Rapport Projet Prog 2 – Rendu 2

Simon Lukowski - Hugo Fruchet

April 2023

1 Nouvelles fonctionnalités

Pour ce second rendu, nous avons ajouté plusieurs nouvelles fonctionnalités

- Une base constructible et destructible
- De nouveaux ennemis avec différentes cibles (les ogres attaquent les joueurs, les goblins les bases)
- Des points de minage de ressources (une carrière pour récupérer des pierres et une forêt pour récupérer du bois)
- Comme signalé au premier rendu, une centralisation des contrôles à été rajoutée qui rend les contrôles plus élégants et plus flexibles

2 Implémentation

Pour le nouveau système de contrôles centralisé, nous avons crée l'objet `ControlManager` et l'énumération `Control` :

- L'énumération `Control` représente chaque contrôle du jeu. Un contrôle possède une liste de touches qui peut l'activer et une liste de `listeners` (fonctions qu'il faut appeler si une touche est pressée).
- L'objet `ControlManager` est l'objet qui s'occupe de récupérer les événements brut de la SFML et d'appeler les bons `listeners` des contrôles lors du déclenchement des événements.

Ainsi, chaque objet va lors de son initialisation ajouter sa fonction dans les `listeners` d'un contrôle et la liaison avec le contrôle sera directement établie.

Pour implémenter la base, nous avons créer une classe `Base` et aussi une classe d'inventaire pour la structure `StructureInventory` :

- `Base` est l'objet physique dans la scène, il est animé et possède différent états qui représente sa vie

- `StructureInventory` représente son inventaire, il possède les différentes ressources et aussi un HUD de l'inventaire qui est le visuel permettant de manipuler l'inventaire quand on le sélectionne

Pour le nouveau déplacement des entités, nous avons implémenté des déplacements aléatoire avec la classe `scala.util.Random` lorsque l'entité est au repos, et poursuivre (ou fuir) une cible qu'elle stocke dans ses attributs lorsqu'elle en a une. De plus, les entités peuvent chasser différentes entités, ainsi la classe `Enemy` a été munie d'une `target_id`, et chaque classe héritant de `GameObject`d'un `id`, afin de pouvoir implémenter un dynamic dispatch, c'est à dire préciser à l'avance à quels type de cibles vont être appliquées les fonctions, et donc notamment poursuivre et attaquer uniquement ces derniers.

3 Problèmes recontrés

Durant l'avancée de la seconde phase nous avons rencontré de nouveaux problèmes :

- Lors de l'ajout des générateurs de ressources, il fallait pouvoir interagir avec eux, cependant, comme on interagit aussi avec une ressource pour la récolter, on générait et récoltait instantanément la ressource, et la séparation des actions en deux touches n'était pas faisable. Pour régler cela, nous avons du introduire une `InteractionAction` dans la fonction d'interaction afin de signaler à l'objet avec lequel on interagit le but de l'interaction (et ainsi, si le but n'a pas de sens avec l'objet en question, on ne fait rien). Ainsi, nous avons pu séparer ces 2 actions différentes en 2 touches différentes et chaque touche essaye d'interagir avec 2 buts différents.

- Lors des différents tests, nous nous sommes rendus compte que parfois les entités appraissaient dans la hitbox d'une autre entité. De ce fait, avec la méthode de déplacement implémentée qui vérifie les collisions de hitbox, les entités se bloquaient les unes dans les autres et elles ne pouvaient plus bouger. Pour pallier ce problème, nous avons dû implémenter le décalage d'une entité lors de son apparition (ce décalage cherchant à se rapprocher du centre pour éviter de faire apparaitre l'entité en dehors du monde).