Rapport de projet:
Langage à Objet Avancés
Sujet 3

Laila ELKOUSSY (#22025466)

Lucas KETELS (#71800543)

Janvier 2022

Introduction **2**
Le cadre 2
Le Déroulement du Jeu **3**
Suppositions 3
Les Classes Château et Salle **4**
Présentation des classes 4
Suppositions: 4
Les Classes Joueur, Joueur Manuel et Joueur Automatique **5**
Présentation des classes 5
Concepts de POO Utilisés 5
Suppositions: 5
Les Classes Personnage **6**
Présentation des Classes 6
Suppositions 7
Les Classes Objets et ObjectFactory **8**
Présentation des Classes 8
La Classe Arme 8
La Classe Potion 8
La Classe Clef 8
La Classe ObjectFactory 9
Concepts de POO Utilisés 9
Suppositions: 9
La Class Combat **10**
La Classe Utilities **10**
Améliorations Possibles **11**

Introduction

Le cadre

Le but de notre projet était de créer un jeu de rôle interactif qui se déroule dans un château. Au début du jeu, le joueur est placé dans une salle aléatoire du château. Le but du jeu est de tuer, dans des combats, tous les ennemis placés dans le château au début du jeu.
Avant le début de la partie, chaque joueur choisit un personnage (parmi Amazone, Guerrier, Moine et Sorcière). Chaque personnage procure des avantages différents, en termes de type de force et défense. Ceci est mieux expliqué dans la partie Personnage.
Dans les salles du château, un joueur peut retrouver des armes, de force et rareté variée, des potions, et des clefs de téléportation.

Le Déroulement du Jeu

Au début d'une partie, le joueur est prié de choisir un personnage.
Après, la partie est lancée, et les joueurs sont placés de manière aléatoire dans les salles. Les joueurs manuels ont le droit de consulter leur sac et leur équipement, ramasser et jeter des objets, lancer des combats avec les ennemis dans la salle, et finir leur tour et potentiellement changer de salle quand ils ont vaincus tous les ennemis présents.
Le jeu est perdu si le joueur manuel meurt au combat, et le jeu est gagné si le joueur réussit à tuer tous les ennemis présents dans le château.

Suppositions

Voici la liste des suppositions que nous avions fait:

- Au début du jeu:
    - Dans chaque salle, deux objets aléatoires sont placés.
    - Chaque joueur automatique est équipé d'armes de rareté ordinaire
    - Les joueurs sont répartis équitablement dans les salles du château
    - Le choix des joueurs automatiques est fait pour avoir une répartition équitable entre les types de personnages.
- Plus le joueur manuel progresse dans le jeu (et tue des joueurs automatiques), plus les joueurs automatiques deviennent forts.
- Le jeu ne rajoute pas de joueurs automatiques au fur et à mesure du jeu. Le nombre de joueurs automatiques ne fait que diminuer.

Les Classes Château et Salle

Présentation des classes

Une objet de type Salle peut contenir des joueurs et des objets. Selon la position de la salle dans le château, elle a un maximum de quatre salles voisines. La classe Salle gère toutes les opérations qui la concerne, comme placer un joueur ou un objet dans la salle.
La classe Chateau gère les salles, et veille à bien initialiser les vecteurs de voisinage d'une salle.

Suppositions:

- Après qu'un joueur manuel quitte une salle, les objets dans cette salle sont supprimés, et nous rajoutons à la salle deux nouveaux objets.

Les Classes Joueur, Joueur Manuel et Joueur Automatique

Présentation des classes

Nous avons fait le choix de faire une distinction entre les classes Joueurs et les classes Personnage. Les classes Joueur gèrent plutôt les actions et choix possibles des joueurs, alors que les classes Personnage gèrent surtout les statistiques, le sac, etc, des personnages qui sont dans le jeu. Cette distinction permet d'encapsuler les choix (automatisé ou sur entrée) des joueurs.

