MP4 Reflection

Alex Wenstrup, Thomas Jagielski

March 30, 2019

Software Design

Project Overview

For MP4, we decided to make a simplified version of Super Mario Bros in Python. To do this, we used the pygame library, and images of sprites from the original Super Mario Bros found on Google Images. We then built a simple physics engine to model Mario's movement, and added a number of objects for Mario to interact with. We feel that our implementation is not only functional, but smooth and playable.

Results

Our game successfully implemented most of the key aspects of the original Super Mario Bros. We were able to add goombas, pipes, and bricks which Mario can interact with, as well as clouds in the background. Mario's movement is smooth (except for a few edge cases which rarely come up), as is that of the Goombas. Mario is also blocked in the correct direction by both pipes and bricks, which was difficult to implement using pygame.

Figure 1: Mario jumps off the ground, about to land on brick

While we had hoped to add more features, including mushrooms, koopas, more decorations, and even a settings menu, we feel that what we were able to include made our project not only fun and playable, but a challenge.

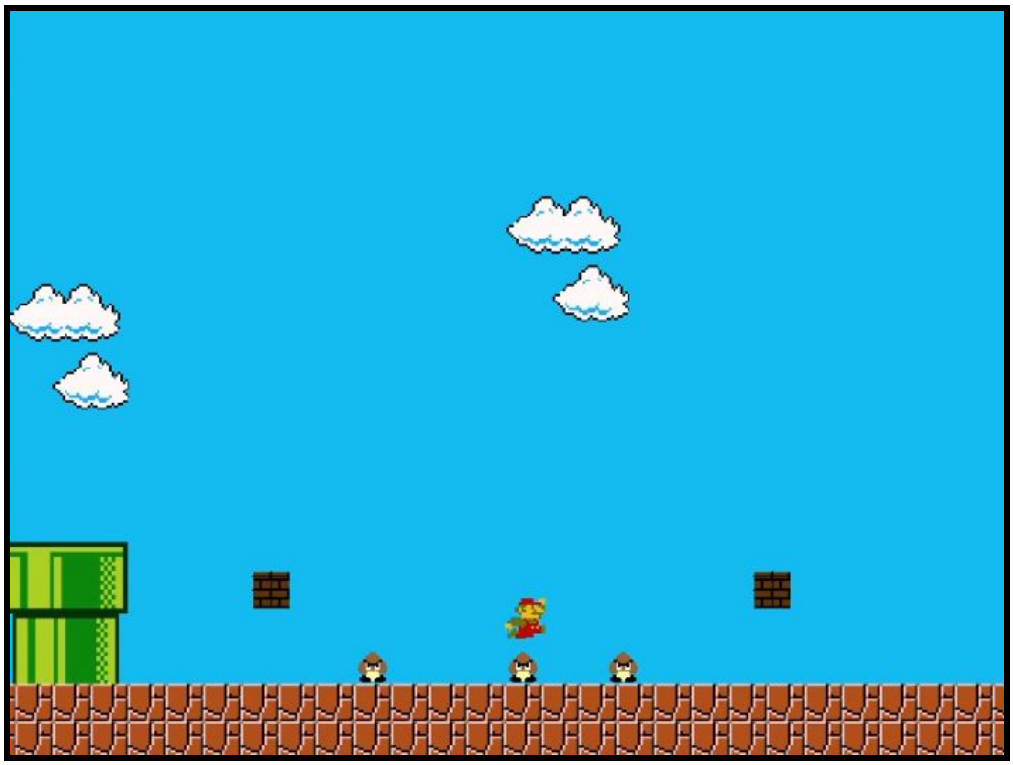

Figure 2: Mario about to land on a Goomba

Implementation

Because our game was so complicated and has so many unique objects, we were very careful about our class hierarchy. This was very useful for us when thinking about and organizing our code, however we didn't actually use most of the features of inheritance. For example, our code contains many redundant functions, because rather than creating functions that can take different parameters, we created near copies of those same functions to save time while coding.

If we had more time, we would have cleaned up a lot of those issues.

Another tricky part of our program to implement was the interaction between different objects. Pygame can tell you if two rectangles are overlapping, but it won't give any information about which sides are overlapping. Furthermore, each object's location is represented by a single point, which made testing for the location of overlap even harder. That said, by using a combination of position and velocity, we were eventually able to get Mario to interact with both bricks and pipes properly.

