CSC-631/831 – Individual FinalTerm Report - Spring 2019

Castle Escape

Sushil Plassar (918818893)

Table of Contents

1. Abstract

Castle Escape tries to explore a different approach to the vanilla multiplayer game, one that is rarely done, but is slowly starting to rise in popularity. Multiplayer games primarily focus on being able to see others that are around them, individually, or communicating cooperatively, to achieve some goal - beating the opponent team, scoring higher than other individuals, ranking higher than others. However, this game will take the core multiplayer gameplay aspects and include a small twist, the idea that players are separated from each other. The players shall not able to see each other whatsoever but must rely on pure communication along with physical cues that occur in-game to cooperatively to escape. This small change of not being able to see your partner dramatically changes how a multiplayer game usually functions.

The creativity that is involved with communication goes above and beyond the usual idea of "attack and defend" that has been popularized by most multiplayer games, which mainly encompass how shooters, sports, or fight based games function. Everyone usually has their own unique way of how to interpret something which is the charm/nature of the game. What may appear blatantly obvious seen through the eyes of one player, will not be magically seen by their partner, simply because of the fact, well, they cannot see each other! How to decipher each puzzle is dependent on how well the players communicate their interpretations with each other to successfully escape. Castle Escape was mainly inspired by the game "We Were Here" but can still be attributed to other titles such as "A Way Out" and "Keep Talking and Nobody Explodes".

2. Introduction

Our game is a first-person puzzle thriller game, similar to how escape rooms work. It requires two players working together to solve a series of puzzles in order to escape their respective rooms. Each player is placed in a separate room, each with their own respective clues and puzzles. In order to solve the puzzles, each player must use the clues found in their room for the other player, therefore, communication between each player is crucial to solving the puzzles to escape with a better time. The theme will include horror aspects with a focus on an eerie and suspenseful atmosphere. The game "We Were Here" is a perfect example of the type of game we're going for.

3. Game Concept

- ❏ Total gameplay time will be around 60 minutes
- ❏ There will be a login screen where player needs to login to the game
- ❏ There will be a main menu that will allow 2 players to connect to start the game
- ❏ There will be two players working together to solve puzzles together in 3 different puzzle rooms

- ❏ The players will be able to move around in their respective rooms
- ❏ The players will have a first-person point of view while playing the game
- ❏ The players will be able to interact with levers, tiles, and switches while in the game to solve the puzzles
- ❏ Interactions in one player's room affect events in the other player's room
- ❏ There is a timer to keep track of how long it takes a duo to complete the game
- ❏ There is a visible timer for the player to see
- ❏ There is an in-game chat for players to communicate with one another. Although third-party communication is recommended
- ❏ A fastest time completion will be tracked so players can see how fast they beat the game. (Database)

4. Motivation

We wanted to deviate from the usual normal multiplayer experience. Though usually seeing other players is considered the norm in any multiplayer game, the game was designed under one of the most important elements about multiplayer - Communication. Castle Escape is heavily focused on communication to be the core functionality of the game, followed by game design and sound to allow for further immersion for the players. The game is still designed to give a horror/thriller vibe to give a sense of terror, a feeling that you want to escape from this place. The most unique aspect of Castle Escape is how everything is still up for interpretation. What two players will communicate/experience will be different than the next two players that play. This is also very rewarding on our part because it is fun to see how people communicate ideas in a stressful situation.

Even more so, the puzzles were designed in such a way to allow both players to be communicating nearly all the time, which may result in talking over your friends in the heat of the moment. This is great because that is exactly how most multiplayer games function. It's not all peace and rainbows for a multiplayer shooter and neither should it be for a multiplayer puzzle. While Player 2 explains what Player 1 is possibly doing wrong, Player 1 may be yelling at Player 2, describing about what they see or what happens in the room because of some interaction. This emphasizes the key aspect of multiplayer in general, that effective communication is the key to progress and achieve your goal.

One could argue that the satisfaction of solving a difficult multiplayer puzzle could be the same or even more so than winning against a troublesome opposing team. That "A-ha!" factor is something we also strived to achieve so players feel rewarded for their efforts. This was very challenging to execute, which was a puzzle itself entirely as we designed this game.

