Projet Programmation 2 : Rapport Partie 1

MANGEL Léo et MICHEL Thomas

1 Etat du projet

Nous avons implémenté les fonctionnalités suivantes :

Affichage :

- Affichage de la carte et du champ de vision du joueur. Seuls les entités dans le champ de vision du joueur sont visibles. De plus, seules les cases de la cartes visibles par le joueur, ou déjà visitées, sont affichées.
- Interface avec les informations sur le personnage du joueur, son inventaire ainsi que les entités dans son champ de vision
- Affichage des événements récents sous forme de logs, comme les actions du joueur ainsi que les attaques subites

Jeu :

- Génération aléatoire de la carte, ainsi que des ennemis et des objets sur cette carte
- Gestion des déplacement du joueur dans 8 directions
- Système d'objet et d'inventaire. Le joueur peut ramasser les objets au sol pour les mettre dans son inventaire. Il peut consommer, équiper, jetter et déposer les objets de son inventaire.
- Système d'ennemis et de combat. Les ennemis ne bougent pas avant d'être activés, ce qui se produit lorsqu'ils entrent dans le champ de vision du joueur. Lorsqu'ils sont activés, ils se déplacent vers le joueur et l'attaquent au corps à corps. Le joueur peut également attaquer au corps à corps en se déplaçant sur la case d'un ennemi. Il y a également des statistiques qui déterminent les dégats faits et reçus. Ces statistiques peuvent être modifiées par les objets.

2 Algorithmes utilisés

Nous avons utilisé plusieurs algorithmes existants pour certaines parties du projet.

2.1 Génération aléatoire de la carte

Pour la génération de la carte, nous avons utilisé un algorithme appelé "Tunneling Algorithm".

2.2 Champs de vision

Pour le champs de vision, nous avons utilisé l'algorithme dit de "recursive shadowcasting".

2.3 Déplacement des ennemis

Dans la version actuelle, les seuls ennemis implémentés dans le jeu combattent au corps à corps, ainsi le plus adapté est d'utiliser un algorithme de plus court chemin. Nous avons donc décidé d'utiliser l'algorithme A*.

3 Difficultés rencontrées

La principale source de difficulté a été d'utiliser Scala.

Par exemple, le fait qu'il n'y ai pas de commande `break`, ou au moins pas qui fonctionne comme dans la plupart des autres langages de programmation, nous a forcé à adapter notre code à cette contrainte.

De plus, ce n'était pas toujours évident de trouver de la documentation compréhensible, tout particulièrement pour les aspects très techniques comme l'interface qui ont du être en partie fait avec du Java. Par exemple, pour afficher une image avec Graphics2D, il existe une demi-douzaine de fonctions qui demande chacune des arguments différents et des classes différentes pour l'image et la différence est difficilement compréhensible.