**Project Overview**

Title of the Project: BrickBreaker

Names of Team Members: Connor Byers, Diana Constantinescu, Nicholas Makharinets

Short Summary of Final Project:

For our final project, we decided to make BrickBreaker. The game requires the player to use the left and right arrow keys to move the sliding block near the bottom of the screen to make contact with a ball. The ball bounces off the block and moves towards the top of the screen where there are three rows of red bricks. When the ball makes contact with a brick, it will bounce off and that brick will disappear. The player needs to move the sliding block to continue bouncing the ball off it so they can delete all the bricks. If the player misses the sliding block with the ball, the ball will fall off the screen, the player will lose a life, and then the ball will reset, moving upwards towards the bricks, so that they have time to re-adjust, instead of quickly losing another life. The player has three lives. If the player loses all their lives, a “Game Over” screen appears. If the player successfully destroys all the bricks, a “You Win” screen pops up. In both cases, players can press any key to exit the game.

**Project Reflection**

When we first started our project, we had multiple goals we wanted to achieve in creating our game, BrickBreaker. We obviously wanted a functioning game, but we hoped to create multiple rows of bricks, instead of just one, and we wanted to make each row a different colour. We also hoped to make sure the sliding block at the bottom does not move off the screen left or right, so we wanted to make sure the sides acted as boundaries or walls, especially so the ball could properly bounce off them. We also had the goal to make the hit detection work whether the ball hit the bricks from the top, sides or bottom. We also wished to make the game as smooth as possible, so that it does not lag or glitch.

When we started the project, we outlined many features and functionalities. We planned to have multiple functions such as a function to move the bottom sliding block, a function to print the bricks, a function to calculate a score, a function to show the starting and exit screen, a function to move the ball, and many many more. When we originally handed in the Gantt Chart, we were at a point in our coding where the ball was bouncing off the walls and the bricks were drawn, but the hit detection was not yet properly working. Since this point, we successfully programmed the hit detection, we added lives, made sure the ball could fall off the screen instead of bounce off the bottom, we also created an introductory screen, a ‘Game Over’ screen and a ‘Winner’ screen. In other words, we reached all our goals, and we even programmed and reached goals that we did not originally plan for, such as printing out the lives. It was possible to reach our goals because we worked on our code from different angles and perspectives and treated each step as its own problem until we figured out the logic to each aspect of the code. Throughout the project, our timeline remained as planned and on schedule, allowing us to effectively finish our program before the deadline. Through our individual skills and collaborative group work, we were able to satisfy all our goals and create a fun game for our final summative.

**Documentation**

As a future direction for our project, with more time, we could implement more features. This could include “power-ups” that could make the ball or the sliding block bigger or smaller like the ones in the original Breakout and/or Brick Breaker. We could add a score that would show how well a person did. The game could also have a time factor where the player needs to break all the bricks before the time runs out. We can also add a feature that prints out how many bricks the player broke, which goes hand in hand with giving them a score at the end of the game. We could add customization so that the user could enter the colors they want for the bricks, sliding block, background, ball, etc.

In terms of issues of ongoing maintenance, we can try to make the program a little smoother in terms of the delay, and reprinting the ball and sliding block so that the program runs without glitches. We can also improve the hit detection to make sure when a brick is hit, other bricks are not partially coloured white. To solve that, we can use the ball radius more in the hit detection part of our code so that the hitboxes are more accurate. We can also change it so that when a brick is hit, the white lines of other bricks are not coloured black.

In terms of structure, we decided to use a variety of functions to make the main code easier to work with and read. This allows for a clean program that can be understood with ease, allowing us to quickly find any issues in our program that need to be changed. We decided to keep our code all in one file because it is nicely split into sections where we declare all variables and functions, we have the main code, and then we have the definitions of all our functions. This way we can easily find what we are looking for in our code, since it is all in one place. We also structured our code this way so that we can run the program to make sure it works properly after each change so that we can easily find bugs and fix them right away, instead of trying to search for problems at the end after having put the whole code together.

In the program, we colour the ball and the sliding block black and redrawing them in a new position in white instead of doing the clear screen method so it’s less choppy. We declare all the variables, draw the ball, sliding block and bricks. Then a while loop is used so the game will continue until all 3 lives are gone or they win. In that while loop, we delete/colour black the ball, sliding block and the lives first. Then, we do the hit detection for the bricks. After that, we do the hit detection for the walls. In that part, we check if it hits the bottom and if it did, we decrease the lives and reset the ball’s position. After that, we move the sliding block based on user input (left/right arrows). After that, we do the hit detection for the sliding block. Then, if the ball hasn’t hit something else, it will add the balls velocities to its coordinates. Finally, it draws the ball and sliding block with their new coordinates.

There are a few suggestions for the future development work that could be done to improve the program such as adding multiple levels to the game, each with an increasing difficulty relative to the previous level. We could potentially even include a second ball to make it harder for the player to keep them both from falling off the screen. We could even make the rows of bricks move down the screen slowly and if they hit the bottom before they user hits them with the ball, they lose the game. This would again increase difficulty and advance the game for elite players.

Any missing and undeveloped features that can be finished include having a feature that either randomly alters the speed of the ball, or gets increasingly faster as the game progresses to make it harder. We tried to include this feature in our program but it caused many bugs and issues so we decided that picking an average velocity of (5,5) will create the best experience for the user.