VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi - 590018

Mini Project Report

on

"GRAPHICAL VISUALIZATION OF CLIENT-SERVER COMMUNICATION"

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE & ENGINEERING

by

ASHWITHA JATHAN	**4MT19CS032**
HRITHIKA BHAT	**4MT19CS059**

Under the Guidance of

Ms. JEEVITHA SAMPATH
(Assistant Professor)

Ms. SHWETHA R J
(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(Accredited by NBA)

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

Accredited by NAAC with A+ Grade, An ISO 9001: 2015 Certified Institution
(*A Unit of Rajalaxmi Education Trust®, Mangalore - 575001*)
Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi
Badaga Mijar, Moodabidri-574225, Karnataka

2021-22

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING
Accredited by NAAC with A+ Grade, An ISO 9001: 2015 Certified Institution
(A Unit of Rajalaxmi Education Trust®, Mangalore - 575001)
Affiliated to V.T.U., Belagavi, Approved by AICTE, New Delhi.

Department of Computer Science & Engineering
(Accredited by NBA)

CERTIFICATE

This is to certify that the mini project entitled **GRAPHICAL VISUALIZATION OF CLIENT- SERVER COMMUNICATION** is a bonafide work carried out by ASHWITHA JATHAN 4MT19CS032 and HRITHIKA BHAT 4MT19CS059 in partial fulfilment for the requirement of 6th semester Computer Graphics Laboratory with mini project (18CSL67). It is certified that all the corrections / suggestions indicated for the Internal Assessment have been incorporated in the project. The mini project has been approved as it satisfies the academic requirement in respect of the 18CSL67 prescribed for the 6th Semester B.E in Computer Science & Engineering Program by the **Visvesvaraya Technological University, Belagavi**, for the academic year 2021 – 2022.

.....	.....	.....
Ms. Jeevitha Sampath	**Ms. Shwetha R J**	**Mr. Ravinarayana B**
Assistant Professor	Assistant Professor	Associate Professor & HOD
MITE, Moodabidri	MITE, Moodabidri	MITE, Moodabidri

Name of the Examiners	**Signature with Date**
1. ……..	**……..**
2. ……..	**……**

ABSTRACT

Client-Server communication demonstrates the methods used to communicate between client and server and also between two clients. Through this project the aim is to show the various methods of how the communication can take place along with the steps for the same. The TCP communication stands for Transmission Control Protocol a communications standard that enables application programs and computing devices to exchange messages over a network. It is designed to send packets across the internet and ensure the successful delivery of data and messages over networks. TCP organizes data so that it can be transmitted between a server and a client. It guarantees the integrity of the data being communicated over a network. Before it transmits data, TCP establishes a connection between a source and its destination. It then breaks large amounts of data into smaller packets, while ensuring data integrity is in place throughout the process. The UDP on the other hand stands for User Datagram Protocol is a communication protocol that is primarily used to establish low-latency and loss-tolerating connections between applications on the internet. UDP speeds up transmissions by enabling the transfer of data before an agreement is provided by the receiving party. Thus, beneficial in time-sensitive communication. Also, the communication between two clients where in one client uploads data which is downloaded by the other client and vice versa is demonstrated.

ACKNOWLEDGEMENT

The successful completion of any significant task is the outcome of invaluable aggregate combination of different people in radial direction explicitly and implicitly. We would therefore take opportunity to thank and express our gratitude to all those without whom the completion of project would not be possible.

We express our thanks to **Ms. Jeevitha Sampath, Assistant Professor and Ms. Shwetha R J,Assistant Professor** Department of Computer Science and Engineering for having provided all the facilities that helped us in timely completion of this project.

We express our sincere gratitude to **Mr. Ravinarayana B, Associate Professor, Head of the Department, Computer Science and Engineering** for his support and guidance.

We would like to thank **Dr. M S Ganesha Prasad, Principal, Mangalore Institute of Technology andEngineering, Moodabidri** for his support and encouragement.

We express our sincere gratitude to our institution and management for providing us with good infrastructure, laboratory facilities, qualified and inspiring staffs, and whose guidance was of immense help in completion of this seminar successfully.

Ashwitha Jathan (4MT19CS032)

Hrithika Bhat (4MT19CS059)

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Computer Graphics

Graphics are defined as any sketch or a drawing or a special network that pictorially represents some meaningful information. Computer Graphics is used where a set of images needs to be manipulated or the creation of the image in the form of pixels and is drawn on the computer. Computer Graphics can be used in digital photography, film, entertainment, electronic gadgets, and all other core technologies which are required. It is a vast subject and area in the field of computer science. Computer Graphics can be used in UI design, rendering, geometric objects, animation, and many more.

Computer Graphics refers to several things:

- The manipulation and the representation of the image or the data in a graphical manner.
- Various technology is required for the creation and manipulation.
- Digital synthesis and its manipulation.

Types of Computer Graphics

- **Raster Graphics:** In raster, graphics pixels are used for an image to be drawn. It is also known as a bitmap image in which a sequence of images is into smaller pixels. Basically, a bitmap indicates a large number of pixels together.
- **Vector Graphics:** In vector graphics, mathematical formulae are used to draw different types of shapes, lines, objects, and so on.

Applications

- **Computer Graphics are used for** an **aided design for engineering and architectural system-** These are used in electrical automobiles, electro-mechanical, mechanical, electronic devices. For example, gears and bolts.
- **Computer Art –** MS Paint.
- **Presentation Graphics –** It is used to summarize financial statistical scientific or economic data. For example- Bar chart, Line chart.
- **Entertainment-** It is used in motion pictures, music videos, television gaming.

- **Education and training-** It is used to understand the operations of complex systems. It is also used for specialized system such for framing for captains, pilots and so on.
- **Visualization-** To study trends and patterns. For example- Analysing satellite photo of earth.

1.2 OpenGL

Open Graphics Library (OpenGL) is a cross-language (language independent), cross-platform (platform-independent) API for rendering 2D and 3D Vector Graphics (use of polygons to represent image). OpenGL API is designed mostly in hardware.

