University of California, Merced

CSE-165 Final Report
Spring 2022

By: Neida Alonso, Valentinno Cruz, and Andre Martin

CSE 165-01: Introduction to Object-Oriented Programming
Professor Ammon Hepworth
May 9th, 2022

1. Project description – a short description of the project topic.

Bringing back the retro gaming with Brick Breaker. Specifically, a user-friendly and easy-to-use version. This game is set in a 2D dimension where the ball moves autonomously, and the user can control the platform the ball moves from. The goal is to not let the ball fall or lose a life. Our concept is based on using the CPP files to create a back end that focuses on the main function of players' health, the bounce of the ball, the bricks, and the score. The header .h files would be the bar of the health, the game, score, and main window. The bonus would be music, background, and images

2. Members – description of how each member contributed to the project.

For this project, it was our first time using QT. We filtered the files depending on the required requirements of the project. As we felt either stuck or either confused we were able to help each other in an aspect that was able to help both ends whether one person was working on the same function requirement or if it was different from the other person. We still felt quite shaky and split the work of the source files and header files evenly. We frequently took turns on each other's files as debugging soon became evident.

3. Implementation – short description of your source code.

Source Files

Main.cpp

As the main file, it references the game header file and additionally receives and processes the application to transfer to the appropriate widgets.

Game.cpp + Game.h

To summarize the game file sets the game up. Here one can see the background, scroll bar, the score, health, and player are defined and created. Additionally, the header file has the classes Qgraphicscene and Qgraphicsviews which are used to visualize the game's scene, player, score, and health points of the player.

Enemy.cpp + Enemy.h

This enemy file contains, what we would argue is one of the most important parts of the game. Here we have initialized the enemy to be placed randomly throughout the game and see if the enemy and the player interfere with one another to cause the player to lose health points.

Additionally, the header file calls the classes of QGraphicPixmapItem and QGraphicsItem to add a pixmap and graphics item to QGraphicsScene.

Health.cpp + Health.h

The health file focused on,

Player.cpp + Player.h

For the player file, it revolved around movement. It was crucial to get this part down as without movement there is no game. As one can see, the focus was on speed and movement. We had to do ample test runs to get the speed down as we did not want to be too fast or too slow. If time prevailed we would have spent more time in this section. Additionally, the feature of shooting to destroy the enemy was done here. This ties us to the header files it calls to which is the player, shoot, and enemy header.

Shoot.cpp + Shoot.h

The bounce file aims at the idea of movement and collisions occurring as the enemy rains down onto the player. Additionally, where the score is marked down when collisions occur. A bonus item we wanted to include was upgrades depending on the players score; however, that was more complicated than we thought.

Score.cpp + Score.h

The score file is short and straight to showing the points the player still had. We focused on four components: whether the player gained or lost points, if the same was reset, and to display the total score. This was shown through the class QGraphicsTextItem which can be seen in our header file.

MainWindow.cpp + MainWindow.h

The MainWindows is pretty explanatory at this point. It is the window that displays the interface and where the layout can be created as shown.

4. Lessons/Conclusions – what are the lessons you learned and you want to share with your colleagues.

Neida Alonso: Researching and finding documentation is so important to prevent error codes. Additionally, time management is a must as we had planned to do a few bonus features such as music and allow the user to do upgrades; however, due to timing decided not to. Lastly, make sure you have sufficient space on your computer/laptop to run QT as it takes up so much space.

Andre Martin: The main thing that I came to know is how to format a lot of code through various versions of .cpp files. I learned how many functions get done through the .h files as well as how they get implemented. One thing I would tell my colleagues who attempt at this with a partner is to make sure the Qt application is up to date.

Valentinno Cruz: The main thing I learned was the organization of classes and files. When we first started we tried putting all the classes and widgets into a small number of files. However, it became obvious that it was better to have multiple small folders as it's easier to manage. More importantly, QT is quite frustrating and to make this project doable have ample communication and time set apart with your team to get it done.