Ludogiciel

Chasse au Monstre

1 Spécifications : le jeu de la chasse au monstre

Pour le projet du second semestre, vous allez développer un ludogiciel appelé *la chasse au monstre* (CaM), dont les règles sont les suivantes :

> Le monstre et le chasseur sont sur un espace identique découpé en cases. Dans un premier temps, on peut supposer que c'est une grille carrée (mais ce n'est pas très important, ça pourrait être une grille rectangulaire, une île aux formes mal définies, un tore, on pourrait y ajouter des obstacles, pourquoi pas des étages, des pièges, des passages secrets...).
>
> Le but du jeu pour le chasseur est de trouver le monstre, pour le monstre d'avoir exploré chaque case de l'espace de jeu sans avoir été trouvé. Un tour de jeu consiste en un coup du monstre suivi d'un coup du chasseur.
>
> Le monstre débute la partie : il choisit une case dans l'espace de jeu, et marque la case avec un 1 (qui signifie : j'étais sur cette case lors du premier tour de jeu). Ensuite, à chaque tour de jeu, le monstre a l'obligation de se déplacer sur une case adjacente non explorée, et de marquer cette case avec l'entier correspondant au tour de jeu. Une case adjacente est une case située autour de la case du monstre (au-dessus, en-dessous, à droite, gauche, ou en diagonale). Si le monstre ne peut se déplacer lors de son tour de jeu, il a perdu la partie.
>
> Le chasseur propose une des cases de l'espace de jeu. Il apprend l'état de la case : jamais explorée, explorée lors du nième tour de jeu, occupée par le monstre. Le chasseur peut ainsi, peu à peu, retracer l'itinéraire du monstre et le débusquer. Le chasseur a gagné s'il trouve le monstre. Il perd si le monstre a exploré tout l'espace de jeu sans avoir été trouvé.

Il vous est demandé de réaliser une version électronique de ce jeu. Les fonctionnalités minimales requises pour le logiciel sont :

- une ergonomie qui dépend des joueurs (IA, humains) ;
- une IHM pour contrôler les différents personnages (chasseur, monstre) ;
- de multiples versions des règles (déplacements, plateau de jeu, retour d'information) ;
- gérer une IA pour chaque personnage (chasseur, monstre).

Les travaux se font par équipes de 4 étudiants (ou 3 ou 5, selon la taille des groupes TD) avec cette contrainte que chaque équipe se constitue à l'intérieur d'un unique groupe TD. Ce projet contribuera à vos notes de Projet et Gestion de Projets.

2 Calendrier prévisionnel

Le projet est découpé en deux parties (livrables), chacune devant être rendue à une date précisée dans le calendrier prévisionnel, avec une présentation synthétique de votre travail.

calendrier prévisionnel

Date	Description
Lundi 18 mars (S12) à 12h	Lancement du projet
S12 / S13 / S14	Encadrement standard (1h) + libre (2h+)
S15 / S16	Vacances

S17	Présentation Jalon 1
Mercredi 1 mai (S18) à 23h	Dépôt Jalon 1 sur Moodle
S19 / S20 /S21 / S22	Encadrement standard (1h) + libre (2h+)
S23	Présentation Jalon 2
Mercredi 12 juin (S24) à 23h	Dépôt Jalon 2 sur Moodle

Avant le rendu de chaque jalon, il vous est demandé de préparer une présentation succincte (5 min maximum) de votre travail. Cette présentation doit mettre en valeur votre travail en général mais aussi mettre en exergue les spécificités de votre projet.

Elle sera suivie de quelques questions qui nous serviront à mieux évaluer votre travail. Cette séance (20 min maximum) se fera donc en présence de l'équipe du projet au complet et des deux encadrants.

3 Livrable 1 : le jeu en mode texte

Pour une première version du jeu, nous nous focaliserons sur le modèle. Pour le livrable 1, vous devez donc rendre :
- un programme java exécutable qui permet de jouer à CaM en mode texte ;
- le code source de votre programme
- des tests unitaires pertinents
- une maquette de conception pour l'IHM pour la suite du projet.

2.1 Programme exécutable

Pour le livrable 1 vous devez faire un programme java exécutable (.jar) qui prend en paramètre différentes options (version des règles, plateau personnalisé, *etc.*) ou le chemin vers un fichier texte contenant les options de jeu. Si la liste des paramètres contient des erreurs, alors le programme affichera un message d'erreur le plus précis possible et rappellera sa syntaxe d'utilisation.

2.2 Code source

Votre source doit être structuré en différents paquetages regroupant les unités sémantiques de votre code. Vous pouvez le rendre sous forme d'une archive zip ou mieux dans un fichier jar.

2.3 Tests unitaires

Vous devez écrire des tests unitaires pour toutes les classes. Toutes les classes de test doivent se trouver dans un dossier de code source séparé, nommé `test`, et qui est à côté du dossier `src`.

2.4 Maquette de conception

Il vous est demandé de préparer la réflexion sur votre IHM du jalon 2 dès maintenant. Vous devez rendre l'état de votre réflexion au livrable 1. Cette réflexion n'est pas définitive, ni ne vous engage pour la suite mais doit exposer l'état de votre réflexion quant à l'ergonomie finale du jeu en particulier dans le cas du jeu à plusieurs humains.

2.5 À rendre

Vous devez rendre une archive dont la structuration est la suivante :
- un fichier `readme.md` qui donne un premier aperçu du projet ;
- une archive jar exécutable qui permet de lancer votre programme ;
- un répertoire `data` qui contient les fichiers de données dont vous pourriez avoir besoin (plateaux personnalisés, historique, *etc.*) ;
- un répertoire `doc` qui contient la documentation ;

— un répertoire `test` qui contient les tests.

