Github link: https://github.com/olincollege/pysmash

1. Changes: What changes did you make in response to feedback from the architecture review? What other design or implementation decisions have you changed since the architecture review?

We have made some changes to our code architecture based on feedback from our architecture review. Per our instructor's suggestions, we have changed our possible network implementation from having the game run locally and having a Network Controller control the second player to having a GameState class that would be managed by the Server and updated by any clients. This, however, remains a stretch goal for us as we work to get the bulk of the main game done.

2. Implementation: What parts of the implementation still need to be done? What is the plan and timeline for finishing the remaining pieces of the implementation? What barriers still exist to completing the implementation?

To this point, we have the basic physics simulation of the game's movement essentially finished - the player can run, jump and interact with the stage and platforms. The biggest thing we have left to implement are the attack/damage system (and the knockback that comes with it). Once we are able to get that done, however, then we have essentially all the puzzle pieces we need for a barebones version of PySmash. We are hoping to get this done in the next two days to hopefully stay on track for our stretch goal of playing this game over the network.

3. Hurdles: What bugs or technical problems have been particularly tricky to tackle? What are some possible approaches you could take that would help you make progress in addressing these bugs or problems?

So far, getting the movement of the character to look right has been the trickiest portion of the project. We tried a few different ways of moving the character around, but ultimately settled on a physics-based system with velocity, acceleration and friction. The knockback has been particularly tricky but we have a few equations that we found online and we might include them or try to make our own calculation for knockback. For the bugs print based debugging has been extremely useful. It's also been useful to see how the game behaves in the window we set up for the game. Only Amit's computer works with pygame so we have had him testing those parts of our code but moving forward we can now use replit to do some more in-depth testing ourselves as well.

4. Tests: What parts of your code have been unit tested and what parts have not? How might you test any yet-untested code for correctness?

To date, we haven't done much unit testing as we work on the game's main motion framework. Some basic testing might be to drop the player on top of a platform and ensure they don't fall through, or that they can't have more than two jumps. However, we suspect the bulk of the unit testing will come into play when we integrate damage and knockback into the game, and making sure those values that are calculated are correct.

5. Code Quality: To what extent is your code readable by someone not on the project? To what extent is your code's performance suitable for the way in which you intend people to use it?

We believe the code is readable for someone outside of the project because we have labeled our variables with accurate and appropriate names ie. self.direction for the direction the player is pointing. We have also included docstring for each of the definitions inside the classes and the class docstrings as well. The files are labeled in a way to detail what exactly is in the file and what it is doing.

6. Is the code working as intended? How do we know?

The code is behaving as intended. We can make a character move around a screen with keystrokes. We ran into a few bugs, such as one where the character's downward velocity would continue to increase exponentially even though they were sitting on a platform, which essentially rooted them to the ground. Our methodology for solving these bugs has been using print-based debugging and running chunks of code in isolation to try and nail down the source of the issue.

7. How is the code's performance? Is this acceptable for the ways in which it is intended to be used, and if not, what are possible workarounds?

The code runs pretty well so far. We had to tweak the acceleration and velocity to make the movements of the characters smooth and as quick as we want them to appear in the final product. The key inputs are pretty responsive but there is a slight delay that we will try to get rid of. So far we have only done one character, player, and stage, but we have written it in a way that allows us to easily implement expansions. There are no pressing issues with our code's performance, only small adjusts that we need to make like the movement speeds.