- **Design:** This API is defined as a set of functions which may be called by the client program. Although functions are similar to those of **C** language but it is language independent.
- **Development:** It is an evolving API and **Khronos Group** regularly releases its new version having some extended feature compare to previous one. GPU vendors may also provide some additional functionality in the form of extension.
- **Associated Libraries:** The earliest version is released with a companion library called OpenGL utility library. But since OpenGL is quite a complex process. So, in order to make it easier other library such as OpenGL Utility Toolkit is added which is later superseded by free glut. Later included library were GLEE, GLEW, and gliding.
- **Implementation:** Mesa 3D is an open-source implementation of OpenGL. It can do pure software rendering and it may also use hardware acceleration on BSD, Linux, and other platforms by taking advantage of Direct Rendering Infrastructure.

Chapter 2

REQUIREMENT ANALYSIS AND SPECIFICATION

2.1 Functional Requirements

The functional requirements are the requirements which are needed to develop the software application.

The functional requirements are broadly classified into 2 categories, they are:

1. Hardware Requirements
2. Software Requirements

Hardware requirements:

For the hardware requirements the SRS (Software requirement specification) specifies the logical characteristics of each interface between the software product and the hardware components. It specifies the hardware requirements like memory restrictions, cache size, the processor, RAM size etc. Those are required for the software to run.

- Processor 2GHZ frequency or above
- Hard disk: minimum1GB disk space or more
- RAM size: 2GB or more
- Display: 800x600 or higher resolution display with 256 colors.
- Mouse: Standard serial mouse
- Keyboard: Standard QWERTY Keyboard

Software requirements:

- Operating System: Windows 7 or above
- Software used: Dev- C++
- Programming language: C language
- Graphics Library: GL and GLU/GLUT
- Compiler used: MinGW Compiler

2.2 Non-Functional Requirements

Non-functional requirements are requirements that are not directly concerned with the Specific functions delivered by the system. They may relate to emergent system properties such as reliability, response time and store occupancy. Alternatively, they may define constraints on the system such as the capabilities of I/O devices and the data Representations used in system interfaces. The plan for implementing functional requirements is detailed in the systemdesign. The plan for implementing non-functional requirements is detailed in the system architecture. Non-functional requirements are often Called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral Requirements". Qualities, that are non-functional requirements, can be divided into two Main categories: Execution qualities, such as security and usability, which are observable at Run time.

- Performance-for example: response time, throughput, utilization, static volumetric.
- Scalability
- Capacity
- Availability
- Reliability
- Recoverability
- Maintainability
- Serviceability

Chapter 3

SYSTEM DESIGN

3.1 Description of Project

A typical Internet service is implemented in two parts—a server that provides information and one or more clients that request information. Such client/server architecture has been gaining popularity as an approach for implementing distributed information systems. The client/server architecture typically consists of a collection of computers connected by a communication network. The functions of the information system are performed by processes (computer programs) that run on these computers and communicate through the network. It takes two sockets to complete a communication path. When two processes communicate, they use the client/server model to establish the connection. The server application listens on a specific port on the system—the server is completely identified by the IP address of the system where it runs and the port number where it listens for connections. The client initiates connection from any available port and tries to connect to the server. Once the connection is established, the client and the server can exchange data according to their own protocol.

The sequence of events in sockets-based data exchanges depends on whether the transfer is connection oriented (TCP) or connectionless (UDP). For a connection-oriented data transfer using TCP sockets, the server “listens” on a specific port, waiting for clients to request connection. Data transfer begins only after a connection is established. The server is a program that responds when a connection is attempted at a certain port. For connectionless data transfers using UDP sockets, the server waits for a datagram to arrive at a specified port. The client does not wait to establish a connection; it simply sends a datagram to the server.

3.2 Algorithm

An algorithm is a procedure used for solving a problem or performing a computation. Algorithms act as an exact list of instructions that conduct specified actions step by step in either hardware- or software-based routines. Algorithms are widely used throughout all areas of IT. In mathematics and computer science, an algorithm usually refers to a small procedure that solves a recurrent problem. Algorithms are also used as specifications for performing data processing and play a major role in automated systems.

Step 1: Initialize the Graphics library.
Step 2: Display the Scene 1.
Step 3: Take the user input.
If (input==’A’|| input==’a’)
Proceed to Scene 2.
End if
Step 4: Take the user input.
If (input==’S’|| input==’s’)
Proceed to Scene 3.
End if
Step 5: Left- click mouse

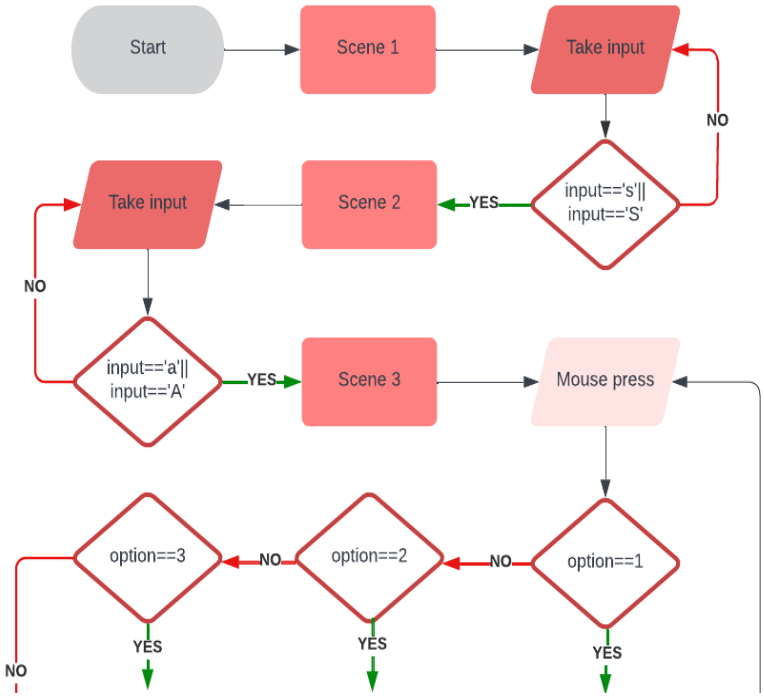
```
If (option ==1) {
    If (op ==1)
        Handshaking_property()
    End if
    If (op== 2)
        Upload_tcp()
    End if
    If (op== 3)
        Download_tcp()
    End if
} End if
If (option ==2) {
    If (op== 1)
        Upload_udp()
    End if
    If (op== 2)
        Download_udp()
    End if
} End if
If (option ==3) {
    If (op== 1)
        Upload_client1()
    End if
    If (op== 2)
        Download_client2()
    End if
} End if
```

```
If (option ==4) {
        If (op== 1)
                Upload_client2()
        End if
        If (op== 2)
                Download_client1()
        End if
} End if
If (option ==5)
        Exit ()
End if
```