5. Game Design

Level Design (Level 1):

- **Goal**: Players must collaborate to solve a simple four-digit code that will allow Player 1 to guide Player 2 with a specific path to get to the other side of his/her room
- **Presentation:** Below are the images of both players rooms

Player 1:

Player 2:

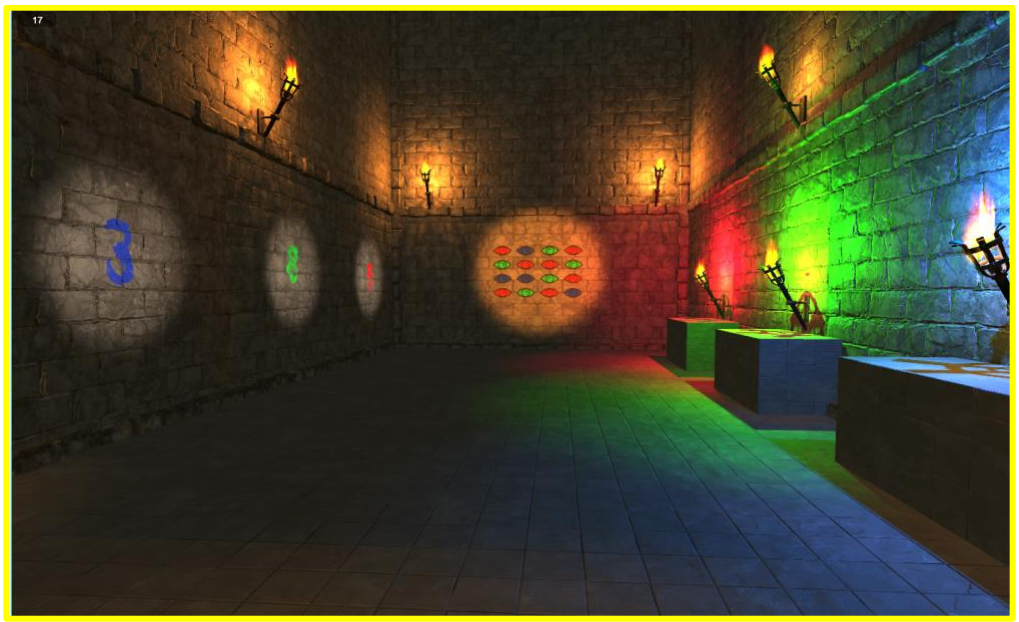
- **Revelation:** The key to this level is that both players need to step on a specific tile so that Player 1 can realize that the symbols on Player 2's room are not really symbolling but numbers. The numbers are then solved by the order of the symbols in Player 2's room, giving a number combination for Player 1 to pull the correct lever. Pulling the incorrect lever will result in the room darkening and eventually turning the room pitch black so that Player 1 can no longer see. Once solved, Player 1 is granted instructions for Player 2. These instructions will be used to guide Player 2 with the correct path to reach the other side of his/her room
- **Audio:** Footsteps sounds, background music, wall tile movement sound, correct/incorrect lever sound

Level Design (Level 2):

- **Goal:** Players must use their respective eye coloured charts on the walls of their rooms to solve a three-digit combination. Both players must hit the correct lever based on their number combinations through the clues found in both rooms.
- **Revelation:** The key to this level is for Player 1 to trigger the coloured torches on their side of the screen by pressing [e] to interact with them. This will reveal a set of eyes for Player 2 to see as well as revealing a number on the wall opposite of Player 1 torches. Player 2 then understands to count the eyes and hit the correct lever. The correct order, however, is based off of reading the number from right to left or red>blue>green and not the obvious red>green>blue. A podium will rise in Player 1's room revealing three more combinations.

