Breakout C200 Final Project

By: Sam Goodin, Jimmy Conway, and Lucas Forbes

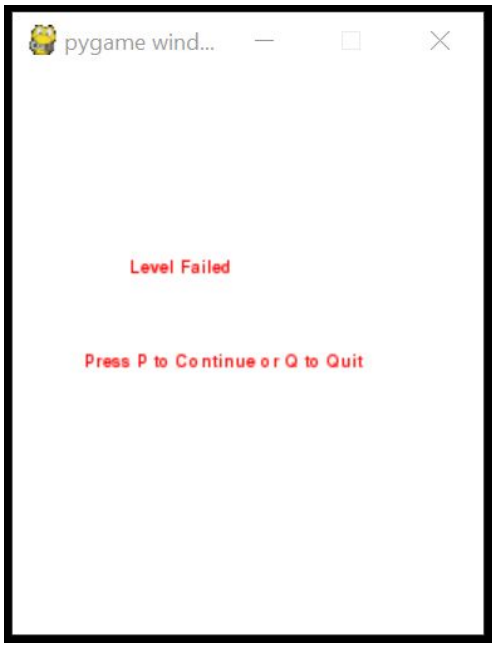
This is the documentation for our project we made for you, Sahiti, our client. Beginning with sprint 1 this document will take you through the entirety of our code and our thought processes behind it. The project you have requested us to make for you is the game “breakout”. This game consists of a user controlling a paddle to hit a ball which then bounces off walls or bricks that are parallel to the paddle. This ball will continue to bounce back and forth so long as the user doesn’t miss it with their paddle or the ball hits the top of the screen. There are many forms of this game and we hope this document will provide you with the insights of why we designed the game the way we did.

Our team is made up of two freshmen, Sam Goodin and Jimmy Conway, and one Sophomore, Lucas Forbes. Sam and Jimmy bring with them multiple years of coding experience while Lucas is relatively new to C.S. Relative to the amount of time they have been coding, Sam and Jimmy can code with speed and precision. Lucas, while not as fast, is able to keep up and provide input in all areas. The biggest hurdle for the team has been learning how to use pygame which has been an interesting and fun task. The group has worked very well and each member has brought their skill set to use. Input is given by everyone when designing the game and writing up the code to produce it. The following paragraphs document the days we worked on the project and the features we implemented.

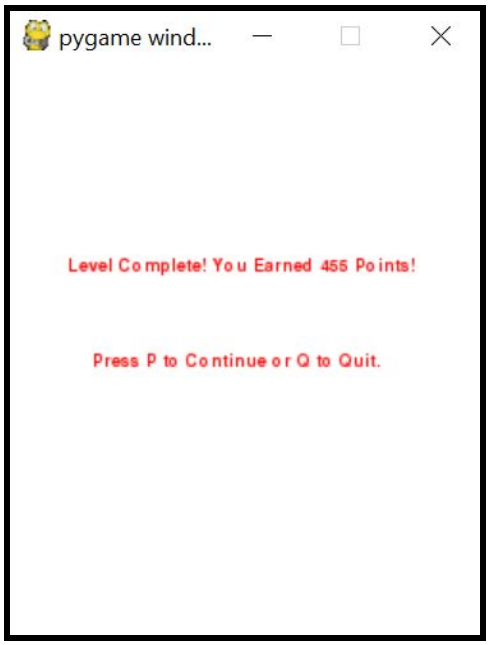
Our group met on Monday, November 27th for four hours to complete the tasks outlined in Milestone 1. Our first decision was to go with a screen size of 250 width and 300 height as we believed these were fair dimensions for the game. We then decided to take on the rest of the features individually while helping each other on certain aspects. When designing the brick layer

