Rapport de projet

Par :

Simon Gagné

Présenté à

Mr. Louis Marchand

Dans le cadre du cours :

420-PRB-DM

CÉGEP de Drummondville

26 mai 2016

Introduction

Le jeu que j'ai programmé est une imitation du jeu Bubble Trouble, disponible à l'adresse suivante : www.miniclip.com/games/bubble-trouble/en/. Dans ce jeu, le joueur incarne un personnage dont le but de tuer des balles mortelles qui bondissent. Ainsi, lorsque l'une d'entre elles entre en contact avec le joueur, il meurt. Pour se défendre, le personnage peut se déplacer à gauche et à droite. Il peut également tirer des flèches sur ces balles afin de les détruire. Une fois détruite, une balle se divise en 2 balles plus petites, qui elles, se diviseront également en 2 balles plus petites jusqu'à être trop petite pour être divisée. À ce moment, la balle disparaît. Lorsqu'il ne reste plus de balles, le niveau se termine. Dans ce jeu, il y a 3 niveaux pour compléter le jeu. Dans ce rapport, je présenterai un guide d'installation et d'utilisation du logiciel, et ce, autant pour Windows que Ubuntu. Je présenterai aussi l'évolution du projet au fil de la session. Je glisserai également le diagramme de classes de mon application dans ce document. Je ferai également une autocritique de mon projet. Finalement, je présenterai quelques « designs » objets utilisés dans mon application.

Un dépôt git du projet est disponible à l'adresse suivante : https://www.github.com/bobbytava/projet_POO2.git.

Guide Installation

Windows

1. Télécharger Eiffel Studio

   Un lien de téléchargement est disponible à l'adresse suivante : https://sourceforge.net/projects/eiffelstudio/. Prendre la version 32 ou 64 bits, dépendamment de l'architecture de votre ordinateur.

2. Installer Eiffel Studio

   On extrait le tout et on exécute l'installateur

3. Télécharger la librairie du jeu

   On se dirige sur la page « https://github.com/tioui/Eiffel_Game2/tree/windows_build » et on clique sur le bouton vert « Clone or Download » et on sélectionne « Download zip »

4. Installer la librairie

   On extrait le fichier « .zip » télécharger. On copie le dossier « game2 » dans le dossier /contrib/library d'Effiel Studio. Ce dossier est généralement à l'endroit suivant : C:\Program Files\Eiffel Software\EiffelStudio 15.11 GPL\contrib\library.

5. Télécharger le jeu

   Le jeu est disponible à l'adresse suivante : https://github.com/bobbytava/projet_POO2.

6. Ajouter les fichiers DLL.

2 dossiers de fichiers DLL sont dans le répertoire de la libraire extraite à l'étape 3 (les DLL 32bits et 64bits). Extraire le dossier .zip correspondant à l'architecture de votre ordinateur. Copier tous les fichiers DLL de ce répertoire dans le répertoire du jeu téléchargé.

7. Lancer le jeu

   Exécuter le fichier « projet.ecf ». Eiffel Studio se lancera et compilera le programme. À la fin de la compilation, appuyer sur la touche F5 ou cliquer sur la flèche verte « Run »

<u>Ubuntu</u>

1. Télécharger Eiffel Studio

   Dans un terminal, lancer les commandes suivantes :

   - sudo add-apt-repository ppa:eiffelstudio-team/ppa
   - sudo apt-get update
   - sudo apt-get install eiffelstudio

2. Installer les libraires C nécessaires

   Dans le terminal, lancer les commandes suivantes :

   - sudo apt-get install git libopenal-dev libsndfile1-dev libgles2-mesa-dev
   - sudo apt-get install git libsdl2-dev libsdl2-gfx-dev libsdl2-image-dev libsdl2-ttf-dev
   - sudo apt-get install libepoxy-dev libgl1-mesa-dev libglu1-mesa-dev

3. Télécharcher la librairie du jeu

   Dans le terminal, lancer la commande suivante :

