1. **Topic**: What is your project? If it is a game, what is the genre and objective? If it is an interactive visualization, what are you visualizing? If it is a tool/application, what does it do?

We are building a super smash bros game that mimics the real videogame. We plan on using less characters in our game. The objective of the game is to knock the other player's character out of the map. This will be done with attacks from character to character and a health bar to control how much knockback each hit does on a character. Thus, overtime the hits become more and more likely to knock a player off the map. The boundaries of the map make a rectangular shape around platforms for the characters to move on. The movements will be determined by the key pads. We will use the arrow keys for movements and some letter keys for attacks/defense. The map is in 2-D. There will be extra platforms besides the main one to allow characters to jump and add complexity to the game. We are visualizing the 2-D map characters inside of it. It's an interactive visualization with characters colliding and platforms on the map holding characters up.

2. **Interactivity**: How does a user interact with the project?

The player uses keyboard inputs to control their character and fight the other player. Our goal is to be able to do this over LAN in which case we will have one computer running the model and two computers which act as controllers and then receive the updated view to be displayed. We will design the project such that it can be played either with two people on one computer (in which case one player would use wasd and another arrow keys) or fully over LAN.

3. **Model**: What information represents the "state" of your visualization, game, or tool? How are you storing this information in the relevant class(es)?

We have an overall game class which represents the overall state of the game. This has attributes such as what map it is, what the characters are, if it's over LAN or local, and it is split up into Character classes as well as Map Classes. The character class is an abstract class which has subclasses with the different character's attributes such as weight, speed, attacks, abilities, attack damage, ect. The Map class also has subclasses that inherit from it which have different types of obstacles/platforms.

4. **View**: What information are you displaying to the user at any given time? What attributes/methods will you use to compute and/or display this information?

We will have a view class that takes the current Game class as an input and uses pygame to draw it to the screen. This will likely be broken up into drawing the stage, drawing the players, and drawing other text/information on the screen (such as damage, stocks remaining, etc.

5. **Controller**: How is the user providing input? What inputs translate to which methods in the model class(es)?

We have an abstract controller class that updates the character. A local keyboard controller determines key inputs and calculates the logic to update the character. A network controller gets a fully updated player over the network and then updates the client side version of the network player.

6. How do the components of your software work together to achieve its functionality?

All the classes work together to create the information needed to display the game and to actually create the display of the game. The controller takes the keystrokes and updates the client player. Then it sends information to the network and receives information from the other client. They both then change all the attributes respectively to represent the state of the game's attributes. These attributes would then be used to visualize the game.

