S2 - Promotion 2024 Toulouse

Runningman

Rapport de soutenance 1

Encadrants de projet : Remi Vernay et Philippe Roussille

N. De Gregorio, T. Dez, M. Leonardon, J. Levasseur

Table des matières

Introduction

Dans le cadre du projet informatique du deuxième semestre qui a lieu pendant notre première année de cycle préparatoire à l'EPITA, nous avons l'occasion de créer un jeu vidéo. Celui-ci sera noté sous forme de soutenance, la première ayant lieu le 10 mars 2020. Ce rapport de première soutenance permettra donc de rendre compte de notre avancé sur le projet ainsi que nos difficultés rencontrées.

Comme vu dans notre cahier des charges, nous réalisons un jeu de plateforme en 2D nommé Runningman se rapprochant de Super Mario Bros. Afin de mener à bien ce projet nous nous sommes fixé un planning à respecter avec des tâches à réaliser avant chaque soutenance.

Pour cette première soutenance nous devions finir à 75% notre premier niveau. Ce qui consiste à avoir commencer l'interface du menu principal, conçu la map du premier monde accompagnée d'ennemis et de notre personnage principal.

Depuis la création du cahier du charge nous avions donc deux mois avant la prochaine soutenance. Afin d'être efficace dans notre travail nous nous sommes donc séparé les tâches avec un système de responsable et de suppléant comme nous pouvons le voir ci-dessous.

	Création de l'interface du menu principal	Création du premier niveau	Création des ennemis	Création du personnage principal
N. De Gregorio	Responsable	Suppléant		
T. Dez			Suppléant	Responsable
M. Leonardon		Responsable		Suppléant
J. Levasseur	Suppléant		Responsable	

Figure 1 - Tableau de répartition des tâches

Ainsi chacun savait ce qu'il avait à faire et pouvait s'organiser de la façon dont il le souhaitait. Cependant, afin d'avoir un regard sur l'avancé globale du groupe, nous avons choisi de nous donner rendez-vous chaque lundi afin de travailler ensemble et régler les problèmes rencontrés.

Grâce à cette méthode de travail nous avons pu avancer dans cette première partie du projet avec une bonne ambiance de travail au sein du groupe ainsi qu'une confiance sur le travail fourni par les autres. Nous pouvions par ailleurs toujours en cas de difficultés apporter de l'aide aux différents membres du groupe grâce au système de suppléant.

I – Reprise du cahier des charges

Durant la réalisation de notre projet nous avons pu nous rendre compte de chose auquel nous n'avions pas pensé lors de la conception de notre cahier des charges. En effet, notamment sur les ennemis que nous allions créer, l'interface du menu principal et sur la création de notre site internet.

Nous avions choisi d'adapter l'ennemi par rapport au monde dans lequel il se trouvait. Par exemple nous avions décidé que lors du premier niveau représenté par un décor de ville, les ennemis représenteraient des rats. Cependant nous n'avions pas pensé à comment nous allions désigner nos différents personnages. Nous avons essayé de le dessiner pixel par pixel mais cela prenait trop de temps. En effet, afin que celui-ci puisse avoir du mouvement et ne pas être fixé sur une seule image, nous devions créer plusieurs images du rat tout en gardant les proportions de départ. Cela nous prenait trop de temps, nous nous sommes donc rétractés sur des personnages déjà tout faits ayant plusieurs mouvements prédéfinis.

Nous nous sommes aussi trompés d'un point de vue de vocabulaire, nous avions utilisé le terme d' « architecture du jeu » pour parler de l'interface du menu principal. Par ailleurs, l'interface du menu principal ne suivra pas le même design qu'annoncé dans le cahier des charges. Celui-ci présentera en évidence le nom de notre jeu, Runningman accompagné d'une image de notre personnage principal ainsi que 2 boutons, « New Game » et « Shop », nous retrouverons dans New Game l'accès aux trois niveaux et dans Shop comme prévu l'accès à la boutique permettant l'achat de nouveaux personnages.

Pour la création de notre site web nous avions dans notre cahier des charges dit que nous le réaliserons avec Wix, un site internet permettant de créer son propre site web gratuitement avec facilité. Cependant étant en école d'ingénieurs en informatique, il serait plus judicieux de profiter de ce projet pour commencer à se familiariser avec le langage HTML et CSS.

