I managed to fix the placement issue of the popup texts by rearranging where the scripts were and how they were structured. The tutorial mentioned before involved making a script for the popup text parent to control the child and a controller script that was static which utilized the Resources.Load() function. I’ve never had any luck with loading resources via script because they would always return null. So I had to make the controller script non-static and create GameObjects to hold references to the Prefab that was supposed to be loaded. I created one for each player because I allowed for the players’ scripts to hold onto references to the GameObjects created and I felt it would be best to allow both player objects to make references that are independent of one another’s active resources.

I tried to extend the popup texts’ use to reflecting the sources of damage they were trying to indicate. This was going to be executed by changing the text color appropriately as well as a damage type keyword being shown in-line with the damage number. Since I have projectile/elemental identification structured based on what types of scripts are used by every projectile, for some reason I’ve been unable to make proper comparisons to those particular types. This was… oversighted… and I’m hoping to find a solution by the time the next week ends, when I will be turning in my final version of this game.

If not, then it is not completely necessary as weeks ago, I also fixed visual effect timing to go along with DOTs and their timings. The visual effects are in fact unique to each type of projectile/element thrown/applied, so in a way those could be fallen back on as the indicators of when a specific type of damage is being applied, but it admittedly is not as directly informative as if the damage number also displayed their damage type beside it.