7. What are the risks or unknowns that you may face as you work through your project, and how can you address them?

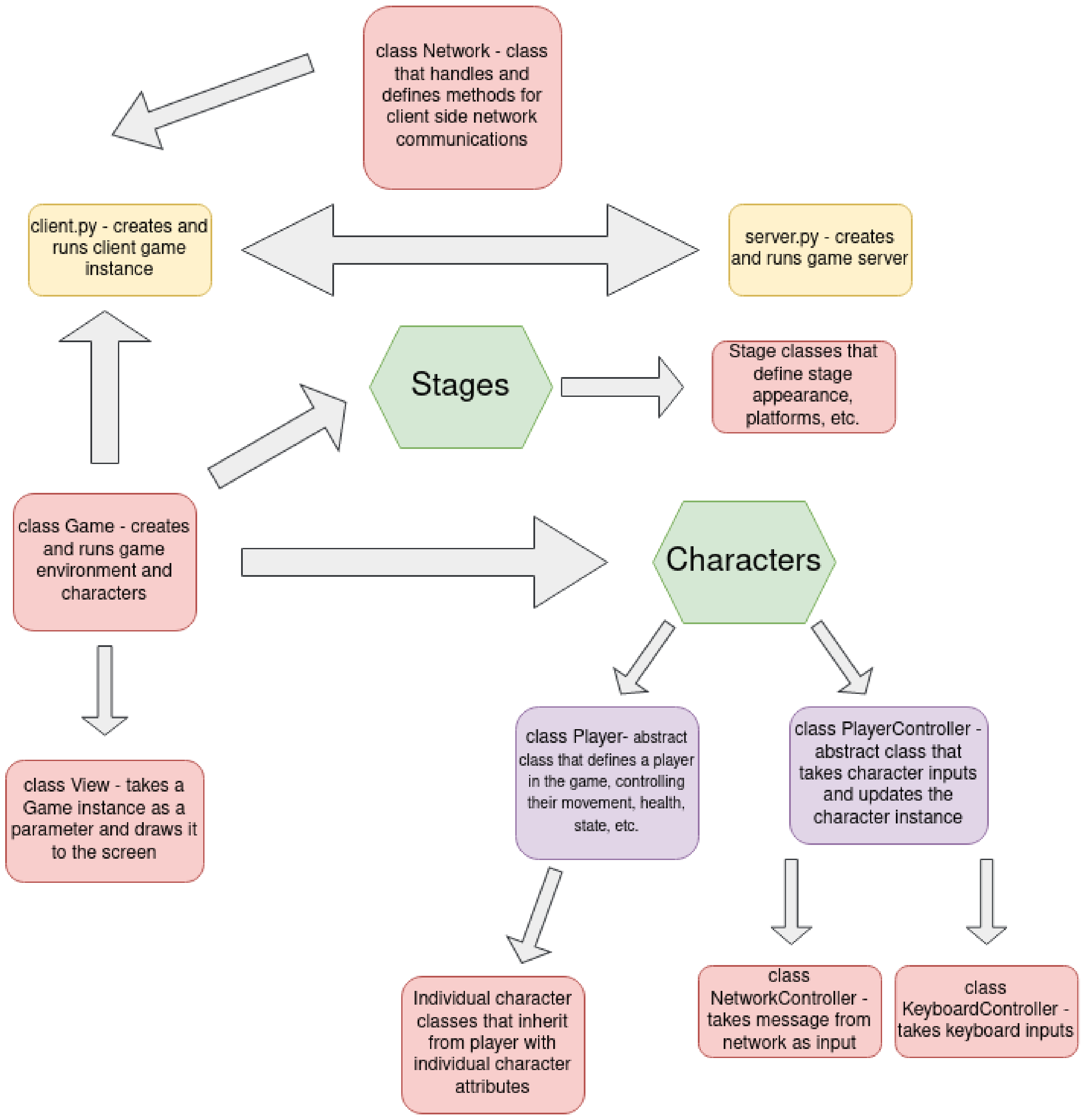
We are taking on a fairly ambitious project in areas most of us don't have a ton of experience in, so there is a very real chance we over scoped this project. To address this, we are spending time up front designating what different goals are, including an MVP and further stretch goals, and structuring our code architecture in a modular way that allows us to start at an MVP and make add-ons to that while being able to stop and have a working product at any point..

8. What are the primary technical problems that you will tackle in this project, and how are you approaching those problems?

The primary technical problem we believe we will encounter is playing the game over a network. We are trying to figure out what information to send and how we will receive the information. We are currently thinking of sending a message with player attributes over the network and using that as an input to a NetworkController that updates the Player Class on each client.

9. How do you envision the user interacting with your software, and how does the design of your project facilitate this interaction?

This game will be keyboard controlled by each client, and there will be a keyboard controller class to facilitate this interaction. As a potential long-term stretch goal, controller input could also be implemented as a separate controller class.


class Network - class that handles and defines methods for client side network communications
client.py - creates and runs client game instance
server.py - creates and runs game server
Stages
Stage classes that define stage appearance, platforms, etc.
class Game - creates and runs game environment and characters
Characters
class View - takes a Game instance as a parameter and draws it to the screen
class Player- abstract class that defines a player in the game, controlling their movement, health, state, etc.
class PlayerController - abstract class that takes character inputs and updates the character instance
Individual character classes that inherit from player with individual character attributes
class NetworkController - takes message from network as input
class KeyboardController - takes keyboard inputs