**How to use the coded files**

**Demo:** <https://youtu.be/DcfJYrF_yjs>

Launch in order: Server--> Client--> Renderer.

Then use Client(Controller) to show commands with !help.

To clarify ‘filename’ does not include the quotations, you just need to put filename name with the extension.

Server only supports txt files.

If you wish to test the product and see if it truly streams the lines in the txt file. Create separate folders for each of them.

**Protocol**

Client talks to server and ask the server to display contents within the server. The server shows contents. Then the user will be able to play the file. Once “play ‘filename’” is started the user has options to pause or resume the said file streaming to the renderer. Renderer just prints what it receives.

**Design**

When we were in our first phases we decided to keep the implementation close as possible to the requirements. In which, we did. The server, controller, and renderer work exactly how it was explained in the scope of the project, and at minimum the text files were supported.

The reasons why we couldn’t implement video is because we would have to use a module called “cv2” in python. It creates a GUI for video playing and creates stream for video playback. While it would be neat to implement that into our code, we had to cut that part out because the group working on the project only consist of two people: Jonny Le and I (Jethjera Silasant). Also, another reasons for cutting “cv2” is because that means if the TA had to run the program he would need to have “cv2” in Mininet to run it on the node(s). So, we took that into account also.

**How the Topology looks**

Switch