Oscar Chng

Chris Fulgencio

Perry To

High Level Architecture

**i) All network communication messages and protocols (e.g. movement, bomb placement, player dying, player joining, etc.)**

All messages sent between clients and servers have a tag at the start of the message signified by <>. This tag is used to identify message type in order to handle it correctly.

Client -> Server  
--

**<GAME> -** Sends position and direction to server which will send to all other players in the session.

**<FOOD> -** Sends a message to the server that the game session needs to spawn food. This message is sent every 3 seconds.

**<LOGIN> -** Sends username , ID, and session to the server. Used when the client presses join session.

**<CONNECT> -** Sends client’s username and hashed password upon connecting. Used by the server to process username and password information

**<SCORE>** - Sends the score of the current player to the server.

**<MESSAGE> -** Sends a chat message to the server so it can distribute it to other clients.

Server -> Client

--

**<CONNECTED>** - Sends a message to the client that they have connected to the server so they can proceed to the lobby scene.

**<P#> -** Lets the client know their player ID so they know which head they are when they connect.

**<GAMESTART> -** Sends a message to the client to know when the session they are connected to is filled. Game will start when the first heartbeat is received.

**<FOOD>** - Sends randomly generated coordinates for food spawning to clients.

**<GAME> -** Sends position and direction to all clients where the original Player class didn’t come from. Ex: player 1 sends to player 2 and 3 but not to player 1.

**<SCORE> -** Sends the score of (example) player 1 to player 2 and 3.

**<DISCONNECT>** - Sent to clients if a player has disconnected so it is handled seamlessly.

**<GAMEOVER> -** Sent to clients if a player has won the game so that the game stops.

**<MESSAGE>** - Forwards a received chat message to all other clients in the lobby.

**<HEARTBEAT>** - A message the server sends every 5 seconds when a session has been started to see if clients have been disconnected or not. Also used by clients so they start the game at the same time.

**ii) Textual description of every class/script written (i.e. non-standard). The text should be able to inform me what the class/script is for and how it works. It must also inform the relationship between objects (e.g. a GameObject has Player and Enemy. Players has Bomb and Sword …)**

**Client** - A modified client class changed to handle the processing of messages. The client checks for message protocols and will forward the information to respective functions and classes.

**Chatbox** - A unity script class that handles the chatbox. Messages can be sent from a client and the server will send it to all other clients in the lobby and messages will be displayed in the chatbox. Inputfield for sessions is a part of the chatbox and will allow users to join a game session.

**endgame** - A short unity script that basically controls the screen that displays what player won when the game is over.

**Snake** - A unity script class that controls the player’s snake. Contains variables for score keeping and controls hit detection. Also contains code for receiving keyboard inputs to control the snake. Contains a Current\_Player class that just stores information about the snake. The snake will constantly move in the given direction and will respawn upon death after a few seconds.

**Snake2** - A unity script class that is largely identical to Snake except for movement control and score handling. Snake 2 instead receives its inputs and scores from the server. This controls the snakes for the other players.

**SnakeManager** - A unity script class that handles the loop of the game state. This class contains the Snake and Snake2 objects for the game. The SnakeManager calls code for receiving from the client and has functions for controlling the game state such as controlling Snake2 and score.

**SnakeSpawn** - A unity script class that controls the assigning of Snake and Snake2 scripts to the snake head game objects in unity. Depending on the info received from the server, the SnakeSpawn class will assign the Snake script to the snake head that will be the player and Snake2 scripts to all other heads.

**SpawnFood** - A unity script class that contains a food class which contains x and y coordinates. This script will send a request to the server when the game is started which will let the server know that food is needed. A function will be called to create the food object in the game every 3 seconds based on the coordinates generated by the server.

**Startgame** - This unity script handles the initial login screen. It contains the client object that will be used connect to the server and has functions for sending and receiving using that client. Contains the function for the button used to connect to the server.

**Server** - A modified version of the Server class. The class is modified to detect messages based on protocols and to respond according to the clients. The server also stores various dictionaries in order to sort out clients by ID and sessions to ensure that all messages are handled correctly and to allow multiple game sessions. The class also contains a database class that it uses to store users and scores and check logins.

**Database** - A database class that utilizes sqlite. The code has functions for checking for correct username and password combinations, inserting new users, and updating the users’ scores. If the database file does not exist, it will be created by the server when run.

**Program** - A class that contains a server class and starts it up.

**iii) System requirements/Tutorial: Which libraries, platforms, .NET version, Unity version, PyGame/Python version is required, and a small tutorial on how to start the client and server.**

Libraries that are needed for the game to run and function correctly are: Newtonsoft.Json and Sqlite for C# files to be able to store and load database files. These files are included in the github repository. However; in order to run the game and edit the game files, Unity 5.0 or higher is needed. To run the game client side, you need to build an exe file from Unity client. After that, open the newly built .exe file three times. Before logging in on those clients, the server needs to be runned. In order to open and run the server, you need to have Microsoft Visual Studios 2010 or higher and in the snaketest\Server Code\Snake Server double click to open the Snake Server solution file. Before opening the file, make sure to have the MyDatabase.sqlite inside \snaketest\Server Code\Snake Server\Snake Server\bin\Debug is deleted first due to the bug of database locking which in returns disconnects the clients when they are being played. After opening the file, click F5 or start to run the server. You can now then log in to the clients with different usernames and passwords. To run the game, you need to have at three players, no more no less because the game requires three players to start so make sure to have three clients opened. The game client also handles multiple sessions, so you can play as many games in different sessions as you want but make sure to have at least 3 clients joining the same session in order to start the game. The game client also have a chat lobby so players can communicate to each other to which session they want to join. The first snake to reach 10 points wins the game, the client will change the scene to a win screen to show which player scored the first 10 points. The username, encrypted password and the scores is located at the \snaketest\Server Code\Snake Server\Snake Server\bin\Debug with the file name MyDatabase.sql. You need an Sqlitebrowser program in order to open the sqlite file to see the data inside. Thats basically it! Have fun and enjoy playing the game.