Name of Project: FlappyBird

Team Members: Ashley Weaver, Jason Husted, Tristan Hunt

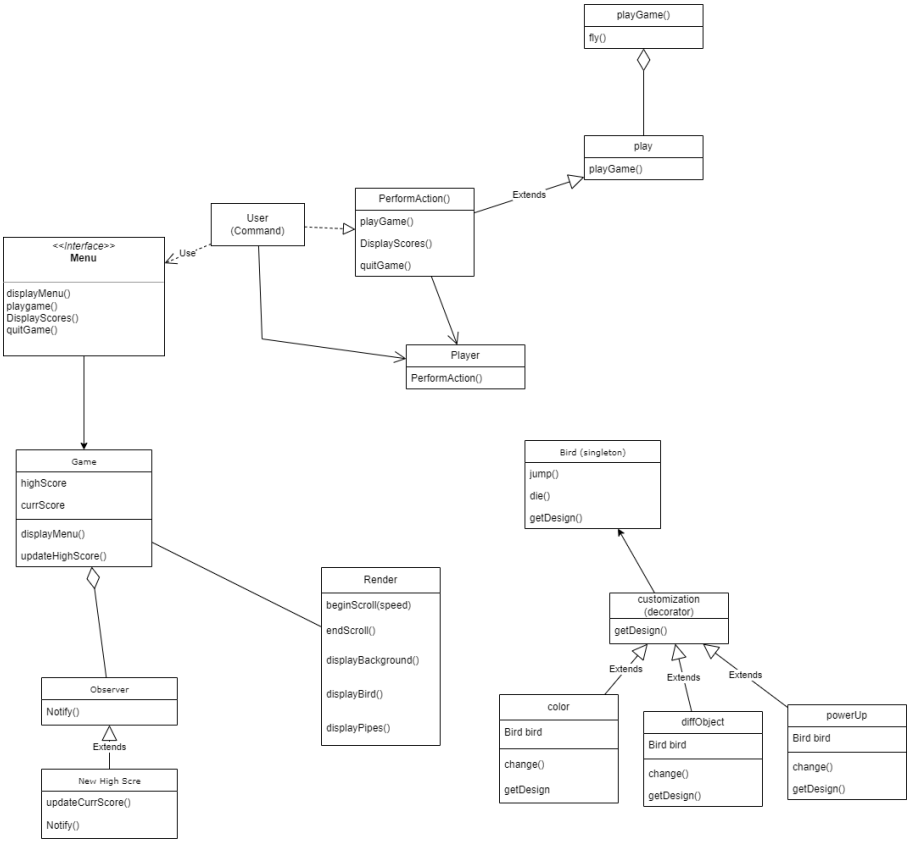
Final State of System Statement:

Our game features a fully functional Flappy Bird game. The game starts off with a start window where the user can choose to either start, options, or exit. If the user presses options, they can change the landscape they are playing on or the color of the bird. Once a selection is made, they can play the game. The game is a bird that has to fly through a series of pipes. The pipes are randomly placed to make it more difficult for the player. Every time the player/bord dies, they have the option to retry. Their most recent score and their high score is displayed at the top of the screen before the play each time.