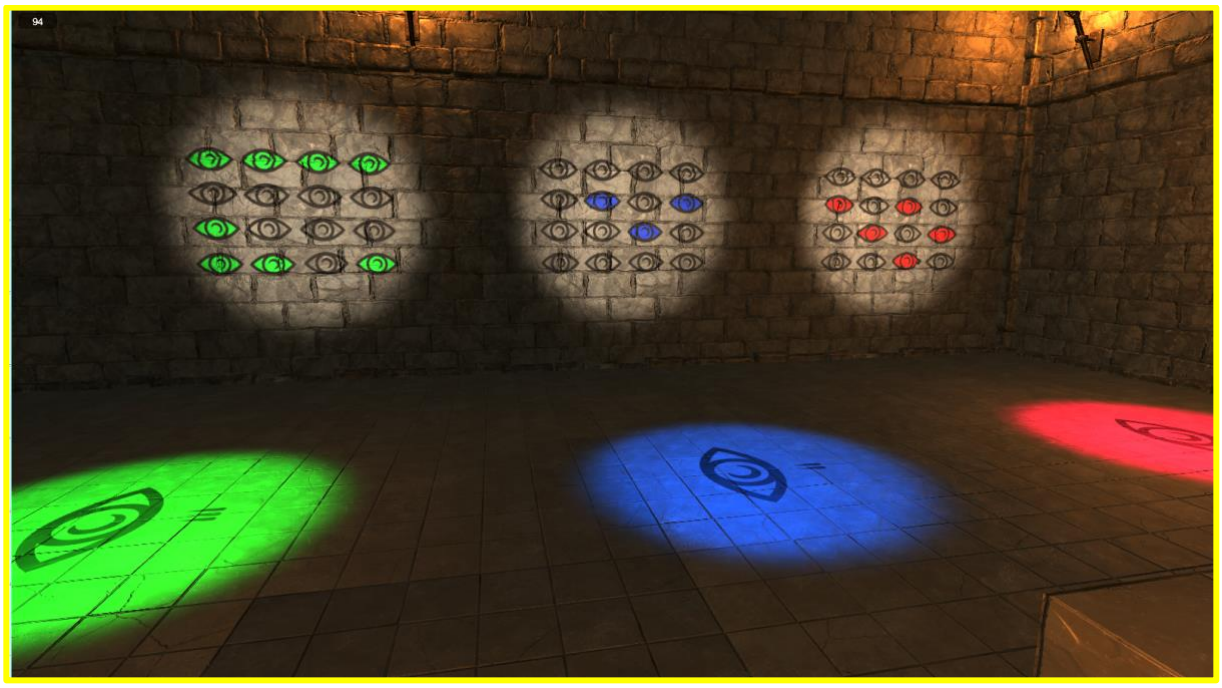
  After Player 2 communicates to Player 1 that they must count the coloured eyes in a specific order, Player 1 can figure out the combination from their own coloured eye chart. After this is complete, the door to the final puzzle will open in both player's room.
- **Audio:** Footsteps sounds, faster pace background music, light switch clicking sound, podium rising sound, wrong lever pull sound, wall coming down from the ceiling sound.

Player 1:



Player 2 (before player 1 activates lights):

Player 2 (after player 1 hits the light in his room):



Level Design (Level 3):

- **Goal:** Players must collaborate together to transverse one final maze together. Player 1 has access to the entire maze's layout while Player 2 can see nothing but walls, doors, and switches (the maze itself). Players must communicate clearly about which way to head first and in which order of triggers to open the correct sequence of doors.
- **Revelation:** The key to this level is in Player 1's map; The colour coded rooms (big rectangles) that correspond to their colour coded gate (skinny rectangles). Each of these rooms has a switch that opens their respective gate. (For example, the blue room will open the blue gate). When Player 2 steps on a switch a loud gate opening sound is played informing Player 2 that a gate has opened. In Player 1's room, a light will turn on shining on the wall and the colour coded room that Player 2 has just triggered. The arrow on the top of the map signifies the exit so it is just all a matter of opening the doors in a correct sequence and then navigating Player 2 out of his maze.
- **Audio:** Footsteps sounds, even higher tempo background music, gate opening sound, light turning on sound, tile being stepped on sound.

Player 1:

Player 2:

6. Architecture

Below is the technology stack used in the project:

- **Game Engine/Client**
  - Unity 3D
  - C# (Unity underlying scripts)
- **Game Server - Programming Language**
  - Java (Server)
- **Database**
  - MySQL Database
- **Cloud Infrastructure**
  - Amazon EC2 (hosting the server)
  - Amazon RDS (hosting the MySQL database)