II – Tâches réalisées

L'interface

Réalisé par Nora De Gregorio

Pour commencer, afin d'accéder à notre jeu nous avions besoin d'une interface d'utilisateur qui est le menu principal. Sur celui-ci est présent le nom du jeu ainsi que notre héros mais aussi trois boutons permettant de commencer une nouvelle partie ou reprendre une partie en cours ou même d'accéder à la boutique.

Figure 2 - Interface du menu principal

Nous avons commencé à travailler l'interface du menu principal que lorsque la création/design du personnage principal étant terminé car celui-ci est présent sur la majeure partie du menu principal. Nous avons donc commencé par créer une scène de type "control", dans celle-ci nous avons ajouté son nœud fils de type HBoxContainer (renommé Menu). Les nœuds BoxContainer permettent l'alignement horizontal ou vertical de chaque caractéristique de notre menu principal. Un des fils du nœud Menu est un TexturRect qui représente le logo soit le nom de notre jeu "Runningman", l'autre fils est un nœud de type VBoxContainer, Buttons qui contiennent les boutons permettant d'accéder aux interfaces.

Remarque : Ceux-ci ont la particularité d'avoir été créé à partir d'un nœud bouton modèle, enregistre comme une scène à part entière ce qui facilitera la création de nouveau de nœuds de ce type. En effet, il suffit de modifier le texte représentant le bouton dans la scène originale, d'enregistrer la nouvelle scène et d'insérer celle-ci dans le nœud souhaite.

Pour continuer, le nœud frère de Buttons est un CenterContainer contenant un TexturRect qui est l'image de notre héros.

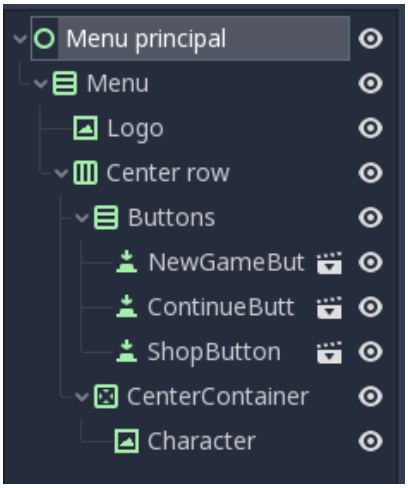

Figure 3 - noeuds utilisés pour l'interface

Le premier monde

Réalisé par Maya Leonardon

Jusqu'à cette première soutenance mon rôle était de commencer à créer le premier niveau.

Chaque niveau correspond à différents mondes ce dernier représente les rues d'une ville, en effet notre personnage court pour rattraper celui qui lui a volé son téléphone rien de plus réaliste que cette scène se produit en ville.

Pour ce faire nous avons utilisé le logiciel Godot, très utile pour créer un jeu de plateforme en 2D.

Godot est un moteur de jeu multiplateforme, c'est-à-dire un logiciel permettant de créer des jeux vidéo qui sont compatibles avec Rider que nous allons utiliser pour écrire le code de notre jeu.

Pour apprendre à utiliser ce logiciel nous avons regardé de nombreux tutoriels expliquant comment créer un jeu vidéo en deux dimensions. De plus pour mieux comprendre certaines fonctionnalités nous nous sommes référés à la documentation Godot sur leur site web qui donne de nombreux exemples.

Tout d'abord pour le design du premier monde il nous a fallu une planche assez complète d'image en format png que nous avons trouvé sur Kenney.com, un site web possédant des milliers de sprites, de modèles 3D et d'effets sonores que l'on peut utiliser gratuitement pour nos projets.

Figure 4 - Planche utilisée pour le niveau 1

Remarque : le format png est primordiale pour Godot car cela permet à l'image qu'elle n'est pas de fond.

Dans un premier temps, Godot fonctionnant exclusivement avec des nœuds, nous avons ouvert une scène avec un Nœud 2D nommé « Principale », puis nous lui avons ajouté un *Sprite*[1] dans lequel nous lui avons inséré la planche d'image. Cependant nous avons remarqué que pour peindre seulement avec des copiés-collés de chaque forme cela allait être assez long.

Dès lors, afin de faciliter la création de notre monde nous avons ouvert une nouvelle scène avec un nœud 2D ainsi qu'un *Sprite* comme dans la première. Puis nous avons inséré la planche d'image dans la case texture du *Sprite*.

Pour que nous puissions sélectionner chaque forme de la planche indépendamment nous avons activé la région du *Sprite* et ainsi fait la coupe automatique de toutes les formes.

