Nom 1 : TEKPA
Prenom : Max
Groupe : 1
———

Nom 2 : Tran
Prenom : Thien-Loc
Groupe : 1

Projet de POO : The Big Adventure

L3 informatique, Université Gustave Eiffel

Rapport : The Big Adventure

I. Introduction

Bienvenue dans notre jeu, un jeu où l'aventure et la créativité se rencontrent. Notre objectif est de créer une expérience de jeu, on veut se rapprocher d'un mode de jeu à la zelda moderne (Breath of the Wild) pas dans son histoire/scénario, mais dans la capacité à créer sa propre histoire.

II. Fonctionnalités

1. Lecture du fichier

Nous avons commencé par l'écriture du package lecture dans lequel se trouve le parseur, qui a pour but de parser la carte (map) du jeu et les éléments constituants les personnages, items, obstacles…

Afin de pouvoir compter les lignes du fichiers, nous avons ajouté dans l'enum Token un nouveau élément NEWLINE pour permettre cela. Comme nous avons décidé d'importer les éléments du fichier séparément c'est-à-dire que nous faisons un parcours pour collecter la taille, un autre pour l'encoding, la data et en fin les éléments et pour cela pour nous avons du revenir à plusieurs fois au début du fichier ainsi nous avons fait une méthode dans le Lexer utilisant Reset pour accomplir cela.

Les éléments du fichier sont stockés dans des listes et maps afin de permettre en cas de validation du fichier leur importation dans les classes et records dédiées.

2. Importation des éléments du fichiers

Après avoir analysé le fichier, nous avons développé une classe LoadingData qui englobe une liste d'inventaire, de personnages et d'obstacles. La liste d'inventaire comprend InventoryItem, une

interface implémentée par les éléments tels que les armes (épée, bâton...), les objets utiles et nutritifs (box, key, pizza ...) . La liste de personnages contient une interface Personnage, qui est représenté par les classes Friend et Enemy. La classe LoadingData comprend également un objet Player qui représente le joueur. Ainsi, cette classe possède des méthodes permettant de créer le joueur, les obstacles, les objets, etc.

La classe LoadingData se concentre uniquement sur l'importation des éléments de la carte. Pour stocker la data de la carte, nous avons également créé une classe Grid qui utilise un tableau bidimensionnel de caractères pour représenter la grille du jeu, ainsi que les encodages associés. Après l'enregistrement des données dans les types appropriés (classes et records), nous avons mis en œuvre les fonctionnalités du jeu, notamment la gestion des personnages et l'affichage.

3. Gestion du jeu et affichage

Après avoir lu et importé les données, nous avons mis en place les classes GameController, Elements et Affichage.

La classe GameController, c'est le chef d'orchestre du jeu. Elle gère la fenêtre du jeu et la boucle principale. Ces méthodes appellent des méthodes de la classe Affichage, et aussi des méthodes de la classe PersonnageMoving et Elements.

La classe PersonnageMoving, elle s'occupe des mouvements de tous les personnages et de leurs actions, comme échanger des items ou utiliser des armes. Elle reprend également des méthodes de la classe PlayerAndItem, qui gère la collecte d'objets, la nourriture du joueur ainsi que la lecture de bouquins.

À cause de la complexité, plutôt que de faire une interface pour tous les éléments du jeu , nous avons délégué les classes qui représentent ces éléments (grid, Allpersonnage, Allobstacles, Allitems) dans la classe Elements. Ça simplifie l'utilisation dans GameController et d'autres classes qui en ont besoin.

Pour l'affichage des éléments, on a créé des petites méthodes privées pour chaque chose à afficher. Toutes ces méthodes sont appelées dans une méthode nommée displayAllElements, qui à son tour est utilisée dans GameController.

Nous avons également stocké toutes les images, y compris celles qui changent avec la direction du joueur et des items, dans une map au début du jeu. Nous avons fait cela afin éviter de charger les images à chaque fois, ce qui consommait des ressources et faisait clignoter l'écran (même avec un seul renderFrame). Mais, nous avons remarqué qu’au début du jeu, les éléments prennent quelques secondes à s'afficher. Nous pensons que cela est du au chargement des images.

4. Améliorations et Corrections

Depuis la soutenance bêta, nous avons crée une classe parser dans laquelle nous avons refait l’entier-té de notre parseur, avec ce nouveau parseur, nous sommes en mesure de détecter les erreurs (peut-être pas toutes), de trouver les lignes d’erreurs. Nous avons renommé quelques classes, interfaces et noms de packages (y avait pas les bons noms).