INFO-H-200 : Programmation orientée objet
Projet

Professeur : Hugues Bersini
http://cs.ulb.ac.be/public/teaching/infoh200

Année académique 2016-2017

---

1 Sujet

On vous demande de réaliser un jeu du type Hack and Slash / Dungeon Crawler en Java s'accompagnant d'une GUI. Le délivrable devra présenter une interface graphique et être implémenté en orienté-objet. Par ailleurs, votre code devra mettre en pratique l'ensemble des concepts OO vus durant les séances de travaux pratiques.

Un jeu de type Dungeon Crawler peut se décrire comme suit : Il s'agit d'une sorte de jeu de rôle dans lequel un héro parcours un labyrinthe (a.k.a. Le donjon) en tuant des ennemies et en ramassant les objets qu'il peut trouver ou ceux résultants du décès d'un enemmi.

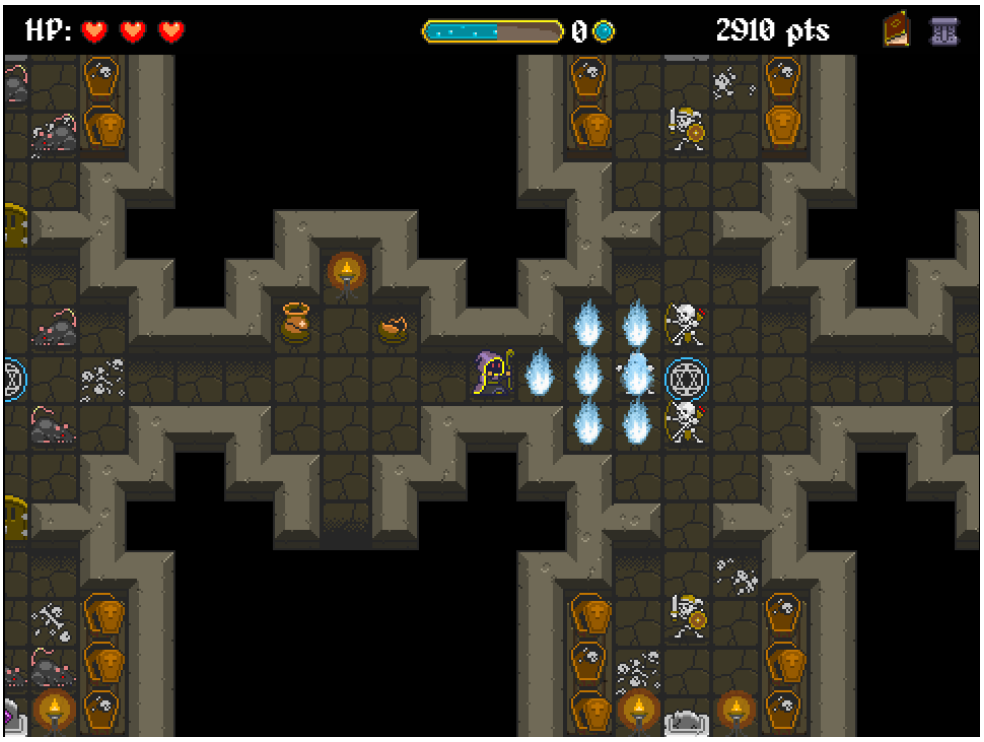

FIGURE 1 – Exemple : The Wizard https ://youtu.be/XvnUZ_kHcaw

De nombreux exemples peuvent être cités et nombreuses sont les variations existant autour de cette thématique. Nous nous contentons ici (Voir Figure 1) de vous fournir un petit exemple simple illustrant le principe sur un cas simple implémenté en 2D sur une carte vue du haut, soit ce que nous attendons précisémment pour ce projet.

2 Cartes

La plateau de jeu devra être d'une taille paramétrable par l'utilisateur, par exemple 50x50. Chaque plateau pourra soit être généré aléatoirement, en plaçant différents objets, ennemis et portes sur la carte au chargement, soit en lisant la description de ceux-ci directement depuis des fichiers. Le joueur pourra ainsi se déplacer suivant quatre directions : Nord, Sud, Ouest et Est. Nous vous laissons choisir le mode d'enchaînement des cartes parmi les trois propositions suivantes :

- Carte statique : Une fois l'ensemble des monstres tués, une nouvelle carte est générée. (e.g : Survival)
- Labyrinthe : Plusieurs cartes existent et sont liées par un ensemble de portes. Ces cartes sont mémorisées pour assurer la cohérence des passages d'une pièce à l'autre. (e.g : Pokemon ou Hack and Slash classique)
- Fenêtrage : La carte sera d'une taille bien plus importante, par exemple 10.000 x 10.000 mais le joueur ne pourra visualiser qu'une partie de 50x50 dans une même fenêtre. Le déplacement de celui-ci fera se déplacer la fenêtre. (e.g : Command and Conquer)

Le personnage pourra donc se déplacer, ramasser des objets, tuer des ennemis, etc...

3 Objets / Inventaires

Différents objets doivent être placés sur la carte ou peuvent être récupérés au sol après avoir éliminé un ennemi. On vous demande de pouvoir gérer un inventaire d'une taille fixe. Les objets pourront donc être ramassés, utilisés ou jetés.

4 Armes

Dans sa version la plus simple, le personnage aura une seule attaque à sa disposition. Une épée ou pour un mage, une attaque directe et non à distance.