De plus, nous avons ajouté un *Staticbody* qui permet la détection des collisions, mais empêche l'objet de bouger en réponse à celle-ci, et enfin à ce *Staticbody* nous ajoutons un nœud enfant *CollisionShape2D* qui va définir où cette collision doit être sur la forme, pour que notre personnage puisse se déplacer sur le sol ou ne pas traverser les murs, par exemple. Par ailleurs, pour les formes qui représentent des plateformes dans les airs nous avons opté pour un *one way collision* comme condition au *CollisionShape2D* afin que le personnage puisse traverser la forme par le bas mais marcher sur le dessus.

Cette deuxième scène que nous avons appelée « Forme » nous l'avons convertie vers un Tileset pour avoir un fichier de format « forme.tres », afin de pouvoir mettre les formes réalisées dans notre première scène.

Enfin dans la première scène « principale » nous avons rajouté un nœud *TileMap*. Ce dernier est une grille utilisée pour créer la présentation/le fond graphique d'un jeu. Il y a plusieurs avantages à utiliser le nœud *TileMap*. Premièrement, cela rend possible de dessiner le layout en « peignant » les tuiles sur la grille, ce qui est beaucoup plus rapide que de placer individuellement chacune des tuiles sous formes de Sprite. Secondement, les tuiles permettent d'avoir des niveaux bien plus larges car elles sont optimisées pour être dessinées en grand nombre.

Ensuite nous avons donc mis le fameux fichier forme.tres dans la case TileSet, et automatiquement les formes réalisées précédemment seront à disposition dans le *TileMap*. Nous n'avons plus qu'à les sélectionner et peindre avec celles-ci notre premier monde.

[1] Element graphique qui peut se déplacer sur l'écran

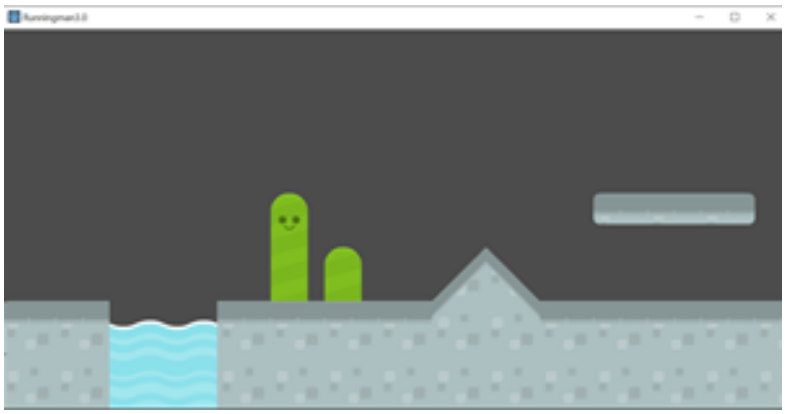

Figure 5 - Extrait du monde 1

Après avoir fini notre monde, nous avons voulu le rendre un peu plus esthétique en ajoutant un paysage en arrière-plan. Pour cela il nous a fallu trois nouvelles images que l'on a superposées. Le but étant que lorsque le personnage se déplace le fond se répète indéfiniment.

Pour se faire nous avons ajouté un nœud a la scène qui se nomme *ParallaxBackground*. Le *ParallaxBackground* est pratique car il s'adapte et se met en fond directement par rapport au monde ou aux personnages présents. Il utilise un ou plusieurs nœuds *ParallaxLayer* pour créer un fond de défilement. Chaque *ParallaxLayer* peut être réglé pour se déplacer à des vitesses différentes par rapport au mouvement de la caméra, ceci peut être utilisé pour créer une illusion de profondeur dans un jeu 2D.

Nous avons donc ajouté trois *ParallaxLayer* à ce dernier. Pour chaque *ParallaxLayer* nous avons rajouté un *Sprite*, le premier se nomme « ciel », le second « nuages » et le dernier « immeubles » nous avons cependant remarqué que les deux premières images sont très petites par rapport à la scène du monde, donc nous avons agrandi ces images dans la région scale[2] où nous avons ensuite mit une valeur de 4 pour les axes x, y de l'image ciel et une valeur de 2 pour l'image nuages. En revanche l'image immeuble est bien trop grande nous mettons alors une valeur de 0,15 pour l'axe x et 0,098 pour l'axe y.

