance et le whatIsSolid qui sont particulierement interessants. Lifetime définit le temps de “vie” du projectile. Dans Start nous appelons la fonction DestroyProjectile après la durée de lifetime. En d’autres termes, à partir de son apparition, il disparaîtra par exemple au bout de 10 secondes si on défini lifetime sur 10. Ce système permet de ne pas surcharger le jeu avec des objets inutiles se promenants hors caméra.

C’est après que ça devient plus technique :

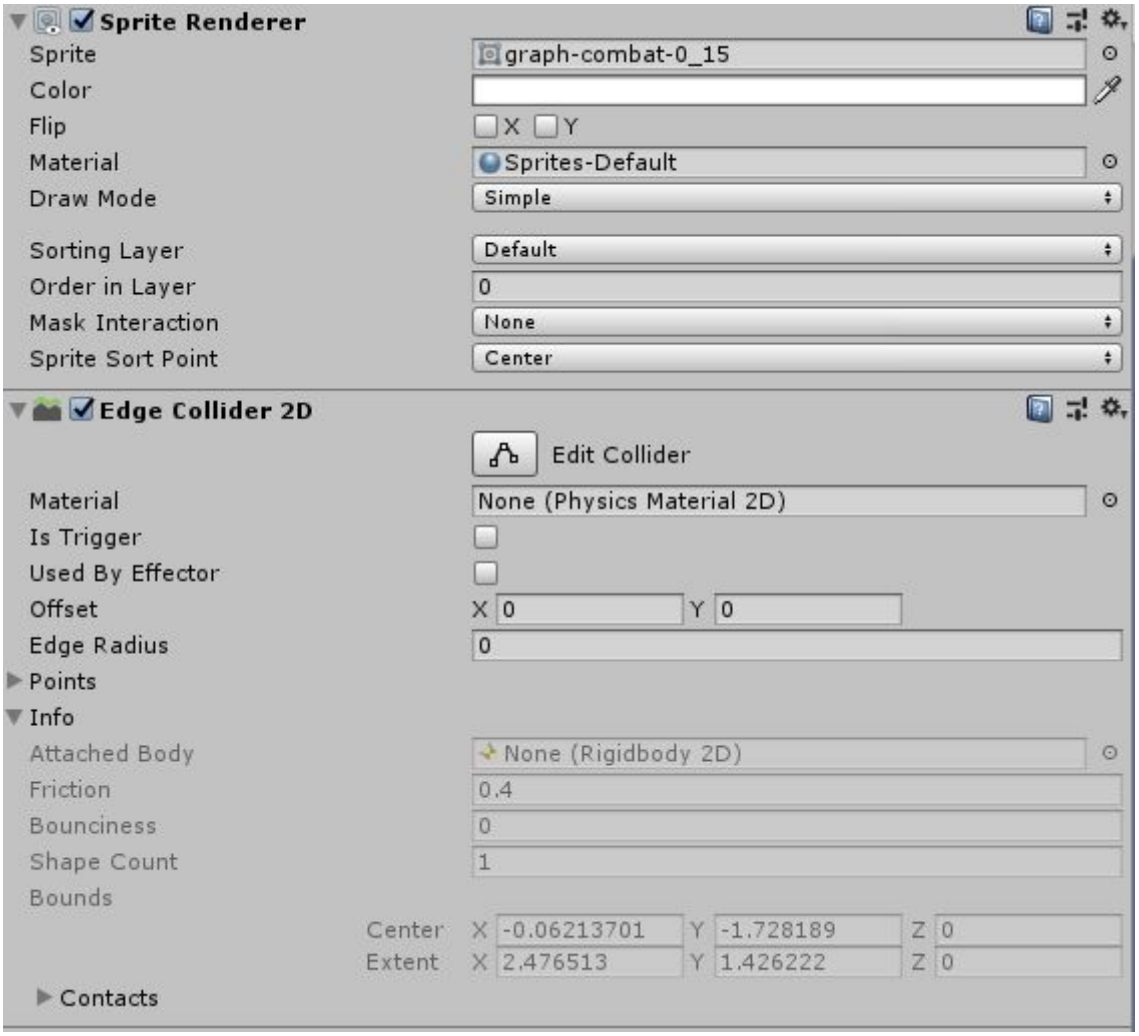
On va créer une variable qui permet de récolter des infos sur ce qu’il heurte en fonction de la distance de l’objet, de la position du projectile, ainsi que de l’appartenance de l’objet a un layer particulier. S’il se cogne à un objet alors on vérifie que l’objet porte le Tag “Player”.

