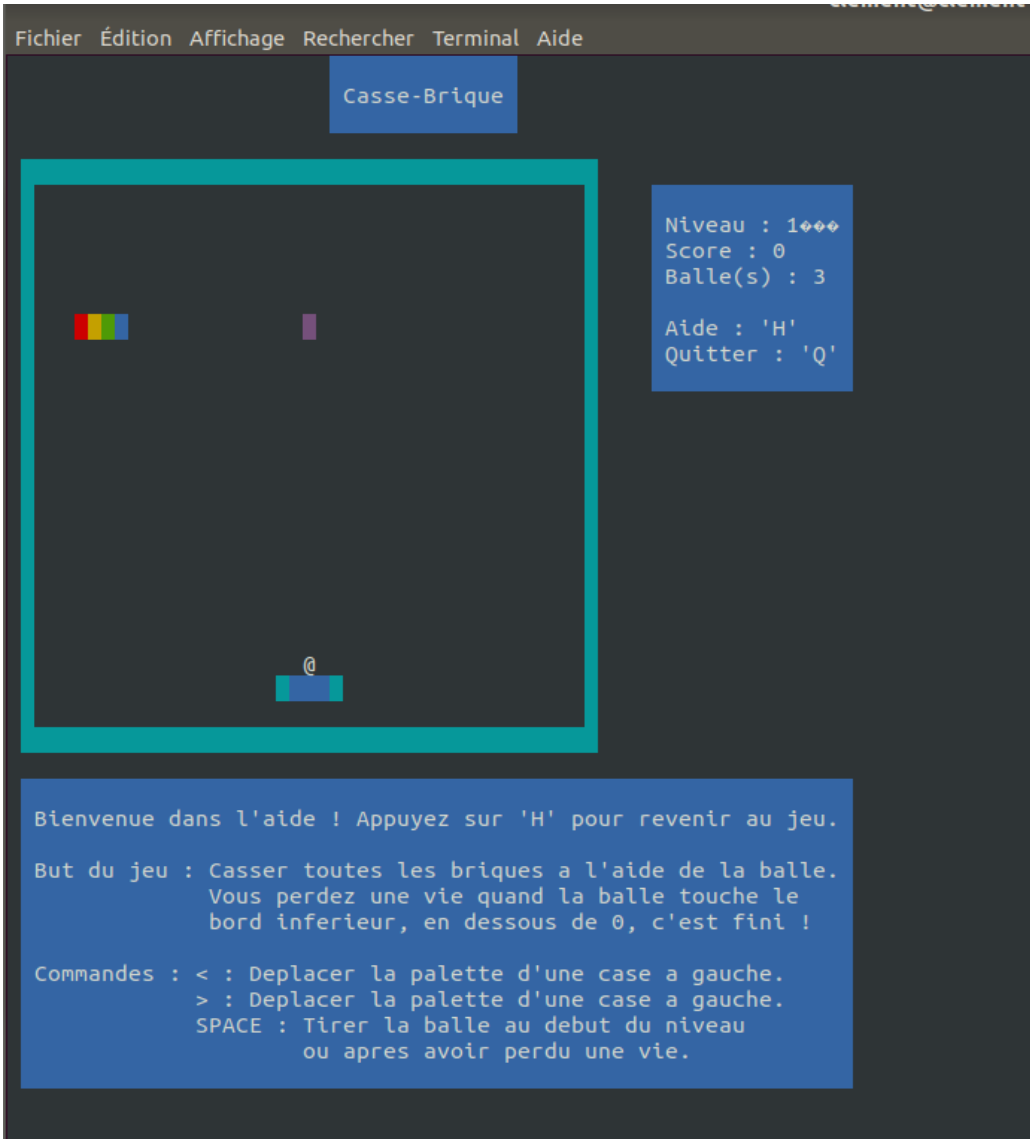
Cassan Mélodie, Wyszynski Anthony
Galinier Lisa, Ahamed Eliza

Rapport de projet HLIN302 : Programmation Impérative avancée

Casse-Brique



Introduction

Dans le cadre de l'unité d'enseignement HLIN302 "programmation impérative avancée", il nous a été demandé de coder un Casse-Brique dans le langage C++. Nous avons donc utilisé les connaissances abordées tout au long de ce semestre pour réaliser notre programme, dans le but de nous exercer à mettre en application ce que nous avions appris, mais aussi pour nous familiariser avec le travail en groupe.