Passons désormais à la répétition du fond au fur et à mesure que le personnage progresse dans le niveau, pour cela nous nous sommes placés dans le nœud *ParallaxLayer* et dans le champ Mirroring et nous avons ajouté sur l'axe x la valeur de la distance pour laquelle l'image va se répéter. De ce fait pour le *ParallaxLayer* du ciel nous avons inséré une valeur de 1440 pour celui des nuages 960 et pour les immeubles 700. Ainsi nos trois images sont bien calées par rapport au monde.

[2] Se référer à la figure 11 suivant le même principe

Figure 6 - Background du premier monde

Enfin pour avoir ce fameux effet de profondeur et de mouvement nous avons ajouté de la vitesse à nos *ParallaxLayer*. Ainsi pour que les immeubles bougent plus vite que les nuages. Nous nous plaçons dans le champ scale des *ParallaxLayer* et a chacun nous modifions les valeurs (x, y) pour :

- Le ciel (0.1, 0.1)
- Les nuages (0.6, 0.6)
- Les immeubles (0.8, 0.8)

Remarque : Il faut décocher la case « Ignore camera Z » sinon le background ne va pas prendre en compte le facteur zoom de la caméra du personnage, ce qui va rendre en effet étrange lors de ses déplacements.

Voici donc la manière dont nous avons créé notre paysage de fond pour le premier niveau avec une illusion de profondeur et de mouvement.

Les ennemis

Réalisé par Johanna Levasseur

Comme convenu dans notre cahier des charges afin d'augmenter la difficulté de notre premier monde nous avons décidé d'y ajouter des ennemis. Le but est qu'à leurs contacts notre personnage principal perd des points de vie et que le seul moyen de les détruire et de leur sauter dessus.

Nous avons tout d'abord commencé par travailler les ennemis en même temps que le personnage principal car en effet ceux-ci ont les mêmes caractéristiques de base. Les différences auront lieu au niveau des mouvements qui sont dans un cas généré automatiquement et dans l'autre dirigés grâce aux touches du clavier.

Pour commencer, afin de créer notre ennemi nous avons choisi de prendre la tuile de mouvement que nous pouvons voir ci-dessous. En effet, ceux-ci vont permettre de créer les bases de tout notre personnage.

Figure 7 - Tuile de mouvement de l'ennemi

Afin d'exploiter cette tuile nous créons un nœud *KinematicBody* à notre scène principale suivi de deux nœuds enfants *AnimatedSprite* et *CollisionShape2D*. *AnimatedSprite* est un nœud qui va permettre d'ajouter les différents mouvements de notre personnage, ici ce sera simplement l'animation d'un déplacement verticale. Mais afin que le mouvement demeure fluide nous devons initialiser le nombre d'image par seconde.

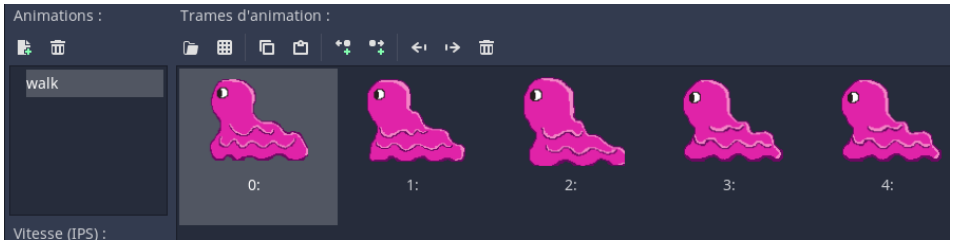

Figure 8 - AnimatedSprite de l'ennemi

Sur la figure 8 nous pouvons voir que notre tuile à était découpée en chaque image de notre bonhomme dans sa réalisation. Par ailleurs, nous pouvons penser qu'avec cet *AnimatedSprite* nous ne pouvons aller qu'à gauche, cependant il existe une fonction permettant à l'*AnimatedSprite* d'être inversé, en soit d'aller à droite. Mais lorsque que nous utilisons la fonction *AnimatedSprite.FlipV* cela permet justement d'inverser notre *AnimatedSprite*.