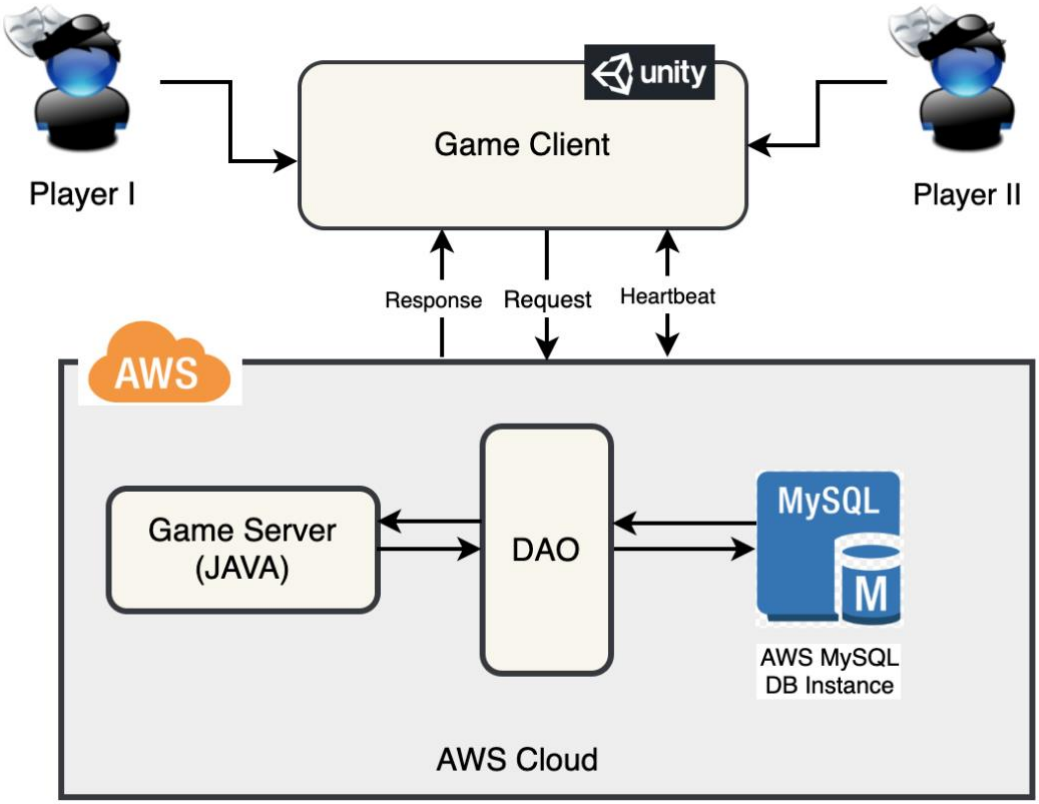


Figure-1 Castle Escape Architecture Diagram

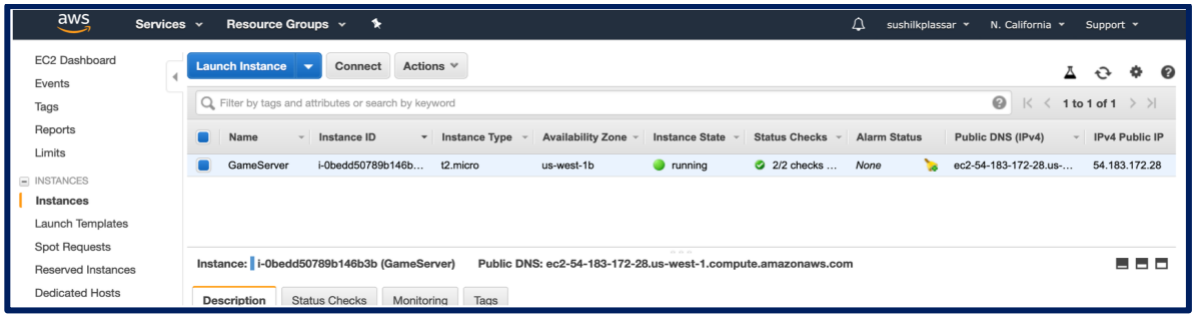
7. Individual Contribution

My major contribution was to work on the server so as the game can be played in a multiplayer mode. In addition to this, I worked on Unity client side to test the protocols developed and developed functionalities like Chatting and Game scoring feature for the game. The major tasks I was involved as part of this are:

1) Cloud Infrastructure

The first step was to create an instance on amazon cloud for hosting the web-server. An amazon free tier account on aws.amazon.com was set up and an instance of EC2 was created. The instance is running and can be accessed via IP - 54.183.172.28

