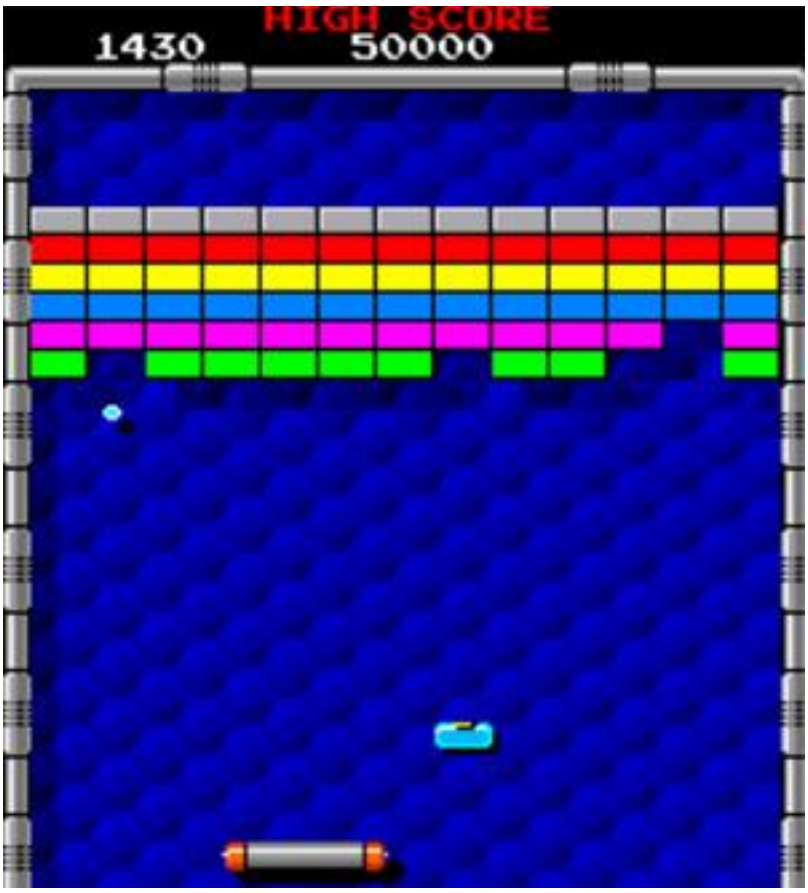
CASSE-BRIQUES

Rapport de mon projet



01 DECEMBRE 2020

BROQUET RAYAN
DIVTEC - EMT

Table des matières

Biographie

Je m'appelle Rayan Broquet. Je suis né à Delémont. J'habite un petit village du jura qui se nomme Movelier. J'ai 2 grands frères qui n'habitent plus avec nous (mes parents et moi), j'ai passé ma scolarité de 4 à 12 ans à Movelier et Soyhières. Je suis ensuite allé au collège de Delémont, et actuellement je suis à l'EMT à Porrentruy en tant qu'apprenti informaticien 3ème année.

But du projet

Le but du projet est de réaliser un petit jeu informatique à l'aide d'un langage de développement comme le C++ ou le Java, ce qui nous permet de gérer nous même un projet et de le développer entièrement. Cet atelier nous aide aussi à percevoir le « background » des jeux vidéo que nous connaissons actuellement et de leur complexité, à prendre connaissance des différents bugs qui peuvent intervenir et de trouver une solution pour les gérer.

Glossaire

C++	Selon Wikipedia, Le C++ est un langage de programmation compilé permettant la programmation sous de multiples paradigmes.
Sprite	Selon Wikipedia, un sprite est dans le domaine du jeu vidéo un élément graphique qui peut se déplacer sur l'écran. En principe, un sprite est en partie transparent et il peut être animé par plusieurs images s'affichant les unes après les autres.
Tick	Signal qui est envoyé à intervalle régulier (généralement plusieurs fois par secondes) et appelant un slot connecté, Permet de définir les IPS (images par secondes).
Les signaux et les slots sous Qt Creator	Selon Wikipedia, les signaux et les slots permettent de connecter des objets entre eux via des signaux qui sont émis et reçus par des slots. Pour un développeur, les signaux sont représentés comme de simples méthodes appartenant à la classe émettrice, dont il n'y a pas d'implémentation. Pour sa part, le slot connecté à un signal est une méthode appartenant à la classe qui reçoit le signal, qui doit avoir les mêmes paramètres que le signal auquel il est connecté et dot être implémenté par le développeur.
Gameframework	Le gameframework est un ensemble de code qui permet la réalisation plus simplifiée d'un jeu ou d'un projet.

Description des étapes du projet

1) Choix du projet

Nous avons dû commencer par choisir jeu dans lequel on allait se lancer pour le reproduire au mieux, ou en créer un s'il convenait à l'enseignant. Le but était de trouver un jeu suffisamment complet sans trop être un jeu trop complexe. Nous devions faire valider l'idée de projet que nous voulions faire à notre enseignant. Aimant bien le jeu Pong étant petit mais étant trop simple à réaliser, j'ai donc choisi une version dérivée du Pong, le fameux Casse-briques.

2) Découverte du code fourni par l'enseignant

Ayant la chance d'avoir fait du C++ en première année, j'avais déjà quelques notions de bases pour commencer. De plus, notre enseignant nous a mis à disposition un GameFrameWork qu'on pouvait sans autre récupérer pour utiliser des fonctions/bouts de code pour notre projet. Une documentation était fournie avec, ce qui m'a aidé à mieux comprendre le code qui était fourni pour ensuite l'implémenter dans mon code.

3) Création de la fenêtre de délimitation

Pour commencer, j'ai commencé par créer une zone rectangulaire dans laquelle ma boule et mon personnage (le rectangle) n'allaient pas pouvoir sortir de la zone. Le principe repose sur une zone de délimitation qui gère les collisions.