Le *CollisionShape2D* quant à lui permet de fixer les limites de notre personnage, c'est ce qui va permettre que notre ennemi ne traverse pas les murs par exemple. Lorsque nous avons commencé à vouloir que notre ennemi se déplace automatiquement nous avions tout d'abord créer une fonction random qui permettait que la direction d'un personnage change au bout de quelques secondes, seulement cela paraissant un peu brouillon et nous nous sommes rabattus sur la mise en forme que nous rencontrons souvent dans les jeux vidéo 2D de nos jours, c'est-à-dire un ennemi qui change de direction seulement à partir du moment où il rencontre un obstacle. Pour cela nous faisons appelle à la fonction *MoveAndCollide* qui donne des informations sur la collision et le corps en collision. Si celle-ci est nulle cela veut dire que nous avons rencontré un objet, nous faisons donc changer de direction notre personnage. Nous pouvons donc grâce à cela faire en sorte que notre personnage suive une trajectoire rectiligne et change de direction une fois arrivé devant un obstacle.

Une fois que nous avions les mouvements de notre, nous nous sommes penchés sur la disparition de notre ennemi lorsque notre personnage principal va lui sauter dessus.

Afin de faire ceux-ci, nous avons ajouté un deuxième *CollisionShape2D* à notre ennemi comme le montre la figure ci-dessous.

Figure 9 - CollisionShapre2D de l'ennemi

Ici, nous avons fait en sorte que lorsque notre personnage rentre en contact avec ce CollisionShape2D, notre ennemi disparait de la map.

Notre monde ne pouvant pas se contenter d'un seul ennemi, nous avons tout d'abord voulu les faire apparaître aléatoirement sur la map avec une fonction random, cependant cela ne nous convenait pas car ils n'apparaissent pas là où nous le voulions. Nous avons donc décidé de créer chaque ennemi un par un afin de pouvoir les placer selon nos envies et ainsi à certains moments augmenter le niveau de difficulté en positionnant plusieurs ennemis dans un même périmètre.

Le personnage principal

Réalisé par Théo Dez

Nous voulions afin de marquer notre jeu, un personnage à la fois réel de par le fait que ce soit un personnage humain et en même temps nous voulions qu'il s'assimile à notre ambiance de fond qui mêle le dessin animé et les paysages réels tels que la rue. De plus, il était important que ce personnage puisse marquer les esprits étant donné qu'il est la représentation du jeu, il apparait comme le premier personnage de base que le joueur obtient dès l'installation de Runningman, étant présent sur l'écran d'accueil, notre personnage apparait comme l'icône du jeu.

Le plus dur a été de trouver un personnage étant libre de droit, facilement intégrable dans notre fond (taille, manière de se déplacer...) et comprenant différentes façons de se déplacer. Comme le fait de rester sur place, de sauter, de marcher et de courir.

Tout comme pour l'animation de l'ennemi nous avons décidé d'opter pour une tuile de mouvement pour chaque type de déplacement ce qui rendra les déplacements du joueur plus fluide au dépend d'une demande plus importante de puissance pour lancer le jeu.