on the top of the screen we went with standard sized rectangles 24 in width and 10 in height to make a nice even row of 10 bricks across. This however created a small gap in between the bricks which we later fixed. We used two classes, one for the design of the brick and one for the design of the brick rows. We colored the bricks using a random integer in a 5-255 range. Moving on to the paddle we decided to have the paddle width be 55 and the height be 5 as it is displayed on the bottom of the screen. We have the paddle being controlled by the user arrow pad for ease of use. While working on the paddle we split the paddle into 5 sections, each of which change the direction and the speed of the ball upon contact. We placed the ball randomly in between the bricks and paddle using a random x and y coordinate setup that would not allow the y coordinate to be greater than 220. This specific height coincided with our random direction for the initial ball to not let it shoot directly past the user. The detection of when the ball hits one of the walls was as simple as adding to the x coordinate of the ball to either positive 1 or negative 1 to change the direction of the ball to its opposite. The same thought process was applied for when the ball hits the bricklayers and the y coordinate. In later levels with multiple rows we went back and changed our code to account for when the ball hits the side of a brick. The indication of a level complete was done by a simple if statement to check if the x value is <= 0 (the ceiling of the user screen). The same process was applied to the bottom of the screen. The destruction of a brick was done by first creating a method to assign a true or false value to the visibility of each brick. The bricks begin with True as their value which allows a later nested for loop to draw all the bricks with True as their value. If the brick is hit with the ball its value is turned to False which means it is skipped in the next frame. We decided to use 20 as our time delay as this creates a fair game difficulty.

This screenshot displays the screen when the ball is missed by the user



This screenshot displays the screen when the user completes a level (with score from sprint 2)



Our group met on Tuesday, December 5th for four hours to complete the tasks outlined in Milestone 2. Our first requirement we built into our game was the level progression. We did so using a levels class and creating 5 different methods inside the class, one for each of the levels. The information held in each of these levels effects the difficulty and final score for the game. We then worked on the amount of lives the user has. We decided 3 was a fair number for the amount of levels in our game and stored the information in a lives variable which is decreased every time the ball reaches the bottom of the screen. We added a user screen for when a life is lost, when the game is over, or when the game is won allowing the user to either continue or quit. The scoring function we built is designed in a way to allow the highest score to go to the user who completes the game in the fastest time while hitting as few blocks as possible. This is done by decreasing the user's score by 10 points every time a brick is hit and by 1 point every 40 frames. The top 10 scores are updated and stored in a flat txt file every time the game is

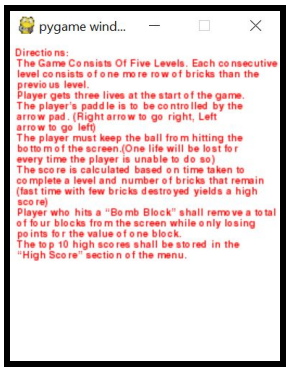
played. At completion of the game the user shall input it's three letter (ABC) initials to have their code be added to the high score list. After searching for a long period of time were not able to find a way for the user to input their initials using the display. We decided having users enter their high scores through the console would suffice. The score, number of lives, and timer were all added to the bottom of the screen underneath the paddle to allow the user to see this information while playing the game. We then created the game screen which will show at the beginning and end of the game. We allowed for three options to be selected via arrow pad: Play, Directions, and High Scores. When selected, Play game will allow the user to start Breakout on level 1, Directions will take the user to a new screen with the directions of how to play the game, and High Scores will take them to a new screen that shows the top 10 high scores. We had to fix a few bugs at this stage including fixing the loss of a life to keep the user on the same level when the user resumes playing and having all user information reset upon start of new game. The instructions for the game are as follows:

1. The game consists of five different level layouts. Each consecutive level consists of one more row of bricks than the previous level. Upon completion of the fifth level the user will keep playing the fifth level until they run out of lives.
2. Player gets three lives at the start of the game.
3. The player's paddle is to be controlled by the arrow pad. (Right arrow to go right, Left arrow to go left)
4. The player must keep the ball from hitting the bottom of the screen.(One life will be lost for every time the player is unable to do so)
5. The score is calculated based on time taken to complete a level and number of bricks that remain (fast time with few bricks destroyed yields a high score)

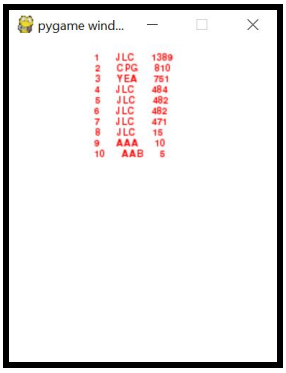
6. Player who hits a “Bomb Block” shall remove a total of four blocks from the screen while only losing points for the value of one block.
7. The top 10 high scores shall be stored in the “High Score” section of the menu.