Figure 3.2.1 Algorithm for Client-Server Communication

3.3 Flowchart

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams. Flowcharts, sometimes spelled as flow charts, use rectangles, ovals, diamonds and potentially numerous other shapes to define the type of step, along with connecting arrows to define flow and sequence.



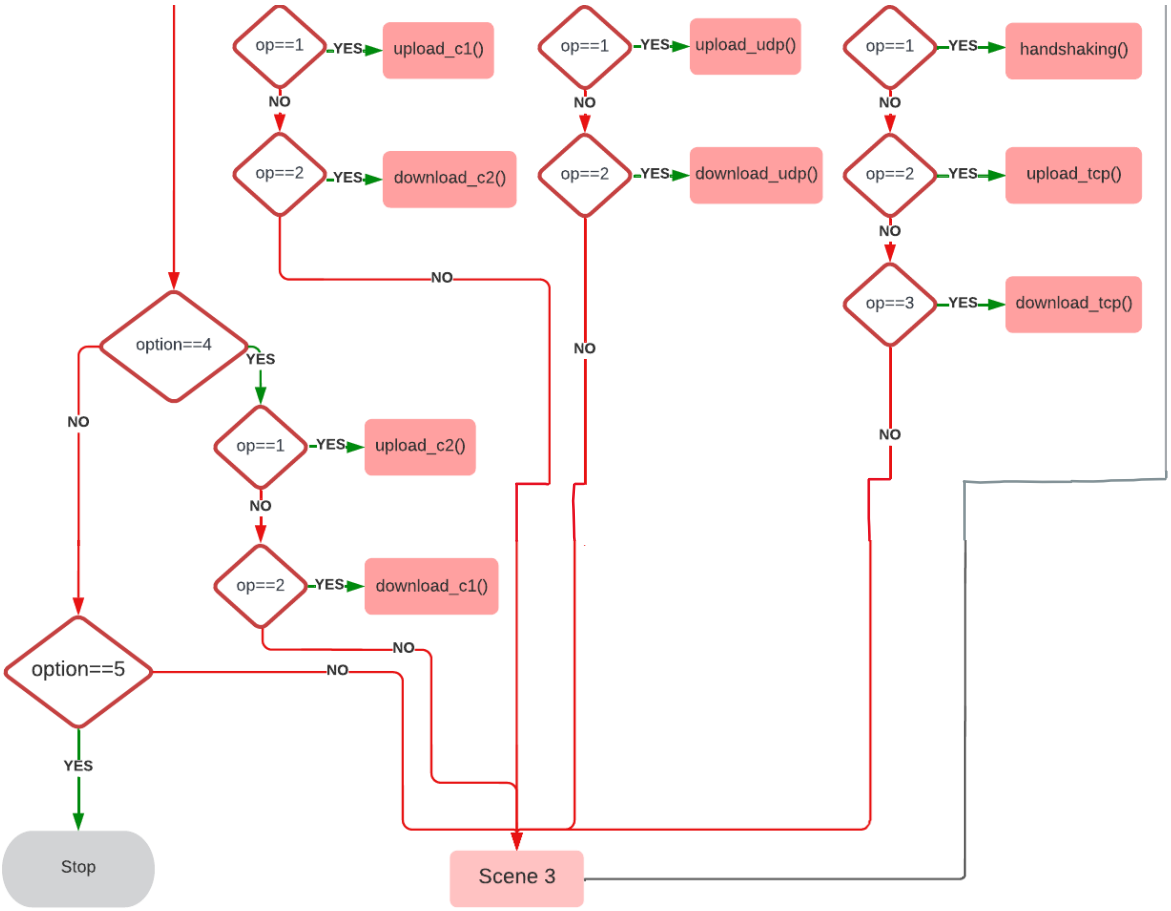


Figure 3.3.1 Flow Chart for Client-server communication algorithm

Chapter 4

IMPLEMENTATION

4.1 Overview

Implementation is the carrying out, execution, or practice of a plan, a method, or any design for doing something. As such, implementation is the action that must follow any preliminary thinking in order for something to actually happen. In an information technology context, implementation encompasses all the processes involved in getting new software or hardware operating properly in its environment, including installation, configuration, running, testing, and making necessary changes. The word deployment is sometimes used to mean the same thing. This project has been developed using OpenGL provided by Dev C++.

4.2 Built-in Functions

The following are list of OpenGL functions used in the project:

- **Void glutInit(int *argc, char **argv):** The glutInit will initialize the GLUT library, the arguments form main are passed in and can used by the application. This will negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to user if GLUT cannot be properly initialized.
- **Void glutInitDisplayMode(unsigned int mode):** The glutInitDisplayMode sets the initial display mode. It requests a display with the properties in mode. The value of mode is determined by the logical OR of option including the color model (GLUT_RGB), buffering (GLUT_DOUBLE) and (GLUT_DEPTH).
- **Void glutInitWindowSize(int width, int height):** This function specifies the initial height and width of the width in pixels .
- **Void glutInitWindowPosition(int x, int y):** This function specifies the initial position of the top left corner of the windows in pixels.
- **Void glutCreateWindow(char *title):** The glutCreateWindow create a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

