

Rapport de Projet : Jeu d'Exploration de Salles

Introduction

Dans le cadre de ce projet, nous avons développé un jeu d'exploration de salles en 2D où un personnage représentant le joueur peut se déplacer de salle en salle, et de niveaux en niveaux. Les salles contiennent des monstres et des objets. Ce rapport présente nos choix de modélisation et de réalisation, ainsi que les différentes fonctionnalités implémentées.

Diagramme de Classes

Nous avons élaboré un diagramme de classes (UML) pour représenter la structure de notre programme. Voici le lien du diagramme : <https://symko.net/Autre/img/uml.png>

Explication de la Modélisation

Le jeu est implémenté de la façon suivante :

- **Classe Main** : représente le point d'entrée du programme. Elle crée une instance de la classe GameGUI qui lance le jeu.
- **Classe GameGUI** : représente la classe principale du jeu, elle gère les affichages (des exceptions, du jeu et des écrans de fin de partie) ainsi que les inputs envoyés.
- **Les classes PanelJeu, PanelAlerte, Panel Start** : représente les Panel qui affichent le jeu et les informations à l'utilisateur.
- **La classe Complication** : la classe permettant de créer une exception.
- **Classe Game** : représente le jeu dans son ensemble. Elle contient les méthodes principales pour gérer le déroulement du jeu comme les déplacements etc.
- **Classe Level** : représente un niveau du jeu, grâce à un tableau de caractères à deux dimensions. Chaque caractère spécifique représente un élément de la salle (joueur, monstre, objet, etc).
- **Classe Player** : représente le personnage du joueur. Il possède des attributs tels que la vie, l'argent, l'attaque, etc. Le joueur peut se déplacer de case en case grâce à des méthodes de déplacement.
- **Classe Monster** : représente les monstres présents dans les salles. Chaque monstre possède des attributs tels que la vie, l'attaque, etc. Les monstres attaquent le joueur lorsqu'il se trouve dans la même salle qu'eux.

- **Classe Door** : représente les portes délimitant les sous-section du niveau.
- **Classe Object** : représente les objets que le joueur peut ramasser dans les salles. Chaque objet possède un nom et une méthode **effet()** qui définit l'impact de l'objet sur le jeu.

Comment Jouer au Jeu

Détail des Touches

Le joueur peut exécuter certaine action dans le niveaux en utilisant les touches du clavier suivantes :

- **Z** : Se déplacer vers le haut.
- **Q** : Se déplacer vers la gauche.
- **S** : Se déplacer vers le bas.
- **D** : Se déplacer vers la droite.
- **E** : Utiliser la compétence spéciale
- **O** : Quitter le Jeu

Interactions avec l'Environnement

Les interactions se déclenchent automatiquement lorsque le joueur entre en contact avec un élément spécifique dans le jeu :

- **Ramasser des Objets** : Lorsqu'un joueur se déplace sur une case contenant un objet, cet objet est automatiquement ramassé. Les objets peuvent être des potions, de l'argent, des équipements ou des améliorations.
- **Ouvrir des Coffres** : En entrant en contact avec un coffre, celui-ci s'ouvre automatiquement et le joueur récupère son contenu, qui peut inclure de l'argent ou des objets spéciaux (si le coffre est dans le shop il fait aussi payer le joueur).
- **Utiliser la Compétence Spéciale (Freeze)** : En appuyant sur la touche **E**, le joueur peut activer la compétence spéciale freeze.
- **Combats** : Voir ci-dessous.

Combats

Les combats se déclenchent de manière instantanée lorsque le joueur entre en contact avec un monstre. Voici comment ils se déroulent :

- **Déclenchement du Combat** : Dès que le joueur se déplace sur la case d'un monstre, le combat commence automatiquement.
- **Résolution du Combat** : Le combat commence par l'attaque du joueur, puis par l'attaque du Monstre et ainsi de suite. Chacun à son tour va attaquer l'autre jusqu'à ce que l'adversaire meurt. A chaque attaque on regarde si l'adversaire est mort et si cela n'est pas le cas on

continue d'attaquer. Ainsi c'est pour cela que le combat termine instantanément aux yeux du joueur. Mais le combat reste du **tour par tour**.

Progression dans le Jeu

- **La Sortie** : Chaque niveau contient une sortie qui apparaît lorsque tous les monstres de la salle ont été tués. Lorsque la sortie est prise, elle nous emmène au niveau suivant.