Concepts de POO Utilisés

Nous avons une classe Joueur qui est abstraite, puis deux classes JoueurManuel et JoueurAutomatique, qui en héritent. Cela nous permet, grâce au polymorphisme d'héritage, d'appeler des fonctions différentes mais qui servent à la même chose: faire des choix de joueurs. Dans le cas du JoueurManuel, les fonctions demandent le choix au joueur, mais dans le cas d'un JoueurAutomatique, ce choix est automatisé.

Suppositions:

Les joueurs manuels:

- Ne peuvent pas quitter une salle avant d'avoir tué tous les ennemis dans cette salle.

Les joueurs automatiques:

- Peuvent utiliser des potions au milieu de combat
- Ne peuvent pas ramasser des objets dans les salles
- Font tomber tous leur équipement/sac dans la salle suivant leur mort
- Peuvent changer de salle

Les Classes Personnage

Présentation des Classes

La classe personnage gère le personnage qui est physiquement dans le jeu, et non le joueur qui contrôle le personnage. Donc, le personnage est celui qui a un équipement et un sac, etc.
Un personnage a des statistiques différentes, chacune servant un but précis:

Nom	Description
Sante	Valeure initiale: 100 Valeure maximale: gérée par MAX_SANTE Peut être augmentée uniquement par des potions Diminue uniquement dans les combats Dès qu'elle devient 0, la mort du joueur est déclenchée
Habilete	Valeure initiale: 10 Peut être augmentée uniquement par des potions Ne diminue pas Permet de porter le premier coup dans un combat, et augmente les chances d'effectuer un coup critique dans un combat
AttaquePhysique	Valeure intiale: dépend du type de personnage Peut être augmentée par des potions, ou par équipement Type de force
AttaqueMagique	Valeure intiale: dépend du type de personnage Peut être augmentée par des potions, ou par équipement Type de force
resistancePhysique	Valeure intiale: dépend du type de personnage Peut être augmentée par des potions, ou par équipement Type de force
resistanceMagique	Valeure intiale: dépend du type de personnage Peut être augmentée par des potions, ou par équipement Type de force

Nous avons quatre types de personnages: Amazone, Guerrier, Moine et Sorciere. Les statistiques de ses personnages varient selon leur catégorie.

Suppositions

Nous avons quelques suppositions concernant les classes personnage:

- La taille de l'équipement est gérer par la constante préprocesseur TAILLE_EQ, et est fixée à 2
- La taille du sac est gérer par la constante préprocesseur TAILLE_SAC, et est fixée à 4

Les Classes Objets et ObjectFactory

Présentation des Classes

Chaque objet a un nom, un identifiant de type, une rareté, et est soit équipable ou utilisable. La classe objet est une classe virtuelle.

Il existe trois classes d'objets qui héritent de la classe Objet:

- Arme
- Potion
- Clef de téléportation

Les armes et potions sont classés en catégories selon leur rareté et leur force: ordinaire, extraordinaire, et légendaire, avec quelques armes hors norme, qui sont beaucoup plus rares.

La Classe Arme

La classe arme est la seule catégorie d'objets équipables. Chaque arme a quatre catégories de statistiques, identiques aux catégories des personnages, donc dommage physique, dommage magique, défense physique, et défense magique. Quand un personnage s'équipe d'une arme, les statistiques du personnages augmentent de manière correspondants aux statistiques de l'arme.

La Classe Potion

La classe potion, avec la classe Clef, est dans la catégorie des objets utilisables (ou consommables). Une potion a un type, qui prend valeur dans les catégories de statistiques possibles d'un personnage, un boost, et un booléen indiquant s'il s'agit d'un poison. Le type de la potion indique sur quelle statistique elle va agir. Le boost indique la quantité qui va être ajoutée ou enlevée à la statistique. Si la potion est un poison, la valeur du boost va être retirée à la statistique, sinon elle va lui être rajoutée.
Les potions peuvent être utilisées au milieu d'un combat.