- **void glClear(GL_COLOR_BUFFER_BIT):** Clears all buffer whose bits are set in mask. The mask is formed by the logical OR of values defined in gl.h GL_COLOR_BUFFER_BIT refers to color buffers.
- **void MatrixMode(GL_PROJECTION):** Projection matrixes are stored inOpenGL projection mode. So, to set the projection transformation matrix. That mode is invoked through the statement, glMatrixMode(GL_PROJECTION).
- **void LoadIdentity():** Sets the current transformation matrix to an identity matrix.
- **void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdoublenearVal, GLdouble farVal):** Sets up a three - dimensional orthographic viewing region.
- **void glutKeyboardFunc(void(*func)(unsigned char key,int x, int y)):** glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback.
- **void glutDisplayFunc():** glutDisplayFunc sets the display callback for the current window.
- **void glutMainLoop():** Cause the program to enter an event-processing loop. It should be the last statement in the main.
- **void glColor3f(GLfloat red, GLfloat green, GLfloat blue):** This function sets present RGB colors. Different color is given to object using the colors parameters such as red, green, blue. The maximum and minimum values of the floating-point types are 1.0 and 0.0 respectively.
- **void glRasterPos3f ():** specify the raster position for pixel operations.
- **void glutBitmapCharacter(void *font, int character):** glutBitmapCharacter renders a bitmap character using OpenGL.
- **void glutPostRedisplay():** Mark the normal plane of current window as needing to be redisplayed. The next iteration through glutMainLoop, the window's display call back will be called to redisplay the window's normal plane. Multiple calls to glutPostRedisplay before the next display call back opportunity generates only a single redisplay call back. GlutPostRedisplay may be called within a window's display or over lay display call back

to remark that windows for redisplay.

- **void glutTimerFunc():** glutTimerFunc registers a timer callback to be triggered in a specified number of milliseconds.
- **void glPushMatrix():** Push the current matrix stack.
- **void glPopMatrix():** Pop the current matrix stack.
- **void glTranslatef(GLfloat x, GLfloat y, GLfloat z):** Multiply the current matrix by a translation matrix.
- **void glScalef(GLrotate angle, GLfloat x, GLfloat y, GLfloat) :** Multiply the current matrix by a general scaling matrix.
- **void glLineWidth(width):** Line width is set in OpenGL with function glLineWidth(width). Floating point value is assigned to parameters width, and this value is rounded to nearest nonnegative integer.
- **void glBegin() and void glEnd():** The glBegin and glEnd functions delimit the vertices of a primitive or a group of like primitives. Syntax of glBegin is as follows: glBegin(GLenum mode);
- **void glVertex2f(x,y):** The glVertex function commands are used within glBegin/glEnd pairs to specify point, line, and polygon vertices.
- **void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks):** glutSolidSphere and glutWireSphere render a solid or wireframe sphere respectively.
- **void glutSwapBuffers(void):** glutSwapBuffers swaps the buffers of the current window if double buffered.
- **void glFlush(void):** The glFlush function forces execution of OpenGL function in finite time. This function has no parameters. This function does not return a value.

4.3 User-defined Functions

The following are list of user-defined functions used in the project:

- **void upload (), void download ()**: These functions display the messages “Uploading” and “Downloading” respectively.
- **void uploaded (), void downloaded ():** These functions display the messages “Uploaded” and “Downloaded” respectively.

- **void handshake ():** This function displays the message “Handshaking property”.
- **void setFont(void *font):** This function sets the current font style.
- **void drawstring(float x,float y,float z,char *string):** This function sets the position/transform for the string for it to be displayed.
- **void frontscreen(void):**
- **void lines (int x, int y), void lines1(int X, int Y):** These functions draw the lines for handshaking property at the given position pointwise.
- **void lines2(int x, int y), void lines3(int X, int Y):** These functions draw the lines for uploading and downloading data for TCP at the given position pointwise.
- **void lines4(int x1, int y11), void lines5(int x2, int y2):** These functions draw the lines for uploading and downloading data for UDP at the given position pointwise.
- **void lines6(int x3, int y3), void lines7(int x4, int y4):** These functions draw the lines for uploading and downloading data for client1 and client2 respectively at the given position pointwise.
- **void lines8(int x5, int y5), void lines9(int x6, int y6):** These functions draw the lines for uploading and downloading data for client2 and client1 respectively at the given position pointwise.
- **void op (int id):** This function is used to check cases for the menu attached to the left mouse button that is visible in the scene 3.
- **void systems ():** This function designs the scene 3 which consists of all the four client systems involved in the client-server architecture demonstration, along with the intricate detailing of the systems.
- **void server ():** This function designs the scene 3 which consists of the server involved in the client-server architecture demonstration, along with the intricate detailing of the server.
- **void names ():** This function is used to design the scene 1 which consists of the bitmap string displaying the college, department, subject, project title and student details.
- **void display1():** This function calls the functions for scene 2, scene 1 along with the creation of the menu for the scene 3.
- **void display(void):** This function designs the scene 2 with the detailing related to the same.

Chapter 5

SNAPSHOTS

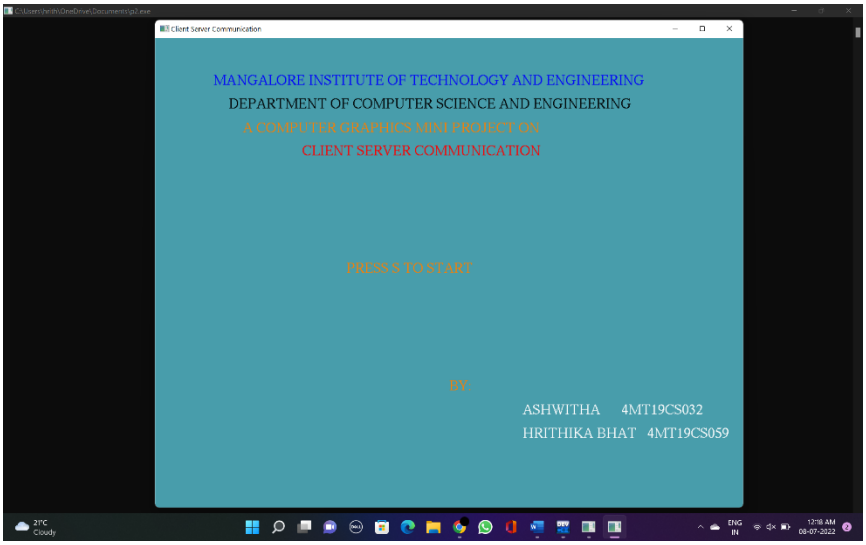


Figure 5.1 Screenshot Of Scene 1

Figure 5.1 indicates Scene 1 showing the project and student details.