Vous trouverez dans ce rapport une première partie comprenant notre analyse afin de comprendre ce qui était attendu de nous et les choix que nous avons fait pour débuter ce projet. La deuxième partie portera sur les tests que nous avons effectués, ainsi que nos remarques et commentaires. La troisième partie concernera l'algorithme qui calcule la direction et la vélocité de la balle ainsi que ce que nous avons pu faire au sujet du fichier de configuration. Enfin, la dernière et quatrième partie fera un bilan des étapes que nous avons réalisées et leur degré d'achèvement.

I - Analyse du projet et choix effectués

Pour comprendre comment mener à bien notre projet, nous avons bien évidemment commencer par revoir les règles générales du jeu Casse Brique. Nous avons ensuite tous ensemble découvert le fonctionnement la classe Window qui nous était fournie, et discuté de l'exploitation que nous pouvions en faire. Finalement, une personne a commencé à se concentrer sur la classe de la balle, deux autre sur la classe brique, et une dernière personne sur la classe joueur. Au fur et à mesure de l'avancement de notre projet, nous avons réorganisé la répartition des tâches au mieux pour arriver jusqu'au résultat final. Pour ce faire, nous avons dû faire certains choix :

Premièrement, nous avons décidé d'utiliser la plateforme github pour mettre notre code en commun et pouvoir le partager rapidement et efficacement. Nous avons donc pu travailler à distance durant une partie du projet grâce à ce moyen de partager notre code.

Notre deuxième choix important porte sur l'utilisation d'un Makefile. En effet, après nous être rendus compte que nous ajoutions de plus en plus de classes au projet, la compilation devenait de plus en plus fastidieuse. Nous avons donc réalisé un makefile qui nous permet, une fois situé dans le bon dossier, de n'avoir que la commande make à écrire dans le terminal pour compiler l'ensemble du programme.

En ce qui concerne le jeu lui-même, pour aller dans l'ordre de son déroulement, nous avons décidés qu'un menu simple dans lequel nous pourrions naviguer à l'aide de la flèche du haut, de la flèche du bas, et de la touche entrée était de rigueur. Afin de le rendre plus attractif, le titre s'affiche et part lentement, est capable de clignoter, et chaque mouvement de curseur affecte le texte à sa droite.

Pour la « vraie partie jeu », nous avons décidé d'y faire afficher un titre, le plateau de jeu, une fenêtre d'informations, si demandé avec la touche "h" une fenêtre d'aide et enfin si nécessaire une fenêtre « level up », une fenêtre « game over » et une fenetre « classement » aussi accessible dans le menu cité précedemment.

NB: la touche q ne fonctionne pas si l'aide est affichée, pour quitter la partie il ne faut pas qu'elle soit en pause !

Pour le plateau lui-même, la balle reste collée à la raquette tant que la touche espace n'a pas été appuyée, tous les mouvements de la raquette sont appliqués à la balle à ce moment là. La raquette en question à 4 mouvements différents : 5 cases à gauche : "a", 1 case à gauche : la flèche de gauche, 1 case à droite : la flèche de droite, 5 cases à droite : "e".

Pour passer rapidement sur la balle dont nous parlerons plus loin, celle-ci possède 6 directions, toutes les directions cardinales sauf sur les côtés, la balle doit toujours gagner ou perdre en hauteur. Aussi celle-ci ne peut aller à une autre vitesse, nous verrons cela plus tard.

Pour ce qui est des briques, elles sont définies par une position ainsi qu'un nombre de points de vie, allant de 1 à 5, étant chacun représenté en jeu par une couleur différente. Si la balle rentre en contact avec l'une d'elle, elle perd de la vie et change de couleur, jusqu'à être totalement effacée du plateau. Ces dernières sont données au plateau grâce à des niveaux, auquels on peut ajouter des briques ou des coordonnées et un nombre de PV, pourvu que les coordonnées de soit pas déjà utilisées, ni qu'elles soient trop proche de la raquette ou hors du terrain. Ces 3 cas sont les uniques cas provoquant une erreur arrêtant totalement le programme.

Enfin si on est suffisamment doué, ou chanceux, ou que l'on a supprimé les meilleurs scores … pas très légal … , en fin de partie le jeu nous affiche le classement avec la place qui nous à été attribuée, en attendant que l'on entre notre nom, puis que nous appuyons sur la touche "q" après que le jeu nous félicite et affiche notre nom. Il va sans dire que tout autre caractère qu'une lettre est ignoré.