On lance donc un debug pour signaler de faire des dégâts et on lance la fonction TakeDamage() du player avec les dégâts entrés en paramètre. On récolte donc des informations en continue sur le statut du projectile. Ensuite on bouge le projectile de façon basique.

Enfin il y a la fonction DestroyProjectile() qui détruira simplement l’objet.

Donc le collider avec le trigger activé sur le personnage sert à être détecté par le projectile et ainsi permettre au tout de fonctionner.

Voyons rapidement la cage et le texte.
Ici nous pouvons voir la cage :



Cela représente donc un simple mur.

Voici le script du texte :

```
public class life : MonoBehaviour
{
    Text txt;
    private int health;
    public GameObject player;
    // Start is called before the first frame update
    void Start()
    {
        txt = gameObject.GetComponent<Text>();

        txt.text = "HP : " + health + " /100";
    }

    // Update is called once per frame
    void Update()
    {
        health = player.GetComponent<Fightcharacter>().lifepoint;
        txt.text = "HP : " + health + " /100";
    }
}
```

On récupère les points de vie du personnage et on les applique au texte qui va les afficher à l'écran.

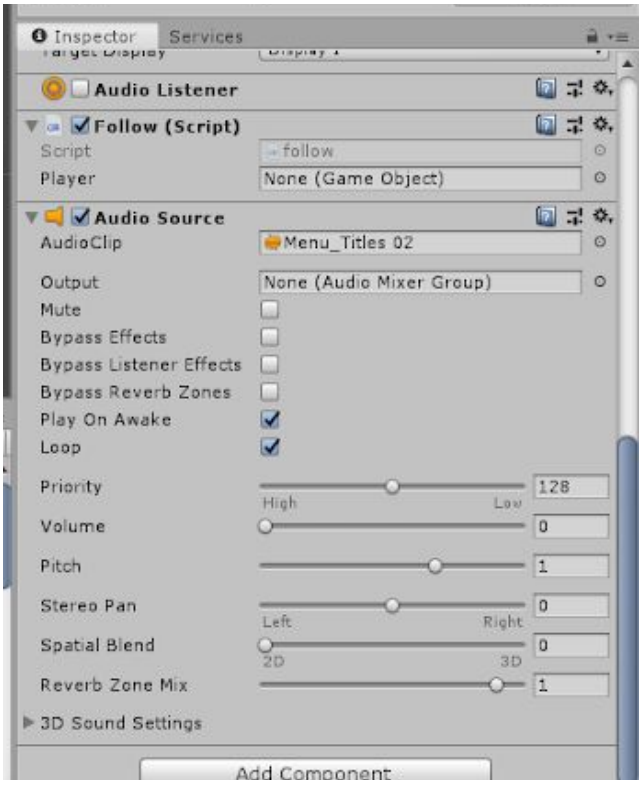
3.4. Les menus

Les menus sont une partie importante du jeu car ils permettent à l'utilisateur de sauvegarder sa partie, changer les paramètres, commencer une nouvelle partie ou encore quitter le jeu. Nous avons regardé des tutoriels sur le site officiel d'Unity pour comprendre comment cela fonctionnait. Nous devions créer une nouvelle scène sur Unity pour chaque menu. Nous avons alors choisi une image de fond pour le menu principale qui pour le moment nous permet de lancer le jeu et de le quitter mais qui dans un futur devrait permettre de charger les parties sauvegardées et de modifier les paramètres du jeu comme le volume ou les touches de déplacement.

Pour le moment le menu reste assez simple mais étant donné nos lacunes sur Unity, chaque pas est une découverte. Même si les menus ne sont pas si compliqués à réaliser, l'apprentissage met un peu de temps ce qui explique le peu développé dans cette partie du jeu.

