Table of Contents I

Table of Contents II

Introduction

This is a 2 player game based written in c++ with the help of SDL2 and ENET libraries. This fun game essentially has 2 game modes.

1. Game mode 1 tests you knowledge on how well you know the nooks and crannies of the campus. It is similar to a treasure hunt across the campus. You will be given 2 hints and you objective is to place 2 flags as close to the correct location possible. The one who overall places the flags closer wins!.
2. Game mode 2 is a time based mode. The one who is faster wins. You have 3 objectives and your goal is to complete them as fast as possible. The one who gets them done first is declared the winner.

General game feature

To make the game more interesting and complex, multiple features common across both the modes have been implemented

Beware of the guards!

Three guards have been scattered across the IITD map. We recommend, you definitely try avoid them. They are actively patrolling around the campus and will chase you if you go near them because you are not wearing a mask. They will literally turn red while following you. To loose them run fast enough, or better, get near the opponent while being chased by the guard. If they are not careful, the guard will start chasing them instead of you! If any guard does manage to catch you, he will impose a fine and return to the main gate to resume patrolling.

Take the Yulu

If you find walking is just too slow, take the Yulu to whiz around the campus in style. You sure get a speed boost, but taking a yulu is still expensive and your funds are limited.

Money

You have limited amount of money. Spend it wisely. Money in the game can be spent for multiple things.

1. Buy food to regain you stamina
2. Taking a yulu will cost you money
3. The guard will fine you.

Travel faster, if you don't mind getting dirty

Two water bodies flow across the campus. These are of course the drains of IITD. If you don't mind getting dirty, use them to travel at a 10 % faster speed.

Hungry ?

Your stamina is also limited. Running across the huge campus can be tiring. Once your stamina runs out, you won't be walking but crawling. Don't worry though. If you have enough money, you can always recharge yourself at the mess or the various eateries across the campus.

Basics of Game Mode 1

To play the Game mode 1, start the server with "./game.out 1" and "./game.out IP_ADDR" for the client side. You will be presented with an initial screen. Press enter to get open the instructions and objectives screen. Press tab once you are ready. The game won't start till both the player are ready.
Go around the campus, placing the flags with the help of "i" and "o" keys for the first and the second flag. If you forget what the various keys do, simply press the "p" key.
Out of a pool of 14 questions, 2 are randomly selected and dispayed. These are the same for both the players.

Basics of Game Mode 2

To play Game mode 2, start the server with "./game.out 2" and "./game.out IP_ADDR" for the client side. The initial screen and the controls screen would be displayed. If you forget what the various keys do, simply press the "p" key.
Out of a pool of 10 objectives, 3 are randomly selected for each player. Only one objective is displayed at any instant. Completing an objective involves going to the place asked and pressing "j". Once an objective is completed the next one is displayed. First to finish all three wins.

