Nicholas Poole

GAM 350

For my final project, I decided to implement a persistent database to store the information on my server. This was to allow for features such as high scores, non-volatile player accounts, and generally greater stability for the game’s information handling. Overall, a persistent database is extremely helpful for any project that intends to have server-based online multiplayer.

I used MySQL to create a database to store unique values for each player. With that database, any player that leaves the game could rejoin to reclaim any amount of points they had already acquired. Players would input a name at the login phase of the client game. After inputting this data and connecting to the server, the server would place the name into a data table, alongside additional values for the score and identifier. If a player is on a team that wins a round of the game, the players would have one point added to the score value associated with their name; this way, players can compete for a high score on a leaderboard.

The primary issues faced by the implementation were the means of writing in new data for MySQL, and the difficulty in actively using the data within the project itself. MySQL has a very tricky interface to use, with an incredibly frustrating editor and its primary method of editing being done through the command line. This was effectively solved by learning how to use the command line for MySQL, and bypassing the editor entirely by writing the queries into the server’s scripts. The other issue, of easily accessing the table data, proved far more evasive. While it was very simple to access the contents of the first column of data, reaching other columns would result in error messages regardless of syntax, to the degree that even finding answers that worked for other users online seemed to fail with my code. Ultimately I was forced to let this one go, and as such I was unable to place the player’s score as a value on the UI.