KLE Society's
KLE Technological University

Course Project Report

On

Covid 19 Game using Socket Programming

Team No. : C26

Name	USN	Roll No.
Saloni Shah	01FE18BCS183	363
Samarth M	01FE18BCS184	364
Samarth R	01FE18BCS185	365
Sanjana Kambar	01FE18BCS190	390

INDEX

1.Introduction

With the start of 2020, we have been experiencing the outbreak of Covid-19. Coronavirus disease (COVID-19) is an infectious disease caused by a newly discovered coronavirus. Most people infected with the COVID-19 virus have experienced mild to moderate respiratory illness and recover without requiring special treatment. Older people, and those with underlying medical problems like cardiovascular disease, diabetes, chronic respiratory disease, and cancer are more likely to develop serious illness.

The best way to prevent and slow down transmission is to be well informed about the COVID-19 virus, the disease it causes and how it spreads. Protect yourself and others from infection by washing your hands or using an alcohol based rub frequently and not touching your face.

The main prevention measures suggested by the doctors and researchers state, wearing masks and maintaining social distance.

The essence of the game we developed simulates the condition of Covid outbreak and the way to stay in the game is to follow the prevention measure to prevent the spread, i.e., maintain social distance from the person who is infected.

Through the game we wish to pass on the seriousness of following the prevention measures so as to reduce the spread of the virus. Games have been an integral part of teenagers and youngsters' lives and this can prove to be a good way to spread the social message.

2.Software Component Details

Socket Programming

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while another socket reaches out to the other to form a connection.
Server forms the listener socket while the client reaches out to the server.

Socket programming in python has become ubiquitous as python provides a bunch of APIs to work with.

The primary socket API functions and methods in this module are:

socket() - One socket(node) listens on a particular port at an IP, while another socket reaches out to the other to form a connection. Server forms the listener socket while the client reaches out to the server.

bind() - The bind() method of Python's socket class assigns an IP address and a port number to a socket instance. The bind() method is used when a socket needs to be made a server socket.

listen() - The listen() in Python calls the underlying listen syscall: listen() marks the socket referred to by sockfd as a passive socket, that is, as a socket that will be used to accept incoming connection requests using accept(2)

accept() - The accept() method of Python's socket class, accepts an incoming connection request from a TCP client.

connect() - The connect() method of Python's socket module, connects a TCP(Transmission Control Protocol) based client socket to a TCP based server socket.

send() - The send() method resumes the generator and sends a value that will be used to continue with the next yield . The method returns the new value yielded by the generator. The syntax is send() or send(value) .

recv() - recv() returns an empty bytes object , then the client closes the connection and the loop is terminated. The with statement is used with conn to automatically close the socket at the end of the block.

close() - close() closes the connection with the client.

The API calls the server makes to setup a “listening” socket are:
socket()
bind()
listen()
accept()

A listening socket does just what it sounds like. It listens for connections from clients. When a client connects, the server calls accept() to accept, or complete, the connection.

The client calls connect() to establish a connection to the server and initiate the three-way handshake. The handshake step is important since it ensures that each side of the connection is reachable in the network, in other words that the client can reach the server and vice-versa. It may be that only one host, client or server, can reach the other.

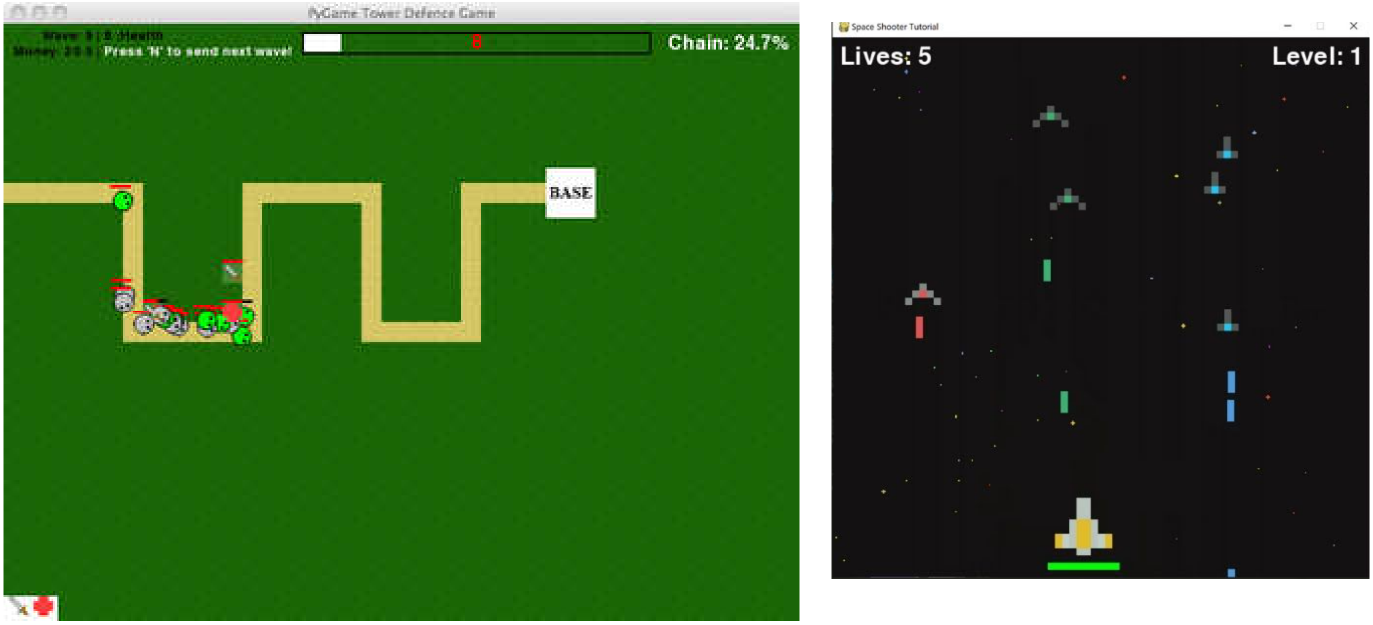
PyGame

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.

Pygame was originally written by Pete Shinners to replace PySDL after its development stalled. It has been a community project since 2000 and is released under the free software GNU Lesser General Public License.