Controls

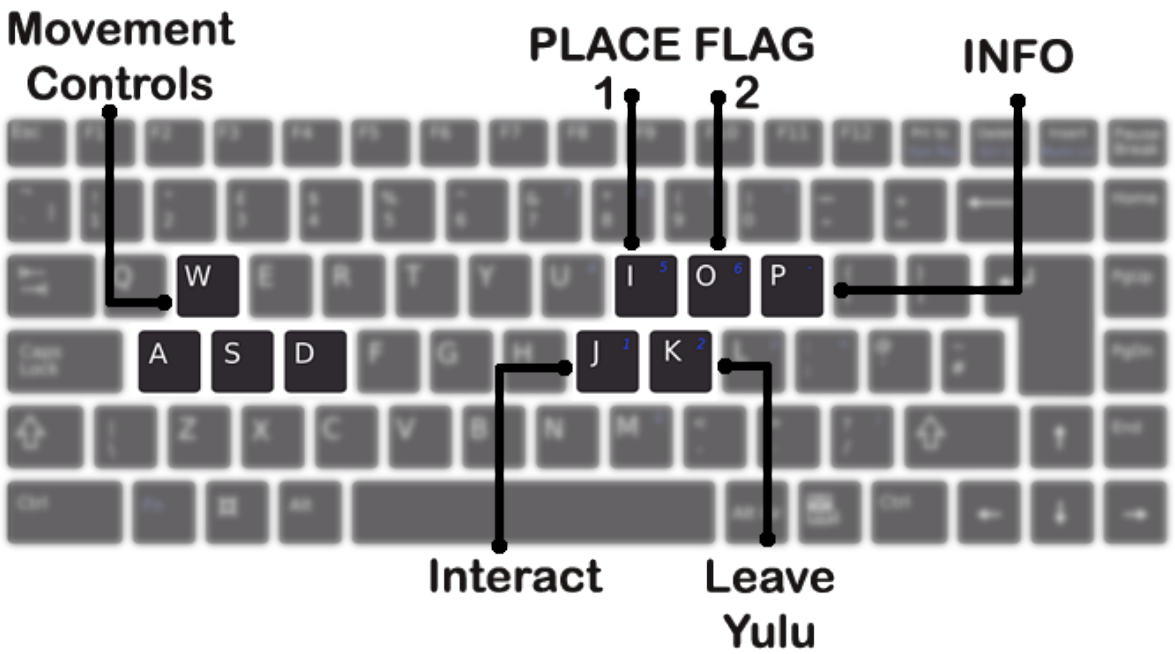

Figure: Controls

Design Decisions

We have used the following libraries

1. SDL2: Library to implement the basic functionality
2. ENet: Library to create connection between the two devices
3. SDL_ttf: Text rendering Library
4. SDL_mixer: Implement the audio system

Rendering the map

The map of IITD is huge. Rendering in all ar once was not viable as it would have caused too much lag. Therefore only the map currently visible on the screen is rendered. We basically divide the entire map into 80*80 pixel chunks to get 225*84 chuncks. The pixelated design has repeatable patterns. Thus not all tiles need to stored. Typically a seperate tile map is generated. We have a matrix of size 225*84, which maps to the id of tiles. We just iterate this matrix and load the tiles in view of the player. Since these are static in nature, we just skip the part of updating/refreshing these tiles. 3 Such maps are used:-

1. A background image for all the roads and grass.
2. A layer to render all the buildings, trees, etc
3. A layer to detect collisions and classify the type of the tile it is. Currently we classify any tile into whether it is a normal tile, a collision barrier, or part of any special landmark like a hostel, eatery, lhc, football ground etc.

Connecting to the peer

For the connection, we used enet, as it provides reliable udp. UDP is much faster then TCP, but does not guarantee packet delivery. Enet does garantee this to some extent. Thus we get the best of both worlds.
In the game we exploit the natural asymmetry that is created when one player is the server and other is a client. A bool am_i_server is set to true, if you are the sever, recognised by the fact, that an ip address was not passed. am_i_server bool is used to decide what sprite to give the character. For the program, the player he controls is player1, and he just needs to send the location of player1. The packet recieved is always the opponent and thus it updates the player to position and animation.
The motion of the gaurds is controlled by the server. Since he has the locations of both the players, it is fairly easy to compute its path. The location of these gaurds is broadcasted to the client. Now this client has gaurd locations, it can locally detect collisions and take action accordingly.
Apart from these packets for the game state, eg the stage at which other player is, who won, the score of other player is exchanged.

Entity Control System

Taking inspiration from this video, we implement an entity control system that takes care of how objects are refreshed, updated and rendered. A manager takes care of all this. It keeps a track of all the entities we have and loops over them to call their individual refresh, update and render functions.Each entity itself has various components like the position Component, sprite Component, and many more. All the various components derive from a common components class. It has functions like init(to initialise), update, render,etc. Each entity calls the refresh, update and render function for each component.
All these entities are put in groups. The order of rendering is decided by the groups.
To improve performance, the maps are taken care separately as described above.