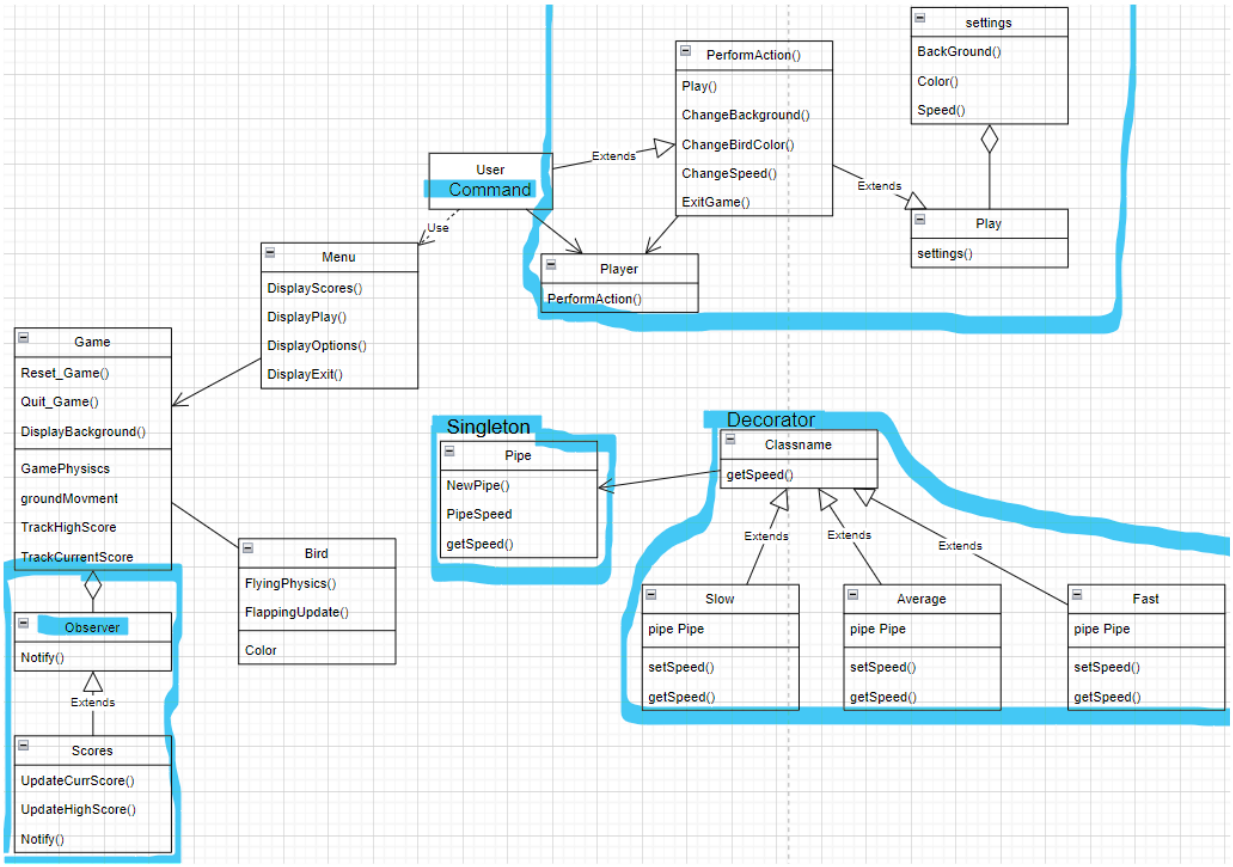
Like we said in our meeting with the TA, the biggest change was the coding language we used. We tried to use Java as our game coding language, but it proved to be too difficult. We then made the decision to use Pygame in python which ended up being a lot easier an fun! It took a little bit of time to understand Pygame because this was our first time working on it, but we figured it out.

Final Class Diagram and Comparison Statement

Initial UML Diagram



Final UML Diagram



There are some key changes that were made from the time of our project 5 to our final project. The Biggest change would be changing the singleton pattern and decorator pattern from the bird to the pipes. It was better to perform the color changes of the bird in the command pattern so we decided to use the decorator pattern for the change of speeds in the game. For the singleton, it could have been implemented either to the pipe class or the bird class; we decided to just use it for the pipe class for the sake of time. Otherwise, there are not that many changes but instead just making each class a little more complicated or adding more functions to the classes. A few examples of this is in the game class. When we first made the UML diagram for project 5, we din't think that much would have to go into making the game. Another example of adding stuff to the classes would be for our command pattern. We went along and added more things the user could do in the game, so we had to reflect that into our final UML diagram.

Third-Party Code vs. Original Code Statement

As stated in the ReadMe file in our github repository, our base code for the game physics, game, bird, and pipes were provided in a tutorial by Coding with Russ (github id: @Russs123). All of the images used for our game were also provided by outside sources (links/citations are in the ReadMe section of our repository).

Everything beyond this was original code made by us. We built off the base code to make a much more complicated and robust game that featured a menu, different bird options, different landscape options, OOAD design patterns, and more.

Statement on the OOAD process for your overall Semester Project

1. Coding Language: Deciding on the coding language that we wanted to use defintley was difficult. We initially agreed that we should stick to a coding language that we had all been working on the entire semester (Java). After hours of trying to set it up, it ended up being really difficult. We made the decision to switch to python and it has been way easier (it wasted a lot of time though trying to figure Java game development out)
2. Scheduling: Since all of us are upperclassmen/graduating students, this made trying to meet pretty difficult. Luckily we all were really communicative over text despite not being able to meet in person that much.
3. Code Structure: We came up with a pretty good UML diagram before we started coding, but we ended up having to change it a bit to fit what we actually ended up doing. It was really helpful though.

Presentation Recording Link

https://drive.google.com/file/d/1ZAb_hCOygWq0N0pekuqE0_5HB7BqXclB/view?usp=sharing