Rapport Projet Programmation 2

Nicolas Dumange, Gabriel Jeantet

1 Présentation du projet

1.1 Contenu du projet

Notre travail se compose de 20 fichiers :

— monsters.scala et artificialIntelligence.scala, définissant respectivement les caractéristiques et le comportement des monstres
— entity.scala, définissant les caractéristiques et les propriétés d'affichage des éléments non régis par une IA, telles que le personnage du joueur et les objets
— inventory.scala et item.scala, définissant la gestion des objets, et leur utilisation
— render.scala, tilesetHandler.scala, sprite.scala et spritetable.png définissant les options graphiques de la partie
— env.scala et map.scala, définissant la génération du niveau, en terme de murs/sol et de biomes
— main.scala,dimension.scala, game_panel.scala, log.scala, mainloopvars.scala et mainloop.scala, définissant l'affichage de a fenêtre de jeu et les interactions entre le joueur et le jeu
— position.scala, des objets encapsulant des coordonnées dans le plan
— game_object.scala, définissant une partie de jeu

1.2 Choix techniques

Pour les déplacements des monstres poursuivant le joueur, nous utilisons l'algorithme A*, et le champ de vision utilise l'algorithme de Bresenham's line. Le champ de vision de la plupart des monstres leur permettent de détecter le joueur, et celui du joueur lui permet de voir les cases à une distance illimitée, et garde en mémoire la dernière position observée d'un élément tel qu'un objet, un monstre, ou encore l'agencement du niveau.

Les intelligences artificielles suivent pour le moment le même modèle :

— à partir du moment où un monstre voit le joueur, il se déplace vers la dernière position du joueur observée.
— si le joueur est à portée, le monstre l'attaque
— si le joueur n'a pas encore été observé, ou si le monstre a atteint la dernière position connue du joueur, il se déplacera de manière aléatoire dans le niveau

A cette base s'ajoutent quelques variations :

— une action tous les deux tours
— deux actions par tour
— une connaissance permanente de la position du joueur

La génération de la carte se constitue d'un placement aléatoirement des murs sur un niveau vide, puis une harmonisation par un automate cellulaire, qui place des murs sur les cases vides entourées de murs, et des cases vides à la place des murs isolés, puis par une détection de plus courte distance entre les salles pour creuser des couloirs, afin de garder un niveau connexe.

La disposition des biomes et le test de connexité s'effectue en utilisant la méthode des pouring buckets, où une case transmet à ses voisins vides son contenu, c'est à dire son numéro de salle ou le biome par lequel elle est remplie. S'il n'y a qu'une salle, alors le niveau est connexe.

Les objets existant sont la gelée de soin (healing goo), rendant 5 points de vie au joueur, et le trophée de victoire, mettant fin à la partie lors de l'utilisation, et apparaissant lorsque le joueur a vaincu tous les ennemis du niveau. Ces objets se ramassent lorsque le joueur marche sur la même case, et sont utilisable en cliquant dessus dans l'inventaire.

1.3 Jouabilité

1.3.1 Contrôles

Le jeu s'effectue en tour par tour, un tour correspondant à une action du joueur.

Le joueur se déplace en utilisant les flèches directionnelles du clavier, ou bien en utilisant la souris , en cliquant dans la direction voulue. La souris permet de plus de faire des déplacements diagonaux. Si la case visée est occupée par un monstre, l'action de déplacement devient une attaque, infligeant des dégâts au monstre en question. Lorsqu'un monstre meure, celui-ci disparaît de la partie, lorsque le joueur meurt, un message de défaite est affiché, et la partie est terminée.
Les objets sont utilisables en cliquant dessus dans l'inventaire.
Le joueur peut attendre un tour en utilisant la touche w.

1.3.2 Monstres

Les monstres présents correspondent chacun à une IA :
— le gobelin (goblin) possède une IA basique
— le hobelin (hoblin) n'agit qu'un tour sur deux
— la chauve-souris (bat) agit deux fois par tour
— le chasseur (hunter) connaît l'emplacement du joueur, et lors de sa première action, un message indique au joueur qu'il est poursuivi

1.4 Graphismes

La librairie utilisée est swing.

La fenêtre est composée de trois parties, un affichage du niveau, l'interface utilisateur, où sont visibles l'inventaire et la vie du joueur, ainsi que l'historique des événements, affichant les 5 derniers événements, sous forme de message. Le niveau est divisé en case carrées définissant les positions et les distances, au sein desquelles sont affichés les "sprites" des éléments.

2 Problèmes et difficultés rencontrés

Scala est un langage nouveau pour nous et assez différent de ceux que nous avons l'habitude d'utiliser, il a donc fallu un temps d'adaptation assez long pour pouvoir commencer à travailler efficacement.
De plus, la documentation de Swing est assez peu fournie en ce qui concerne Scala, elle l'est surtout pour la version Java de Swing.

3 Perspectives

Pour le moment, la condition de victoire est de vaincre tous les ennemis. Nous comptons modifier cela, par l'ajout de boss, et implémenter un système d'étages.
De nouveaux ennemis seront ajoutés pour rendre le jeu plus diversifié.
Nous comptons aussi ajouter des objets permettant au joueur d'avoir plus d'options offensives.