Scoring in game mode 1

To calculate the points of a player, we use the formula

$$\text{Score} = \frac{1}{a|t_1 - d_1| + b} + \frac{1}{a|t_2 - d_2| + b}$$

This is done so that placing as close yo the correct location is rewarded accordingly. If you place too far away, the scores should be negligible. Coefficients a and b are chosen, so that the scores don't fall off too drastically with distance and don't get too high near the exact location. The d_i is set to infinity if the ith flag is not placed. That is 0 score is given. The one with a higher score of course wins

Economics of the game and managing your health

Each player starts with Rs. 1000. If you eat in your hostel mess it costs Rs. 20, every other meal costs Rs. 50. The charges for Yulu are Rs. 6 per second. If you are driving your Yulu over the football ground, basketball ground, tennis court, garden etc, a fine of Rs. 30 per second will be imposed. If you crash your Yulu in the sewer a fine of Rs. 500 is imposed and the Yulu would be lost. If a guard catches you then you are fined Rs. 200 (Remember you are roaming without a mask).

Your speed is linearly dependent on your stamina remaining. If your stamina drops to 0, you speed is half of the initial speed. Eating food restores your stamina to 100%. A yulu increases your current speed by an amount of your initial speed. Swimming speed is 10% more than your walking speed.

Glimpses of the game I

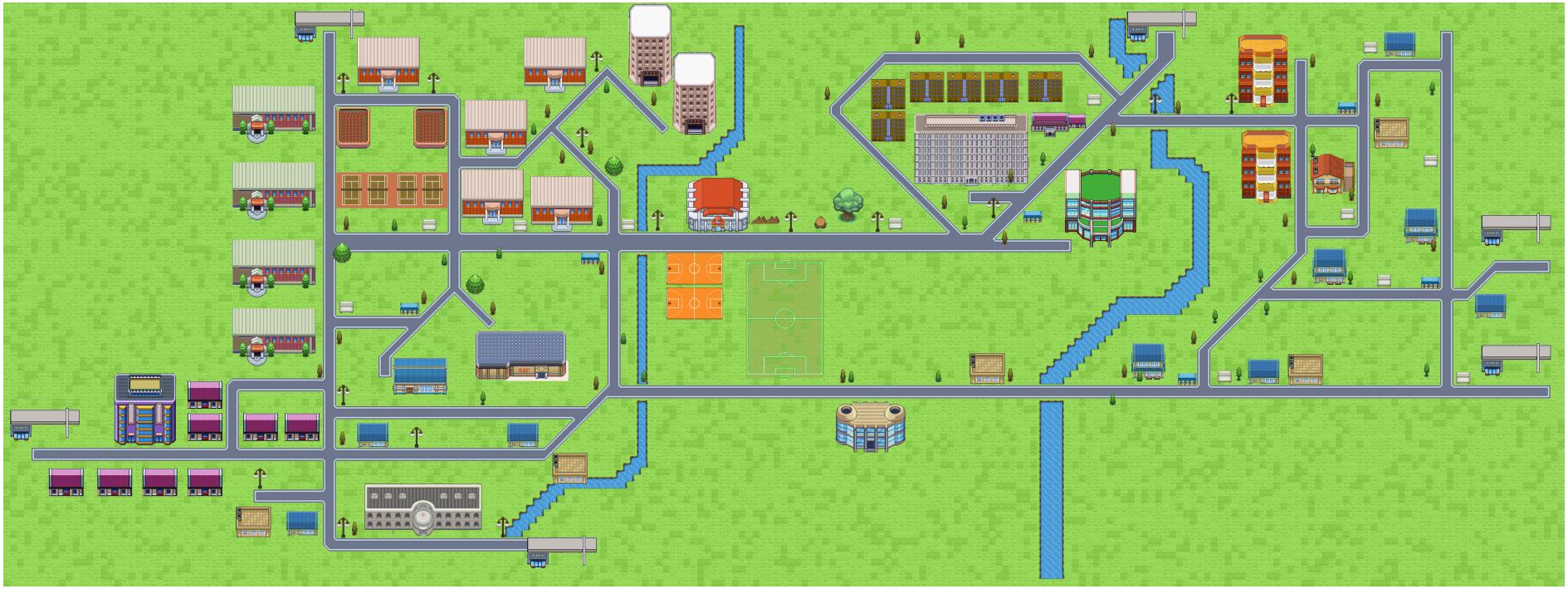

Figure: Map

Glimpses of the game II

Figure: Loading screens

Glimpses of the game III

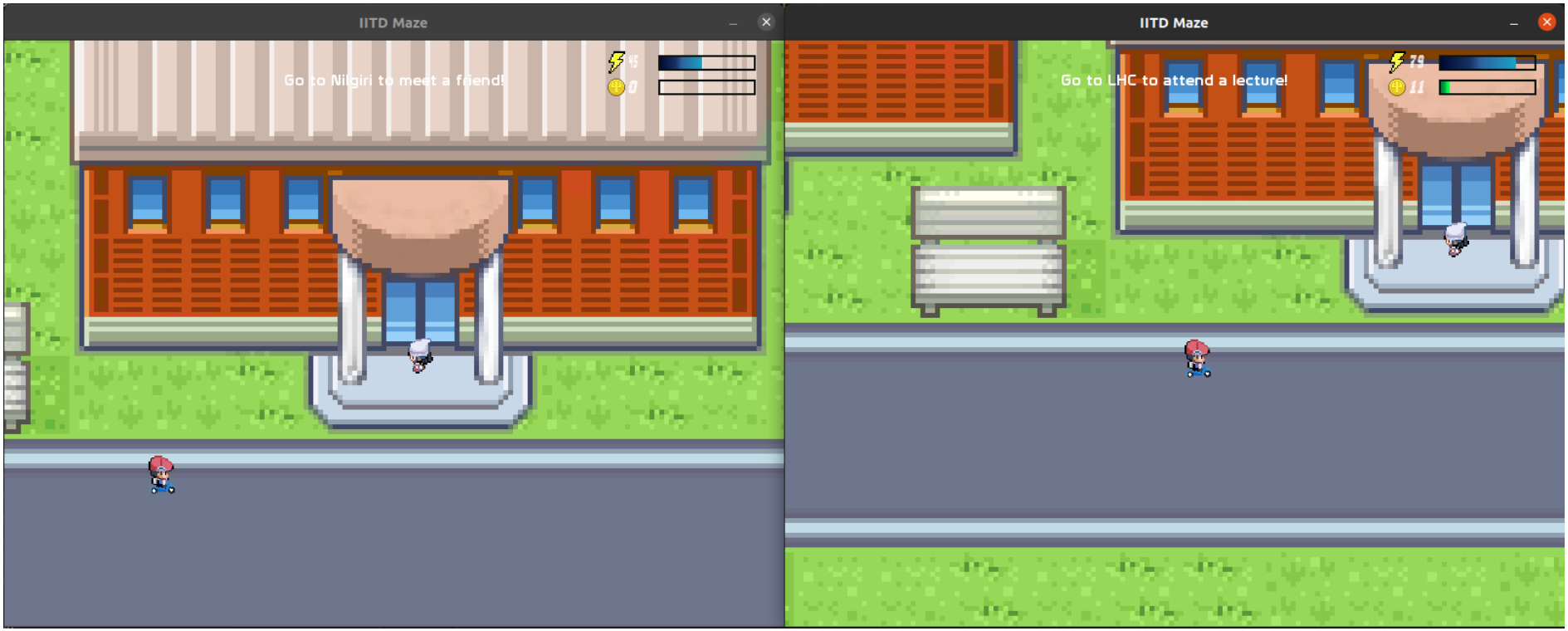

Figure: Sample images

Glimpses of the game IV

Figure: Sample images

Glimpses of the game V

Figure: Sample images

Glimpses of the game VI

Figure: Sample images

Glimpses of the game VII

Figure: Sample images

Glimpses of the game VIII

Acknowledgement I

1. Deviant art: https://www.deviantart.com/davian-art
2. Pinterest: https://in.pinterest.com/
3. Open game art : https://opengameart.org/
4. LibSDL Documentation: https://www.libsdl.org/
5. YouTube Playlist - How To Make A Game In C++ SDL2 From Scratch! : https://www.youtube.com/playlist?list=PLhfAbcv9cehhkG7ZQK0nfIGJC wSLrx
6. Enet example : https://github.com/zpl-c/enet
7. Pyxel Edit Software