Brick Breaker, by Lisa Hachmann and Maggie Jakus

Project Overview

For Mini Project 4, we decided to make Brick Breaker, a classic arcade game. We did this in Python using Pygame and Sprites. The game has 5 levels, during which the user uses a paddle to control a brick at the bottom of the screen to hit a randomly positioned ball that destroys bricks that are randomly positioned and colored. We used several classes to represent the level, ball, paddle, and bricks.

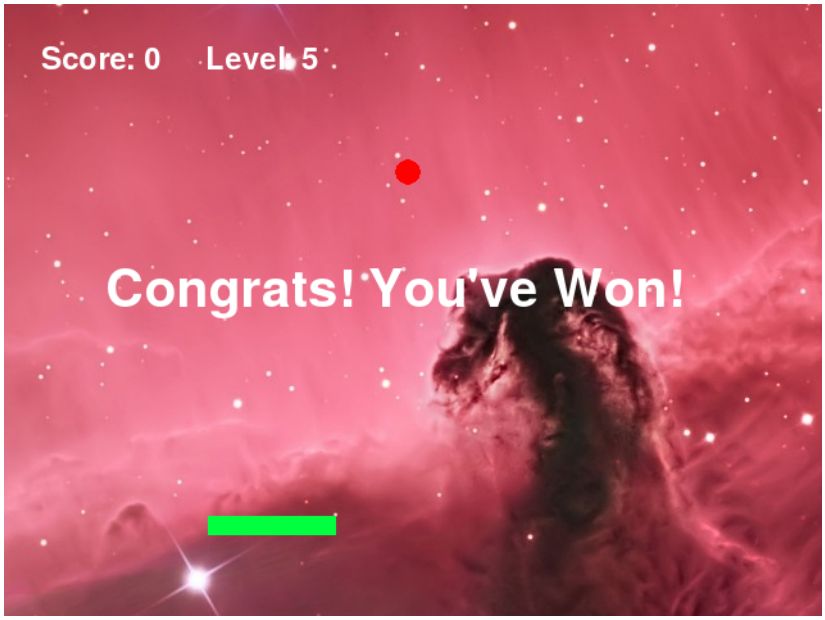
Results

We were able to successfully create the game of Brick Breaker. The user operates a paddle at the bottom of the screen to hit the ball, which bounces off the paddle to hit the bricks at the top of the screen. When the ball collides with a brick at the top, the brick disappears and the ball continues to bounce. Once the user has cleared all of the bricks from the screen, the user progresses to a new set of randomly positioned and colored bricks. This occurs for five levels.

We have a soundtrack in the background (the Star Wars soundtrack) and update the score as the user plays. Every time the user breaks a brick, the score is increased by one. The score resets for every level. We wanted to have sounds for breaking the bricks, but we didn't have enough time to do so. We also ran out of time to add specialized bricks (bricks that would take multiple hits to break or bricks that would release a second ball when hit).


Score: 4
Level 1

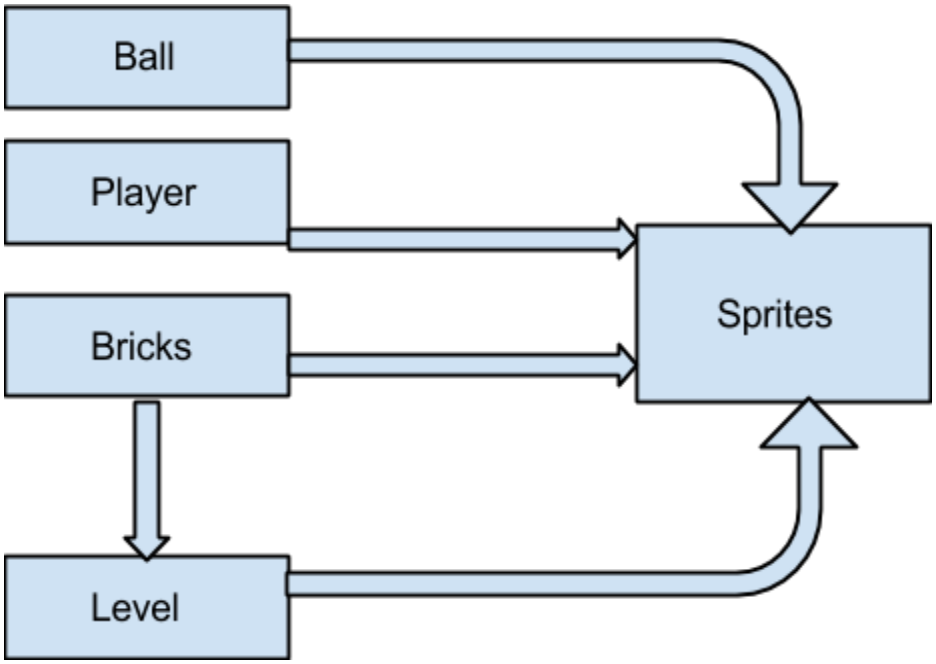

Score: 8
Level 1



Implementation

We created classes for the paddle, the ball, the bricks (that need to be broken), and levels. We chose to make these all sprites. Going into this project, we did not know what sprites were, but we had previously heard that they would be useful for this project. We decided to use sprites, and overall we think this was a good decision. We decided to not strictly use a model-view-controller method for our program. In retrospect, this was not the best decision. However, we did not well understand this concept, so when we began this seemed like the best idea. It made creating our program more difficult and messier.

The program searches for events -- collisions between the ball and the paddle and the ball and brick(s). If the ball collides with either the paddle or a brick, an event is triggered and a collision occurs. Once all of the bricks have been cleared, the level is reset and a new set of randomized bricks are created. This runs for five levels, unless the ball goes off the bottom of the screen.



Reflection

We thought the project went very well because we scoped it appropriately. We were able to create the basis game early enough in the week. This allowed us to be able to add new features and adjust our goals. For example, we decided to add music and a score for each level, as well as randomly colored and positioned bricks. For unit testing, we always tried just adding one new item to the screen at a time. We began by simply creating an image, rather than starting with fully moving and colliding sprites. Looking back, we wish we had understood objects and classes better before we started the project. We also wish we had understood the model-view-controller method earlier than the day before the project was due so that we could have implemented it. Going forward, we both have learned these topics to an extent, and have a bigger base plate of knowledge of objects, pygame and using classes to work from in the next project.

For our team process, we liked to do pair programming and switch off who was navigating and who was driving. We would also switch who was trying to implement something new into the code on their own, and normally we met later to fully implement and debug that code. We met very often in order to create Brick Breaker, but due to the length of code we had, pair programming was the best option and it worked well for us.