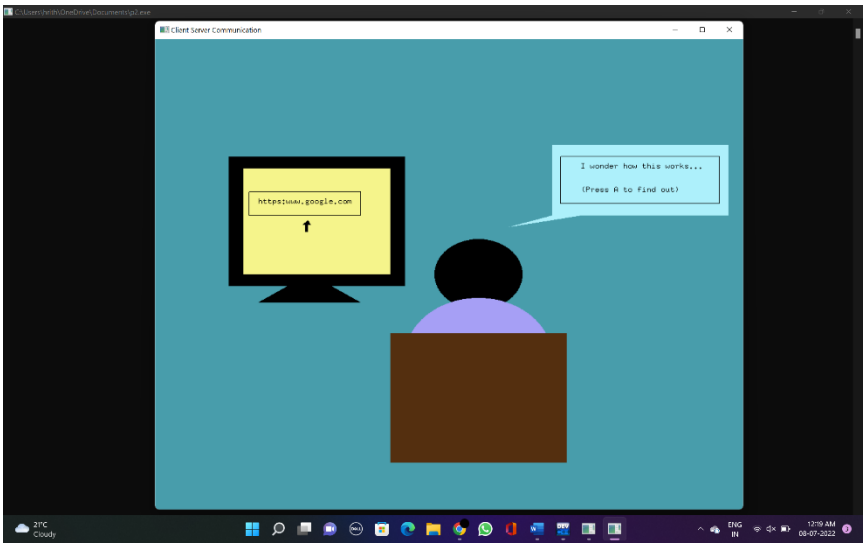


Figure 5.2 Screenshot Of Scene 2

Figure 5.2 indicates Scene 2 wherein a student wonders how the internet works while browsing.

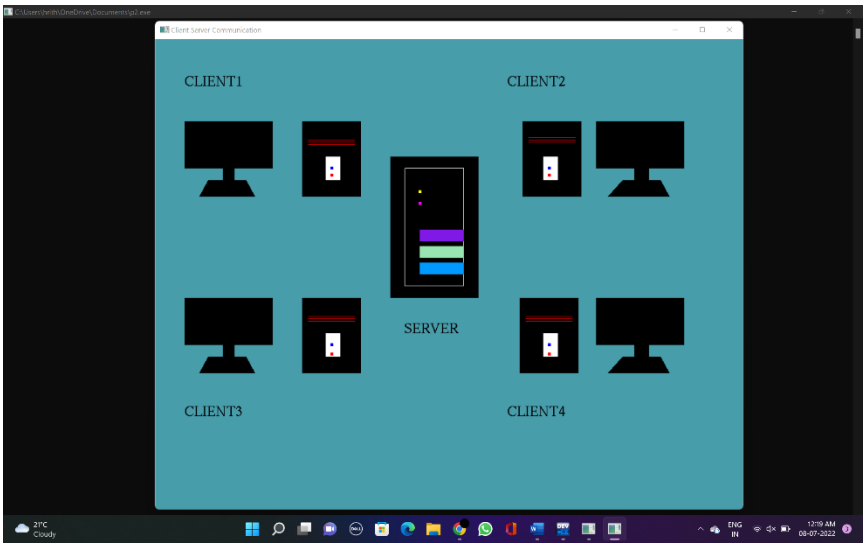


Figure 5.3 Screenshot Of Scene 3

Figure 5.3 contains the Scene 3 which includes the client systems along with the server system.

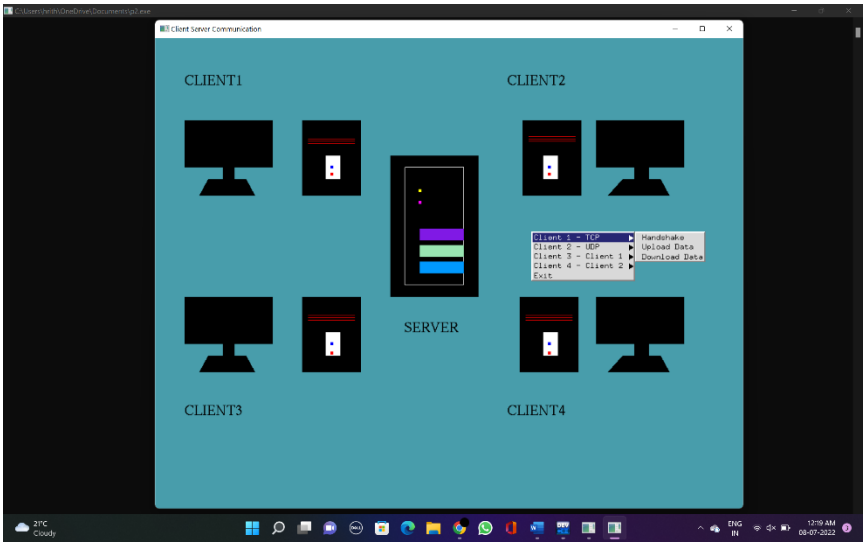


Figure 5.4 Screenshot Of Menu-TCP options

Figure 5.4 indicates the main menu option 1 that is the TCP which has provision for uploading and downloading data from the client to the server and vice versa.