5 Bonus et améliorations

D'autres fonctionnalités doivent être implémentées, quelques exemples sont fournis ci-dessous. Attention, un bonus ne sera validé comme apport supplémentaire qu'à la condition qu'il implique un effort de développement. Par exemple, proposer différentes potions 100HP, 200HP,... ne compte pas comme bonus car il s'agit, non d'un nouvel objet avec un comportement différent, mais d'un objet avec une valeur d'attribut modifiée.

- Différents objets : Potion de vie instantanée ou heal-over-time, armes, armures, potion d'invisibilité, potion d'invulnérabilité,...
- Des objets particuliers permettant d'accroître la taille de l'inventaire. Des modifications de l'équipement courant (i.e : Changer d'arme en passant d'une épée à un arc)
- Différents effets : Attaques à distance, Damage over time, Area damage, Ralentissement (i.e : Snare), Etourdissement, Etat d'ébriété (i.e : les commandes claviers seraient inversées), attaques spéciales ne pouvant être employées qu'à une fréquence donnée (i.e : 1x par minute),...
- Des emplacements de la carte pourraient soumettre l'utilisateur à un ralentissement de déplacement. D'autres pourraient présenter des pièges.
- Les ennemis pourraient se déplacer selon un schéma prédéfini, attaquer le joueur ou le fuir, les ennemis pourraient aussi bien être des kamikazes,...
- Téléportation, Retour dans le passé,...
- Apprentissage de compétences, Choix d'un classe de personnage,...
- Sauvegarde, Chargement, Checkpoint.
- Implémentation de voleurs (i.e : Un objet pourraient vous être dérobé mais serait récupérable sur la dépouille de celui-ci)
- Intelligence artificielle : les déplacements des ennemis se seraient alors plus simplement aléatoires mais prendraient en compte certains facteurs.
- Multi-joueur

Le principe du jeu est simple et nous n'offrons qu'un seul sujet, libre à vous de choisir les fonctions que vous désirez implémenter ! Toutefois, certains fonctionnalités sont refusées :

— Codes de triche
— Zoom sur la carte
— Minimap
— Implémentation d'un scénario de jeu ou de dialogues pour les PNJs.
— Choix de caractéristiques physiques (i.e : altération de l'apparence de votre personnage)
— Système de fabrication d'objets (i.e : Craft)

6 Consignes

On vous demande d'implémenter une application graphique 2D en Java en illustrant les concepts orientés-objets vus durant les cours et aux séances de travaux pratiques (e.g : Héritage, Encapsulation, Exceptions, Interfaces, Généricité, Polymorphisme, Design patterns, Threads, etc...).

Le projet peut être réalisé individuellement ou par groupe de **2 étudiants** au maximum. Vous n'êtes pas obligés de faire partie de la même série de TP pour former un groupe. La défense aura lieu en salle informatique et comportera, outre une démonstration de votre jeu, des questions individuelles. Chaque étudiant devra donc connaître et comprendre l'ensemble du programme présenté et sera interrogé sur sa conception. La démonstration pourra se faire soit sur les ordinateurs des salles, soit sur votre ordinateur personnel.

7 Exemple de jeu

Afin de vous permettre de démarrer l'implémentation de jeu rapidement et ce, avant la fin des TPs sur les Interfaces Graphiques, les Threads et les Designs Patterns (MVC & Observateur), un exemple vous est fourni. Celui-ci consiste en une version minimaliste d'un **Bomberman**. Vous y trouverez une structure pré-établie ainsi que des exemples d'utilisation du clavier, de la mise à jour d'une carte à l'écran ainsi que un exemple d'utilisation de threads. Le code vous est fourni ainsi qu'un diagramme UML de Classes mettant en avant sa structure.

8 Délivrables

8.1 Rapport préliminaire

Un rapport de maximum 5 pages portant sur la phase d'analyse fonctionnelle, de conception et de modélisation vous sera demandé pour le **30 Mars**. Ce rapport devra contenir les éléments suivants :

1. Une description de votre architecture orientée objet accompagnée d'une explication sur les choix posés.
2. Une description des différentes fonctionnalités qui seront implémentées.

Au moins dix fonctionnalités de votre choix doivent être détaillées. Leur pondération sera discutée avec chaque groupe après la remise afin de garantir une charge de travail équitable. Un bonus de maximum 2 points sera accordée dans le cas où le nombre de fonctionnalités additionnelles implémentées à l'issue du projet dépasse ce qui avait été défini dans le rapport préliminaire.

8.2 Rapport final

Le rapport final devra décrire l'ensemble du travail réalisé avec les éléments suivants :

1. Une description synthétique des fonctionnalités implémentées
2. Les diagrammes UML : Classe et Séquence (boucle de jeu principale)
3. Une description complète et justifiée de votre architecture finale en indiquant les Design Patterns qui ont été employés.

Le rapport final devra être envoyé par email à **mwaumans@ulb.ac.be** pour le **14 Mai** accompagné du code source complet et commenté. Le sujet de votre email devra indiquer précisément **INFOH200_Projet** afin de faciliter de traitement de l'afflux massif d'emails attendu durant cette période.

9 Evaluation

La défense du projet consistera en une brêve démonstration de votre programme ainsi que quelques questions relatives à votre code. La défense aura lieu durant la semaine suivant la remise et se fera suivant un horaire à déterminer.

L'évaluation du projet portera les critères suivants :

- L'architecture du projet
- Le respect des règles de bonnes pratiques de la programmation orientée objet
- La maîtrise des concepts vus aux cours et aux travaux pratiques
- La qualité et la clarté de votre implémentation (e.g : Structure claire, commentaires appropriés,...)
- Les possibilités de paramétrage de l'application
- Le contenu du rapport
- La pertinence de vos réponses aux questions lors de la défense.