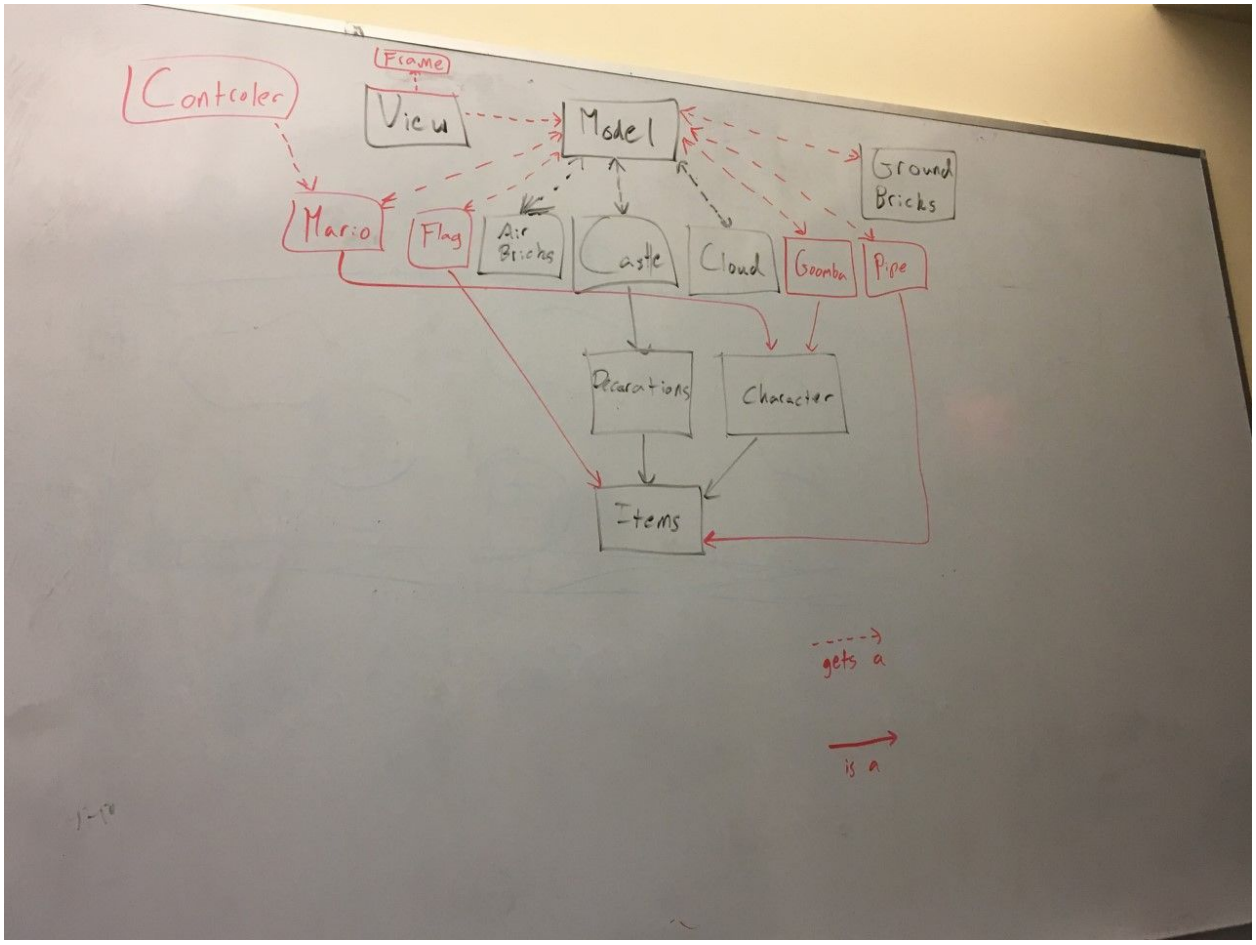

Figure 3: Class Diagram

Reflection

I (Alex) think this project went really well. Thomas and I both started working early, and made a point of having a lot done by the beginning of spring break. Our project was easily scalable, so we were able to put in work the whole time without worrying about running out of things to add. One tool I had barely used before was Teletype for Atom, which allows two people to simultaneously edit code, and it proved to be really helpful in our project.

The project we created was very successful. Although we learned much during the process, I (Thomas) think we could have implemented the ideas of inheritance better. In the future, I would like to use inheritance more so I further my understanding of it. Prior to beginning, I wish I would have taken advantage of the documentation for pygame in the beginning. In doing this, I think we would be able to take advantage of more of the pygame functions. With respect to how we programmed, much of it was done as pair programming with Teletype in

Atom. This allowed us to only edit the files on one computer at ones and avoided merge conflicts. This proved to be a successful method of working for us during this project.