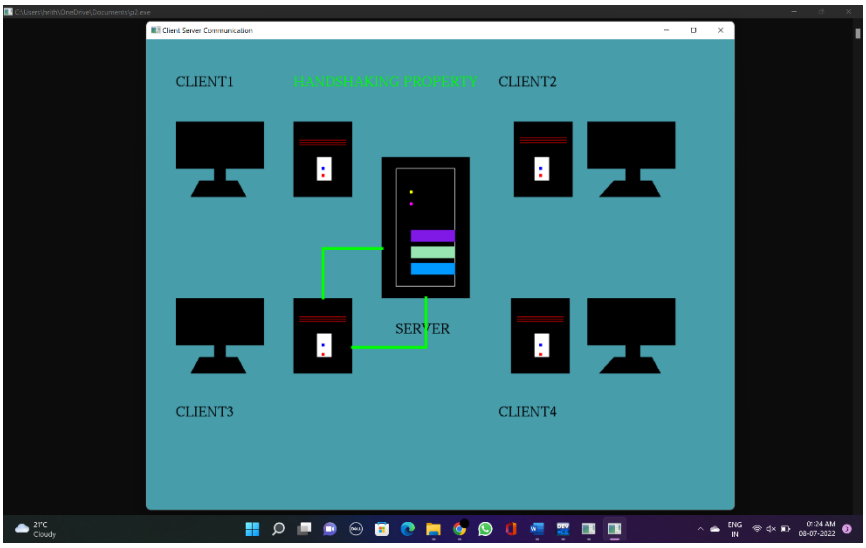


Figure 5.5 Screenshot Of TCP- Handshaking property

Figure 5.5 indicates the handshaking property for establishing connection for the TCP client-server communication.

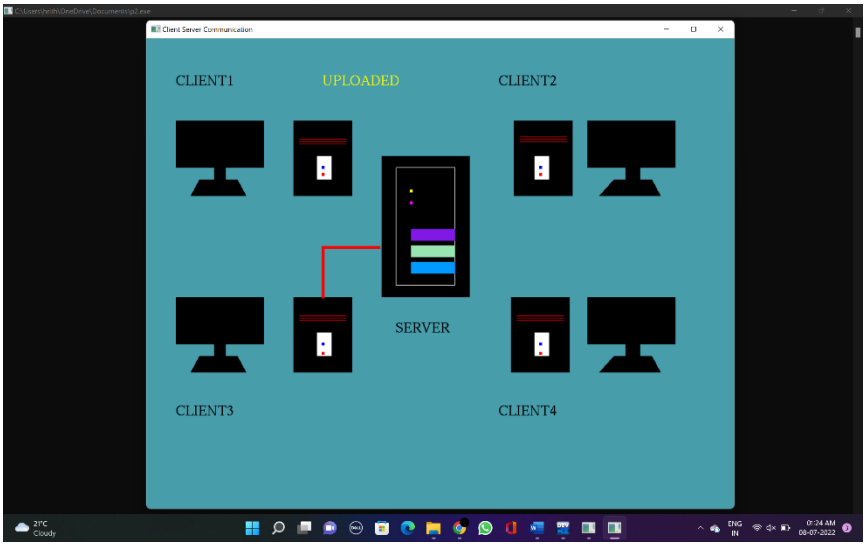


Figure 5.6 Screenshot Of TCP- upload data

Figure 5.6 indicates uploading of the data for the client in TCP client-server communication.

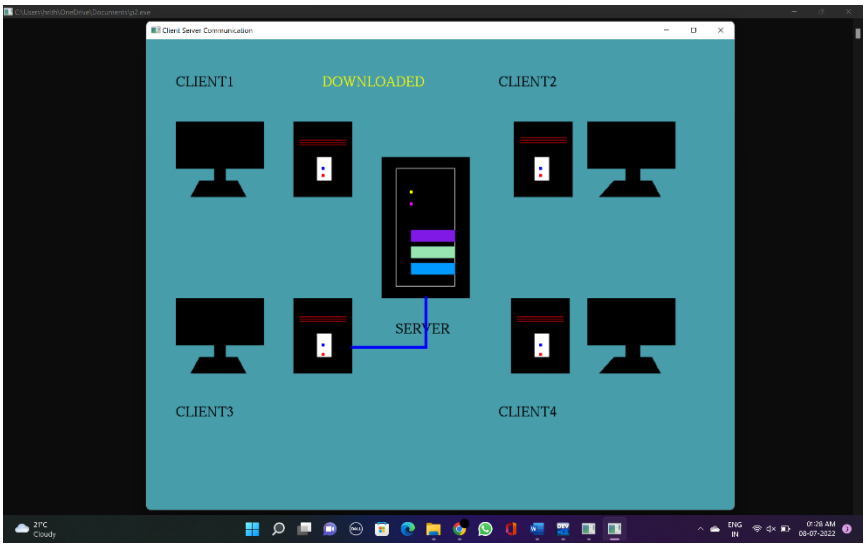


Figure 5.7 Screenshot Of TCP- download data

Figure 5.7 indicates the download of data from server to the client using TCP connection.

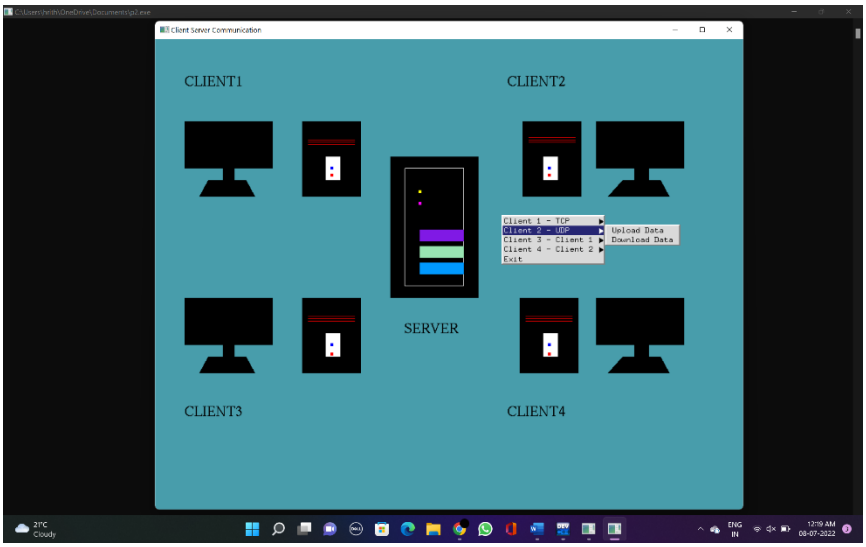


Figure 5.8 Screenshot Of Menu-UDP options

Figure 5.8 indicates the main menu option 2 that is the UDP which has provision for uploading and downloading data from the client to the server and vice versa.