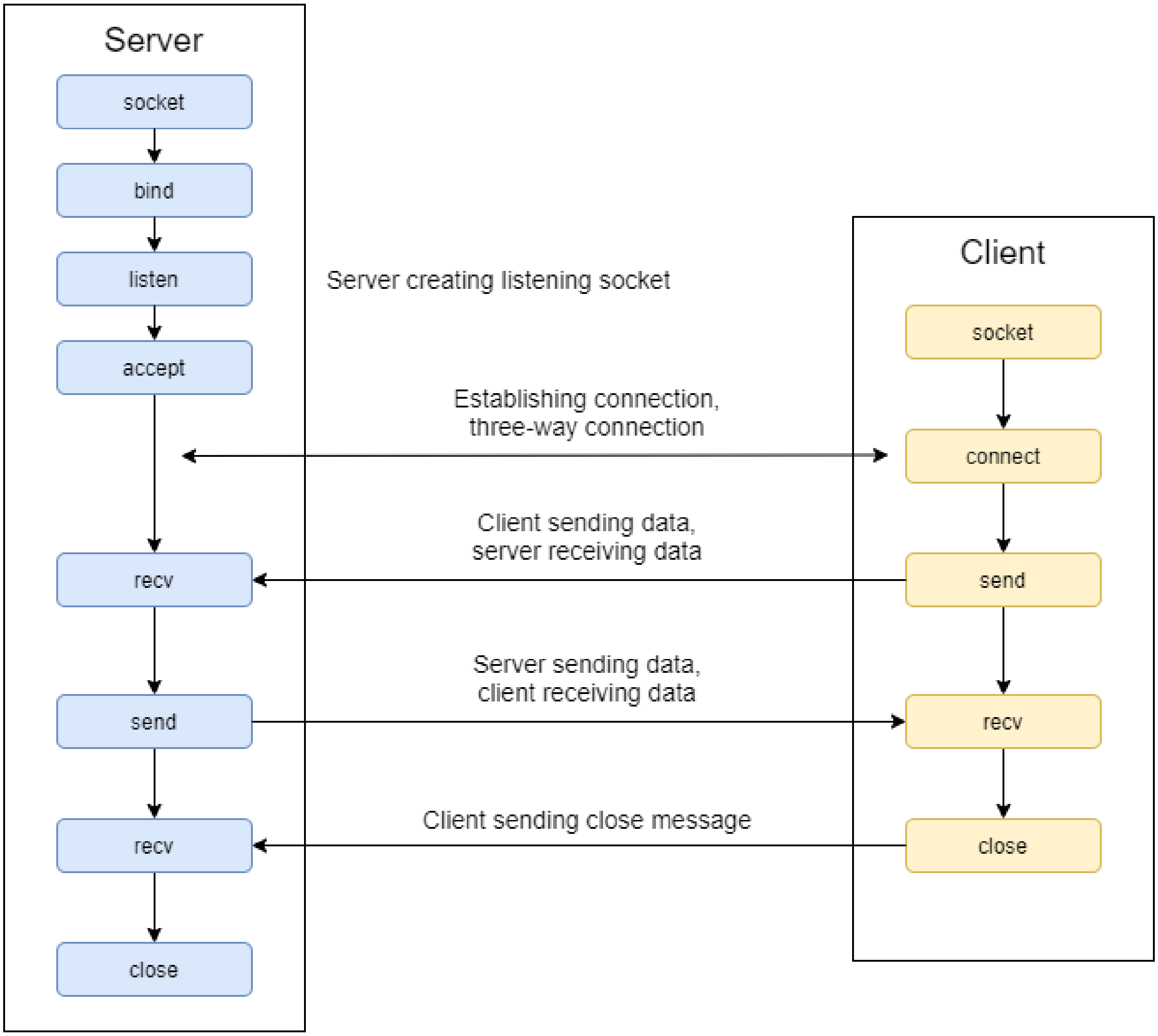
Pygame uses the Simple DirectMedia Layer (SDL) library with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. This is based on the assumption that the most expensive functions inside games can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.

Other features that SDL doesn't have include vector math, collision detection, 2D sprite scene graph management, MIDI support, camera, pixel-array manipulation, transformations, filtering, advanced freetype font support, and drawing.



Some examples of games using PyGame

3. Schematic Diagram



4. Implementation Details

The game makes use of pygame and socket programming library, where initially a server is started and then the players acting as the clients connect to the server to play the multiplayer game.

The whole game is split into several files as : server, client, network and player.

Initially, we create a player object, containing the x and y coordinates in the pygame window, a health variable to indicate the current health of the player, a variable named ‘far’ containing the array of boolean values where far[i] indicates whether the current player is far from the i-th player and a variable named vel to indicate the number of pixels it need to move at a time. This object has the method to draw the circle for a given color, or to load an icon indicating the infected and uninfected players. If the health of the player is zero, then they are considered as infected players, and can infect other players as well. For simplicity, we have made the first player joining the server infected and others uninfected.

The network object is used to send and receive information between the player (client) and the server.

Whenever a server is instantiated by running server.py, a TCP socket with IP 127.0.0.1 (localhost) and port 5555 starts listening for the incoming clients. As soon as the client.py is started, the server creates threads and sends the pre-fed player information based on the client counts to the client. The client, after receiving the player object, draws it in the pygame window and allows the user to move across the pygame window. Now, since it’s a multiplayer game, it is necessary for any player to have the location information about the opponent or other player as well. Thus in order to receive this information, the client sends the current player information to the server, where the server after receiving it, updates in it’s list and sends the information of the remaining player.

In the meantime, the client continuously iterates through the other player's information and updates the 'far' information with respect to the infected player i. If the Euclidean distance between the two points in the pygame coordinate indicating the player position is less than the radius with which they were instantiated, then the far bit of that player is set to false, causing the health of the current player to decrease by a count 1. If the health count of a player becomes zero, that player becomes infected and can infect the neighbouring healthy players if they are in a close proximity. As a result, the color of the player changes to black or the icon indicating the healthy player changes accordingly.

At the same time, the leaderboard indicating the health of all the active players in the server will be updated in the pygame window, as well as the caution "Maintain Distance" will be shown whenever a healthy player comes in contact with the infected player. We have also added a message "Wear mask & be safe" to show constantly throughout the game.