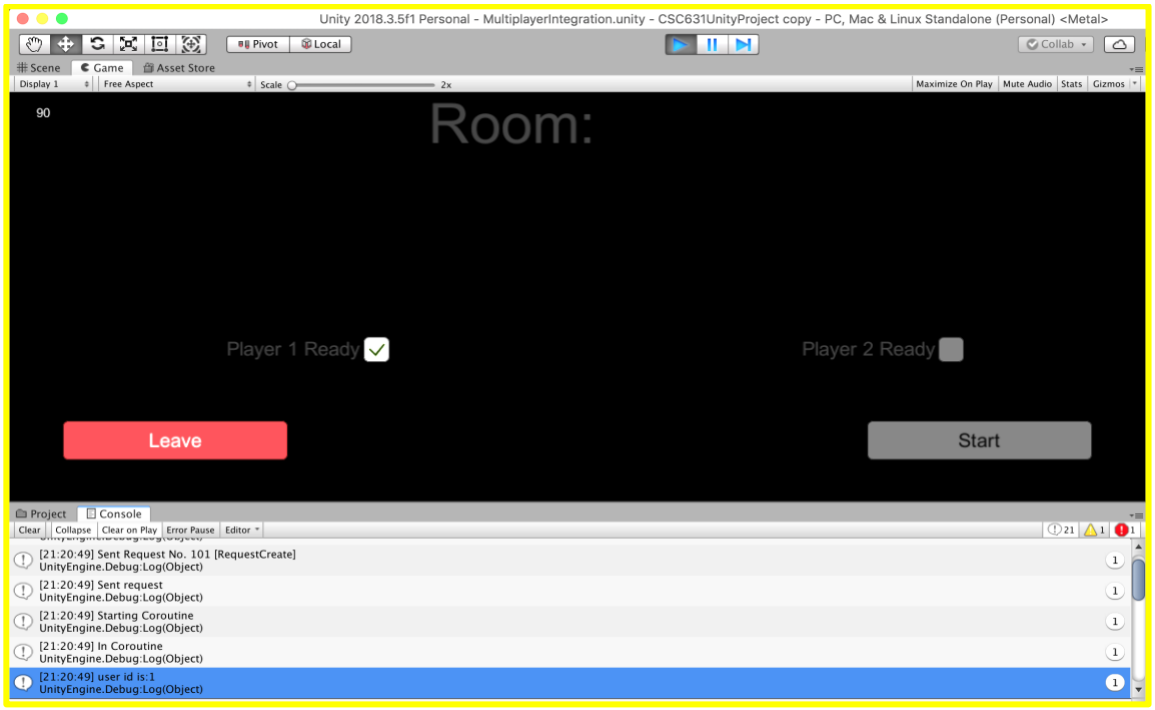
Server Code deployment: A Jar file was created each time with the latest server changes and deployed directly to ec2 instance via scp command.



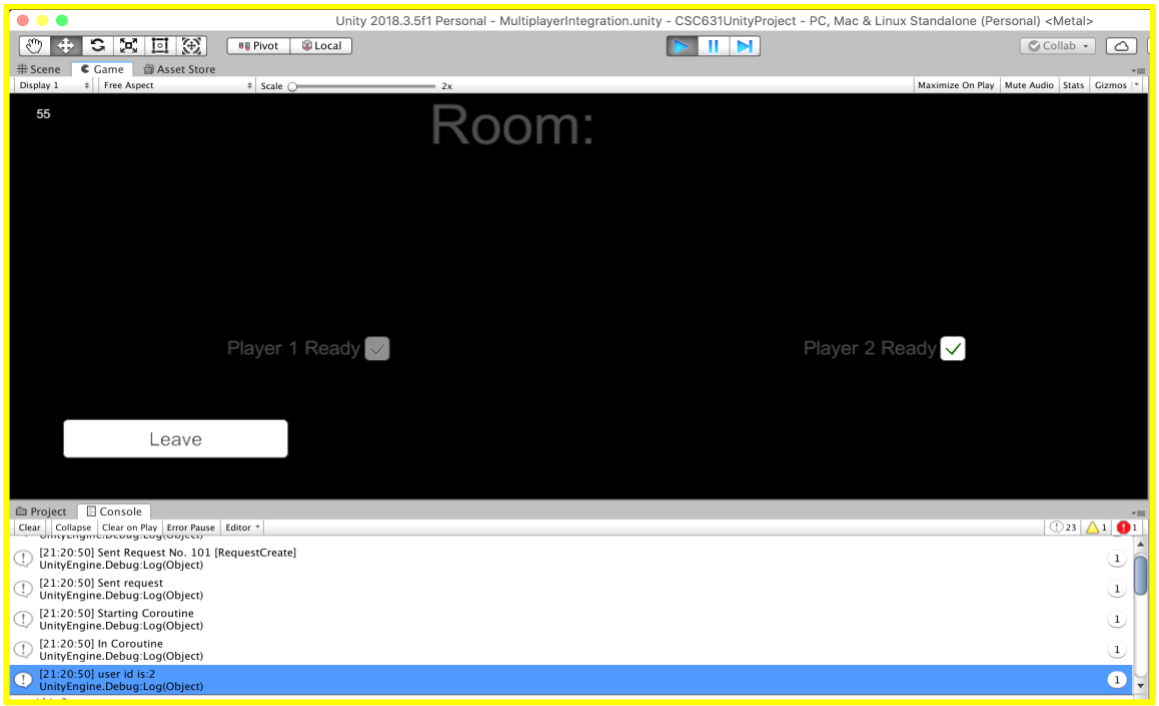
2) **Migrating initial server code to cloud for testing connectivity**