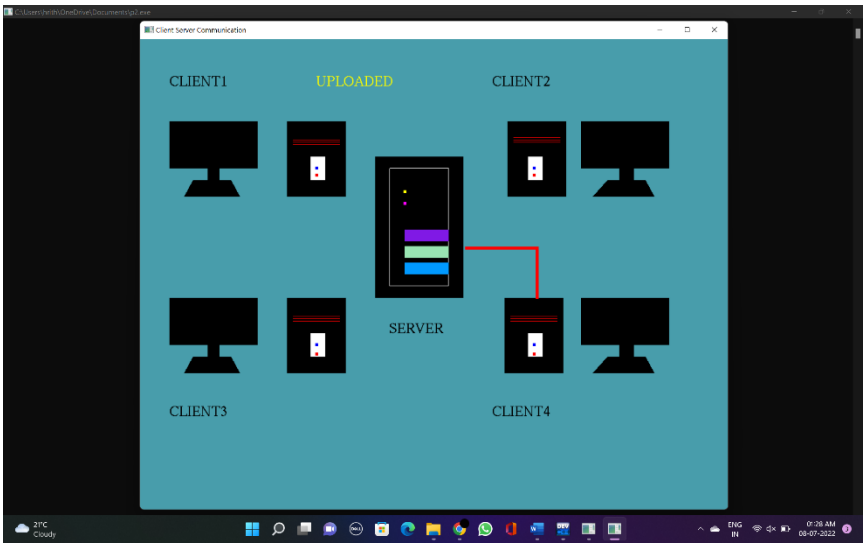


Figure 5.9 Screenshot Of UDP- upload data

Figure 5.9 indicates the upload of data from the client to server using UDP connection.

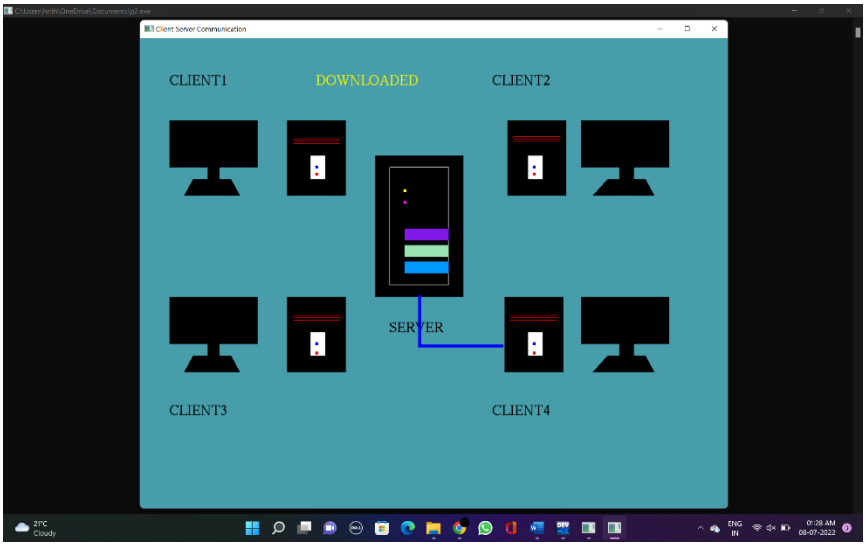


Figure 5.10 Screenshot Of UDP- download data

Figure 5.10 indicates the download of data from the server to the client using the UDP connection.

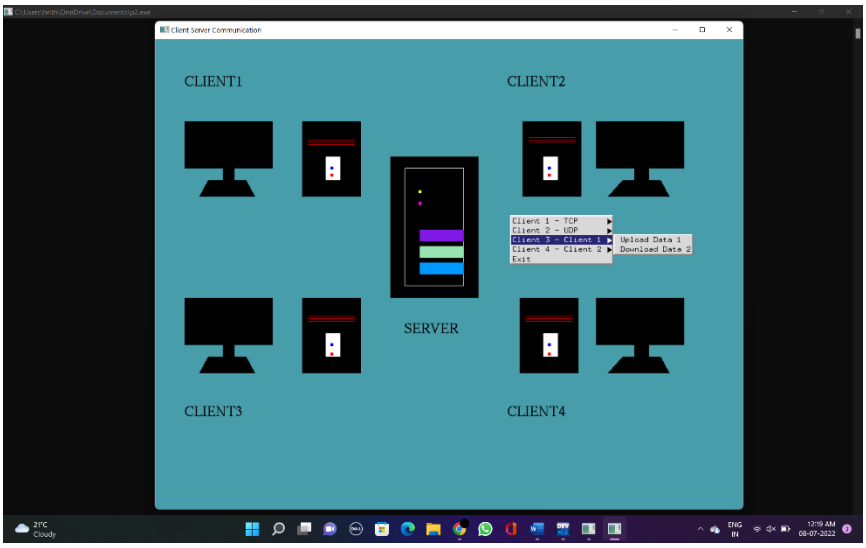


Figure 5.11 Screenshot Of Menu- Client 1 options

Figure 5.11 indicates the main menu option 3 that is the client 1 which has provision for uploading data from client 1 and downloading data sent by client 2.