5. Screenshots

When the players are indicated by the circle:

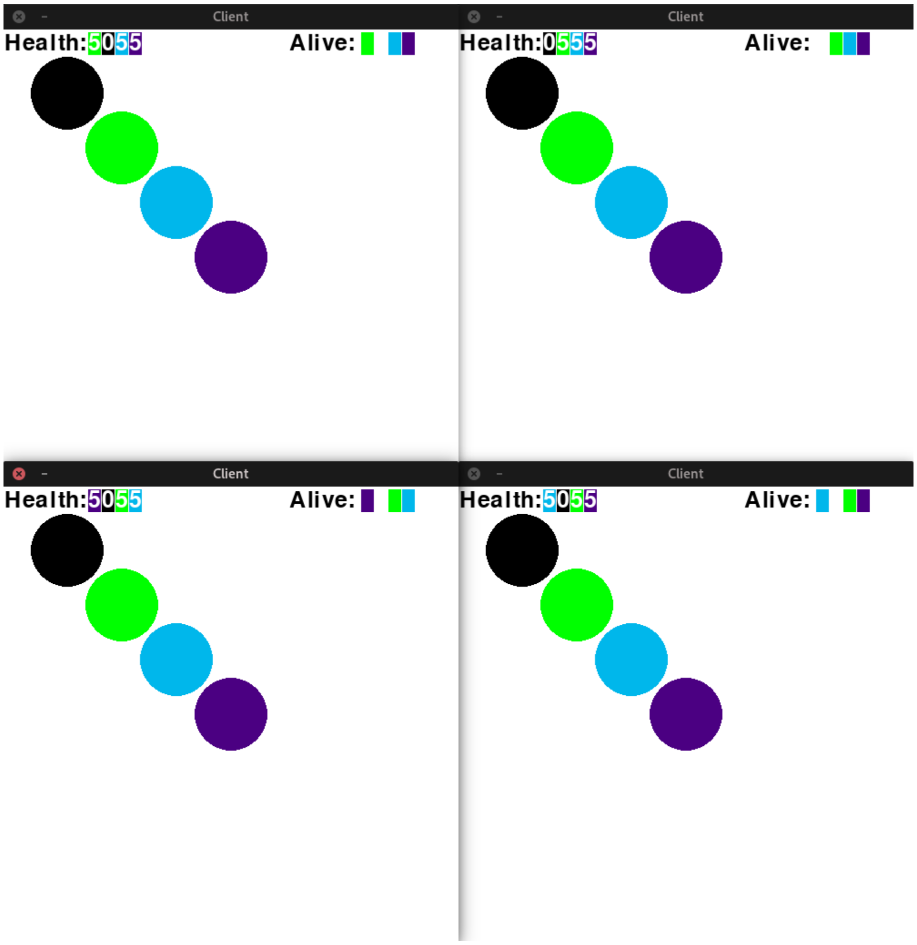


Fig 1. Four players, Initially one player is infected (shown in black color)

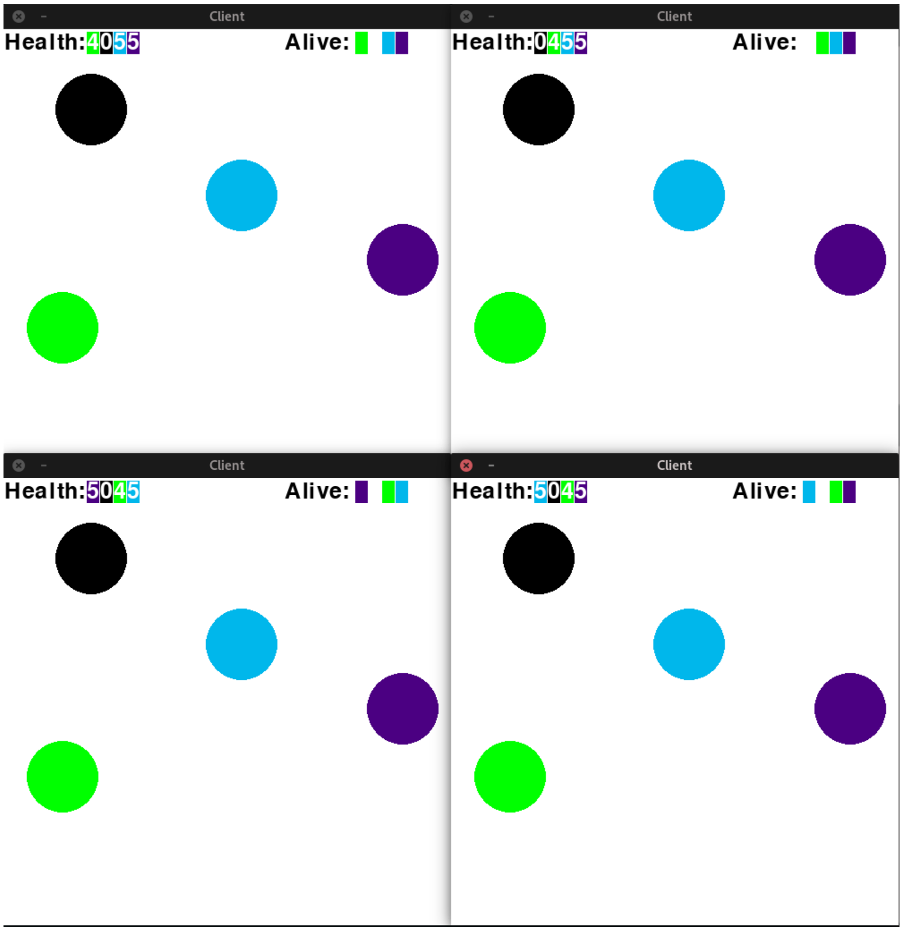
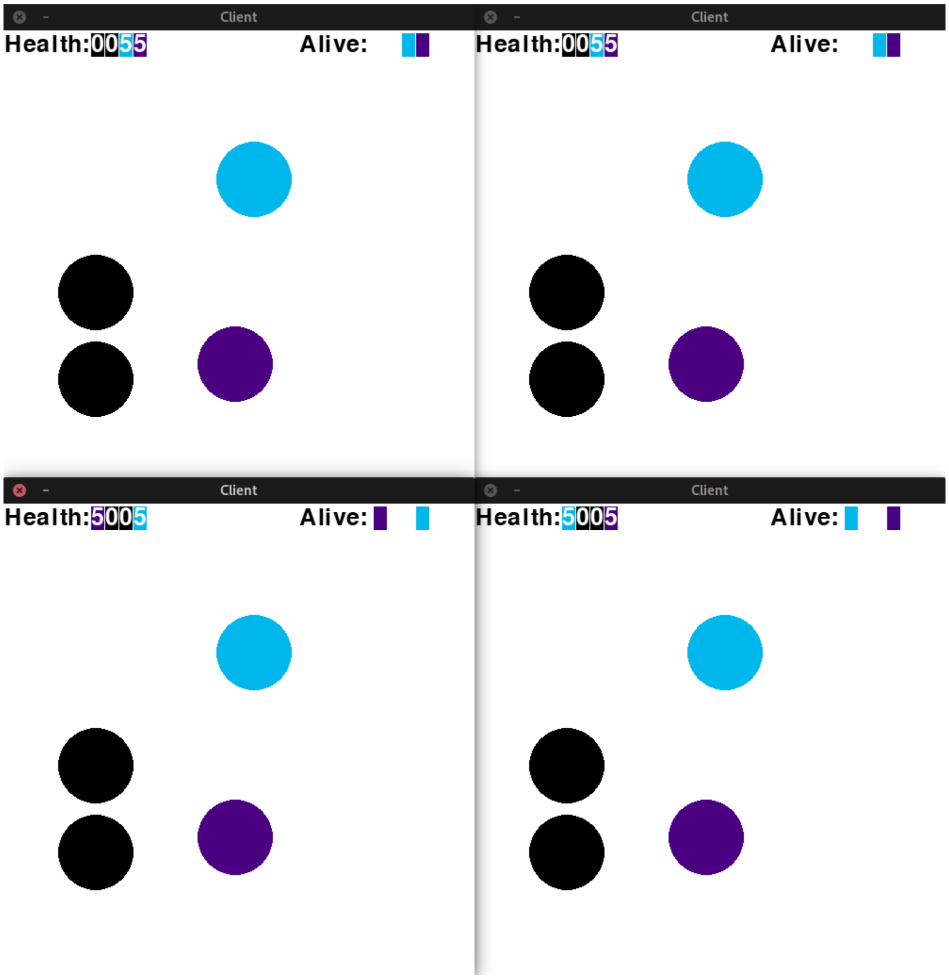


