Projet : Programmer le prototype d'un RPG en mode texte

I. Jeu à développer

L'objectif de ce projet est de développer le prototype d'un jeu vidéo de type « RPG[1] ».

Le personnage joué évolue dans un environnement hostile, où il rencontre des ennemis qu'il doit combattre et vaincre afin de gagner de l'expérience. Il peut également trouver et récupérer différents objets lui permettant d'améliorer l'équipement ou les capacités de son personnage.

Votre jeu devra proposer plusieurs thèmes distincts, le thème sera choisi au lancement d'une partie. Par exemple : médiéval fantastique, préhistoire, *steampunk* post-apocalyptique, etc.

Le thème choisi aura une incidence sur toutes les caractéristiques du jeu : spécificités de l'environnement, ennemis rencontrés, objets récoltés, classes de personnages jouables, etc.
Par exemple :

- Si le thème « médiéval fantastique » est choisi, le personnage pourra :
  - utiliser des pouvoirs magiques : boules de feu, éclairs, etc. ;
  - récupérer des épées, des lances, des baguettes magiques, des casques, des armures, etc. ;
  - ramasser des potions de vie (pour se soigner) et des potions de mana (pour la magie) ;
  - rencontrer des chevaliers errants, des dragons, des chauves-souris, des rats enragés, etc.
- Si le thème « préhistoire » est choisi, le personnage pourra :
  - récupérer des bâtons, des pierres, des frondes, etc. ;
  - rencontrer des mammouths, des tigres à dents de sabre, etc. ;
  - ramasser de la viande et des fruits pour se nourrir, ou des brindilles pour faire des feux.

Vous êtes libres de choisir les thèmes que votre jeu proposera. Votre jeu devra proposer au moins 2 thèmes différents dès la mise en œuvre du prototype.

Dans la suite de ce sujet, pour illustrer le déroulement du jeu, nous prendrons l'exemple du thème « médiéval fantastique ». Pour autant, gardez en mémoire que votre jeu devra proposer plusieurs thèmes distincts.

1 RPG : Role-Playing Game. Voir : https://fr.wikipedia.org/wiki/Jeu_vid%C3%A9o_de_r%C3%B4le

II. Consignes de réalisation

1. Interface de jeu

Le jeu devra être jouable *dans un terminal*, avec une interface *textuelle* permettant de choisir le thème du jeu, créer son personnage, puis effectuer un certain nombre d'actions avec son personnage. Toutes les interactions avec le jeu s'effectueront au travers d'un menu textuel ; il n'est pas demandé de faire d'interface graphique.

2. Conception et développement

Pour réaliser ce projet, vous devez utiliser le langage **`Java`**, et faire appel à vos connaissances des **designs patterns** pour la phase de conception.

Votre code suivra une architecture MVC, de manière à répartir le code en trois *packages* :

- Le package `modele` contiendra les classes et interfaces permettant de modéliser et manipuler l'ensemble des données du jeu.
- Le package `vue` contiendra une seule classe `Ihm`, permettant d'interagir avec le jeu et afficher les informations.
- Le package `controleur` contiendra *a minima* la classe `Controleur`, qui sera en charge du déroulement du jeu, et qui communiquera avec le modèle et la classe `Ihm`.

En dehors de ces trois *packages*, vous ajouterez une classe `Main` qui contiendra le `main`.

3. Constitution des groupes de projet (avant le 27 novembre 2023)

Ce projet est à réaliser en binôme ou en trinôme. Vous pouvez constituer votre binôme ou trinôme avec des étudiant · es d'un autre groupe que le vôtre.

Avant le 27 novembre 2023, sur la page Célène du cours, vous devez ajouter les noms des membres de votre groupe dans le wiki nommé « **Constitution groupes projet** » correspondant à votre groupe. Cela nous permettra de créer le dépôt *Git* pour votre projet.

Pour modifier le tableau du wiki, il faut cliquer sur le wiki, puis ouvrir la liste déroulante indiquant « Afficher » et sélectionner « Modifier ».

Inscrivez votre binôme ou trinôme dans l'un des groupes. Nous réorganiserons les groupes pour les équilibrer si besoin.

III. Travail à réaliser

1. Initialisation de la partie : thème du jeu, et classe du personnage

Au lancement du programme, on pourra effectuer plusieurs choix afin d'initialiser une partie.

