Nicolas Devatine – L3 Informatique

Rapport projet PFA

L'objectif de ce projet est de coder un jeu de type metroidvania en utilisant le langage Ocaml et la bibliothèque graphique Tsdl. Dans ce rapport je vais détailler les fonctionnalités que j'ai implémentées ainsi que les difficultés rencontrées.

Pour les images, les musiques et les sons du jeu, j'ai utilisé des créations libres de droit que j'ai trouvé sur le site opengameart.

1 – Les fonctionnalités

Toutes les fonctionnalités demandées dans les différents paliers ont été implémentées ainsi que certaines fonctionnalités additionnelles.

- **Objets :**

  Il y a deux types d'objets qui sont chacun représenté par un module Ocaml. Le premier type d'objet est le « gameobject » qui représente tous les éléments du terrain (plateformes, décorations, portes, etc…) et le deuxième type est le « character » qui représente les personnages non-joueur qui peuvent être des ennemis par exemple.

  Chacun de ces modules dispose de méthode de mise à jour, de mouvement et d'affichage qui peuvent être appelées à chaque tour de la boucle principale et qui permettent de les faire évoluer dans le jeu.

- **Collisions :**

  Pour les collisions j'ai fait le choix de les tester à chaque fois qu'un mouvement est effectué : si une fois le mouvement appliqué il y a une collision alors le mouvement est annulé dans le ou les sens de la collision. Le mouvement est d'abord appliqué sur l'axe horizontale, s'il y a collision on annule ce mouvement, puis pareil sur l'axe verticale.

  Pour savoir s'il y a une collision, comme tous les objets du jeu sont des rectangles, je vérifie pour tous les objets qui bouge s'ils s'intersectent avec un objet de la scène, si oui alors il y a collision

  Le fait que les objets soient des rectangles simplifie les collisions mais les rend aussi moins crédibles, surtout lors d'une collision au niveau des angles d'un objet qui n'a pas l'apparence d'un rectangle.

- **Mouvement :**

  Les objets qui peuvent avoir un mouvement ont une vitesse horizontale et une vitesse verticale associée et à chaque appel de leur méthode de mouvement, leur position horizontale et/ou verticale est modifié en fonction de la vitesse de l'axe en question.

- **Gravité :**

  La gravité n'est pas une gravité « réel » ou physique mais une gravité simplifiée. En utilisant le fait que les objets en mouvement ont une vitesse verticale, j'incrémente d'une constante cette vitesse à chaque mise à jour de l'objet ce qui permet d'avoir une impression d'accélération et de décélération lors du saut par exemple. Même si cette gravité n'est pas fidèle à la gravité physique, elle reste réaliste lorsqu'un objet est en mouvement verticale.

- **Scène :**

  Le jeu est composé de scènes qui sont enchaînées lors de la progression du joueur. Une seule scène est active à la fois. Une scène dispose d'un ensemble d'objets, d'une entrée et d'une sortie ainsi que de méthode de mise à jour ou de mouvement. Lorsque la scène change, l'ancienne scène est détruite. Les scènes sont chargées à partir de fichiers extérieurs.

- **Caméra :**

  Une caméra suit le joueur lors de ses déplacements, elle dispose d'une taille et sa position est calculée en fonction de la position du joueur et de la taille de la scène.

- **Animation :**

  Chaque objet dispose d'une liste d'animations et d'une animation courante qu'il va jouer en fonction de l'action en cours. Une animation est une liste d'image qui vont s'enchaîner. Les animations sont gérées par un module qui dispose de méthodes permettant de les faire évoluer en fonction de différents paramètres comme par exemple le nombre de fois où elles doivent être jouées.

- **Projectiles :**

  Le joueur ainsi que les ennemis disposent de projectiles qui servent à infliger des dégâts. Les projectiles sont liés à leur propriétaire pour éviter de s'infliger des dégâts à soit même. Chaque projectile dispose d'une durée de vie, ils sont détruits lorsqu'ils dépassent cette durée de vie ou entre en collision. Lors d'une collision, si elle a eu lieu avec une entité qui peux prendre des dégâts et qui n'est pas le propriétaire du projectile alors la vie de cette entité est diminuée d'un. Les entités capables de lancer des projectiles disposent d'un temps de recharge pour éviter de lancer trop de projectiles à la fois.

- **Points de vie :**

  Le joueur ainsi que les objets disposent de points de vie qui sont diminués à chaque fois que des dégâts sont subis. Le joueur dispose initialement de trois points de vie. Les points de vie du joueur sont affichés en permanence en haut à gauche de l'écran.

  Lorsqu'une entité subît des dégâts elle devient invincible pendant un certain temps pour lui permettre de partir et pour lui éviter de mourir trop rapidement. Chez le joueur cette invincibilité se traduit par un « clignotement » du personnage pendant cette durée.

- **Menu :**

  Le jeu dispose d'un menu simple permettant de lancer le jeu ou de le quitter. Lorsqu'un joueur perd, il est renvoyé sur le menu.

- **Ennemi (IA) :**

  Le jeu dispose d'ennemis que le joueur doit affronter pour avancer. Ces ennemis ont un mouvement et peuvent tirer sur le joueur. Le mouvement des ennemis est simple, quand ils entrent en collision de manière horizontale ils repartent dans l'autre sens et effectuent donc un mouvement de va-et-vient. Pour leurs attaquent, ils s'arrêtent et tirent dans la direction du joueur si le joueur se trouve à moins d'une certaine distance de leur position.

- **Sons :**

  Le menu, les scènes, le joueur ainsi que les objets possèdent des musiques et des sons. Des musiques d'ambiance sont jouées en boucle dans les scènes et des sons sont déclenchés lors d'actions spécifiques tel qu'un saut. Les musiques et sons sont chargés avec les types auxquelles ils réfèrent, ainsi le joueur et chaque objet dispose d'une liste de sons et le menu et les scènes d'une musique.

- **Autres :**

  J'ai également ajouté certaines fonctionnalités supplémentaires.

  - **Rebond sur les ennemis :**

    Le joueur peut sauter sur les ennemis pour leur infliger des dégâts ce qui le fera rebondir.

2 – Les difficultés rencontrées

La principale difficulté rencontrée a été l'organisation et la planification du développement. J'ai eu trop tendance à me lancer dans le code sans avoir beaucoup réfléchis avant, ce qui a eu pour conséquence de devoir revenir très souvent sur du code déjà écris pour pouvoir l'adapter aux nouvelles fonctionnalités que je voulais mettre en place et donc des pertes de temps conséquentes.

L'autre source de difficulté a été de savoir quand et comment bien utiliser les aspects fonctionnels du langage Ocaml qui n'étaient pas forcément pertinents dans toutes les parties du projet.