RAPPORT PROJET PFA

Léonard RALEIGH

https://github.com/leonard-raleigh/projet-pfa.git

Entités :

player :

On incarne un personnage simplement représenté par un carré rouge (par choix de DA évidemment et non pas par manque évident de talent en graphisme de ma part).

Le personnage joueur implémente des mécaniques de déplacement assez complexes :

Déplacement latéral simple et réactif :

w permet de se déplacer à gauche

e permet de se déplacer à droite

Saut analogique :

p permet de sauter, la hauteur du saut dépendra de la longueur de l'appui sur la touche

Double saut analogique :

Le personnage dispose d'un unique double saut qu'il utilise en sautant en l'air. Après avoir effectué un double saut, le joueur devient immunisé aux dégâts de contact (mais pas aux projectiles !) et effectue même des dégâts aux ennemis en leur rebondissant dessus.

Wall jump :

Le personnage peut "s'accrocher" aux murs en se collant vers eux en l'air. Cela lui permet de recharger tous ses sauts, mais aussi de tomber bien moins vite. Il peut après sauter directement depuis le mur.

Toutes ces mécaniques combinées font que la prise en main du personnage est très agréable, et les déplacement paraissent très "smooth".

Le personnage peut aussi tirer un projectile, vers la droite ou la gauche dépendant de la direction dans laquelle il se déplace, grâce au bouton p. Le joueur soit spammer la touche pour tirer, car garder le bouton appuyé ne tirera qu'un seul projectile (mécanique volontaire). Le temps de rechargement du projectile est fixe. Le projectile repousse un peu le joueur sur le coup et a une durée de vie limitée, disparaissant après une courte distance parcourue, ou si il heurte un mur. La puissance assez faible de ce projectile fait que le joueur doit habilement alterner entre celui-ci et le double saut pour éliminer les ennemis. Le joueur dispose de 5 points de vie pour arriver au bout du niveau. Si le joueur se fait toucher par un projectile ennemi ou bien vient en contact avec l'un d'entre eux alors qu'il

n’est pas en double saut, il perd 1 PV et est repoussé sur le côté. Le joueur subit alors un “hit stun” et le personnage ne répondra plus aux inputs du joueurs avant de toucher le sol. Il dispose aussi d’une période d’invincibilité montrée par un clignotement du sprite.

enemy :

On dispose de 3 types d’ennemis distincts, chacun avec leur apparence et leur fonctionnement.

- La grenouille :

Elle se balade simplement sur la map, sans se soucier de la position du joueur. On pourrait penser que ça n’en fait pas un ennemi dangereux. Mais ses sauts sont très rapides et leur trajectoire peut être imprévisible, car elle rebondit sur les murs.

- La chauve-souris :

Elle vole, non affectée par la gravité et pourchasse directement le joueur. Sa vitesse est raisonnable et elle ne sait pas éviter les murs, mais elle reste un ennemi redoutable.

- La mouche :

Elle virevolte de droite à gauche sur la carte, et tire occasionnellement des projectiles assez lents dans la direction du joueur. Ni elle ni ses projectiles ne sont affectés par les murs.

Tous ces ennemis disposent de 2 PV, et perdent 1 PV par coup (double saut ou projectile du joueur).

bullet :

Entité représentant les projectiles du joueur comme ennemi. Il sont différenciés lors de la gestion des collisions par leur tag (Bullet true pour les projectiles du joueur, Bullet false pour les projectiles ennemis.)

wall :

Les murs dans ce projet ne sont pas différenciés en murs verticaux et horizontaux : ils partagent l’unique tag Wall. La gestion des collisions est faite en calculant le vecteur de pénétration.

Le niveau du jeu est délimité par 2 immenses murs verticaux qui encloisonnent le joueur, et le niveau se parcoure du bas vers le haut, grâce aux différentes mécaniques de saut.

Systèmes :

Collision_system :

Gère les collisions entre les entités dont les hitbox coincident sur une frame donnée. Si au moins une des entités a une masse finie, le vecteur de pénétration permettant de séparer l'entité avec la masse la plus petite de celle avec la masse la plus importante est calculé, et ce vecteur et les tags des entités sont passées aux fonctions de résolution des collisions des entités.

Die_system :

Supprime simplement les entités dont les point de vie sont tombés à zéro.

Draw_system :

Affiche les textures des entités, en commençant par les murs, puis tout le reste (afin d'afficher proprement les mouches qui traversent les murs par exemple).

Le jeu dispose d'une caméra qui suit le joueur, seulement dans l'axe vertical.
Chaque frame, une coordonnée en y est calculée qui suit le joueur, mais pas strictement afin d'avoir un effet plus lisse.

Le draw system affiche ensuite les entités en décalant leur position sur l'écran selon la valeur calculée pour la caméra.

Foe_com_system :

Système gérant le comportement des ennemis, décrit précédemment.

Ce système s'occupe aussi de cycler périodiquement entre 6 sprites afin de donner un effet animé aux ennemis, tout en gérant également leur orientation.

Gravity_system :

Applique une vélocité vers le bas à toutes les entités qui sont en l'air, jusqu'à atteindre une vitesse de chute maximale. La vélocité appliquée et la vitesse de chute max sont déterminés par un attribut "floatiness".

Move_system :

Applique simplement la vélocité des entités à leur position.

Perish_system :

Supprime les entités dont le chrono de "temps de vie" est écoulé. Notamment utilisé pour les projectiles.

Play_system :

Système gérant tous les déplacements du joueur, mais aussi les changements de sprite selon les événements. C'est aussi ici qu'est calculé le offset de la caméra.

Autres éléments et choses à noter :

Dans le fichier principal game.ml, d'autres mécaniques sont implémentées, comme :

- Spawn des ennemis par vague au moment où le joueur dépasse une certaine hauteur dans le niveau, afin de ne pas surcharger le moteur du jeu.

- Chargement de toutes les textures avant l'initialisation du jeu, les textures sont ensuite misent dans une table de hachage accessible via le registre global.

Le jeu ne semble pas marcher quand on essaie de le lancer via javascript (en tout cas je n'ai pas réussi) à partir du moment où on charge des textures qui sont des images.
Cela pose problème car en SDL, même si tout marche correctement, un problème que moi et d'autres camarades ont remarqué est que le jeu crash en SDL au bout d'un moment si on décharge trop d'entités. Or mon jeu étant en partie basé sur des projectiles qui se suppriment après un moment, cela pose problème. Le niveau est quand même finissable si on s'y prend assez rapidement, mais cet élément hors de mon contrôle réduit un peu l'expérience finale.

Aussi, je tenais à noter que je me suis plus concentré sur les l'implémentation de mécaniques de gameplay, notamment liées aux déplacement du joueur et aux comportement des ennemis, qu'à l'apparence globale du jeu. Même si je tenais quand même à implémenter des sprites animés, ce que j'ai fait pour les ennemis.

Je précise que pour ce projet, je me suis inspiré du jeu Tower Fortress, publié par Nitrome et disponible sur Steam et sur mobile.