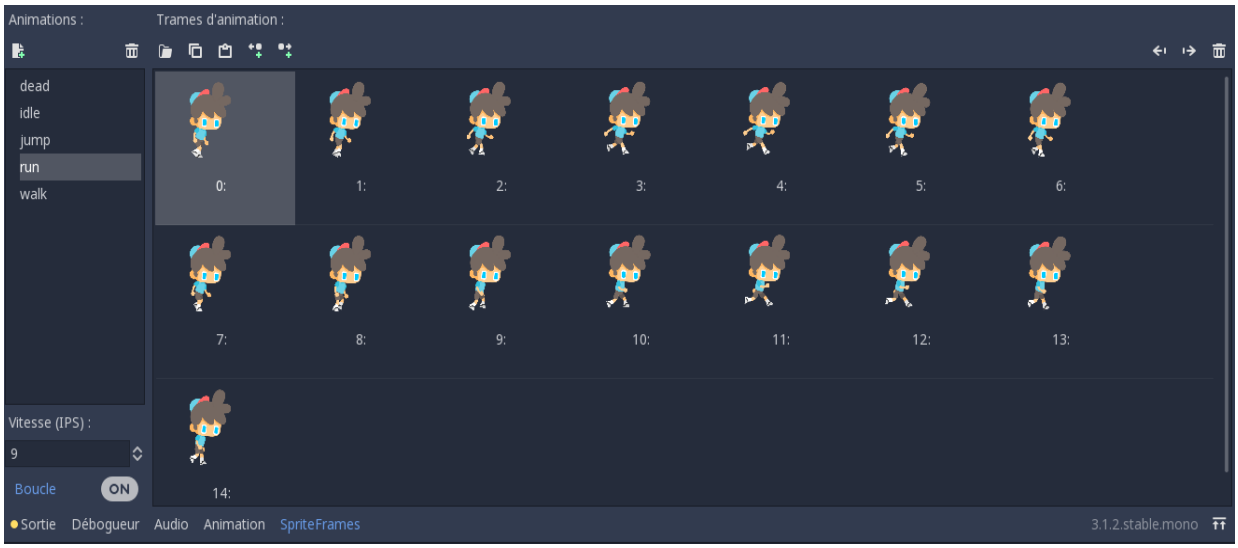

Figure 10 - AnimatedSprite du personnage principal

Sur la figure ci-dessus nous pouvons voir comment se présente les différentes trames d'animations avec l'apparition du personnage image par image. Par ailleurs, nous pouvons remarquer en bas à droite de notre image une case représentant la vitesse en IPS, IPS correspondant au nombre d'image par seconde nous avons décidé de mettre une vitesse de 9 images par seconde car c'est celle qui paraissait être la plus fluide, et qui donnait une démarche réaliste à notre personnage. Il en est de même pour toutes les autres animations du même type (Idle, Dead, jump, run et Walk) qui seront utilisées pour différentes actions rencontrées tout au long des différents niveaux.

Lors de l'insertion du personnage dans la scène nous nous sommes rendu compte qu'il était bien plus grand que les autres éléments de celle-ci. Sa taille était disproportionnée, nous avons donc décidé de changer l'échelle du joueur (scale) en la passant de 1 à 0,1 ou 0,2 nous attendons encore de voir ce qui concorderas le mieux avec nos trois mondes et nos différents ennemis.

Figure 11 - Représentation du scale

Passons désormais au codage de notre personnage, sur le logiciel godot, toute création fonctionne par la création de nœud, les nœuds sont l'architecture principale de notre jeu. Afin de l'intégrer à notre monde, nous nous sommes donc placés dans le nœud du monde principale pour créer un *KinematicBody2D* qui permet de contrôler le personnage par du code.

Cette méthode permet de détecter les collisions avec d'autres corps et permet donc un contrôle plus précis sur leurs réactions et leurs mouvements.

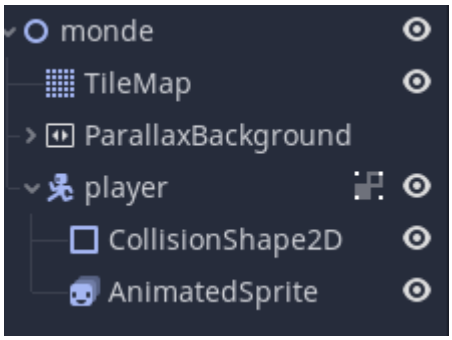

Figure 12 - Noeuds du personnage principal

Dans ce *KinematicBody2D* nous pouvons remarquer sur la figure 12 que nous avons ajouté comme nœud enfant, un *AnimatedSprite* c'est la fonction de godot qui permet d'avoir à l'aide d'une succession d'image une impression de mouvement pour chaque actions comme vu précédemment.

Tout comme la création de notre ennemi que nous avons vu plus tôt, il est nécessaire d'insérer un *CollisionShape2D*. Nous avons choisi un format rectangulaire car c'est celui qui semblait le mieux s'adapter à la forme de notre personnage. De plus, il fallait que ce *CollisionShape2D* soit parfaitement aligné au pieds du héros car nous l'utilisons dans un premier temps quand le joueur se déplace sur la map mais aussi lorsqu'il doit sauter sur un ennemi pour l'éliminer.

Pour finir la création de l'architecture des nœuds, nous avons décidé de donner l'impossibilité aux nœuds enfants de la classe *Kinematicbody2D* de bouger seul afin que qu'ils soient impossibles de les décaler les uns des autres.

