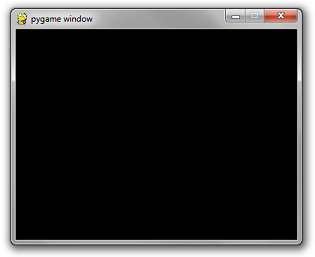
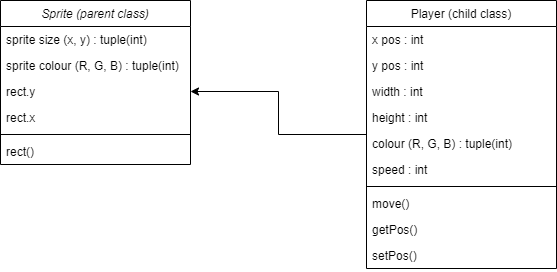
**Project Design Document – Joseph Henry**

**Prototype 1 – (The multiplayer)**

**Problem decomposition**

1. Create a client program that will act as the main game loop. This client should define and render a basic display for the game to be displayed within.
2. Create a new file that will run a local server. This file should open a new socket that will listen on a given port for connections. This will then create a connection on a child socket with the client. This connection will be used to send data between the local server and the client.
3. Create a new function that will take in an ip address and port of the server wanting to be joined. It should then take this information and attempt to send data to the requested server address. The game server should show the connection to the host so that they not only know that someone has connected but also who it is (via their ip).
4. Create a parent class of sprite and a child class called player. This will be used to define the character each client will be playing as, initialise them onto the screen and control all the movements. The parent class will also be useful later in the generation of platforms.



In pygame there is a default class for a sprite, I have used this to create my player class.

1. The local server should be able to hold data from clients in the form of an object.
2. The client should ask the network function to send its player’s objec data to the local server. The local server will then send back the other player’s position to the original client. This will allow for both clients to have both players positions at all times which means both clients can keep track of the other client’s movements respectively and show them on screen by updating after each communication with the server.

**Network Diagram**

Client 2



Client 1

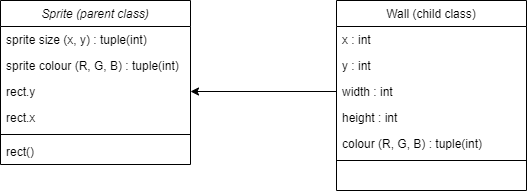
Switch



Local server host

**Prototype 2 – (Spawn zone and physics)**

**Problem decomposition**

1. Create better suited movement controls. I will do this by implementing jumping and removing the up and down motions previously used (get rid of: w to move up and s to move down and replace with w to jump).
2. I will then need to make a new class called a wall. This will act as the platforms and the floor/roof/walls.
3. Make the client render a spawn area on the game window by creating a border around the screen out of the new wall objects.
4. Implement collision between the player and the wall objects.
5. Add gravity to the player so that they do not float but rather fall to the nearest platform/floor below them.

**Prototype 3 – (Map generation)**

**Problem decomposition**

There should be a simple title screen with the name of the game and a few buttons for the users to interact with including a start server button and a join server button. This is to allow for players to be able to choose when they wish to start a match rather than being thrown straight into a game. In addition, the map should still generate the same spawn area; however, there will also be procedural generation to make an infinite yet diverse map for players to run through on each play of the game. This will increase the replay ability of the game for its users.