For our final project game, we extended the 2D game that we created for assignment 7. This game is similar to agar.io in which players control a circle and consume smaller circles to become larger.

Rules:

1. The controlled circle can increase in size by coming into contact with smaller circles and absorbing them.
2. Smaller and larger circles move about the screen independently of the controlled circle.
3. If a larger circle comes into contact with the controlled circle, the game ends.
4. If the controlled circle consumes all other circles, the game ends.
5. All circles move at the same speed.
6. If a circle comes into contact with a small, red, stationary circle, they gain red circles for a limited time that revolve around the circle and slowly reduce the size of enemy circles that come into contact with them.

The W, A, S, and D keys are used to move the player circle.

The P key is used to pause the game and unpause the game.

The way by which the player understands the game state is the size of the player circle relative to the other circles on the screen.

There is an in-game GUI system that shows how many smaller circles the player has consumed for that round, and how many total rounds the player has gone through (i.e. how many times they have won or lost).

The pause screen GUI shows a list of high scores from the player’s previous attempts. The high score list is implemented through the getHighScores function which uses an XML file to return saved scores and the addScore function which adds the users score to the XML file every time a game is finished.

The player wins when they have consumed all 50 circles that spawn at the beginning of the round. When this happens, a victory screen displays, showing how large the player circle is. The lose condition is when the player circle is consumed by a larger one. A lose screen then displays, also showing how large the player circle was at the time of loss.

The player hears a sound when they consume a circle or are consumed. Press the Q key to mute and unmute the sound.

Small, red power-ups are generated randomly on the screen every few seconds. Circles can obtain the power-ups by consuming them the way they would normally consume other circles. Power-ups extend the Circle class. After consuming a power-up, the circle will have 2 small red circles that revolve around the circle (animation hierarchy) for a limited time that slowly reduce the size of other circles when in contact with them.

The player can press the spacebar during the game to use another power-up that increases their speed for 1 second. This power-up has a cool-down of 8 seconds before it can be used again. A GUI for this cool-down is included in-game at the top center of the screen. When the white bar is full, the power-up can be used.

The classes include the player’s circle and the randomly generated enemy circles. Player is a subclass of Circle since it includes all the same functionality, plus the controls WASD for direction. Functions of Circle allow the object to consume smaller circles and to be destroyed by larger circles upon contact, and to move about randomly. The Circle class also has functions that turn the power ups on and off and determine whether or not a circle is being touched by a hostile power up.

Jack implemented the motion and consumption functionality, power ups and power up GUI, and high score functionality. Braxton wrote win/lose conditions and worked with Jade on the win/lose screens. Braxton also implemented the restart game and pause screen functionalities with the high score list. Jade created the main in-game GUI and implemented the sound, as well as improved the aesthetics of the game.