Répartition du Travail

Le projet a été réalisé en binôme, avec une répartition des tâches adéquate. En effet, nous avons commencé le projet ensemble en discutant de la modélisation et des fonctionnalités à implémenter. Nous avons ensuite débuté l'implémentation en créant les classes principales et en définissant les méthodes et attributs nécessaires. Chacun de nous a ensuite pris en charge une partie du code, en se concentrant sur les fonctionnalités qui lui étaient assignées.

- **Tristan** : Gestion de la logique du jeu, implémentation des mécanismes de déplacement, de combat, de gestion des niveaux et de l'interface graphique.
- **Erwan** : Implémentation des objets, des monstres, de la gestion monétaire du jeu, du freeze et de la génération aléatoire.

Nous nous sommes ensuite réunis pour peaufiner le jeu et pour tester le programme dans son ensemble.

Idées Uniques

1) Système Monétaire

Gagner de l'Argent

Les joueurs peuvent accumuler de l'argent de deux manières principales :

- **Les Coffres** : Des coffres sont disséminés dans les salles des différents niveaux. Le joueur doit explorer les salles pour trouver ces coffres et les ouvrir.
- **Tuer des Monstres** : Chaque monstre vaincu par le joueur donne une certaine somme d'argent (20 pièces ici).

Dépenser de l'Argent

L'argent accumulé peut être dépensé dans une boutique accessible au joueur à certains points du jeu (tous les trois niveaux). La boutique offre une variété d'objets et d'améliorations qui peuvent aider le joueur dans sa quête :

- **Vie**: 20 points de vie pour 50 pièces.
- **Épée**: 10 points d'attaque supplémentaire pour 50 pièces.
- **Compétence spéciale**: Compétence spéciale Freeze pour 120 pièces.

2) Compétence Spéciale

En plus des articles classiques, notre jeu propose une compétence spéciale disponible à l'achat dans la boutique : le **freeze**. Cette compétence offre un avantage stratégique crucial pour le joueur en situation de combat.

Description de la Compétence Freeze:

La compétence freeze permet au joueur, en appuyant sur la touche **E**, de geler tous les monstres présents dans la salle pendant une durée de 3,5 secondes. Pendant cette période, les monstres sont complètement immobilisés et incapables d'attaquer. Les combats engagés contre eux durant le freeze ne font perdre aucun point de vie au joueur, permettant ainsi de les vaincre sans subir de dommages.

Génération Aléatoire de Niveaux

Après les 4 niveaux prédéfinis, notre jeu bascule sur une génération aléatoire de niveaux. Cette génération aléatoire se fait via la méthode **createRandomLvl()**, qui décide aléatoirement entre trois configurations de niveaux possibles (**createLvl1()**, **createLvl2()**, et **createLvl3()**). Voici comment cela fonctionne :

Fonctionnement de la Génération Aléatoire

1. **Sélection aléatoire du Niveau** : La fonction **createRandomLvl()** utilise un générateur de nombres aléatoires pour sélectionner l'une des trois configurations de niveaux. Elle fonctionne comme suit :
2. **Création des Niveaux** : Chaque méthode de création de niveau (**createLvl1()**, **createLvl2()**, **createLvl3()**) initialise la grille du niveau et place les différents éléments (murs, joueur, monstres, coffres, portes) de manière semi-aléatoire. Voici un résumé des étapes communes :
 - **Initialisation de la Grille** :
 - La grille est d'abord remplie d'espaces vides.
 - Des murs sont placés pour entourer le niveau et parfois créer des sections internes aléatoires.
 - **Placement du Joueur** :
 - Le joueur est toujours placé en (7,5) pour avoir une position de départ fixe.
 - **Placement des Monstres et des Coffres** :
 - Un nombre aléatoire de monstres et de coffres est déterminé.
 - Les monstres et les coffres sont placés à des positions aléatoires dans la grille, en évitant de bloquer la position de départ du joueur.

Ces méthodes garantissent que chaque niveau généré est unique tout en suivant des structures de base préétablies pour assurer la jouabilité.

Gestion des exceptions

Les exceptions sont créées par la classe `Complication` qui permet de créer des exceptions. Ces dernières sont transmises dans toutes les méthodes et remontent jusqu'à la classe `GameGUI` qui affiche alors le problème via la classe `PanelAlerte`.

Dans notre cas, les uniques problèmes qui peuvent causer des exceptions sont les méthodes de la classe `Level`. En effet ces dernières renvoient des complications si des exceptions `IOException` sont détectées au moment de l'ouverture, de la lecture, de l'écriture et de la fermeture des fichiers txt du level.