Names: Corey Nguyen, JiMin Lee, and Warren Urbina.

Network Communication and Protocol:

(ALL the messages are sent to the server from the client then echoed back to the rest of the clients so that they can update the game state on their client)

- Player Movement - a message is sent from the client to the server in the form of:
  player_index(x,y,z)<EOF>
  where player index is a number from 0 to 3 indicating which bomberman each player is, and x,y,z are the coordinates.
- Bomb Placement - a message is sent from client to the server in the form of:
  player_index,B(x,y)<EOF>
  where B denotes that a Bomb has been placed on a client.
- Death - a message is sent from client to the server in the form of:
  player_index,D,<EOF>
  where D denotes that a player has died.
- Disconnection - a message is sent from the client to the server in the form of:
  player_index,X,<EOF>
  where X denotes that a player has disconnected.
- Player Connection - a message is sent in the form of:
  player_name has successfully connected<EOF>
  where player name is the name of the player that logged in.

Game Object Relationships:

Bomberman can place Bombs.
Bombs explode and Spawn Powerups.
Bombs destroy “destroyable” blocks.
Bombs chain Explode other bombs.
Bombermans can pick up powerups to increase their attributes.

Scripts:

AutoDestroy:

This is a script that just has a timer and destroys an object after a certain amount of time. We have this script attached to the explosion animation so that the animation will go away after 0.6 seconds.

Block:

This is a script that has 3 different functions to get a blocks position and one to destroy a block at a certain position. One is to just get a blocks position, another is to get a position of a block that is indestructible. The last one only destroys a block at a position if it is destroyable. There is a boolean attribute that denotes if a block is

destroyable or not. There is also a trigger in this script that detects if an explosion hits a block and then destroys it accordingly. This script is attached to all the block objects.

Bomb:

This script defines the behavior of a bomb. When a bomb is placed it finds all the remaining game objects with the tag “Destroyable” and stores that into an array. Then the script invokes the Explode function. The explode function causes the blocks around it, that are destroyable, to get destroyed so that they are no longer in the game scene. It does this by checking where the bomb is placed and adding its “range” to all four cardinal direction to see if a destroyable block is there, and if there is the block is destroyed. After a block is destroyed a function called SpawnPowerUp is called, that randomly spawns one of our three powerups (speed, range, and number of bombs) where the block was destroyed. There is also a trigger on this script that allows a bomb to explode other bombs. If a bomb is collided with an “explosion” then it would cause the bomb to call the Explode function and explode causing a chain reaction of explosions. This script is attached to our bomb prefabs.

BombDrop:

This script gets components from the Synchronous Client script so that it can send the message when a bomb is placed. The script mainly just detects if the space bar is placed and if a player is allowed to place another bomb (there is a bomb limit restriction). After the space bar is placed a bombPrefab is instantiated that the players current position and then a message is sent from the client to the server. This script is attached to each bomberman game object.

Death:

This script handles the collisions that the Bomberman can run into. These collisions include colliding with an explosion, and the three power ups. If a bomberman collides with an “Explosion” then a message is sent to the server indicating that the bomberman has died and then the gameobject is destroyed. If the bomberman collides with the powerups then the speed of the bomberman will go up, or the range of the explosion will go up, or the number of bombs a bomberman can place will go up. This script is attached to each bomberman game object (we have 4).

DisconnectButton:

This script is attached to an empty game object so that we can use that game object and attach it to a gui button in Unity. The script has a single function Disconnect() that sends a message to the server saying that a player has disconnected and closes the application or stops the Unity editor. The function only runs when the button it is attached to is clicked.

GameOverScript:

This script is attached to the main camera in a scene and basically all the GUI objects are written in the script itself. It has a button that displays “Load Main Menu” and will load the application back to the main menu of the project allow players to play the game again.

Login:

This script runs with CreateAccountT.php and LoginAccountT.php which are

the database scripts implemented as WWWForm. Login script includes two of graphical interface functions LoginGUI() and CreateAccountGUI(). First function allows player to login with the account that already exsits in the database. If logging in is successful, then it leads the game session scene. Second function helps player to create a new account and store that data into the database. There are two IEnumerator functions called CreateAccount() and LoginAccount(). These functions create WWWForm, sends message to php scripts, add values into the php scripts, and allow php to send back the message and wait for php to find Unity. If statements are also included to make sure the players are logging with existing account and password.