Fig 2. Other player’s health decreases when it comes in contact with infected player



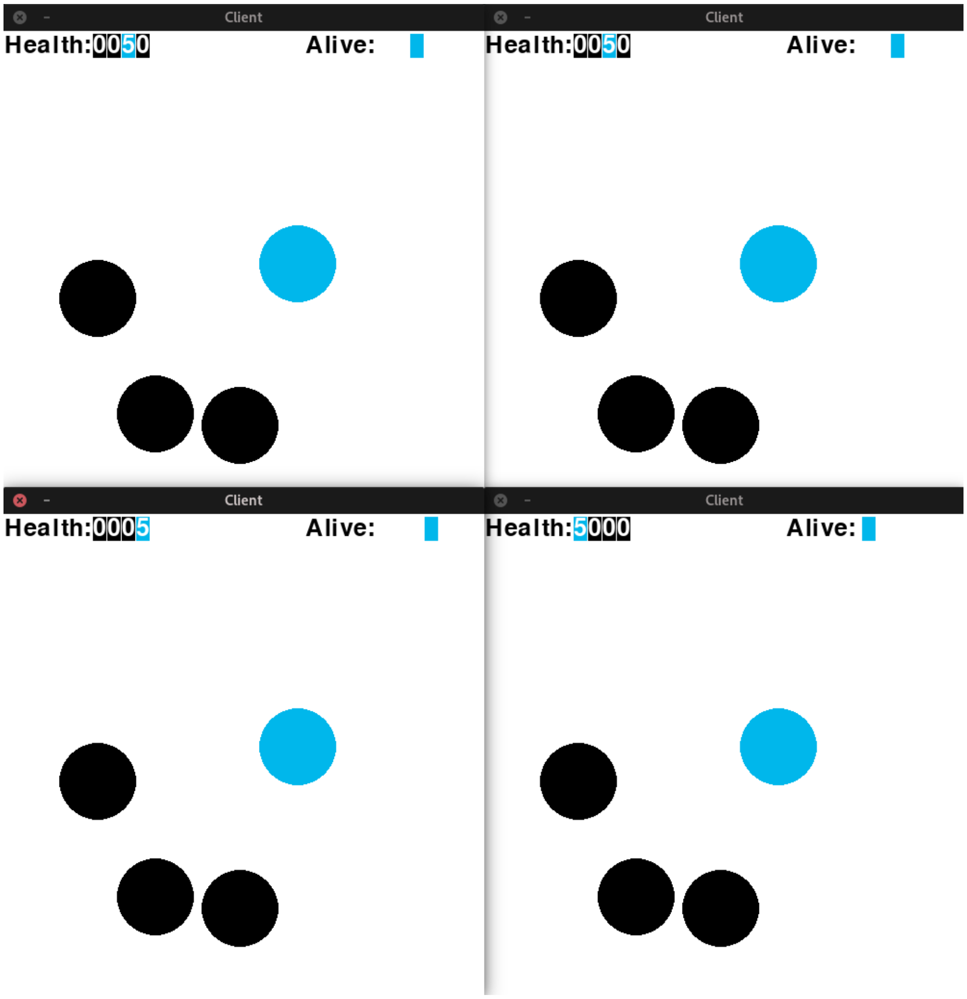
Health:0055 Alive:

Health:0055 Alive:

Health:5005 Alive:

Health:5005 Alive:

Fig 3. Green player gets Infected. And can infect non infected players



Client

Health:0050 Alive:

Client

Health:0050 Alive:

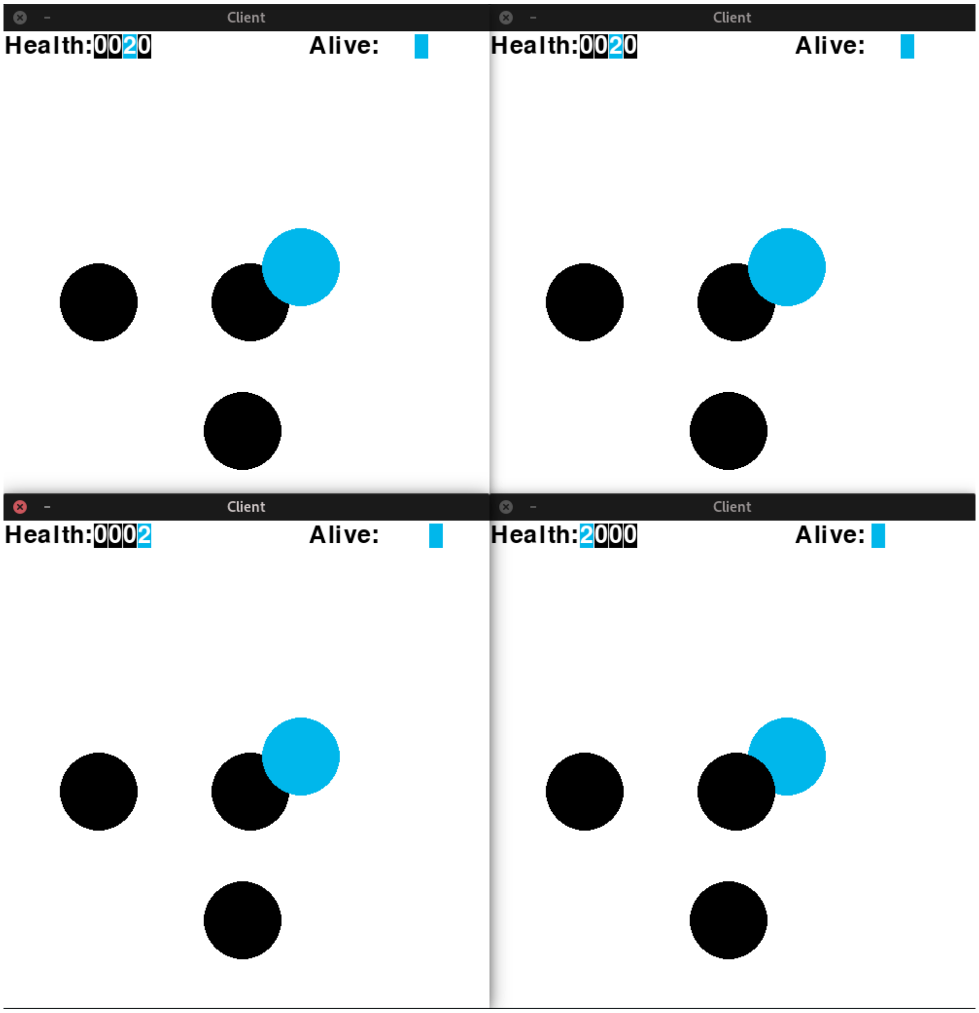
Client

Health:0005 Alive:

Client

Health:5000 Alive:

Fig 4. Purple player gets infected from infected green player



Health:0020 Alive:

Health:0020 Alive:

Health:0002 Alive:

Health:2000 Alive:

Fig 5. Infected purple player decreases non infected blue player’s health

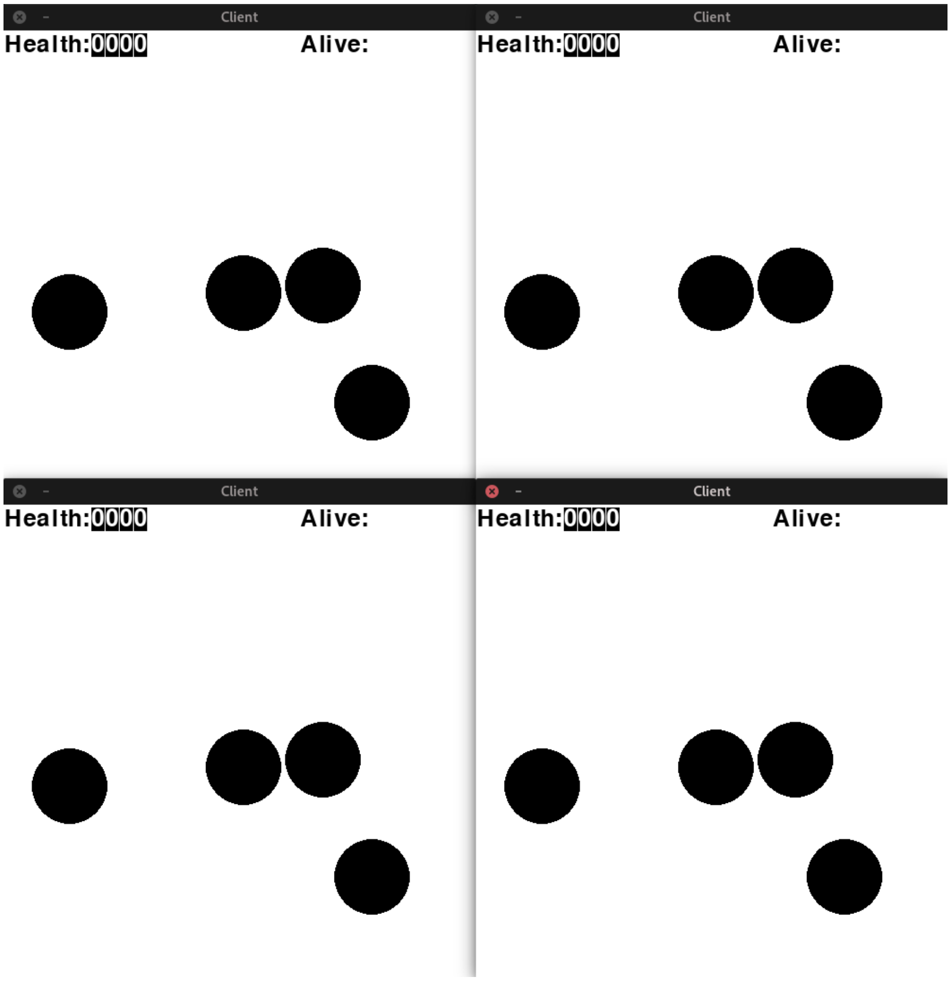


Fig 6. Blue player becomes infected when health reduces to zero

When the players are indicated by the icon:

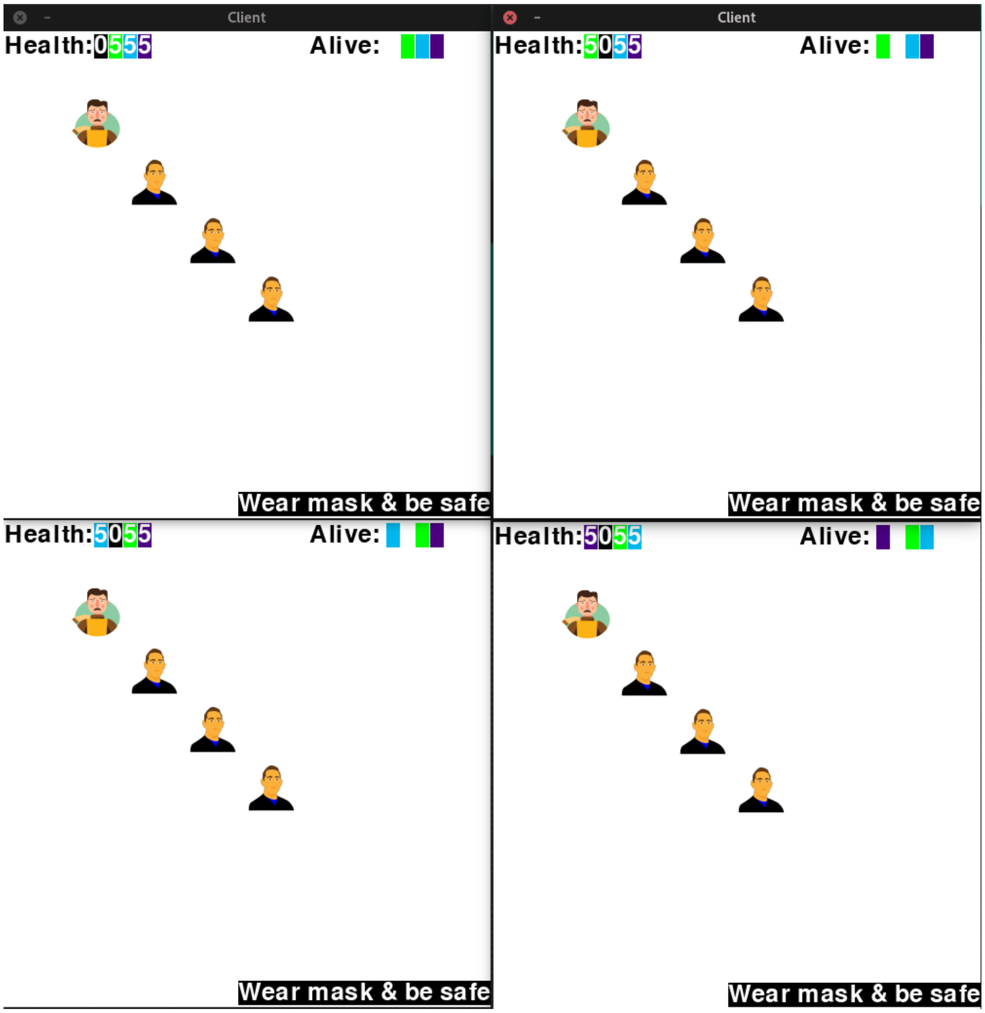
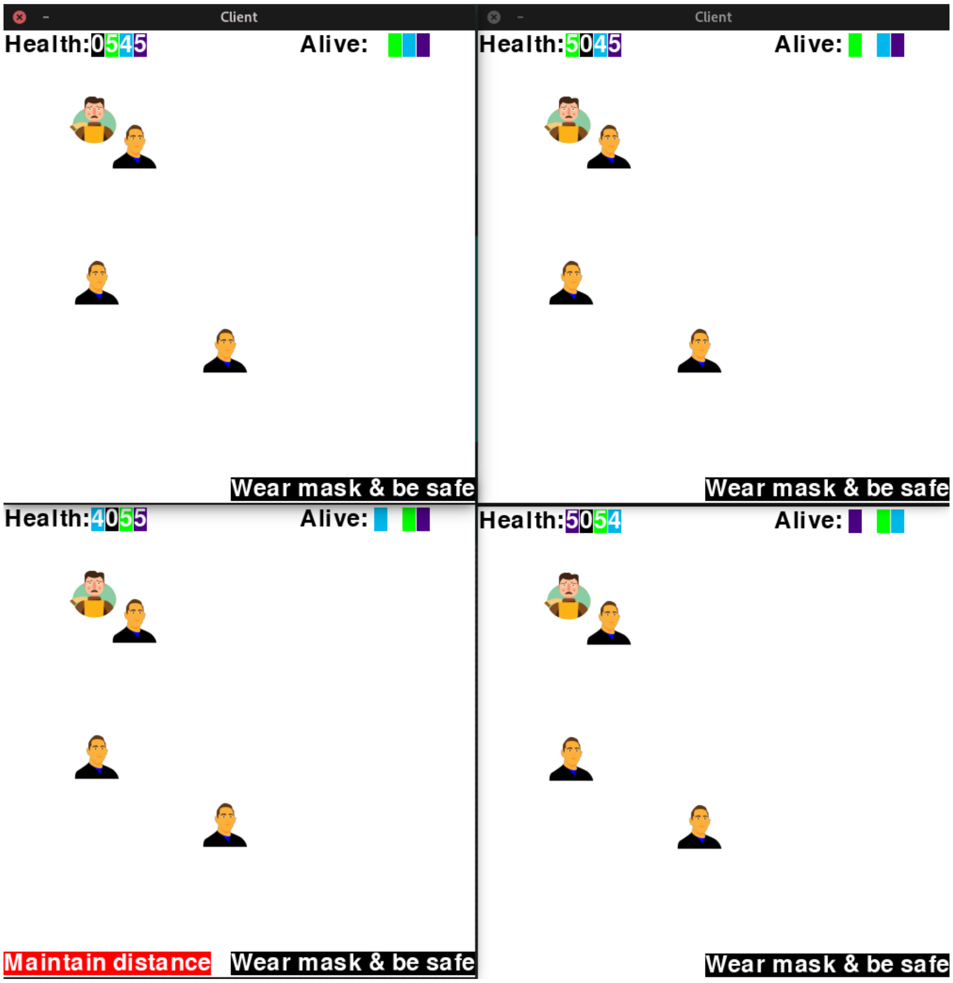


