Armand Chaplain-Duret **14/01/2022**

Ethan Leblanc

SAE Développement

Analyse

1._Présentation du jeu

Le jeu que nous avons programmé est un labyrinthe dans lequel le but principal est de trouver la sortie de trois labyrinthes consécutifs classés par ordre croissant de difficulté, en commençant par un labyrinthe de tutoriel.

Dès le lancement du jeu, un tuto en explique au joueur son fonctionnement. Dans chaque labyrinthe, se trouvent d'une part, des ennemis à combattre et d'autre part, un marché où il possible d'acheter des équipements permettant d'améliorer le personnage. Cela est possible grâce à l'argent gagné en combattant les ennemis ou en réussissant les labyrinthes précédents.

Chaque labyrinthe est rempli de brouillard. Le joueur a donc une visibilité limitée. Les sorties des labyrinthes sont représentées par des carrés pleins. Les touches de base sont : z pour avancer, d pour aller à droite, q pour aller à gauche, s pour reculer, e pour interagir et o pour ouvrir le menu des options en jeu. Ces touches sont modifiables via le menu principal ou le menu option en jeu.

Le combat contre un ennemi se fait aléatoirement en 2 tours en fonction de la probabilité de victoire du joueur. Suite à une défaite, le joueur revient au début du labyrinthe avec un nombre de cœurs en moins, en fonction de sa résistance. Dans le cas où le joueur ne comprend pas les principes essentiels au bon déroulement du jeu, il peut accéder aux aides dans le menu option.

2._Structures de données :

Nous avons utilisé deux types de structure de données :

1. **Structures finies**

   _Variables Int : Remplissage des tableaux (lignes et colonnes) : indexages, limite de la vue, probabilité de victoire, nombre de pièces, numéro du dialogue et de la page

   _Variables double : Nombre de vies et résistance

   _Booléens : Tuto fait ?, lancer le jeu, vérification d'entrées (touches existantes), obstacles ?, emplacement dans l'inventaire vide, résultat du combat, en jeu ?, gagner ?

   _Char : Vérifier s'il y a un obstacle à l'emplacement demandé

   _String : Savoir quelle action le joueur fait, trouver un élément dans un tableau, stocker les items, description des items, quelle interaction est possible

2. **Structures indexées**

   _Tableaux : Cordonnées de la position du joueur, items achetables (disponible en magasin et possédés), prix, description des items, quantité des items, interactions, lister les obstacles.

   _Tableaux multidimensionnels : stocker le labyrinthe, le plateau de jeu, boites de dialogue (bardinfobas), brouillard.

   Avantages : Meilleure utilisation de la mémoire quand on utilise un tableau : Réduit la redondance de l'information. Quadrillage des informations représentant la labyrinthe et la position du joueur. Facile d'utilisation.

   Inconvénients : Taille fixe

3._Algorithmes :

Autre solution de placement des murs : format CSV

Flush : Eviter d'importer d'autres modules java : quelques bugs : Actualisation.

Autre possibilité de fonctionnement de la vue du joueur : affiche un tableau différent en fonction de la position du joueur. Performances en utilisation du stockage : Moins performants. Plus rapide en jeu.

Utilisation possible de listes pour la difficulté : Taille variables selon la difficulté

Performance en temps de la liste : Plus rapide

Performance en utilisation mémoire : Moins lourd

Liste : plus facile d'implantation et de compréhension

4._Circuit algorithmique du JEU :

Plateau de jeu -> remplit de brouillard -> contour du plateau de jeu -> rentre le joueur dans plateau -> vue du joueur -> affiche le plateau de jeu -> vérification interactions possibles -> affiche barInfoBas (interactions possibles ou non) -> demande au joueur de faire une action -> vérification de l'action (possible ou non) -> traitement de l'action

L'apport de chacun :

Armand : Déplacements joueur, ennemis, interactions, description, nombre de vies, main, remplissage et affichage labyrinthes, barInfoBas, affichage barInfoBas, vue joueur.

Ethan : Menus, Placements Mur, le marché, Affichage (Menu et combat), Ennemi, Items

Pourcentage : Armand : 50% Ethan : 50%