The next step was to check for the connectivity after the server code was deployed on cloud instance. Below is the screenshot when player try to connect with the server

Player 1 trying to connect:



Player 2 trying to connect:



3) Protocols

As part of handling and responding to client requests, I developed various request and response protocols. They are:

a) RequestLogin and ResponseLogin: This protocol was developed to enable the user to login into the game. The client requests by entering the username and password and gets logged in after providing the right credentials
b) RequestSaveTopScore: This protocol was developed to save the top score of the playing team after they complete the game or the moment they die during the game. The timer at the client side was sent to server which saved this score into the database as score for the playing team
c) RequestGetTopScore and ResponseGetTopScore: This protocol was developed to retrieve the top five best timing scores for the playing team. The team can view these scores from the menu screen and can try to play better to improve their timing score for completion of game
d) RequestChat and ResponseChat: This protocol was developed to enable the chat functionality in the game. Once the game starts, the players can communicate with each other using this chat functionality.
e) RequestTwoPlayer and ResponseTwoPlayer: This protocol was developed to ensure a minimum of two players are needed before starting the game.

4) Integration of client with server

After the basic structure was built on the client side and server was deployed on cloud, I worked on integration of client-side components with server code. The Remote_Host value was modified to point to cloud instance IP address and connectivity was done. After this, the rest of the things were easy to integrate as they were initially tested locally before being deployed on cloud.

5) Chat

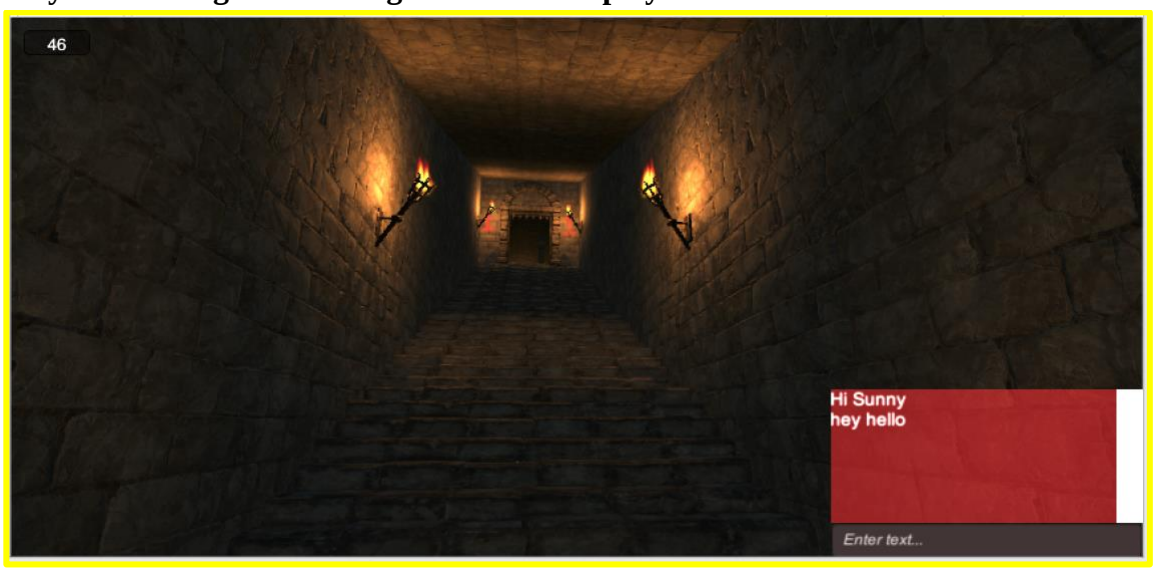
In addition to working on server, I created the end-to-end chat functionality for the game. By using this, the players could communicate with each other since our game is based on effective communication between the players to solve the puzzles. This was done by:

a) Developing a chat manager object
b) Developing client side requestChat and responseChat protocols
c) Developing server side requestChat and responseChat protocols