   - git clone --recursive https://github.com/tioui/Eiffel_Game2.git

4. Configurer la librairie du jeu

   Dans le terminal, lancer les commande suivante :

   - sudo ln -s `pwd`/Eiffel_Game2 /usr/lib/eiffelstudio-15.12/contrib/library/game2
   - cd Eiffel_Game2
   - ./compile_c_library.sh

5. Lancer le jeu

   Exécuter le fichier « projet.ecf ». Eiffel Studio se lancera et compilera le programme. À la fin de la compilation, appuyer sur la touche F5 ou cliquer sur la flèche verte « Run »

Guide d'utilisation

Lorsque qu'on démarre le jeu, nous arrivons dans un menu où il y a 3 options. Une option est sélectionnée par défaut par une flèche à la gauche de de l'écran. L'usager peut changer l'option sélectionnée à l'aide des touches « haut » et « bas ». L'usager confirme son choix à l'aide de la touche « Enter ». Les 3 options sont les suivantes :

1. Play

Lance le jeu.

2. Host/join a game

   Dirige vers un menu où différentes options de jeu en ligne sont disponibles.

3. Quit

   Quitte la partie

Play :

Avant le début de chaque niveau, il y a un décompte de 3 secondes pour laisser au joueur le temps de se préparer. Comme mentionné dans l'introduction, le personnage peut se déplacer à droite et à gauche à l'aide des touches « droit » et « gauche » respectivement. Cela est très utile pour éviter les balles, ou les chasser. Le personnage ne peut quitter l'écran. Comme vu précédemment, le but du jeu est de détruire les balles. Pour ce faire, le joueur peut appuyer sur la touche « haut » afin de tirer une flèche. Cette flèche est lancée verticalement. Si cette flèche touche une balle, celle-ci meurt. Nous reviendrons sur le fonctionnement de la mort d'une balle. Une seule flèche peut être lancée par le personnage à la fois. La flèche est réinitialisée de 2 façons : si la flèche touche une balle ou si la flèche touche le plafond. La flèche est lancée du centre du personnage. Une fois lancée, si le personnage se déplace, la flèche reste sur place. Une balle pour être tuée autant par la pointe de la flèche que le corps de la flèche. Ceci est une astuce pour tuer les petites balles qui ne rebondissent pas très haut. En effet, on peut lancer une flèche devant la balle et s'éloigner afin que la balle touche le corps de la

flèche. Voici le fonctionnement des balles. Il existe 3 grosseurs de balles. Plus la balle est grosse, plus elle rebondit haut. Une balle rebondit toujours à la même hauteur (sauf si une balle est le résultat d'une autre balle est tuée très bas. Nous en reparlerons plus tard). Une balle peut autant se diriger vers la gauche que la droite. Toutefois, elle conservera la même direction jusqu'à temps qu'elle touche un mur, où elle changera de direction. Tuer une grosse balle donne 50 points, tuer une moyenne donne 25 points et tuer une petite balle donne 10 points. Ces points se cumulent jusqu'à la fin de la partie. Lorsqu'une balle se fait tuer, à moins qu'il ne s'agisse d'une petite balle, elle se divise en 2 balles plus petites. L'une d'entre elles se dirigera vers la gauche tandis que l'autre se dirigera vers la droite. Les nouvelles balles crées sont des balles de la taille inférieure à la taille de la balle qui a été tuée (une balle grosse balle crée 2 moyennes et une moyenne crée 2 petites). Ces balles crées un absorbent un petit momentum de la flèche et sont ainsi un petit peu projetée vers le haut. Lorsque ces balles retoucheront, elles rebondiront à la même hauteur que les autres balles de leur taille. Ainsi, lorsqu'une balle est tuée très basse, même après sa projection, il arrive qu'elle soit plus basse que la hauteur à laquelle elle est supposée rebondir. C'est pourquoi les premiers bonds d'une balle crée par une balle tuée près du sol peut surprendre. Si la balle tuée est une petite balle, elle disparaît. S'il n'y a plus de balles de le tableau, le joueur passe au niveau suivant. À la fin du troisième tableau, la partie se termine et le joueur est victorieux. Toutefois, si une balle entre en collision avec un joueur, la partie est terminée et le joueur est considéré perdant. Peu importe la conclusion, à la fin de la partie, le joueur a 2 options : appuyer sur « Enter » pour retourner au menu principal,

ou appuyer sur « Esc » pour quitter le jeu. À noter qu'à tout moment, même dans les menus, le joueur peut appuyer sur Esc pour quitter le jeu.

