Philip Schiffrin
pjs2186
5/17/2015

The program Co-Op-Pyration is a template for a game mechanic where one player dynamically creates a board of blocks and passes it to a second player, who tries to maneuver his way through the board. The board is passed by a server program that simply waits for two players to connect and makes a thread for those two players. The thread tells the two connections who is the player and who is the board maker and passes the board from the maker to the player. Once the player is finished, they are prompted to play again. The maker exits.

The game also has single player functionality where, if the program cannot connect to the server, then it prints an error message and creates a board at random for the player to play. If the player collides with a block then they are prompted to play again.

The server's ip address is hardcoded into the program due to a difficulty I ran into getting the local ip of my ubuntu server dynamically. Therefore, in order to run the program with the server on your local machine, you need to go into the program and actually change the ip address in the function connectToServer(). Once this ip address is changed, you should be able to start up the server, start two instances of the program, and pass a game board from one instance to the other.

The main functionality that is not implemented is the continual passing back and forth of the game board. The hope was that once the player finishes playing the game, they will then take on the role of the game maker and pass the board to the other player, who will get a chance to play. It is clear, however, that this functionality is possible within the given framework. Other functionality that I had hoped to implement includes a database for player statistics, a user profile for any given player, and better gameplay (something other than a flappy bird clone).

The most exciting thing about the project was how easy it was to implement. The pygame library provided an easy way to create shapes with images on top of them, get events from the user, process those events in a game loop, and update the objects on the screen, all of which make up the basic functionality for a game. My hope was to provide a proof of concept to myself that this dynamic game making model could work and I believe, under that criteria, it was a success.

The least exciting this about the project was the possibility of scaling and releasing the game for a wide audience. My hope was to be able to, if not release something downloadable by the general public, then at least host it online. While I do think this is possible, python does not seem to provide an easy or robust way to manage this kind of release. For mainly this reason I believe that, if I were to attempt this again, I would use either Java, C++, or Unity.