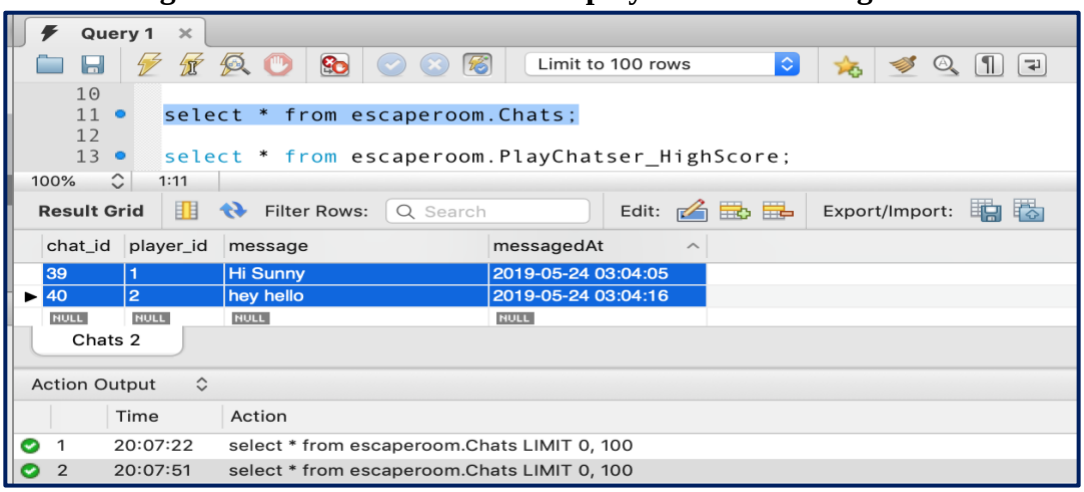
In addition, these chat messages are also saved into the database by using the DAO provided by the team member working on DAO. This was developed as part of parallel development of the project components and later integrated into the game

Player 1 sending chat messages to the other player:

Player 2 sending chat messages to the other player:



Chat messages saved in database based on player ID and timing:

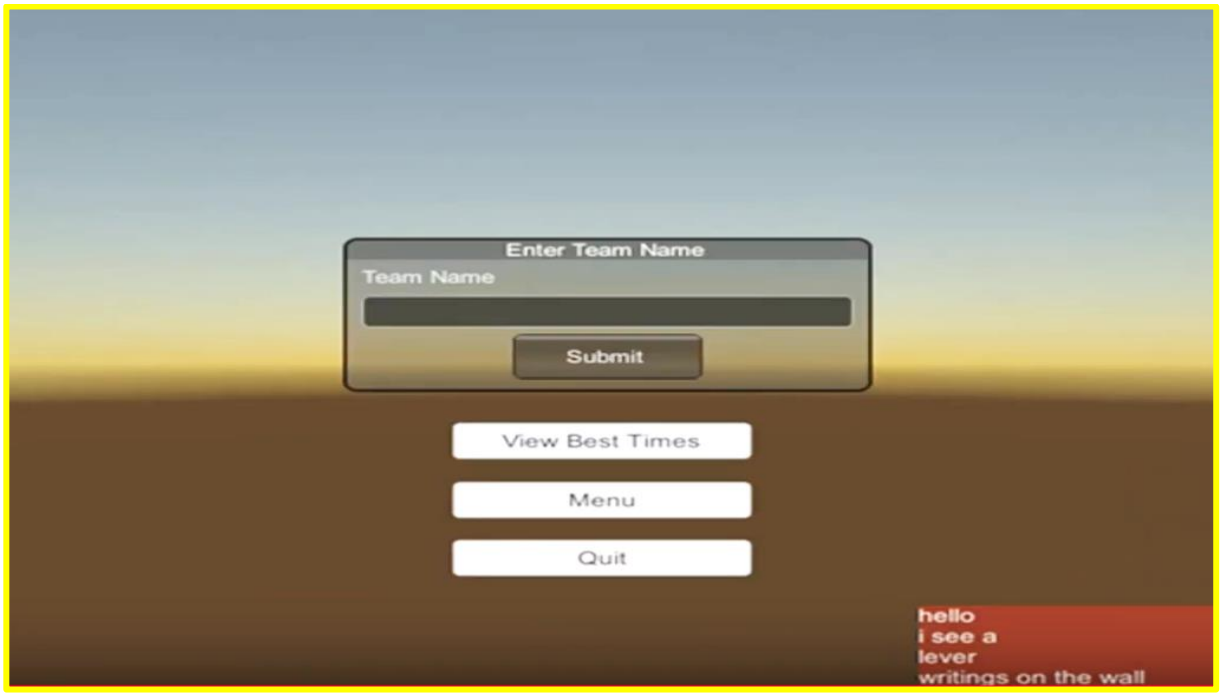


6) Scoring

I created the end-to-end scoring functionality for the game. This helped to make the game more competitive as players would like to better their timings while playing the game and solving the puzzles. This was done by saving the Team name and Team score in the database:
For saving the scores to DB:

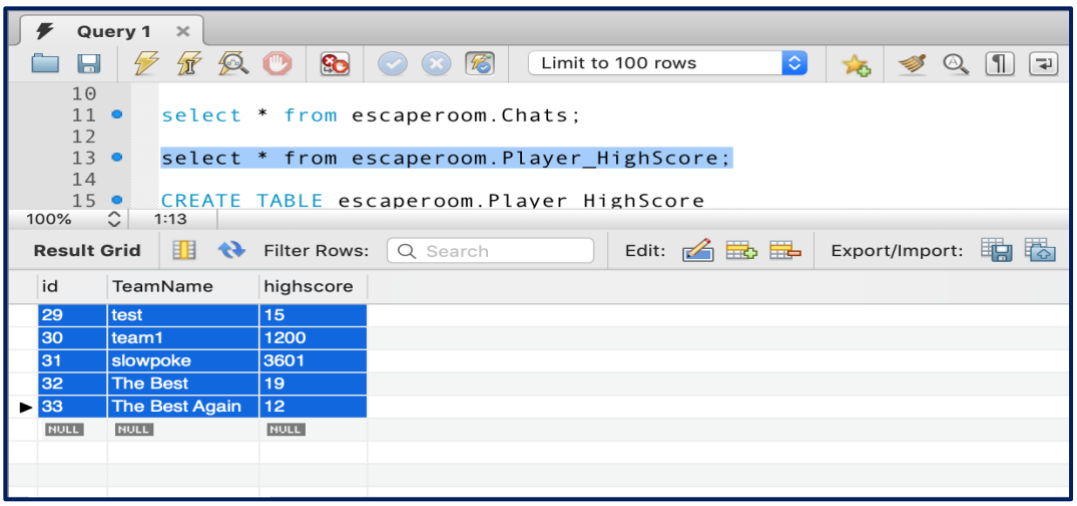
a) Developing client side requestSaveScore protocol
b) Developing server side requestSaveTopScore protocol
For retrieval of scores:
a) Developing client side requestTopScore and responseTopScore protocols
b) Developing server side requestGetTopScore and responseGetTopScore protocols
c) Modifying the Games DAO object to save and retrieve the scores into the database
The time running at the client side was is calculated in seconds and stored in database as an integer value. But when client asks for retrieval of scores, the scores are passed as best of top five timings and the time value is displayed in HH:MM: SS format. This component was developed parallelly and later merged into the actual game.

Game screen to enter the team name and view the best timings for the team:



Game screen showing the top five timings with team name finishing the game:

Database showing the Team-name and their high score saved in the database:



8. Challenges

As part of developing this game there were few major challenges we faced:

1) Integration of client component with the server, understanding how call backs work after getting responses back from server.
2) Initially, it was a bit tricky to understand how to connect two players via heartbeat and how to send updates from one player to others
3) Hosting RDS MySQL DB and Amazon EC2 instance on different accounts. This was a major challenge as I didn't know that amazon doesn't allow to connect with instances that are opened under different AWS accounts. After searching over internet, I came to know that it requires VPC (Virtual private cloud) pairing which is complex to implement and time consuming.
   As a quick fix, I created the EC2 instance and RDS database under same AWS account to resolve this issue

4) With no prior knowledge of Unity and game development, there was a learning curve to get everything work together. Initially, it was bit tough to understand the intricacies involved but later on we became comfortable handling the complexities involved around the client and server model

9. Learnings

1) Working with people coming from different background and knowledge level and doing team coordination.
2) Technical challenges of dealing with client-server architecture and understanding how protocols work.
3) Intricacies of game development i.e. details about how actual game-flow works and how small details matter to make a huge difference.
4) Getting to know how things actually work behind the visuals and graphics that are seen while playing multiplayer games.
5) How debugging and testing plays a vital role in solving so many bugs ahead of time. Debugging both server and client at the same time would many times hang the system since Unity is very resource-intensive so I learned the importance of logging and exception handling.
6) Importance of continuous integration, as continuously integrating code in github didn't let arise much of the merge conflicts and made the process of testing very smooth.