Majok Ring

Luke Moore

Zachary Gill

CSE 165

May 9th, 2016

Object Oriented Programming

Final Project Report

Our final project is a brick breaker video game. The game is played using left and right arrow keys to move the paddle and can also use the mouse motion to move the paddle left and right. Within this game, the player only has control of the paddle and all other objects move independently. The objective of the game is to keep destroy as many blocks as possible and increase your score until your lives goes to zero. A life is lost if the ball passes the players paddle at the bottom of the screen. During the creation of each level, certain blocks have powerups enabled and once hit, will drop powerups that the player can catch using the paddle and use. Once the lives goes to zero, the player can decide to play again with a new set of lives and the high score is kept on the screen. The work load between the three of us was evenly split. Majok implemented the core mechanics of the game along with the power up drops. He was able to make the collisions work properly to make blocks disappear and have the ball to change directions and speed when deflected off of the paddle or walls. Luke worked on the game interface such as the score and lives interface, hit animations/color, and also worked on mapping the controls of the game to the player’s mouse and keyboard. Zachary implemented the scoring and points system along with the lives system logic. All of these parts of the game were able to work in unison which made a functional brick breaker game.

The logic of the game was wrapped in our Elements.h class. In the class we created a Ball object, a Paddle object, a Brick object, and a Game object which held the state of the game. The game class holds our vector of elements, and every elements inside of its draw function. The game class also creates blocks and also creates the menu/score interface. All of these objects inherited from the GrRect class and some classes would override the draw function for functionality and capabilities. We used the glutBitmapString function to print out text for our game interface such as the score board, lives, and points on screen. In order to implement sound, we used python and also a python extension called pyaudio in order to play.wav files on Linux. This code was ran through a terminal by the following lines, system("cd sounds; python playwav.py loseLife.wav > /dev/null 2> /dev/null &");

As for the results, we were able to create a working game that illustrates Object Oriented concepts such as inheritance, encapsulation, and polymorphism. Our game makes use of virtual functions and allows correct object types to call the corresponding functions. All of the mechanics of the game work correctly with the exception of some glitches such as the ball phasing through two blocks instead of one and the mouse needs to be double clicked in order to begin moving the paddle. The scoreboard and lives system was successfully implemented and makes our game easy to use/understand for anyone as the text is readable and updates in real time.

The lessons we have learned through the making of this game has taught us the more important and fundamental aspects of programming such as portability. We learned that making your implementation simple is the best way to program as you reduce the risk of causing difficulties if you want to port your program to another machine or OS. By making all of our objects inherit GrRect, we were able to have full control over each object as they had the same members to manipulate and we did not have to worry about compatibility. In our project, playing sound was an issue on Linux as we did not have access to Windows.h in order to use the playsound function. We were able to bypass this by using another programming language, python, in order to successfully run/compile the playwav.py file through our main program and play sound during special events. Overall, an important lesson is to make sure that you are hiding your implementation in a clean and organized way so that other programmers can read and collaborate without issue.