Host/join a game

Cette option dirige vers un autre menu où les différentes options réseaux sont offertes. Voici les options disponibles :

1. Host

   Cette partie, présentement non-fonctionnelle serait supposée créer un serveur qui attend la connexion d'un autre usager. Lorsque ce nouvel usager se serait connecté, ils seraient supposés jouer une partie ensemble (chaque joueur aurait son personnage). Pendant que le joueur hôte, attendrait un autre usager, il aurait la possibilité d'appuyer sur la touche « Enter » pour retourner au menu précédent (ce menu-ci). Toutefois, en ce moment, lorsque qu'on sélectionne l'option « Host », le jeu cherche indéfiniment et plante.

2. Join

   Cette partie se déplace vers un autre menu ou l'on demande à l'usager d'entrer l'adresse du server à joindre. Lorsque l'usager entre une adresse et appuie sur « Enter », il serait supposé joindre la partie et jouer avec l'autre personne. Toutefois, cette partie ne fonctionne pas et la connexion avec le serveur n'arrive jamais puisque le programme plante. L'usager qui tente de joindre tente de joindre la partie d'un autre peut toutefois, avant de confirmer

l’adresse à joindre, descendre sa sélection et appuyer sur « Enter » pour retourner au menu précédent (Host/join a game).

3. Back

   Sélectionner cette option retourne au menu précédent (menu principal)

Quit

Quitte le jeu.

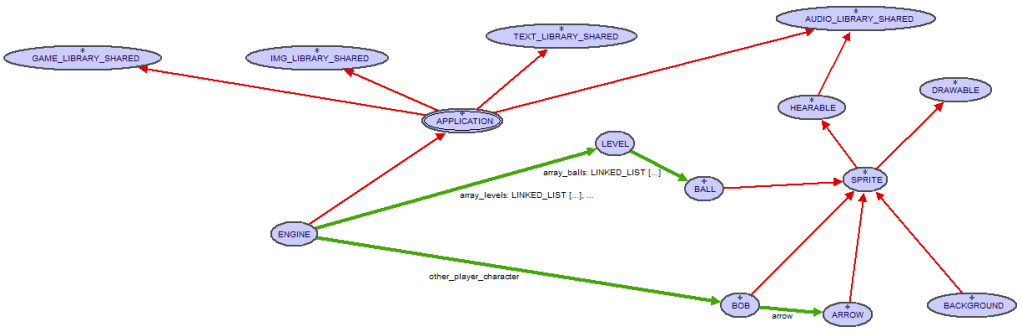
Évolution du projet

Ce projet s’est déroulé en montagne russe. Puisque j’avais abandonné ce cours l’an dernier car je m’étais pris vraiment, vraiment en retard, j’ai commencé la session avec la volonté de faire ce projet du début à la fin. C’est pourquoi j’étais à jour lors de la remise 1. J’avais alors un jeu qui faisait du son avec mon personnage qui se déplaçait en faisant une animation. Toutefois, plus la session avançait, mes vieilles habitudes ont repris le dessus j’ai arrêté de travailler sur ce projet. C’est pourquoi je n’ai pas remis la remise numéro 2 car il s’agissait du même programme que la remise 1. J’avais trop d’orgueil pour avouer que je ne faisais rien dans ce cours. Ainsi, j’ai recommencé à programmer pour la remise 3. J’ai gérer le mouvement des balles. Cela m’a pris beaucoup de temps car j’avais, à la base, une approche qui gérait la vitesse des balles et lieu de sa position, ce qui créait des problèmes lorsque je tuais une balle. J’ai aussi créé ma flèche. Ainsi, j’ai remis à la remise 3, au travail qui aurait été à peine suffisant pour la remise 2. Par la