Votre jeu affichera donc plusieurs menus successifs :

1. *Choix du thème.*
   Par exemple : « Médiéval fantastique », « Préhistoire », « Steampunk ».
   Pour rappel, le thème a une incidence sur l'ensemble des spécificités du jeu.
2. *Création du personnage.*
   On devra choisir une classe de personnage parmi plusieurs classes disponibles. Les classes proposées dépendront du thème choisi. Par exemple, pour le thème « médiéval fantastique », on pourra proposer les classes « barbare », « archer · e », « sorcier · e ».
   La classe choisie déterminera l'équipement et les capacités initiales du personnage.
   On devra également choisir un nom pour le personnage joué.
   Vous pouvez ajouter des paramètres supplémentaires (cri de guerre, etc.) à la création s'ils vous semblent pertinents pour votre jeu.

Le personnage est créé avec des caractéristiques déterminées essentiellement par la classe à laquelle il appartient. Par exemple, pour le thème « médiéval fantastique », un personnage « barbare » aura plus de points de force que les autres classes, mais moins de points d'intelligence qu'un · e sorcier · e, ou moins de dextérité qu'un · e archer · e.

Les capacités d'une classe sont définies par des attributs de base, qui influent sur divers aspects du jeu. Dans la majorité des RPGs, on retrouve les capacités suivantes :

- la **force**, qui quantifie la force physique du personnage : dégâts infligés, charge physique qu'on peut porter…
- la **dextérité**, qui détermine l'habileté au combat, et la probabilité d'échapper aux pièges.
- la **constitution,** qui influe sur la capacité à récupérer les points de vie avec le temps, ainsi que sur le maximum de points de vie.
- L'**intelligence**, qui mesure la capacité à manipuler la magie.

Vous pouvez ajouter d'autres capacités si vous le souhaitez, voire proposer des capacités différentes selon le thème.

2. Menu principal du jeu

Une fois initialisé, le programme affiche le menu principal du jeu. Ce menu propose deux options :

- **Afficher l'inventaire du personnage**, c'est-a-dire tous les objets qu'il possède.
  Certains objets sont fournis dès le démarrage du jeu, tandis que d'autres peuvent être récupérés durant le jeu. Parmi ces objets, il peut y avoir des aliments ou des potions, qui peuvent être consommés, de l'équipement (armure, casque, chaussures, arme, etc.).
  Le personnage peut alors utiliser cet équipement de plusieurs façons :
  - Consommer un objet consommable : manger une pomme, boire une potion, etc.
  - Jeter un objet, afin de libérer de l'espace dans son inventaire.
  - S'équiper avec un objet de son inventaire.
    Attention, le personnage ne peut porter qu'un casque, qu'une paire de chaussures, qu'une (ou éventuellement deux) arme(s) à la fois, même si son inventaire en contient davantage. Ainsi, même si le personnage dispose de plusieurs paires de chaussures dans son inventaire, il ne pourra porter (au plus) qu'une seule paire.
- **Entrer dans le donjon.**[2]
  Une fois entré dans le donjon, le personnage va devoir traverser un certain nombre de pièces, et les débarrasser de tous les ennemis qui s'y trouvent.

3. Phase de combat

a) Déroulement de la phase de combat

À l'entrée dans une pièce, une phase de combat se déclenche. Elle se déroule à tour de rôle :

1. Le personnage attaque un ennemi, choisi parmi ceux encore vivants dans la pièce,
   ou il récupère un objet présent dans la pièce.
2. Tous les ennemis *encore vivants* dans la pièce portent une attaque au personnage, un par un.
3. On retourne au point 1.

Pendant le combat, et à chaque coup porté ou reçu, l'interface *dans le terminal* doit afficher précisément l'évolution de l'état de santé de tous les participants au combat, au fur et à mesure.

2 Nous prenons ici l'exemple du donjon, dans le cas où le thème choisit serait « médiéval fantastique ».

b) Fin de la phase de combat

La phase de combat se termine dans deux cas :

1. Le personnage joué n'a plus de points de vie. Dans ce cas, le combat est perdu.
   Deux options se présentent alors :
   a) Retourner dans le passé, et donc à revenir dans une précédente pièce du donjon par laquelle il est déjà passé. Dans ce cas, le personnage joué récupérera le même état qu'il avait en entrant dans cette pièce et le même inventaire. Par contre, la pièce pourra contenir à nouveau des ennemis. *Le personnage n'a le doit de retourner dans le passé que deux fois au cours d'une traversée du donjon.*
   b) Quitter le donjon. Dans ce cas, le programme revient au menu principal du jeu. Le personnage sort du donjon avec un seul point de vie ; il a perdu tous les objets pouvant être consommés de son inventaire, mais pas son équipement ni les autres objets ramassés.

