**Sprint Report: Milestone 1**

It was decided very early on that we wanted to make the game we choose from scratch in Unity. After weighing the pros and cons of using open source code versus creating the code ourselves, we decided that creating our own source code would ultimately be faster and give us more freedom to customize the game as we please. As a result, we decided to go for Bomberman, a game most of us had played before and had pieces of the code already written from previous projects. The tools we’re using for this project are Google Drive for documents, GitHub, Unity, and Visual Studio for source code, and Discord for communication.

**What we planned to do**

Our initial version of single player Bomberman involved 2 players, one was controllable and the other was a dummy static player. To win, the player had to kill the dummy player and be the last man standing. In terms of movement and overall level layout, we decided that grid-based levels and movement would be consistent with the older Bomberman games. Typical controls would use the WASD or arrow keys, and the Spacebar or Enter key for deploying bombs. In terms of gameplay, we decided against adding too many features because of the amount of time we had. As such, the only power-up we wanted for the initial build was additional bombs. In terms of aesthetics, we wanted to focus more on the game logic and preparing the code for future milestones so we didn’t add too much. But we had planned to add a camera effect that zooms in and out based on where each player was on the screen.

**What we accomplished**

We accomplished a majority of the game features we wanted in the initial build. This included grid-based levels and movement, multiple bombs, and player controls. We imported a map generator script that one of us had already previously wrote and used that to generate our grid-based maps. However a bulk of the workload was focused on preparing the source code for future milestones. Part of this preparation included creating our own Input Manager that would manage multiplayer controls better than Unity’s built-in manager. We also imported a Window Manager script that one of us previously wrote to make switching between UI windows and HUDs easier. In addition to Bomb, Explosion, and Player scripts to supplement the game logic, we also created a Game Manager that keeps track of victory conditions. To anticipate for multiplayer support, we decided our clients were going to be thin clients, and so the Game Manager was written to be what our server would eventually be in future milestones.

**What we didn’t accomplish**

Some of the things we decided not to include in this build were dummy players and the camera script. The dummy player was scrapped because we realized we wouldn’t be using it in future milestones so there was no motivation to write scripts for it. As a result, our victory condition devolved into killing yourself to end the game. Additionally, the camera script was a low-priority feature initially, and time constraints dissuaded us from adding it in for this milestone. Lastly, our game manager doesn’t yet control the timer or keep track of the players.

**What we plan to do next**

Apart from what future milestones require, aesthetically we plan to add sprites to represent players and bombs. If time allows, we’ll also be adding sound effects. In terms of the game manager, we plan to have it control the round time as well as keep track of the players as well. Additionally, we plan to look into ways to add networking into the game without using Unity’s built-in networking capabilities. We also plan to add in a simple SQL database to keep track of the player’s progress. The login window in the initial build was made with this database functionality in mind.