4) Création du personnage

J'ai ensuite créé mon personnage qui est le rectangle sur laquelle la balle doit rebondir, je lui ai donnée une vitesse en pixels par tick qui vérifiait l'emplacement suivant s'il pouvait s'y rendre ou s'il rentrait en collision avec mon rectangle de délimitation. J'ai dû gérer la répétition de touches pour pouvoir se déplacer aisément, grâce à KeyPressed(). Je lui ai ajouté une animation de changement de couleur pour rendre le « personnage » plus attractif, à l'aide de addAnimation() et startAnimation().

5) Implémentation de la boule et des murs

J'ai repris du code qui nous était à disposition dans le GameFrameWork, une boule qui se déplaçait, je l'ai modifié pour qu'elle me corresponde et je l'ai implémenté dans mon projet.
Je la fais réapparaître sur le rectangle si elle touche le mur du bas, en suivant cet algorithme :

- La boule en position Y est en dessous de la valeur 685 ?
- Si oui → passe le boolean *isWaiting* à vrai.
- Si *isWaiting* → Positionne la boule au centre du personnage (rectangle)
  → Si l'utilisateur appuie sur espace ou clic gauche de la souris
  ➔ Réinitialise les valeurs et la boule continue sa trajectoire normalement

---

Pour les différents cubes à détruire j'ai opté pour une boucle dans une boucle pour une création simple rapide et efficace de mes différents sprites. J'utilise les conditions suivantes pour la création de mes différents blocs à détruire :

- **Créer une boucle i de 0 à 3**
  - **Créer une boucle j de 0 à 18**
  - **Créer un bloc à chaque fois en lui attribuant un ID**
  - **Ecarte de X pixels pour le suivant blocs**
  - **Fin de la boucle j.**
- **Ecarte de Y pixels pour la prochaine rangée de blocs**
- **Réinitialise l'écartement X pour que les blocs soient alignés.**
- **Fin de la boucle i.**

Améliorations futures envisageables

Quelques améliorations qui pourraient être ajouter au projet avec un peu plus de temps, ce qui le rendrait beaucoup plus attractif et amusant :

Gestion de score

J'ai pensé à faire un tableau des scores en prenant en compte le temps du joueur pour détruire tous les blocs, les scores seraient enregistrés dans un fichier JSON et un affichage dans une autre scène qui trierait les scores en fonction du meilleur temps obtenu.

Briques bonus/malus

Des briques qui apporteraient un bonus ou malus au joueur pour lui permettre d'augmenter comme diminuer son score pourrait rendre le jeu plus intéressant pour le joueur. De plus, des bonus de rapidité ou de lenteur pourraient rendre le jeu plus vif et dynamique.

Gestion de niveaux / Affichage graphique

Créer plusieurs niveaux différents avec plus ou moins de blocs à détruire et en formant des différentes formes/dessins qui seraient formés dans les blocs, ce qui rendrait le jeu plus beau grâce à ses affichages et plus amusant en ayant plusieurs niveaux différents.

Différentes vitesses de la boule

En fonction du nombre de vie restant au joueur ou du nombre de blocs restants, la boule irait de plus en plus vite, ça permettrait de rendre le jeu plus difficile et donc créer un « challenge » pour obtenir la première place du podium.

Ce que le projet m'a enseigné

En général

J'ai appris énormément de choses sur la programmation orienté objet ce qui va m'aider pour le futur, connaître plus en détail le background d'un jeu, la compréhension d'un code, que ce soit concernant les variables, les constantes, les nouveautés qu'on peut faire que je ne connaissais pas comme laes bibliothèques *QDebug*, *QSettings*, *QColor* et *QPainter*.

Ça m'a rappelé quelques notions de bases de la programmation qu'on peut oublier comme la bonne notion des noms de variables, les commentaires qui doivent définir clairement ce que fait un bout de code ou encore ne pas mettre des noms de variables ou fonctions en franglais.

Sur Qt Creator

Je n'aurais jamais imaginé pouvoir faire un jeu à l'aide d'un autre logiciel que Unity ou Unreal Engine. Qt Creator offre beaucoup de possibilités de faire son propre projet et il ses avantages propres à lui,

comme l'utilisation de slots pour détecter la fin d'une animation, QPaint pour dessiner sur notre projet ou encore pouvoir gérer ses propres tick et sprites.

Compétences de gestion de projet

Ordre dans lequel travailler

Je comprends mieux pourquoi il faut penser à faire des schémas de son projet avant de se lancer à l'aveuglette, il est très important de se représenter comment réaliser et que faut-il faire au début avant de commencer à trouver des animations graphiques et autres options graphiques non nécessaire à la partie logique du jeu.

Conclusion

Créer un jeu grâce à Qt Creator m'a appris et remémorer beaucoup de choses, comme la création d'un schéma avant de se lancer, les différentes règles de programmation pour les variables, constantes, fonctions, etc. Et ça va m'aider pour la suite à comprendre mieux les jeux et le code en général et en C++, je reconnais que c'est un petit projet pas non plus complexe à gérer, mais ça m'aide à maîtriser les bases avancées de la programmation et la conception d'un projet. Un point négatif est la création d'un fichier d'un **exe** qui selon moi est assez complexe comparé à d'autre IDE. Pour conclure, je dirais que Qt Creator est très utile pour se lancer dans des petits jeux pas trop complexes, des mini-jeux sympathiques, mais je ne développerai pas un jeu comme un MMORPG avec des graphiques en 4K et des animations complexes sur ce logiciel, cependant il m'a été très utile et enrichissant pour la compréhension de code en C++.