UNIVERSITÉ DE MONTPELLIER

L2 Informatique

HLIN302 – Rapport de projet noté « Casse-Briques »

Programmation impérative avancée

Alban MANCHERON et Pascal GIORGI

Etudiants :

Laila CHAMROUK

Ayoub LAAFOU

Kaoutar MAJDOUL

Soukaina MESSAOUDI

Année : 2018-2019

Sommaire

Introduction

Dans le cadre du module programmation impérative avancée du semestre 3 de notre deuxième année d'informatique à la faculté de Montpellier nous avons eu pour projet le développement d'un casse brique en C++ en groupe. Avec la participation de Laila CHAMROUK, Ayoub LAAFOU, Kaoutar MAJDOUL et Soukaina MESSAOUDI nous allons dans un premier temps énumérer les motivations du projet puis le cahier des charges. Suite à cela nous allons présenter notre organisation de travail ainsi que sa conception ainsi que les choix faits durant le projet afin d'exposer nos réalisations. Enfin, nous terminerons sur une conclusion objective de notre travail détaillant les difficultés rencontrées ainsi que les améliorations possibles du jeu proposé.

Motivation du projet

Le but de ce projet est de développer un casse brique. Le casse brique est un jeu vidéo consistant à casser des briques se trouvant en haut de l'écran de jeu. Le joueur contrôle une raquette qui lui permet à la fois d'expédier les balles vers les briques et de les réceptionner en la déplaçant latéralement. Lorsque la partie commence la balle prend une direction afin de rencontrer les briques sur lesquelles elle rebondit. Lors du rebond sur celles-ci le score du joueur doit augmenter. De plus quand la balle rencontre un obstacle tel que le bord gauche, droit ou supérieur du terrain elle rebondit et change de trajectoire. Lorsque la balle rebondit sur une brique, la brique est abîmée et finit par être éliminée après plusieurs contact avec celle-ci. Si la balle arrive au bord inférieur du terrain, alors celle-ci sort du terrain et est perdue. Le but du jeu est donc de détruire toutes les briques, et cela avec un nombre de balles limité. Quand le joueur n'a plus de balle, la partie est perdue.

Cahier des charges

Tout d'abord, le terminal doit ouvrir une fenêtre permettant le début du jeu. Au sein de cette fenêtre nous devons retrouver des messages fonctionnels affichant le score, le nombre de vies restantes ainsi que les interactions avec le clavier nécessaires afin de quitter la partie. De plus les briques peuvent avoir des formes variées et leurs destructions peuvent être causées par plusieurs tirs. Enfin, le placement du ou des terrains doit être défini dans un fichier de configuration.

En somme l'utilisateur doit pouvoir :

— Lancer la balle à chaque début de partie en appuyant sur la touche espace de son clavier.

— Déplacer sa raquette sur le terrain en utilisant les flèches de son clavier : ← et → .

— Quitter la partie en cours en appuyant sur la touche Q de son clavier.

— Consulter son score et son nombre de vies restantes.

Une fois lancée, l'orientation de la balle doit dépendre de l'obstacle qu'elle a rencontré. Lorsqu'une brique a été touchée par la balle celle-ci change de couleur afin de confirmer à l'utilisateur qu'elle a bien été touchée par sa balle et ainsi sa résistance diminue. Lorsque le nombre de balle est épuisé la partie s'arrête et le score du joueur est affiché.

D'un point de vue ergonomique, l'exécution du programme doit être simplifié à l'aide d'un *makefile* s'assurant de la compilation de tous les fichiers nécessaires.

Organisation du travail

Durant ce projet nous avons mis en place un dossier GitHub regroupant tous les fichiers nécessaire au développement du jeu. Cet outil collaboratif permet l'hébergement du code et facilite la gestion de version des fichiers. La deadline étant le 6 janvier 2019, nous avons eu recours durant le temps imparti, à plusieurs réunions afin de se répartir le travail, mettre en avant les tâches réalisées et celles restantes à faire.

Avant toute chose, nous nous sommes renseignés sur les différentes bibliothèques proposées en particulier *ncurses.*

Conception

Lors de la conception de ce projet nous avons décidé de créer, pour chaque classe, un fichier *.cpp* et un *.h* afin d'optimiser notre code. En déclarant tes fonctions dans un *header*, nous avons pu les utiliser dans n'importe quels *.cpp* simplement en l'incluant à celui-ci et réduire le temps de compilation des fichiers. Le *header* nous a également permis de commenter le rôle des fonctions.

Dans un premier temps nous avons défini les dimensions du terrain de jeu. Afin de garder la même taille qu'un terminal à son ouverture nous avons choisi ses proportions pour le casse brique. Suite à cela nous avons décidé de placer la raquette en bas au centre du terrain et la balle au centre de celle-ci. La taille de la raquette varie selon le niveau choisi, en effet plus le niveau est élevé plus la taille de celle-ci est basse afin d'augmenter la difficulté de la partie.