- Tous les ennemis présents dans la pièce sont morts. Plusieurs options se présentent alors :
  - Quitter le donjon, et revenir au menu principal. Dans ce cas, il faudra recommencer le donjon depuis le début, et la progression dans le donjon sera perdue.
  - Accéder à l'inventaire, pour consommer un élément ou changer d'équipement. Dans ce cas, le personnage ne quitte pas le donjon, il reprendra sa progression là où il s'est arrêté.
  - Passer à la pièce suivante.

Une fois que le personnage a atteint la dernière pièce du donjon et tué tous les ennemis qu'elle contenait, on revient au menu principal du jeu, et le personnage gagne des objets dans son inventaire.

Toute fonctionnalité supplémentaire, qui peut impliquer un patron de conception pertinent, sera la bienvenue !

4. Visualisation de l'environnement et des personnages

Quand le personnage entre dans une pièce du donjon, on doit pouvoir visualiser le personnage, les objets présents dans la pièce, ainsi que tous les ennemis. Vous pourrez représenter chaque élément avec une couleur et un symbole.

Voici quelques exemples de représentation que vous pouvez utiliser :

Symbole	Élément	Type
@	Personnage	Joueur
$	Coffre	Objet

IV. Rendu du travail

1. Première partie à rendre avant le 22 décembre 2023

Pour ce premier rendu, vous utiliserez la zone appelée « Dépôt Conception Projet » sur Célène.

Vous y déposerez une archive au format « .zip » contenant les fichiers suivants au format PDF :

- un ou plusieurs fichiers contenant les diagrammes de classes de conception ;
- un fichier explicatif justifiant vos choix de conception, et plus particulièrement vos choix de designs patterns.

2. Deuxième partie à rendre avant le 8 janvier 2024

Pour ce deuxième rendu, vous utiliserez le dépôt *Git* qui vous aura été fourni au début du projet.

Le dépôt devra contenir :

- un document indiquant si des éléments de conception ont changé depuis le dépôt précédent ;
- un dossier contenant les sources de l'implémentation de votre jeu.

V. Annexes

1. Gérer les couleurs dans les messages affichés en console

Pour afficher de la couleur dans la plupart des consoles, voici quelques informations.

Vous pouvez mettre de la couleur dans la sortie standard en utilisant les valeurs ci-dessous :

```
// Revient aux couleurs par défaut du terminal
public static final String ANSI_RESET = "\u001B[0m";
// Couleurs pour le texte
public static final String ANSI_BLACK = "\u001B[30m";
public static final String ANSI_RED = "\u001B[31m";
public static final String ANSI_GREEN = "\u001B[32m";
public static final String ANSI_YELLOW = "\u001B[33m";
public static final String ANSI_BLUE = "\u001B[34m";
public static final String ANSI_PURPLE = "\u001B[35m";
public static final String ANSI_CYAN = "\u001B[36m";
public static final String ANSI_WHITE = "\u001B[37m";
// Couleurs pour le fond
public static final String ANSI_BLACK_BACKGROUND = "\u001B[40m";
public static final String ANSI_RED_BACKGROUND = "\u001B[41m";
public static final String ANSI_GREEN_BACKGROUND = "\u001B[42m";
public static final String ANSI_YELLOW_BACKGROUND = "\u001B[43m";
public static final String ANSI_BLUE_BACKGROUND = "\u001B[44m";
public static final String ANSI_PURPLE_BACKGROUND = "\u001B[45m";
public static final String ANSI_CYAN_BACKGROUND = "\u001B[46m";
public static final String ANSI_WHITE_BACKGROUND = "\u001B[47m";
```

Voici un petit exemple qui affiche : `Hello, World!`

```
System.out.println(ANSI_PURPLE_BACKGROUND + ANSI_WHITE
     + "Hello, World!" + ANSI_RESET);
```

Pour plus d'informations sur les couleurs : ANSI Escape Code (Wikipedia)