II - Tests, problèmes rencontrés, remarques

A) Progression générale et tests effectués

Nous avons réalisé les étapes du projet dans l'ordre qui suit, avec les tests associés à chacune d'elle :

- tests sur la classe window, création de fenêtres pour afficher l'aide
- création de la classe brique et fin de prise en main de window en affichant une seule brique sur le plateau
- création du message d'aide, tests d'affichage de la raquette
- tests sur la balle, des chocs avec les bords, d'abord sans prendre la raquette en compte puis en la prenant en compte.
- tests des messages de fin de partie, de réussite d'un niveau en commençant par y associer des lettres du clavier
- création d'une classe niveau pour stocker les informations d'un niveau
- gestion de l'inclinaison et de la direction de la balle avec les bords de la raquette
- utilisation de la classe niveau pour afficher plusieurs briques sur le terrain
- tests des chocs de la balle sur les briques, gestion du score et de la résistance des briques
- création d'un menu, avec les options de choix de niveau, de clignotement du titre, de vision des meilleurs scores
- gestion des meilleurs scores en fin de partie, selon si un score doit être ajouté au tableau ou non

B) Problèmes rencontrés

La lecture de fichier de configuration nous a fait nous poser beaucoup de question sur la façon d'afficher un niveau et sur les informations indispensables à connaître pour en initialiser un. Nous en sommes donc arrivé à la conclusion qu'il fallait autant de briques que possible sur plusieurs lignes, avec un certain espacement entre les briques, et un nombre de vies de toutes les briques du niveau identique. C'est la dessus que nous nous sommes donc basés pour créer la classe niveau et le fichier de configuration optimal. Nous avons de plus fait le choix de faire un fichier de configuration par niveau.

La réalisation des algorithmes permettant de lire le fichier de configuration fut assez difficile, et a nécessité beaucoup de tests simples pour être comprise. Au final, nous n'avons malheureusement pas pu mettre plusieurs niveaux dans le même fichier texte.

Nous avons aussi décidés par rapport au fait que la classe Config était encore assez obsolète de ne pas la mettre dans le programme principal, étant donné que le but étant de fournir un fichier pour créer un groupe de niveaux et non pas un seul.

L'organisation du groupe a elle aussi été un challenge, car malgré la bonne volonté générale, le manque d'implication de certains aura sûrement pénalisé l'ensemble du groupe de telle sorte à ce que des classes telles que Config n'aient pu être finies, et que des idées de suite pour le jeu n'aient même pas pu voir le jour, par manque de temps.

III - Explications des algorithmes codant la balle et le fichier de configuration

A) La balle

La balle à été un des plus gros défis de ce projet. Au tout début nous n'avions AUCUNE idée de comment faire un algorithme qui prendrait à la fois en compte les mouvements d'une balle, ses interractions avec le terrain, les briques et la palette plus le mouvement de cette dernière. Au final après s'y être mis entièrement, ce n'était pas si compliqué

La seule chose sur laquelle nous avons le contrôle est la raquette par le getch(), la balle bouge toute seule par des méthodes reliant sa position et celles de tous les objets, et surtout sa fonction « update », changeant ses coordonnées selon sa direction. Pour essayer de faire court, il nous faut premièrement savoir qui sont les objets, d'où les méthodes « bord », « palette », « brique » et leur variantes comment « bordBas », « bordPaletteG » ...

Ensuite il nous fallait un premier cas, ce fut un coin et une direction vers ce coin, donc en diagonale, les cas droits étant les plus simples. Selon la présence de ces 3 objets, la balle allait rebondir contre un coin, un mur, un plafond/sol, ou rien. Facile vous me direz, mais il peut y avoir 4 coins, d'où les paramètres "i" et "j". Ils définissent à la fois un angle, mais aussi les possibilités d'objet : (X+i,Y), (X+i,Y+j) et (X,Y+j).

Maintenant que nous avions ces paramètres, il fallut faire un peu du cas par cas dans la méthode « direction » , afin de définir la direction selon que côté la balle touchait. Ensuite viens la fonction « update » regroupant tout : les directions dans les angles vues juste avant, les simples directions en haut ou en bas, et les cas spéciaux contre les bords de la palette (voire ces derniers et les bords de terrain).