Fig 7. Four players, Initially one player is infected



Health:0545 Alive: 🟩🟦🟪

Wear mask & be safe

Health:5045 Alive: 🟩 🟦🟪

Wear mask & be safe

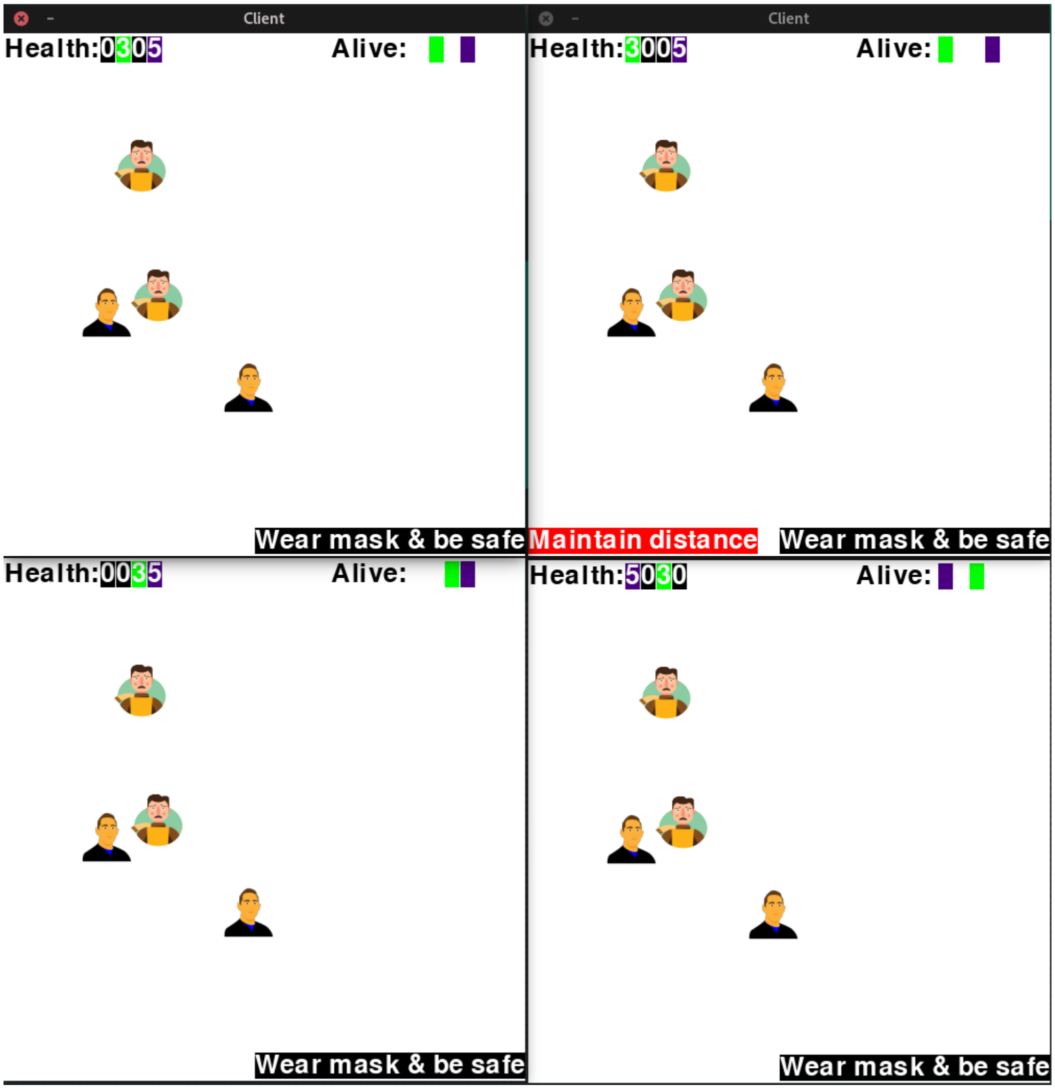
Health:4055 Alive: 🟦 🟩🟪

Maintain distance Wear mask & be safe

Health:5054 Alive: 🟪 🟩🟦

Wear mask & be safe

Fig 8. All the players are free to move. A caution is displayed to healthy player whenever they come in contact with infected player



Health:0305 Alive: 🟩 🟪

Wear mask & be safe

Health:3005 Alive: 🟩 🟪

Maintain distance Wear mask & be safe

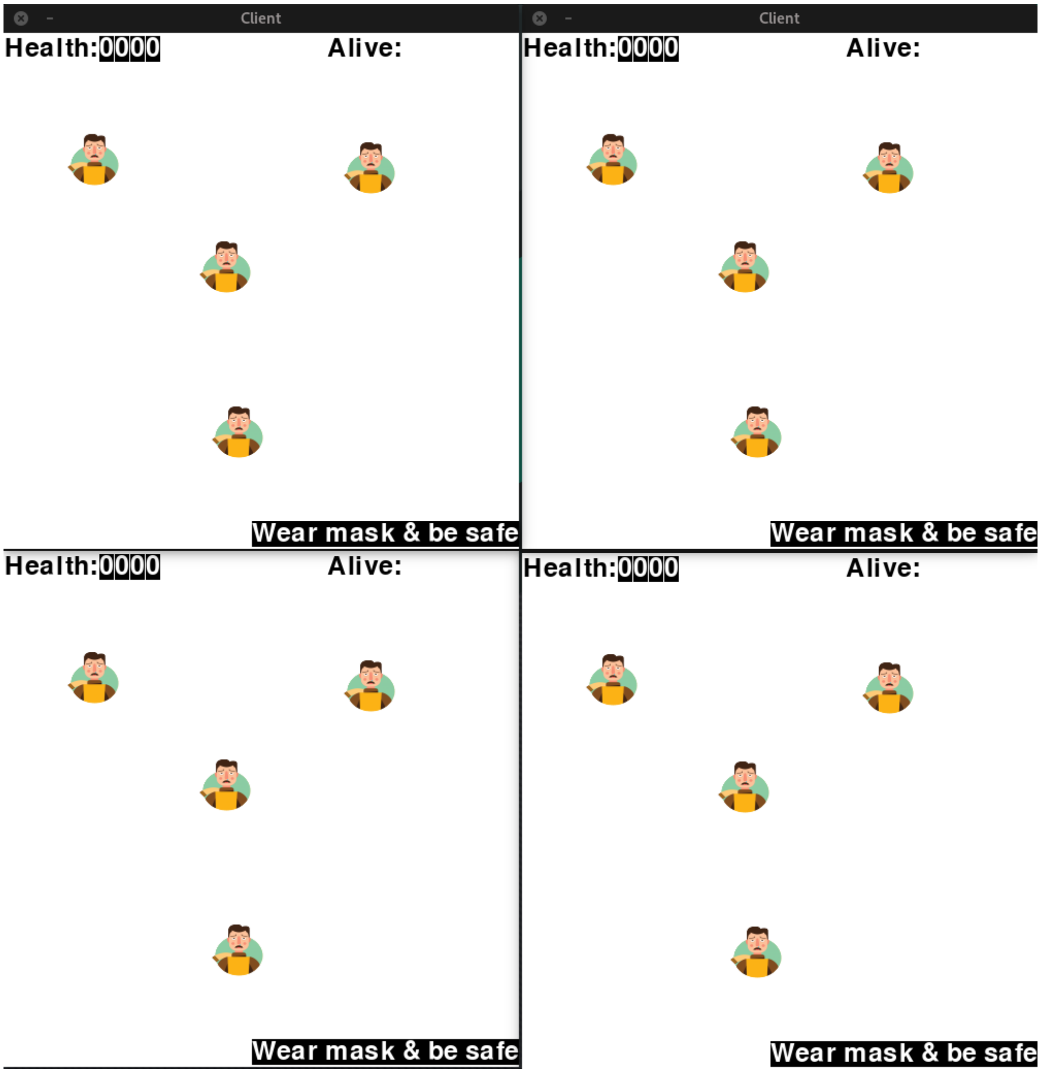
Health:0035 Alive: 🟩🟪

Wear mask & be safe

Health:5030 Alive: 🟪 🟩

Wear mask & be safe

Fig 9. Similar caution shown for player 2 and health being reduced



Health:0000 Alive:

Wear mask & be safe

Health:0000 Alive:

Wear mask & be safe

Health:0000 Alive:

Wear mask & be safe

Health:0000 Alive:

Wear mask & be safe

Fig 10. All players are infected and health reduced to zero

6. References

Understanding socket programming

- ➔ https://www.geeksforgeeks.org/socket-programming-python/
- ➔ https://realpython.com/python-sockets/
- ➔ https://www.edureka.co/blog/socket-programming-python/