suite, il est arrivé quelques complications dans un contexte hors-projet : j'ai été malade, j'ai eu beaucoup de travaux dans d'autres matières et un changement dans ma vie personnelle. Ainsi, je n'ai pas touché au programme jusqu'à la remise 4, où je n'ai pas remis de travail, encore une fois par honte de ne pas avoir progressé. C'est à ce moment, où à travers tous mes examens et mes nombreux travaux dans mes autres matières, avec un travail ayant 1 mois de retard, que j'ai décidé d'en faire un peu tous les jours. Mais ce ne fut pas suffisant, loin de là. Alors me voici avec mon travail, avec un mode solo fonctionnel mais avec une implémentation réseau non-fonctionnelle. Cela une semaine que j'essaie de le faire fonctionner sans résultats. J'ai également tenté de faire des test automatisés mais j'ai manqué de temps pour rendre le tout fonctionnel.

Diagrameme de classes

Ce diagramme est également disponible dans le dépôt git



Autocritique

Je m'autocritique un peu comme mon projet s'est déroulé. Je suis très satisfait de moi jusqu'à la remise 1. Tout ce déroulait super bien. Par la suite, j'ai arrêté un moment et tout à dérailler. La partie suivante du programme (les balles) ont été un peu compliqué car je n'avais pas la bonne vision face au problème. Mais en ayant continué à travailler, j'aurais réussi à trouver ma solution bien avant la remise 2 et j'aurais pu alors rester à jour pour la programmation réseau de la remise 3. C'est le fait que j'aie retourné à mes vieux démons de paresseux. Mais en ayant procrastiné, la seule amélioration que je me suggère aurait été de demander de l'aide à la suite de la remise 4 lorsque j'ai recommencé à travailler sur le projet et que je me suis rendu compte ma partie réseau ne fonctionnait pas. J'aurais ainsi pu passer plus de temps sur le reste. Mais j'ai tenté de faire le plus possible jusqu'à la fin au lieu d'abandonner, comme l'année dernière, et j'en suis fier.

Design objet

Constructeur :

Un bon exemple de constructeur est ma classe LEVEL où il n'y en a 3. En effet, cette classe possède un constructeur différent selon le niveau que l'on veut créer.

Héritage simple :

Un bon exemple de l'héritage simple est ma classe HEARABLE qui hérite de AUDIO_LIBRARY_SHARED lui permet d'avoir toutes les fonctionnalités de cette dernière.

Héritage multiple :

Un exemple d'héritage multiple est ma classe SPRITE qui hérite de HEARABLE tout en héritant de DRAWABLE. Ainsi, tout ce qui découle de SPRITE peut autant être affiché que jouer un son.

Polymorphisme :

Je n'ai pas trouvéd'utilisation pertinente de polymorphisme dans mon programme. Mais j'imagine de l' « array_balles » de ma classe ENGINE aurait pu être un liste de SPRITE au lieu d'être une liste de BALL dans l'optique où une autre classe n'étant pas une balles aurait besoin d'être passé une par une dans une boucle

Classe abstraite :

La classe DRAWABLE est un bon exemple de classe abstraite puisque qu'elle contient des fonctionnalités de bases intéressantes. Toutefois, elle est abstraite car on ne veut pas l'implémenter puisqu'il s'agit seulement d'une fondation pour les objets pouvant être affichés.

Méthode redéfinie :

Un exemple de méthode redéfinie est un la méthode play_sound de HEARABLE. Puisque certaines SPRITE qui font du son gère leur son différemment (BACKGROUND joue en boucle) il faut redéfinir cette fonction dans chacune des classes héritantes.