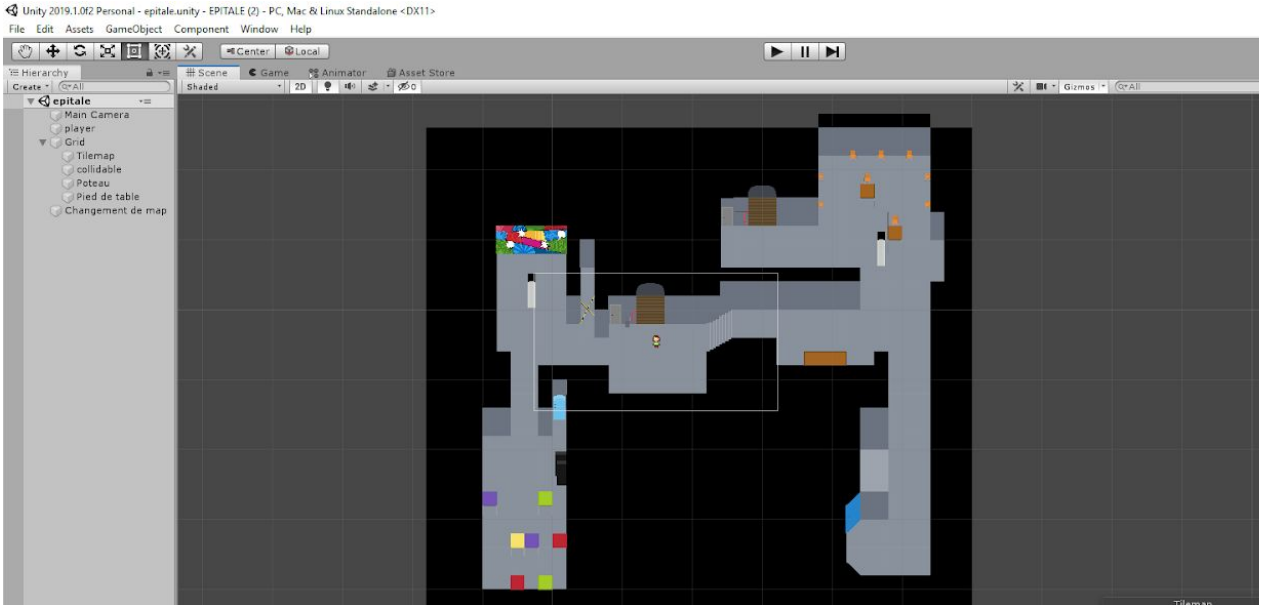
Screenshot du menu principal

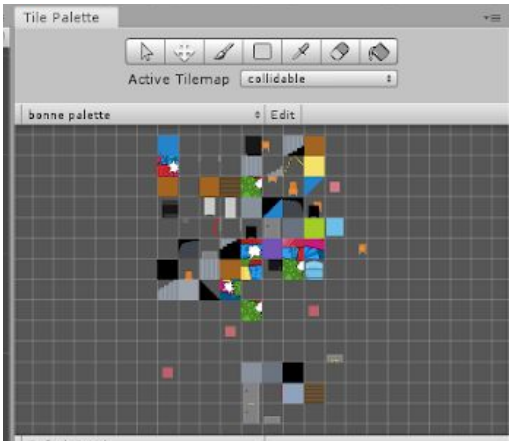
Nous avons inclus une musique de fond pour le menu que nous avons téléchargé depuis le site d'Unity où on peut retrouver une grande variété de musiques, sons et bruitages différents. Nous avons inclus cette musique grâce à un "audio source" ou source audio en français qui était ajouté comme un composant de notre caméra. Nous avons ajouté à celui-ci une musique qui est reproduite en boucle.