Figure 13 - Présentation du ColiisionShape2D

Comme vu lors de la partie de l'ennemi, la création du personnage principal et de celui-ci reste assez similaire, la différence principale étant que le joueur doit pouvoir contrôler ses mouvements à l'aide du clavier numérique. Nous avons donc commencé par programmer le mouvement principal, qui est de courir. Pour cela, nous avons tout d'abord commencé par insérer de la gravité afin que l'on soit attiré par le sol. Les *CollisionShape2D* du monde vont donc ici permettre de stopper la chute du personnage, sans cela le personnage traverserait le paysage. Ensuite, nous faisons en sorte que le mouvement de la course vu précédemment dans l'*AnimatedSprite2D* se lance en même temps que notre personnage progresse, pour cela nous ajoutons par exemple un « si la flèche droite est pressée » alors l'animation se lance notre personnage progresse sur l'axe des x avec une certaine vitesse. De plus, comme vu plus haut, nous utilisons la fonction FlipV afin que nous puissions utiliser le même mouvement mais inversé par rapport à un axe vertical pour aller à droite et à gauche.

```
public class ennemi : KinematicBody2D
{
    private const float gravity = 200.0f;
    private const int walkSpeed = 200;

    private Vector2 velocity;

    public override void _PhysicsProcess(float delta)
    {
        velocity.y += delta + gravity;

        if (Input.IsPressed("ui_left"))
        {
            GetNode<AnimatedSprite.FlipV>("AnimatedSprite").Play("run");
            velocity.x = -walkSpeed;
        }
        else if (Input.IsPressed("ui_right"))
        {
            GetNode<AnimatedSprite>("AnimatedSprite").Play("run");
            velocity.x = walkSpeed;
        }
        else
        {
            velocity.x = 0;
        }
        MoveAndSlide(velocity, new Vector2(0, -1));
    }
}
```

Figure 14 - Code pour faire courir le personnage

III – Difficultés rencontrées

Durant ces premiers mois de réalisation de notre projet nous avons pu commencer à nous familiariser avec Godot. Une application qu'aucun d'entre nous n'avait déjà utilisé avant ça. Notre principale difficulté reste même aujourd'hui la recherche de la façon dont écrire nos scripts.

En effet, nous trouvons régulièrement des tutoriels nous expliquant comment créer ce que l'on veut mais seulement en GDscript, le langage informatique de Godot et non en C# comme demandé. Nous avons donc passé beaucoup de temps à essayer de retranscrire ce que nous voulions et fait face à de nombreux échecs. De plus, nous pouvons remarquer que les tutoriels trouvés en plus d'utiliser un vocabulaire informatique compliqué, ils sont régulièrement en anglais, ce qui nous a poussé à sortir de notre zone de confort.

Pour ce qui est de l'interface principale nous rencontrons quelques difficultés quant à la disposition des *Sprite* (qui permettent de mettre de la matière), en effet ceux-ci se décalent lorsque que nous voulons compiler notre programme. De plus, nous avons eu du mal avec les images (format png) insérées dans les nœuds de type *TexturRect*, en effet celles-ci étaient trop grandes pour rentrer dans nos BoxContainer, il nous a donc fallu modifier la taille des pixels.

Au niveau du personnage principal nous avons rencontré une difficulté pour comprendre comment fonctionner le script, en effet à nos débuts à chacune de nos tentatives le script aboutissait sur une erreur : "Failed to build project solution". A force de recherches et de tentatives le script fonctionne mais pour l'instant uniquement pour les déplacements simples du personnage.

Quant au script de l'ennemi nous rencontrons aujourd'hui toujours la même erreur vue précédemment lorsque nous ajoutons notre code sur la disparition de l'ennemi lorsque le joueur lui saute dessus. Afin de régler ce problème nous comptons revoir notre code ligne par ligne et le tester afin de trouver à quel niveau nous avons fait une erreur.

Nous nous sommes durant ces deux premiers mois également familiarisés avec Git qui a été par moment déclencheur de petites frayeurs au sein des membres du groupe lorsque nous travaillions en même temps sur le projet. En effet, il nous est une fois arrivé après des semaines de « blocage » sur l'erreur vu précédemment tout perdre 30 secondes après avoir trouvé la solution à notre problème. Heureusement, nous avons pu retrouver la version antérieure que nous avions push de notre projet grâce à la documentation de Git.

IV – Tâches à finaliser

Comme vu précédemment il nous reste quelques détails à régler afin de finir notre premier monde.

En effet, nous pouvons penser au retour au début du monde lorsque le joueur perd tous ses points de vies. Ainsi qu'aux finitions de l'interface du menu principal comme la boutique où nous retrouverons les différents personnages disponibles, et l'accès aux différents mondes. Par ailleurs nous aimerions insérer la scène du monde 1 dans le bouton « NewGameButton », afin que notre joueur à partir du menu principal puisse accéder directement au premier monde. De plus nous allons insérer des animations permettant une transition lorsque nous cliquerons sur un des boutons du menu principal. Nous devrons par ailleurs comme expliqué précédemment rééquilibrer les dimensions de nos *Sprite* et images présentes dans l'interface afin de ne plus avoir de problème de décalage par rapport à l'écran.