MainMenu

The MainMenu script is for the first screen you are presented with in the game. It consists of a start button and will load the next scene which is the login scene (described previously). It also has an audio source that is attached to a sound asset. The source basically serves as a lobby/background music. Additionally there is a text field (called ipAdd) that will allow the user to type in the ip address of the server and the database (which by our design has to be the same ip). It will take the ip that the user typed in and update the local variable of the ip on the Login script and the Synchronous client script so the client will be able to “hard code” the ip address of the server into the client. The same logic is behind the database, “hard coding” the ip address.

Move:

This is a script that handles the commands that trigger the movements of the bombermans. It handles the up, down, left, and right arrow keys as input and translates the bomberman positions accordingly while playing the correct animation. The script is attached to a game object called Mover that takes in the Main camera object because the client is attached to the camera object. This allows the Mover object to gain access to the player index so that the script will know which bomberman is the correct bomberman to move.

Game Session Scene:

The Game session scene has a text field that allows the player to type in a game name. That game name is then updated from a public string in SynchronousClient so that the Client can utilize the game name. The Synchronous client then uses that game name to create separate client connections associated with that game name (stored in a dictionary).

Synchronous Client:

This script connects the client to the server. This script has an array of the bomberman game objects. It contains a Socket client that handles our socket and connects with the server. There is also a messagelistener where we do our data handling. The message listener gets all the messages from the server and parses them. It determines what messages are sent whether it is a disconnection, movement, death, or bomb placement. Then proceeds to do what it should do on the client side. The start function in the script sets up the bomberman array and the animations associated with

each bomberman. Then the update script will determine if a player has disconnected or not through the use of boolean variables. It also will determine if a bomberman has moved far enough to see if a position update needs to be sent to the server. It also will update the positions of the other bomberman. This script is attached to the main camera so that it will always exist on the client side. The Client also has a Dictionary for storing the game names entered from the previous scene (game session scene). The game name is stored in a dictionary along with a list of Client Connections which is a class within our synchronous client. The game name separates the client connections which have all the member variables we need such as the socket handler (connection between the client and server) and the PlayerId or playername that the user typed in during the login session.

System requirements/Tutorial

Unity Version 5.0.1f1
.NET = 3.5 (in accordance with Unity Version 5.0.1f1 which works with .NET 3.5)

Platforms: Windows XP/7/8, make the executable at your discretion.
Libraries: No extra library installations are needed.

Server:

1. Go to Bomberman2D/bomberman/Server
2. Run the ClientServer.sln and start it

*Alternatively compile the SynchronousServer.cs file and then build it as an executable. When you run the executable it will bring up a command prompt and it will print what is queued into the server queue from the Client.

Client:

1. Open the project in unity and make sure you are on the “main_menu” scene and click play
2. Click the “Start!” button
3. Type in the Database/Server IP address
4. Press Start!
5. Create an account (If you don’t already have one)
6. Type in account and password then click “Log In”
7. A new scene will load and a field for a game session will pop up, type in a one word phrase for a game session. By default if you just hit the button it should put you in a game called default0.
8. The Game will load after that. Move around with WASD or the Arrow keys and place a bomb by pressing space bar.

Database:

1. Download Xampp Apache Server + MySQL from https://www.apachefriends.org/index.html
2. Copy CreateAccountT.php and LoginAccountT.php, then paste it into the folder called htdocs (usually located under c:/xampp/htdocs)
3. Run Xampp Control Panel as Administrator
4. Check service for Apache and MySQL and Start. **Note:** Do not turn it off
5. Type '127.0.0.1/phpmyadmin' or 'localhost/phpmyadmin' in your browser
6. Create database: name it as 'accounts'
7. Set Number of columns to 2
8. Name 1st column 'Account' and 2nd column 'Password'. **Note:** Case-sensitive
9. Set Types for both column as 'VARCHAR'
10. Set Length/Values - 50
11. Test Login and Create Account from Unity.