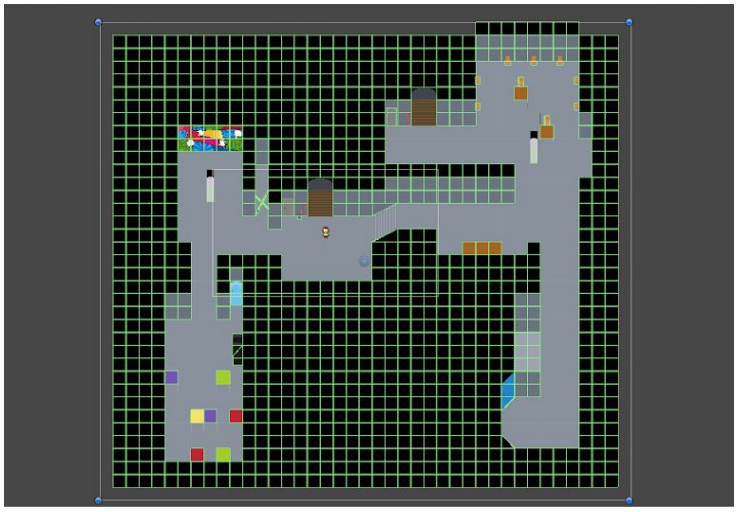
3.6. Les cartes

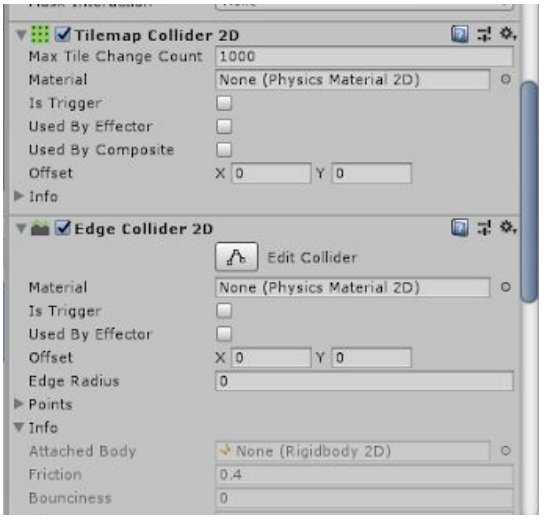
Pour la réalisation des cartes nous avons utilisé une grille qui nous permet de placer des carrés de carte comme si on faisait un puzzle. Lorsque nous avons réalisé l'image en PNG avec tous les éléments de la carte nous les avons découpé à une taille spécifique (205x205 pixels). Ces petits carrés sont ensuite insérés sur différentes couches de la grille. Chaque couche aura une fonction spécifique. Par exemple, la couche "collidable" contiendra tous les éléments du décor que notre personnage ne peut pas traverser. La couche "tilemap" correspond aux éléments sur lesquels notre personnage peut passer.





Nous avons expliqué juste avant que certains objet du décor ne laissent pas passer notre personnage. Pour cela nous avons utilisé un composant que l'on peut ajouter aux couches de cette grille appelée tilemap collider qui permet de rendre compte de la collision du personnage avec un élément de cette couche. Cette collision peut ensuite être traitée comme un obstacle pour le personnage comme les murs ou les tables ou encore les chaises. Nous pouvons aussi utiliser ce composant pour réaliser une action lors de la collision de notre personnage avec la tuile en question. Nous avons donc utilisé cette fonctionnalité pour changer de scène et donc pouvoir déplacer notre personnage entre les différentes scènes.