This screenshot displays the main menu of our game. The diagram at the bottom shows the “Play” page

This screenshot displays the “Directions” page



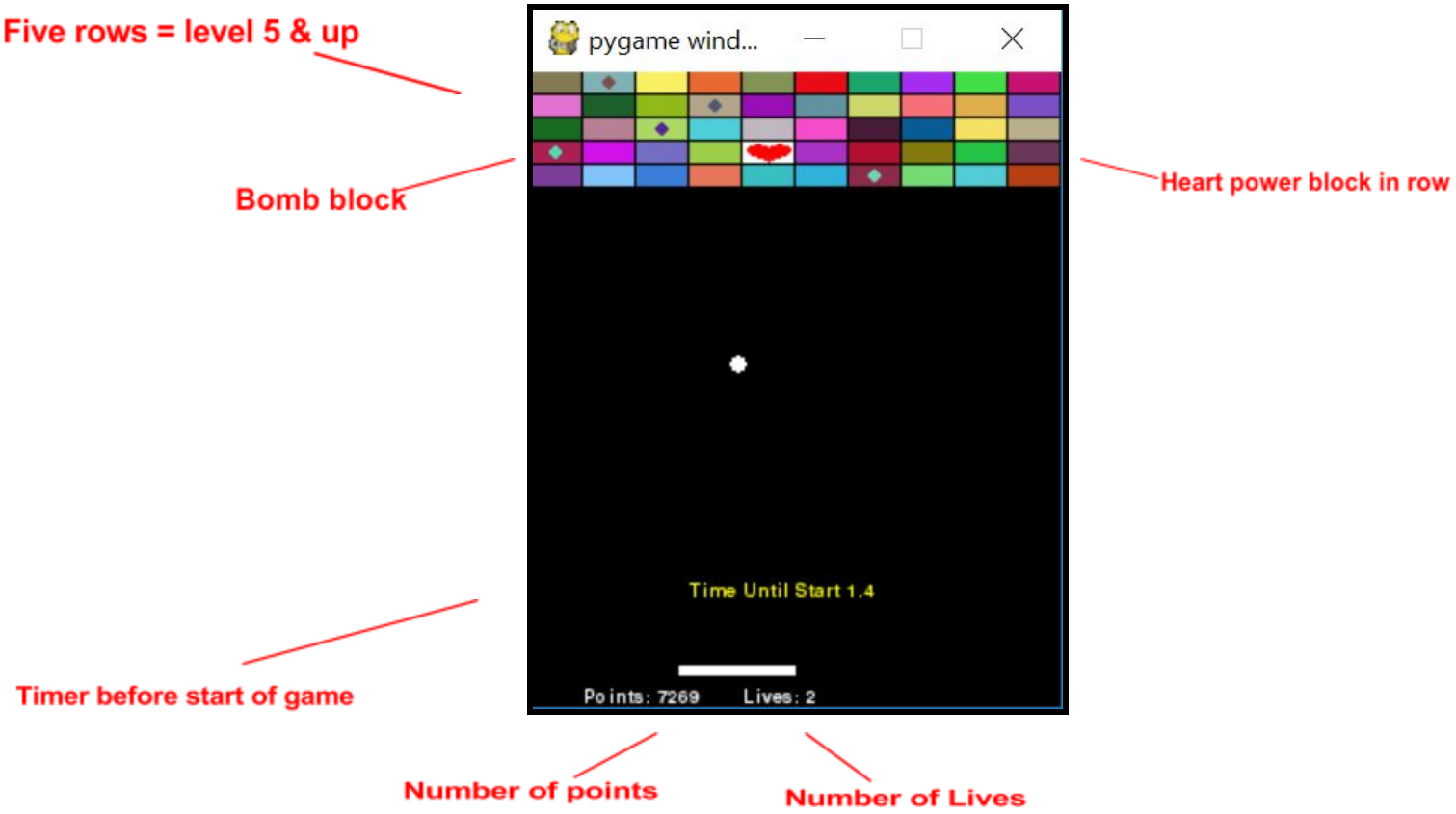
This screenshot displays the “High Scores” page