La Classe Clef

La classe Clef fait partie de la catégorie des objets utilisables. Il n'existe qu'un type de clef (dans le vecteur d'objets possible dans la classe Jeu): une clef de téléportation. Une clef permet au joueur de changer de salle sans que son tour se finisse, et sans tuer tous les ennemis présents dans la salle. Elle n'est pas utilisable en combat.

La Classe ObjectFactory

La classe ObjectFactory gère la création des objets selon leurs raretés. Elle est la seule classe qui peut créer des objets.

Concepts de POO Utilisés

Pour la création d'objets, nous utilisons clairement la Factory Pattern. Un objet de type ObjectFactory est initialisé avec un vecteur d'objets possibles (fourni par la classe jeu). Puis, pour toute création d'objet, il suffit d'appeler la fonction pick() de l'objectFactory (ou une autre de ses variations), et la classe renvoie un pointeur d'objet choisi en prenant compte les raretés des objets.
De plus, notre modèle est très accommodant au rajout d'objets différents. Il suffit de rajouter des instances de ces objets dans le vecteur initialisé dans le classe Jeu.
Pour ajouter des nouvelles classes d'objets, cela n'est pas plus compliqué. Il suffit juste de spécifier s'ils sont utilisables ou consommables, s'ils sont utilisables en combat ou pas, et ajoutez une catégorie pour gérer leur utilisation dans la classe Joueur Manuel.

Suppositions:

- Les potions et les clefs de téléportation sont à usage unique, alors que les armes ne le sont pas.

La Class Combat

La classe combat permet d'encapsuler les mécanismes qui se déroulent lors d'un affrontement entre deux joueurs. Cette classe admet deux attributs de type Joueur* (représentant les deux adversaires) ainsi que les méthodes commencerCombat, calculDegatPhysique et calculDegatMagique. Ces deux dernières permettent évidemment de calculer les dégâts qu'inflige l'attaquant sur son ennemi. La fonction commencerCombat permet quant à elle de gérer le tour à tour du combat (Le joueur j1 attaque j2, puis j2 attaque j1) et indique l'issue du combat. Une fois le combat terminé, la classe Jeu prend le relais et s'occupe de supprimer du jeu et de la mémoire l'ennemi, si celui-ci a été vaincu, ou si vous avez vous-même perdu le combat, de gérer la fin de la partie.

La Classe Utilities

La classe utilities est, comme son nom l'indique, une classe procurant des fonctions récurrentes utiles, auquel nous faisons appel plusieurs fois dans notre projet, tel que: une fonction qui génère un nombre aléatoire selon un intervalle donné, ou une fonction qui valide les entrées du joueur pour s'assurer qu'elles sont conformes.

Améliorations Possibles

Feature Multijoueurs:
Nous avons déjà commencé à implémenter une possibilité d'avoir plusieurs joueurs manuels dans un même jeu. Le but serait de collaborer entre joueurs manuels contre les ennemis.
Pour pouvoir implémenter ceci, il reste à faire les choses suivantes:

- Interdire aux joueurs manuels de lancer des combats entre eux
- Laisser les joueurs manuel changer de salle si elle ne contient pas d'ennemis (et non si le joueur est seul dans la salle).
- Considérer qu'un jeu est gagné quand il n'y a plus d'ennemis (et non quand il reste un unique joueur manuel)
- Considérer qu'un jeu est perdu quand tous les joueurs manuels sont vaincus.

Mode Survie :
Ceci consisterait à créer un mode où le joueur manuel ne peut pas gagner, et le but est uniquement de voir pendant combien de tours le joueur peut survivre. Cela voudrait dire que le nombre d'ennemis devrait être augmenté constamment, et que la condition de victoire soit annulée.

Sauvegarde de Parti:
Ceci consisterait en la possibilité de pouvoir sauvegarder une partie au milieu, et au début du jeu avoir la possibilité de relancer la partie.
Interface Graphique
Malgré le côté rustique et charmant de l'interface textuelle, une interface graphique serait une amélioration considérable au niveau du jeu.