**Your choice of game?**

Bomberman.

**How is the game played?**

The game will be played with the keyboard, using the WASD keys for movement and the spacebar for deploying bombs. Competition will be done with 4-players through a network connection. One laptop will serve as the server while the other laptops can connect to it. Whoever kills the other players will win.

**Game objects**

|  |  |
| --- | --- |
| Player | The player can move in all cardinal directions and can place bombs only on walkable tiles. It can use its bombs to destroy destructible walls and kill other players. When an explosion touches the player, it gets destroyed. |
| Bomb | The bomb can only be used by players. It explodes and turns into an Explosion after a couple of seconds. Its power can be increased when the player picks up a power-up. |
| Explosion | The explosion detects players and destructible walls if valid type is destroyed and replaced with basic tiles. |
| Walkable (Basic) Tile | Walkable tiles are spaces on the level where players, bombs, explosions, and destructible walls can spawn. |
| Destructible Wall | Destructible walls are stationary and can be destroyed by explosions. Sometimes they will drop power-ups when destroyed. |
| Power Up | Currently only increases the range of the bomb’s explosion. Will implement more power ups in the future. |
| Indestructible Wall | Indestructible walls are stationary and can’t be destroyed by explosions. |

**High Level Architecture**

* Dedicated authoritative server
* All game logic will be processed on the server
  + Server will screenshot the current game state then send the image over a UDP channel, streaming the game state to the client
  + Initial version will use Application.CaptureScreenshot to capture a PNG
    - Second version will use a Render Texture to decrease the time to capture screenshots and convert them to a byte array
    - Final version will check for differences between the previous image data and the new image data, and send the differences between the two over the network. This will decrease the load on the network to improve overall speed
  + Image is converted to a byte array, time stamped, and sent across the network
  + Server will send additional messages to control audio cues, which are then processed by the client
* Thin Client
  + Inputs received by the client are written to a message which is converted to a byte array
    - This byte array is send to the server over a UDP channel to the server for processing
  + Images are rendered to the client as soon as they arrive, unless the time stamp on the image pre-dates the last rendered image. In this case, the older image is discarded.
  + Audio cues sent by the server will be processed and played by the client
    - Audio cues will be queued up on the client side, and processed when the incoming image timestamp aligns with the audio cue time stamp

**Persistent User Information**

* User data will be stored in a MySQL database using XAMPP to access the loopback address at 127.0.0.1
  + Stored data will consist of a username and password
* Communication between game and database was done using php
  + Log in
    - Unity sends the username and password to a php script using the WWWForm library
    - Php script receives the username and password and converts it into a SQL query
    - The php script verifies the username exists and if the passwords match and sends back either a “valid” or “invalid” if the login was successful or not
  + Account Creation
    - Unity verifies that both passwords (normal password and verify password) are the same before sending them to the php script
    - Another php script then converts the username and password into a SQL query and inserts the new account into the database