Nous avons également défini la taille des briques et leur résistance en début de fichiers car la résistance des briques diminue chaque fois qu'une balle rebondit sur l'une d'entre elles. De plus, lorsque la balle touche la partie inférieur du terrain, raquette exclue, le nombre de vie du joueur diminue et quand celui-ci est égal à zéro la partie se termine et le score final du joueur est affiché.

Chaque fois que la balle rebondit sur une balle celle-ci passe du rouge au vert et au bout du deuxième rebond elle disparaît et fait gagner des points au joueur. Celui-ci peut consulter, pendant la partie, son score et le nombre de vie qu'il lui reste. Enfin, il peut quitter la partie à tout moment en appuyant sur la touche '*q*' de son clavier.

Avant de commencer une partie il peut choisir un pseudo auquel sera associé son score lors de la fin de sa partie. Après avoir choisi un pseudo, il accède au menu du jeu qui lui propose plusieurs choix comme celui de commencer à jouer et de choisir un niveau par la suite ou l'accès à des informations telles que l'aide, le nom des auteurs, la version du jeu ou encore les 5 premiers scores.

Réalisation

1. Algorithme concernant la balle

Afin de gérer les mouvements de la balle nous avons mis en place les deux fonctions *moveBallx* et *moveBally* qui prennent en paramètre respectivement la coordonnée *x* et *y* de la balle et un booléen *minus* qui gère les rebonds de celle-ci. Comme nous pouvons le voir sur la figure ci-dessous lorsque les coordonnées de la balles atteignent les bords le booléen *minus* passe à *true* ce qui permet le rebond.

```
void moveBallx(int x, bool &minus) {
    if (x == 0) {
          minus = false;    }

    else if (x == SCREEN_WIDTH) {
          minus = true;
    }
}
void moveBally(int y, bool &minus) {
    if (y == 0) {
          minus = false;    }
    else if (y == SCREEN_HEIGHT-1) {
          minus = true;
    }
}
```

Figure 1 : Fonctions du mouvement de la balle

Lorsque la balle touche la brique A, par exemple, par le bas ou la droite alors les booléens *xMinus* et *yMinus* prennent la valeur *false* ce qui permet de faire rebondir la balle vers le bas ou de faire dévier son axe lorsqu'elle tape la brique par la droite. Le même principe est utilisé lorsque la balle touche le haut de la brique ou son côté gauche cependant les booléens prennent la valeur *true* afin de permettre le rebond correcte pour ces cas là comme on peut le voir sur la figure 2.

```
if((bally==A.y+1 && ballx>=A.x && ballx<=A.x+BLOCK_LENGTH) || (bally==A.y && ballx==A.x+BLOCK_LENGTH+1)) { //la balle touche la brique
en bas ou à droite
                    xMinus=false;
                    yMinus=false;
                   moveBallx(ballx, xMinus);
                    moveBally(bally, yMinus);
                    score++;
                    touchA++;
          }
          else if((bally==A.y-1 && ballx>=A.x && ballx<=A.x+BLOCK_LENGTH) || (bally==A.y && ballx==A.x-1)){ //la balle touche la
brique en haut ou à gauche
                    xMinus=true;
                    yMinus=true;
                   moveBallx(ballx, xMinus);
                    moveBally(bally, yMinus);
                    score++;
                    touchA++;
```

Figure 2 : Balle qui touche une brique

Enfin les cas de rebond ne résultant pas de contact avec les briques sont montrés dans l'annexe 1.

2. Organisations des fichiers

A présent nous allons décrire l'ensemble des fichiers et leur contenu sous forme de liste afin d'obtenir l'architecture du programme et d'obtenir des informations sur leur contenance.

- ***JeuCB.cpp*** est le fichier contenant le *main* du jeu. Il permet à l'utilisateur de saisir un pseudo puis de choisir entre différentes options telles que commencer le jeu, obtenir de l'aide ou des informations sur la version, les auteurs ou les scores. De plus, après que le joueur est choisi son niveau, il appel les fonctions gérant l'affichage et les interactions selon les niveaux. Enfin il permet l'enregistrement des scores dans le fichier ***score.txt*** à la fin de chaque partie.

- ***initBall.cpp/h*** contient les fonctions *moveBallx* et *moveBally* qui gérent le déplacement de la balle ainsi que la fonction *init_curses* et *finish_curses* pour la gestion du terminal durant la partie et à la fin de celle-ci.

- ***niveau(1/2/3).cpp/h*** représentent les fichiers de chaque niveau ainsi que leur configuration. Chacun des niveaux possède des paramètres qui influent sur la difficulté du jeu tels que la taille de la raquette ou le nombre de vies.

- ***window.cpp/h*** énumère et initialise les différentes couleurs utilisées durant le jeu.

- ***help.txt / auteurs.txt / version.txt / score.txt*** contiennent respectivement l'aide, le noms des auteurs, la version du jeu ainsi que les scores de chaque joueur.

