Cahier des charges

-

ChessRPG

Pierre-Louis GONDRAS
Robin GREYL
Tiffanie SCHREYECK

1. Description du projet

L’objectif du projet est de créer un jeu à mi-chemin entre les échecs et un RPG. Le plateau et les déplacements de pièces sont inspirés d’un jeu d’échecs mais les combats entre les pièces sont inspirés des RPG. Une IA sera intégrée quand les parties joueur contre joueur seront fonctionnelles.

Chacun son tour, les joueurs vont déplacer une pièce sur l’échiquier selon les déplacements classiques aux échecs. Chaque pièce possède une puissance d’attaque et un nombre de points de vie selon son type. Les points de vie des pièces sont persistants et la pièce bénéficie d’un bonus d’attaque lorsqu’elle est attaquante. Le but du jeu est de tuer le roi adverse.

Pour déclencher un combat, il faut qu’une pièce se déplace sur une case déjà occupée par une pièce adverse. Le combat se termine quand une des pièces en combat meurt. La pièce attaquante inflige des dommages en premier et bénéficie d’un bonus d’attaque.

1. Contraintes

* Durée de développement : environ 2 mois
* Le jeu sera développé en C, compatible Linux et Windows
* Utilisation de la librairie SDL et ncurses pour la version alpha
* Utilisation de Git pour gérer et archiver le code
* Production d’une documentation par Doxygen
* Le code sera présenté proprement
* Un diagramme des modules sera fourni
* Des outils de débogage (valgrind, etc.) seront utilisés

1. Déroulement du projet

Tâche 0 : rédiger le cahier des charges

Membres impliqués : tous

Durée : 1 semaine

Tâche 1 : définir le diagramme des modules

Membres impliqués : tous

Durée : 1 semaine

Tâche 2 : développement d’un prototype avec ncurses

Durée : 3 semaines

Tâche 2.1 : écriture du module « piece.h/c »

Membres impliqués : Robin

Ce module gère les différentes pièces (roi, fou, etc.)

ainsi que leur couleur et leurs points d’attaque et de vie. De base, les couleurs ne sont que Noir ou Blanc, mais on pourra ajouter des apparences de pièces différentes en fin de projet. Les points de vie et d’attaque varie selon le type de la pièce.

Tâche 2.2 : écriture du module « case.h/c »

Membres impliqués : Pierre-Louis

Ce module génère une case avec une couleur de fond : blanc ou noir selon la position sur l’échiquier, et bleu si la pièce sélectionnée peut se déplacer sur cette case. Une case contient aussi un pointeur sur une pièce, qui sera mis à NULL si aucune pièce n’occupe cette case.

Tâche 2.3 : écriture du module « plateau.h/c »

Membres impliqués : Robin

Ce module crée un plateau qui est un tableau à double indice de 0 à 7. Chaque adresse du tableau contient un pointeur sur une case. La position de la case est déterminée par ses indices dans le tableau. On stocke également le nombre de tours écoulés.

Tâche 2.4 : écriture du module « joueur.h/c »

Membres impliqués : Tiffanie

Ce module stocke le nom des joueurs et la couleur qu’ils ont choisi au début de la partie. On utilise la couleur choisie par le joueur pour savoir quelles pièces il peut sélectionner et sur quelles cases il peut se déplacer.

Tâche 2.5 : écriture du module « couleur.h/c »

Ce module contient la liste des couleurs possibles pour chaque pièce et chaque joueur. De base, uniquement Blanc ou Noir mais on pourra ajouter des couleurs supplémentaires par la suite.

Membres impliqués : Robin

Tâche 2.6 : écriture du module « jeu.h/c »

Ce module est le module principal du projet. Il pointe vers les structures Joueur et Plateau. C’est à partir de ce module qu’on choisira le joueur actif, et c’est en grande partie à ce module que les fonctions graphiques et d’IA s’adresseront.

Membres impliqués : Pierre-Louis / Tiffanie

Tâche 2.7 : écriture du module « JeuNCURSES.h/c »

Ce module gère l’affichage du jeu en mode textuel.

Membres impliqués : Pierre-Louis / Tiffanie

Tâche 2.8 : mise en commun des modules et test du jeu

La mise en commun de tous les modules permettra de tester le jeu, et de corriger les erreurs si il y en a. On pourra ensuite passer à la réalisation du jeu en SDL.

Membres impliqués : tous

Tâche 3 :  développement du jeu en SDL

Durée : 3 semaines

Tâche 3.1 : Équilibrage du jeu

Jusqu’à présent, on avait surtout développé le moteur du jeu. Si on veut que le jeu soit jouable, il faudra ajuster de manière cohérente les points de vie et de dégâts des pièces.

Membres impliqués : Robin

Tâche 3.2 : Design des pièces et du plateau

Même si le jeu final sera sûrement en vue de dessus, les pièces seront vu de côté pour qu’on puisse les reconnaître plus facilement. L’objectif de cette tâche est de dessiner des pièces facilement reconnaissables.

Membres impliqués : Pierre-Louis

Tâche 3.3 : Ecriture du module « JeuSDL.h/c »

Ce module permettra l’affichage du jeu en mode graphique.

Membres impliqués : Tiffanie

Tâche 3.4 : test et debug

On teste beaucoup de configurations possible, en vérifiant si il y a des fuites de mémoire ou des bugs dans le programme.

Membres impliqués : tous

Tâche 4 : Ajout de fonctionnalités

Durée : 3 semaines

Tâche 4.1 : Augmentation du côté RPG

Création de sorts particuliers pour chaque pièce, ajustement sur les déplacements ou même création de nouvelles pièces

Membres impliqués : Robin

Tâche 4.2 : Ajout de musique

La musique pourra s’adapter selon les couleurs de pièce des joueurs, la situation dans laquelle est un joueur, etc

Membres impliqués : Pierre-Louis

Tâche 4.3 : Création d’une IA basique

L’IA ne devrait pas être très développé, juste suffisamment pour permettre de jouer seul sans que l’adversaire joue de façon complètement aléatoire.

Membres impliqués : Tiffanie

V. Diagramme de GANTT

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tache / semaine | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | X |  |  |  |  |  |  |  |  |  |
| 1 | X | X | X | X |  |  |  |  |  |  |
| 2.1 |  | X | X | X |  |  |  |  |  |  |
| 2.2 |  | X | X | X |  |  |  |  |  |  |
| 2.3 |  | X | X | X |  |  |  |  |  |  |
| 2.4 |  | X | X | X |  |  |  |  |  |  |
| 2.5 |  | X | X | X |  |  |  |  |  |  |
| 2.6 |  | X | X | X |  |  |  |  |  |  |
| 2.7 |  | X | X | X |  |  |  |  |  |  |
| 2.8 |  | X | X | X |  |  |  |  |  |  |
| 3.1 |  |  |  | X | X | X | X |  |  |  |
| 3.2 |  |  |  |  | X | X | X |  |  |  |
| 3.3 |  |  |  |  | X | X | X |  |  |  |
| 3.4 |  |  |  |  | X | X | X |  |  |  |
| 4.1 |  |  |  |  |  |  | X | X | X | X |
| 4.2 |  |  |  |  |  |  |  | X | X | X |
| 4.3 |  |  |  |  |  | X | X | X | X | X |

VI. Diagramme des modules