De plus nous aimerions que notre monde soit plus difficile à finir, pour cela nous allons rajouter quelques détails tels que des « pièges » tuant le personnage, nous avons pensé à des trous le faisant tomber dans le vide, ou encore à des pierres lui tombant sur la tête. Nous comptons aussi allonger notre map afin que le temps de jeu soit plus long.

Pour le personnage principal la plupart des choses sont terminées, mais il reste tout de même une organisation à faire entre lui et les ennemis afin de savoir combien de « cœurs » représentant la vie du personnage, perd-il en cas d'attaque d'un ennemi. De plus nous devons finaliser la partie du code car nous aimerions à l'aide de la move_and_collide qu'après une collision le personnage puisse rebondir sur l'ennemi et ainsi enchainer plusieurs ennemis.

V – Programme pour la prochaine soutenance

Afin de rester à jour dans notre planning, pour la prochaine soutenance il nous faudra donc finaliser les taches évoquées ci-dessus mais aussi commencer les taches à effectuer évoquées dans le cahier des charges, soit la création de deux autres mondes ainsi que le combat avec notre ennemi final, ainsi que l'insertion d'une musique accompagnant le joueur tout le long de son aventure et évoluant en fonction de l'approche d'une zone de danger.

Afin de mener à bien ce programme, nous gardons la même organisation que nous avions jusqu'à aujourd'hui, soit une répartition des tâches en responsable et en suppléant.

	Création du niveau 2	Création du niveau 3	Insertion de la musique	Création de l'ennemi final
N. De Gregorio		Suppléant	Responsable	
T. Dez	Responsable		Suppléant	
M. Leonardon	Suppléant			Responsable
J. Levasseur		Responsable		Suppléant

Figure 15 - Tâches individuelles pour la prochaine soutenance

La création du niveau 2 et 3 sera plus rapide que celle du niveau 1 où nous étions plutôt dans une phase d'apprentissage. Nous pouvons d'ailleurs remarquer que nous avons fait en sorte que les personnes ayant déjà travaillées sur une tâche similaire ne refassent pas la même chose. Cependant, ayant acquis des connaissances sur la façon dont faire ces tâches, nous avons choisi de mettre les personnes ayant déjà fait une tâche similaire en suppléant du responsable afin qu'elles puissent lui apporter de l'aide.

Par ailleurs, pour la prochaine soutenance nous présenterons un site web contenant une présentation générale de notre projet, de notre groupe, nos rapports ainsi qu'un lien vers les éléments que nous avons utilisés. Afin de concevoir celui-ci, nous avons décidé de tous en faire une partie. Celui-ci nous permettra par ailleurs d'apprendre de nouveaux langages informatiques tels que le html permettant au texte et le css à la mise en forme d'une page.

Conclusion

À l'issue de cette première échéance de projet, nous avons compris que nous nous étions engagés dans une tâche complexe malgré le choix d'un jeu en deux dimensions. Les débuts ont été assez rudes car ils amenaient à la découverte de nouvelles notions auxquelles il fallait s'habituer rapidement. Aucun de nous n'avait jamais utilisé Godot avant ce projet, il a fallu par conséquent un temps d'adaptation pour comprendre le logiciel.

Nous avons également pu remarquer que la création d'un jeu vidéo n'est pas aussi simple qu'il n'y parait et que le simple oubli d'un détail peut empêcher le bon fonctionnement de celui-ci.

Mais cette sensation d'entière création nous procure une fierté conséquente à chaque nouvelle avancée du projet après un dur moment de complications.

Nous avons également pris conscience que l'organisation au sein d'un groupe de travail est primordiale à la réalisation du projet, si cette dernière n'est pas présente dès le départ, il est très facile de se perdre et de rendre notre travail très brouillon. Elle nous permet donc de commencer sur des bases solides.

Pour le moment, nous sommes conscients que nous avons encore beaucoup de travail à faire avant de finir notre jeu vidéo. Mais nous pensons en gardant cette organisation pouvoir arriver à un résultat final à la hauteur de nos attentes.