Bilan et difficultés rencontrées

Ce projet, mené dans le but de concevoir un jeu, a été un exercice enrichissant pour nos compétences dans ce module de programmation impérative. A travers cet entraînement, nous pouvons tirer des connaissances grâce à certaines fonctionnalités accomplis ou non car chacune d'entre elle implique une recherche.

Parmi les tâches accomplis, nous pouvons constater que les trois niveaux, avec une difficulté qui s'élève de niveau en niveau, sont fonctionnels. De plus, l'affichage du score et le nombre de vie restantes sur le côté de la fenêtre sont tous deux opérationnels. A cela s'ajoute la possibilité de quitter la partie à tout moment et de sauvegarder son score lié à son pseudo quand on quitte une partie. Ce dernier est enregistré dans un fichier .txt. Par ailleurs, les briques sont elles aussi fonctionnelles. De plus elles ont une résistance qui diminue à chaque coup de balle. Cette balle qui rebondit sur la raquette a un mouvement cohérent selon l'impact. Enfin, pour conclure sur nos réalisations, nous pouvons également noter que l'affichage de l'aide, des auteurs, de la version, ou alors des premiers scores, sont également des fonctionnalités opérationnelles de ce projet.

Ce projet reste tout de même incomplet. En effet, certaines tâches n'ont pas pu être achevées. Parmi celles-ci, nous avons le passage d'un fichier de configuration qui permettrait la personnalisation du jeu. En ce qui concerne le score, l'affichage d'un top 5 n'a pas pu être aboutit car nous n'avons pas réussi à récupérer seulement la partie contenant le score du fichier afin de trier les lignes associées. Enfin, un dernier point n'a pas pu être finalisé, c'est celui de la sauvegarde en cours de partie avec la possibilité de reprendre la partie à tout moment. Cependant une tentative incomplète a été mise en place seulement celle-ci ne permettait que la sauvegarde mais ne rendait pas la restauration de la partie effective car les briques restaurées disparaissaient au passage de la balle sans provoquer de rebond.

Comme dans tout projet, nous avons fait face à quelques difficultés. La première étant l'utilisation des nombreuses fonctions de la bibliothèque *ncurses*, il était nécessaire de maîtriser ces fonctions avant de pouvoir s'en servir. Ensuite, la gestion du temps mêlant l'apprentissage du langage, l'emploi du temps des cours et le rapport, ont également été compliqués pour le groupe. Cependant grâce l'outil Github nous avons réussi à rester synchronisé dans l'avancé globale du groupe.

Enfin, ce projet nous a fait profiter de nouvelles technologies mais également de l'amélioration dans la pratique de la programmation, notamment avec l'utilisation de fonctionnalités inédites pour nous. De plus, ce projet nous a permis de développer nos capacités de communication dans le groupe, surtout à l'aide des réseaux sociaux, tel que Whatsapp. Enfin, la mise en situation a été comme réelle car le projet disposait d'un cahier des charges précis et une date de rendu fixe ce qui nous a permis d'obtenir un aperçu de la conduite d'un projet en groupe.

ANNEXE 1 : Différents cas de rebond de la balle

```
          if (ballx == 0 && bally == 0) { //la balle touche le coin supérieur gauche
          moveBally(bally, yMinus);
          moveBallx(ballx, xMinus);
          }
          else
          if (bally == SCREEN_HEIGHT-1 && ballx == 0) { //la balle touche le coin inférieur gauche
          moveBallx(ballx, xMinus);
          moveBally(bally, yMinus);
          }
           else
          if (ballx == SCREEN_WIDTH || ballx == 0) { //la balle touche le côté gauche ou droit
                    moveBallx(ballx, xMinus);
           }
          else if (bally == SCREEN_HEIGHT-1) { //la balle touche le bas
          if (ballx >= paddlex && ballx <= (paddlex+PADDLE_LENGTH)) { //la balle touche le paddle
                   moveBallx(ballx, xMinus);
                   moveBally(bally, yMinus);

          }
          else{ //la balle ne touche pas le paddle
                   moveBallx(ballx, xMinus);
                   moveBally(bally, yMinus);


                   vie--; //le nombre de balle(vie) diminue
          }
          }
           else if (bally == 0) { //la balle touche le haut
                    moveBally(bally, yMinus);
           }
          else if(ballx == SCREEN_WIDTH && bally == 0) { //la balle touche le coin supérieur droit
          moveBallx(ballx, xMinus);
          moveBally(bally, yMinus);
          }
          else if(ballx == SCREEN_WIDTH && bally == SCREEN_HEIGHT-1) { //la balle touche le coin inférieur droit
          moveBallx(ballx, xMinus);
          moveBally(bally, yMinus);
          }
          else if((bally==A.y+1 && ballx>=A.x && ballx<=A.x+BLOCK_LENGTH) || (bally==A.y &&
ballx==A.x+BLOCK_LENGTH+1)) { //la balle touche la brique en bas ou à droite
                    xMinus=false;
                    yMinus=false;
          moveBallx(ballx, xMinus);
                    moveBally(bally, yMinus);
                    score++;
                    touchA++;
           }
```