Learning Pygame

- ➔ https://realpython.com/pygame-a-primer/
- ➔ https://www.javatpoint.com/pygame
- ➔ https://www.edureka.co/blog/pygame-tutorial
- ➔ https://www.pygame.org/docs/

Game Development

- ➔ https://www.techwithtim.net/tutorials/python-online-game-tutorial/
- ➔ https://www.techwithtim.net/tutorials/python-online-game-tutorial/online-rock-paper-scissors-p1/
- ➔ https://www.youtube.com/watch?v=-3B1v-K1oXE
- ➔ https://github.com/PlainSight/pygameblog

7. Conclusion

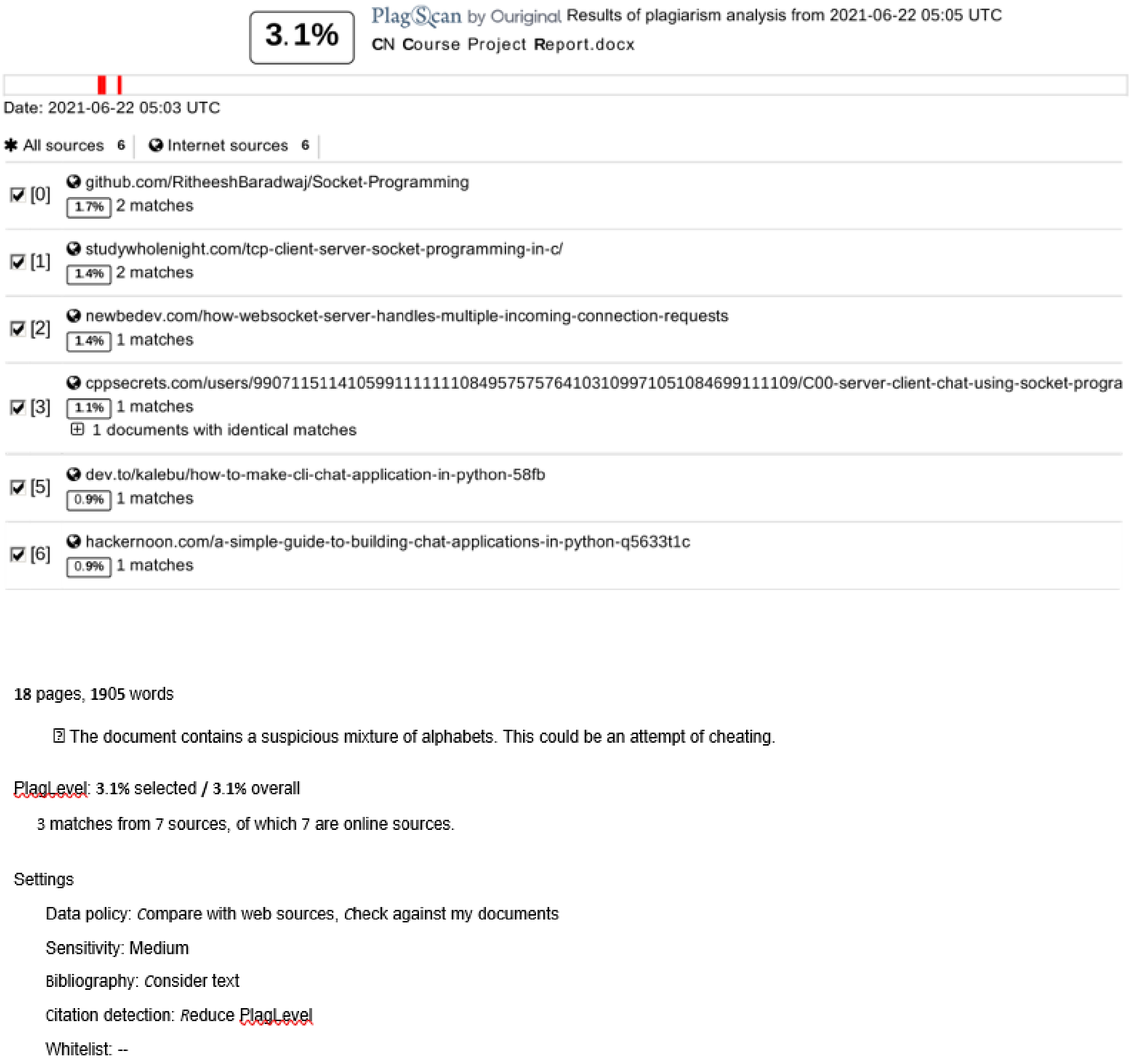
The application very well simulates the Covid-19 outbreak and puts a prime focus on the prevention measures. The player is signalled to maintain social distance if he is not maintaining a minimum predefined distance, alerting ‘Maintain Social Distance’. It also shows a message to ‘Wear a mask and be Safe’.

If followed properly, these prevention measures can bring a significant impact in reducing the spread. We have made an effort to bring out a good social message in the form of a game and hope to create a good impact.

Learnings from the project

In the run of development, we have learnt the concept of how a client/server system works, developing applications to an external server, how data such as objects is sent over ports and how to connect multiple clients to a server. We have also gained considerable knowledge about socket programming and game development.

8. Plagiarism Report



3.1% PlagScan by Ouriginal Results of plagiarism analysis from 2021-06-22 05:05 UTC
CN Course Project Report.docx

Date: 2021-06-22 05:03 UTC

✱ All sources 6 | Internet sources 6

☑ [0] github.com/RitheeshBaradwaj/Socket-Programming
1.7% 2 matches

☑ [1] studywholenight.com/tcp-client-server-socket-programming-in-c/
1.4% 2 matches

☑ [2] newbedev.com/how-websocket-server-handles-multiple-incoming-connection-requests
1.4% 1 matches

☑ [3] cppsecrets.com/users/9907115114105991111111084957575764103109971051084699111109/C00-server-client-chat-using-socket-progra
1.1% 1 matches
⊞ 1 documents with identical matches

☑ [5] dev.to/kalebu/how-to-make-cli-chat-application-in-python-58fb
0.9% 1 matches

☑ [6] hackernoon.com/a-simple-guide-to-building-chat-applications-in-python-q5633t1c
0.9% 1 matches

18 pages, 1905 words

⍰ The document contains a suspicious mixture of alphabets. This could be an attempt of cheating.

PlagLevel: 3.1% selected / 3.1% overall

3 matches from 7 sources, of which 7 are online sources.

Settings

Data policy: Compare with web sources, Check against my documents

Sensitivity: Medium

Bibliography: Consider text

Citation detection: Reduce PlagLevel

Whitelist: --