3 Livrable 2 : version aboutie (IHM, réseau)

Ce livrable contiendra la version aboutie de votre projet. CaM aura alors toutes les fonctionnalités demandées, ainsi que toute autre fonctionnalité qui vous semble enrichissante.

3.1 IHM, ergonomie du jeu

Fonctionnalités requises L'utilisateur doit pouvoir :
- jouer à plusieurs sans possibilités de triche ;
- utiliser une IA pour n'importe quel personnage ;
- choisir les options de jeu (plateau, règles de déplacement, retour d'information, *etc.*).

Fonctionnalité optionnelle L'utilisateur peut :
- charger des cartes de l'utilisateur (définir alors un format de carte) ;
- jouer sur deux ordinateurs dans le cas PvP.

3.2 Tests unitaires

De même, toutes les classes de test doivent se trouver dans un dossier de code source séparé, nommé `test`, et qui est à côté du dossier `src`.

3.3 Documentations diverses

Vous devez produire un **schéma UML**. Le schéma doit être *synthétique*, et aider à comprendre l'architecture du logiciel. Il doit contenir les classes principales, et seulement leurs méthodes principales, doit montrer les associations et leur donner un nom. En aucun cas le schéma UML ne doit être celui qu'on obtient par génération automatique à partir du code.

Vous devez également produire une **documentation de vos classes au format Javadoc**. Cette documentation doit être générée à l'aide de l'outil javadoc sur vos classes convenablement commentées.

Finalement, vous devez faire une **vidéo de présentation du jeu dans ses différentes configurations**. La vidéo durera 2-3 minutes et sera faite par capture d'écran (voir les instructions du premier semestre). Elle doit illustrer l'utilisation du programme CaM. La vidéo peut être sous-titrée ou contenir des explications parlées. Vous veillerez à présenter votre logiciel sous son meilleur jour.

3.4 À rendre

Vous devez rendre :
- une archive jar exécutable qui lance le programme spécifié dans la Section 1, et qui contient tous les fichiers source ;
- si nécessaire, un répertoire [`data`] qui contient les données ;
- une vidéo de présentation, dans le répertoire idoine ;
- un rapport d'une à deux pages qui décrit comment chaque membre de l'équipe a contribué à la réalisation du livrable ;
- un schéma UML ;
- la documentation au format Javadoc.

4 Objectifs, exigences, organisation du travail

Ce projet tutoré poursuit plusieurs objectifs pédagogiques.

D'abord, il y a des objectifs techniques : pratique de la programmation java, description de code grâce à des schémas synthétiques (UML), *etc.*

Nous visons également des apprentissages non techniques, mais tout aussi importants. Le projet est relativement conséquent et sera fait en groupes de 4 à 5 personnes dont la composition est de votre responsabilité. Le mener à bien demandera de bien s'organiser, se coordonner, communiquer. Ainsi, la note de projet tiendra compte non seulement de la qualité technique du logiciel réalisé, mais aussi de votre capacité à vous organiser pour mener ce travail.

Finalement, nous testons votre capacité à utiliser un spécification écrite (ce document). Ceci implique la compréhension complète et non ambiguë du document : relisez le document souvent, mettez en doute votre compréhension, posez des questions !

D'après le programme du DUT, ce projet demande au moins 30 heures de travail individuel par étudiant, soit 3 heures par semaine en moyenne. C'est à chacun d'entre vous de s'assurer qu'il fait sa partie du travail.

Travail régulier et équitable, notation La note de projet tiendra compte de :

- la régularité du travail : un travail régulier témoigne d'une bonne organisation et une bonne coordination et sera récompensé. Vous aurez une meilleure note si vous travaillez sur le projet toutes les semaines (par rapport à tout faire la semaine qui précède la date de rendu) ;
- l'implication équitable de tous les membres du groupe : réussir à impliquer chacun quel que soit son niveau est une compétence très importante, et sera récompensée.
- mettez dans votre rendu tous les documents produits (rapport, maquettes, schémas), pas seulement le code ;
- si une contribution n'est pas visible, la faire figurer dans le rapport.

Toute contribution sera prise en compte. Cependant, un étudiant ne peut pas se consacrer uniquement à des tâches qui ne nécessitent pas de programmer. Chacun doit contribuer en écrivant du code *original*. Il n'est pas demandé que tous les membres d'un groupe écrivent la même quantité de code, mais il ne doit pas y avoir de grands déséquilibres.

Si vous avez contribué avec quelque chose qui n'est pas visible (par ex. recherche de modèles 3D, capture de la vidéo de présentation, préparation de l'archive à rendre, etc.), assurez vous que c'est mentionné dans le rapport.

En cas de **conflits dans le groupe, en parler** à l'enseignant de TP au plus vite ; il/elle pourra jouer le rôle de médiateur pour vous aider à résoudre le problème.

Fonctionnalités requises et optionnelles Un marteau en bois ne permet pas d'enfoncer un clou, même s'il a été sculpté et gravé par le grand maître Paul Gauguin. Pour enfoncer un clou, il suffit d'un marteau ordinaire pourvu que sa tête soit en fer.

Ne faites pas un logiciel qui ressemble à un marteau à tête en bois.

Autrement dit, il faut concentrer vos efforts et donner une grande priorité aux fonctionnalités exigées par le sujet. Seulement après vous pourrez vous faire plaisir en implémentant des fonctionnalités supplémentaires, ou embellir votre logiciel.