Update contient aussi les chocs avec les briques et le bas du terrain selon quel endroit la balle touche, mais pour ce qui est de la balle, tout est expliqué.

B) Le fichier de configuration

En ce qui concerne le fichier de configuration, il a été choisi d'y indiquer les informations relatives à un niveau, soit la résistance de toutes les briques de ce niveau, le nombre de lignes désirées, ainsi que l'espacement entre chaque brique. Nous avons choisi de mettre autant de briques que possible sur chaque ligne. Le fichier de configuration se présente ainsi:

- les commentaires sont précédés du caractère "#"
- les informations qui seront récupérées et utilisées pour configurer le niveau se présentent sous la forme : "clé : valeur" ou "clé:valeur"

Pour ce qui est du traitement, nous avons créé plusieurs méthodes dans la classe config :

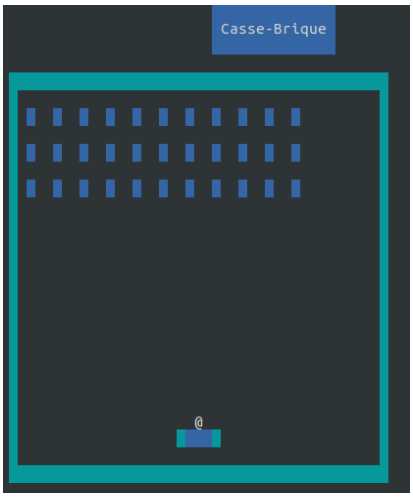
une méthode qui "nettoie" les chaînes de caractère inutiles dans le fichier de configuration, une méthode qui permet de reconnaître une clé et une valeur grâce à leur séparateur ":" , une méthode qui attribue les valeurs récupérées aux attributs correspondant à la bonne clé, et enfin une méthode qui prend un fichier de configuration en entrée et qui combine les méthodes précédentes. Vous trouverez ci dessous un exemple de fichier de configuration, ainsi que les terrains de jeu obtenus suite à l'exploitation des fichiers de configuration du niveau 1 et du niveau 2.

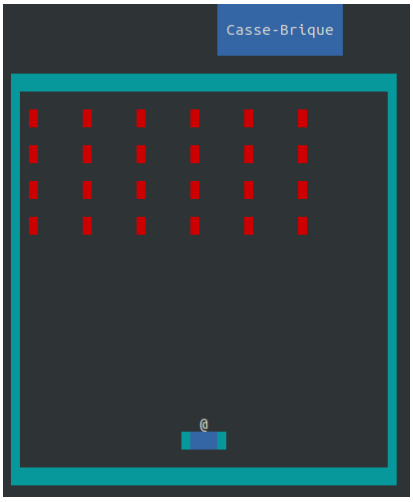
```
# FICHIER DE CONFIGURATION DU NIVEAU 1 DU CASSE-BRIQUE

#nombres de lignes désirées dans le terrain, nombre entier
nbLignes : 4 #si on met un dièse sur la méme ligne il sera quand même enlevé

#L'espace entre les briques, nombre entier
espaceBriques :7

#les points de vie de toutes les briques du niveau, nombre entier
pvBriques:1
```





IV - Bilan

Finalement, nous pouvons résumer l'ensemble des étapes qui ont été réalisées :

Nous avons créé un jeu de Casse-Brique fonctionnel, avec toutes les classes demandées aucune variable globale. De plus, le jeu permet le passage de paramètres au lancement du programme pour connaître l'aide, la version du jeu et le nom des auteurs. Le positionnement des briques est réalisé de telle manière qu'elles ne peuvent ni se chevaucher ni dépasser la bordure. Les message de choix d'un niveau, d'affichage des scores, d'aide du jeu, de fin de partie, d'erreurs dans les fichiers de configuration ou problème de placement des briques sont fonctionnels. Le fichier de configuration est clair et permet de représenter un niveau dans son ensemble, même s'il ne peut pas changer le type de brique, de balle ou de raquette. On peut sauvegarder les 5 meilleurs scores, entrer le nom des joueurs et les afficher. Enfin, un menu permet de choisir diverses options telles que le choix des niveaux, l'affichage des 5 meilleurs joueurs ou alors de tout simplement quitter le jeu.

Ce projet nous a permis de réaliser un jeu à l'aide de nos connaissances, et fut extrêmement formateur pour nous. Il nous a permis de mieux comprendre les notions vues en cours et en TD, et de trouver une façon concrète de les appliquer.