Our group met on Thursday, December 7th to finish fixing the bugs discovered in sprint 2 and to begin working on the tasks outlined in Milestone 3. We decided one of our three extended features should be a “bomb block” which, when hit with the ball, the block and the surrounding 3 blocks are eliminated. We went about implementing this feature by drawing a circle inside the block with the color that is opposite to that of whatever block it is in. We decided that in any given row there should only by 1 bomb and that each block in the row should have a 20% chance of being a bomb block so long as another block in the same row is not already a bomb block. The next extended feature we implemented was that of the life power-up. Similar to a previous “starlord” homework assignment we had in our C-200 class we were able to download an image of a heart from the internet and place it inside a block on the screen using pythons “blit” function. We ran into trouble with the size of the object and so we had to look up how to resize the image to fit inside the block. When the block containing the heart image is hit the user’s amount of lives will be increased by 1. We gave the probability of there being a life block in each level only 5%. That means that any given block has a 5% chance of being a life

block. However, we only allow for one life block per level so if there is already a life block assigned to a block then the subsequent blocks have a 0% chance of being a life block. The final extended feature we included was sound effects because after all what's a video game without noises? We met on Tuesday, December 12th to work further on the sound effects. We did so by using the pygame.mixer.Sound function and pulling from a flat txt file. We were able to find sound packages online which were created for games like ours. We then worked on little ways we could make the game more appealing and interesting for the user. We put our finishing touches on the project and committed the project.

Game Play Screen



List of Features

- Brick layer on top of screen
- Player paddle controlled by arrow keys
- Random initial placement and direction of ball
- Detection of ball hitting ceiling meaning level passed
- Detection of ball hitting floor meaning level failed / game over
- Change of direction when side walls are hit
- 5 way paddle direction split
- Removal of bricks when hit by ball
- 5 game level progression
- Level variation between different levels
- Scoring function
- Top 10 score list saved in txt file
- Main menu
- High score page
- Directions page
- Number of lives
- Timer
- Reset game after level complete / live lost
- Bomb brick
- Life brick
- Sound Effects
- Colorful menu
- Rick and Mortaaaay

Overall this has been a very valuable project and we are happy you chose us to design it for you. We were able to learn a ton about pygame and the certain nuances of of the module. There were many things we learned that we could do with pygame such as the size requirements and how to implement sound into a game, how to change different object's colors to make the game more visually interesting, and how to move objects around on the screen relative to other objects positions. There are numerous more aspects of pygame we came to learn however these were just a few of the more challenging ones. If we had more time on this project we as a group discussed implementing different power-ups similar to the extra lives and bombs. If granted more time our group would focus on implementing a "laser" power up which, when the laser block is hit, would allow the user to shoot some of the blocks using the spacebar. We also discussed creating an "Acid" ball power up which, when the acid block is hit, would turn

the ball a green color and would be able to go right through a certain amount of bricks. The most challenging part of working in a team is communication. Without extensive communication people are bound to work on the same parts of a project, completing them in different ways, and ultimately creating an inefficient team. We as a team have no suggestions to the software process however we would like to acknowledge that we enjoyed the freedom we were given to build our game how we wanted, when we wanted. Forcing us to come to specific team meeting times would not of been productive since we were diligent about meeting on our own.