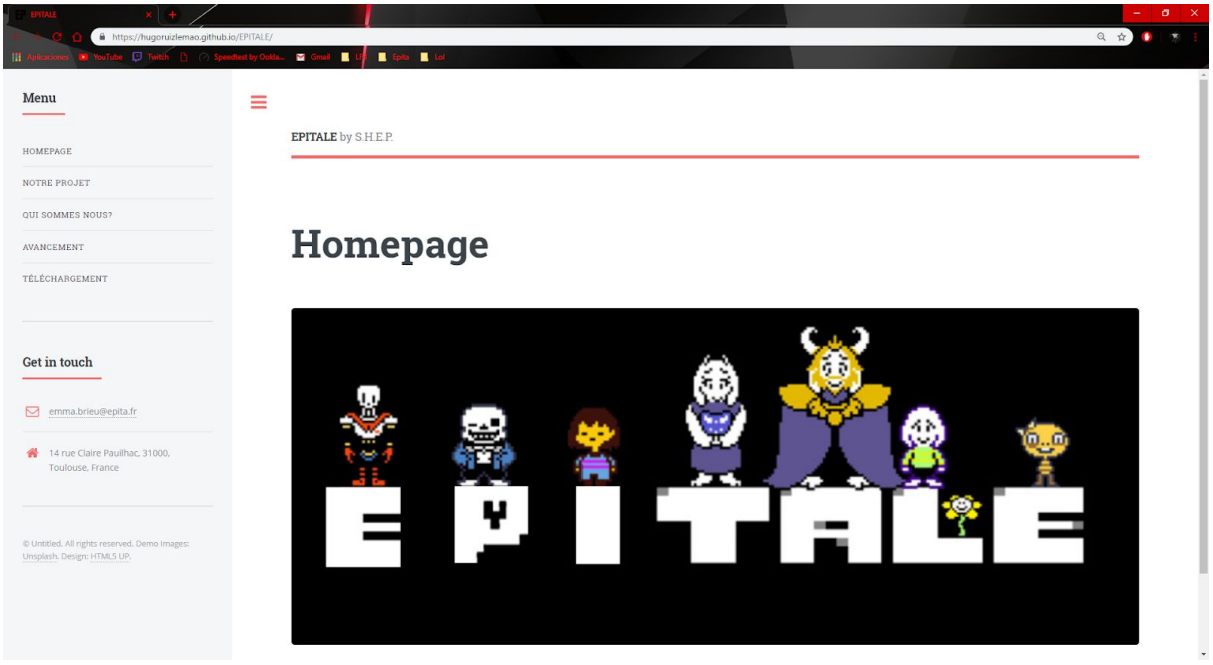
Sur l'image précédente on observe en vert tous les carrés correspondant aux obstacles que le personnage ne peut pas traverser. Comme certaines parties de l'escalier occupent une demie-tile nous avons utilisé le composant Edge collider 2d qui permet de délimiter des contours et nous a permis d'être plus précis lors de la confection de l'escalier central.

Pour pouvoir changer de scène et donc de carte il fallait créer un box collider au dessus des deux cages d'escalier qui permettent de détecter quand le personnage arrive sur cette zone et donc le déplacer sur une autre scène. Sur cet aspect là du projet, pour la prochaine soutenance nous aimerions terminer toutes les cartes, améliorer les transitions entre les scènes pour les rendre moins brusques, ajouter les bruitages et ajouter plus d'éléments de décors sur nos cartes actuelles.

4. Site Web

Le site web n'a pas reçu de modifications depuis la dernière soutenance car nous l'avions bien avancé et il ne restait plus qu'à le mettre à jour au fur et à mesure des avancées du projet. Pour résumer nous avons suivi des cours sur Openclassroom à propos du langage HTML et le CSS qui permettent de respectivement donner un fond et une forme à un site web. Après avoir mis en place le fond du site, en faisant quelque chose de simple, pour rendre notre site plus attractif, nous avons téléchargé la mise en forme en CSS depuis un site web qui les proposait de façon gratuite: ***https://html5up.net*** .

On retrouve sur notre site une page d'accueil, une présentation du projet, une présentation du groupe, l'avancement soutenance après soutenance du projet, un onglet téléchargement ou nous retrouvons les indications à suivre pour la réalisation du projet et les rapports de soutenance. Nous pourrons également y retrouver le jeu terminé dès que possible.



Screenshot du site web de EPITALE

5. Prévisions pour la future soutenance

Nous avons maintenant réussi à répartir les tâches en arrangeant l'intérêt de chacun des membres de l'équipe. L'organisation n'ayant plus de failles désormais, nous pouvons prévoir l'avancement qui sera atteint lors de la soutenance finale.

	Emma	Hugo	Paul	Stevie
Graphismes	100%		100%	
Gameplay				100%
Effets sonores	100%		100%	100%
Site Web		100%		
Interface		100%		

Fig. 5 : Avancement des tâches lors de la soutenance finale.

6. Conclusion

Pour conclure, nous pouvons affirmer que nous avons atteint les objectifs que nous nous étions fixés. Nous avons pris plaisir à réaliser ce projet, mais surtout d'observer l'évolution de celui-ci. Selon nous, c'est assez gratifiant de voir que nos efforts ont payés, et d'appliquer concrètement ce que nous avons travaillé pour notre projet, même si nous avons encore du pain sur la planche.

Enfin, nous pouvons dire que nous sommes conscients qu'il nous reste pas mal de travail sur les graphismes, malgré le fait que nous ayons bien avancé là-dessus. Nous avançons doucement mais sûrement. Grâce à notre avancement, nous avons une idée précise sur la façon dont les différentes parties vont interagir ensemble…