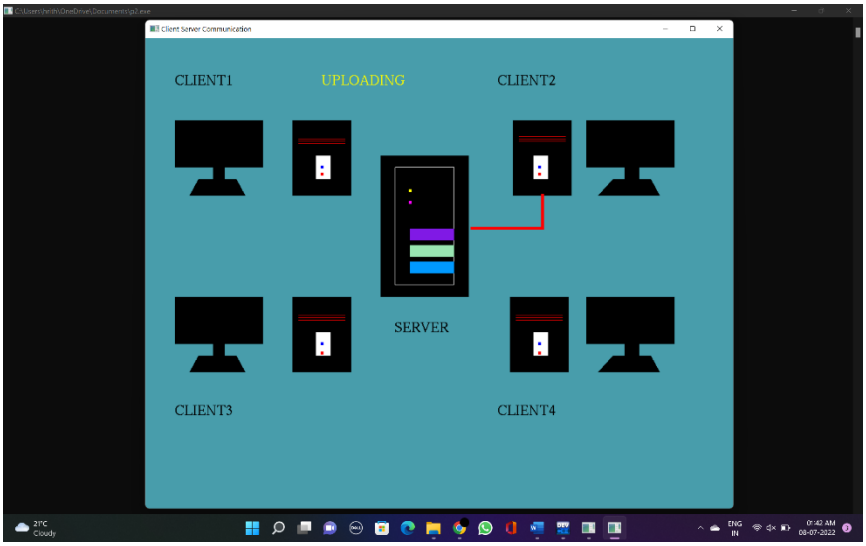


Figure 5.12 Screenshot Of Client 2- upload data

Figure 5.12 contains the uploading of data by the client 2.

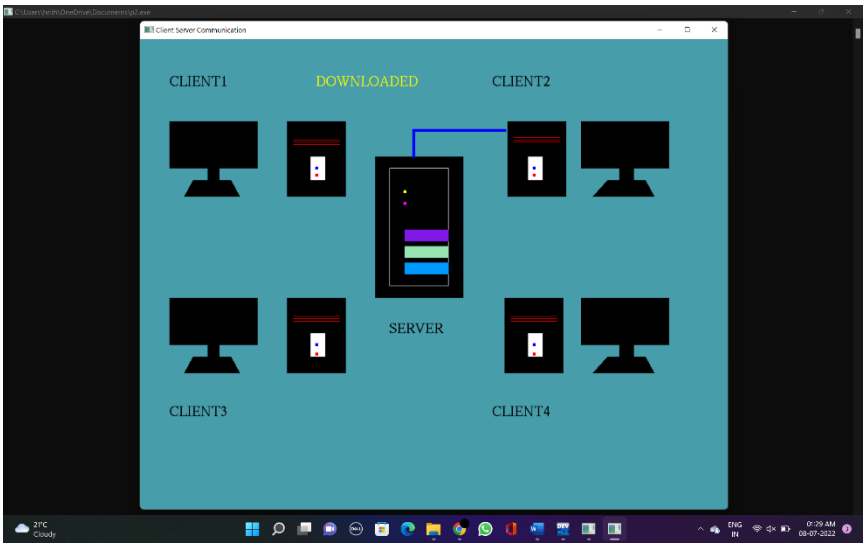


Figure 5.13 Screenshot Of Client 2- download data from Client 1

Figure 5.13 indicates the download of the in client 1 sent by client 2.

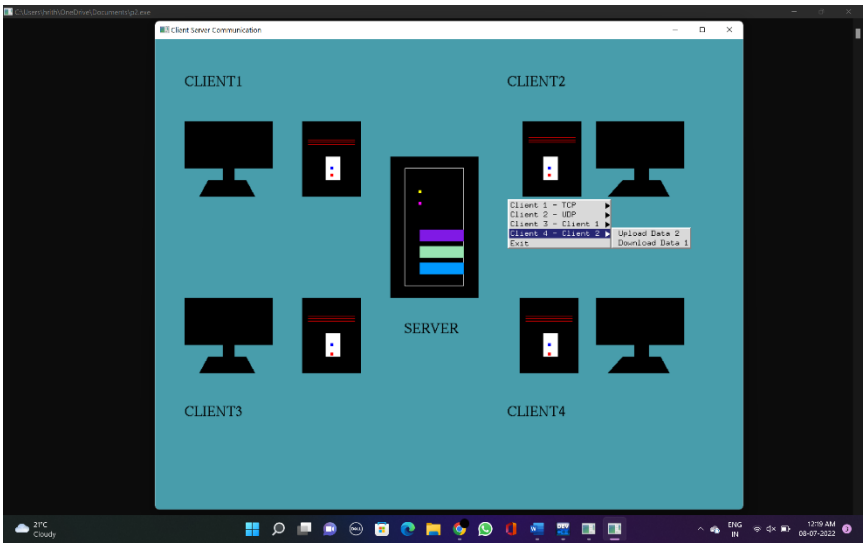


Figure 5.14 Screenshot Of Menu- Client 2 options

Figure 5.14 indicates the main menu option 4 that is the client 2 which has provision for uploading data from client 2 and downloading data sent by client 1.

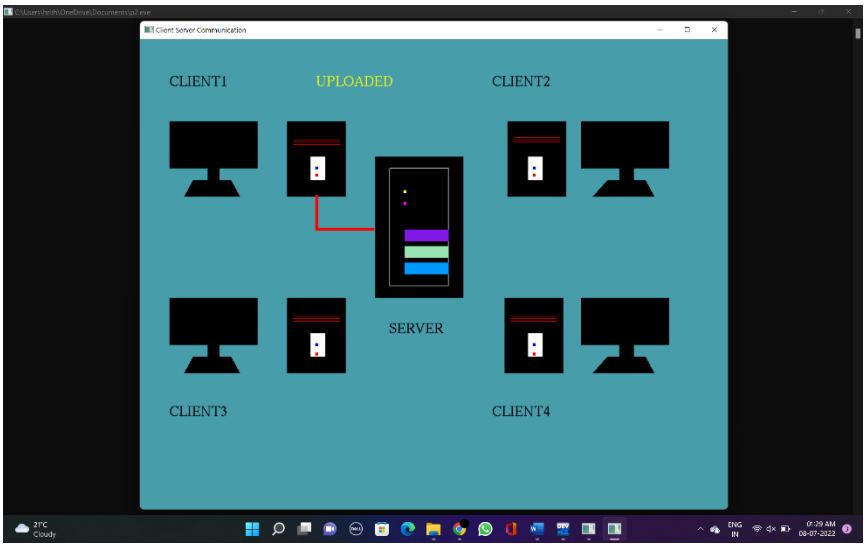


Figure 5.15 Screenshot Of Client 1- upload data

Figure 5.15 contains the uploading of data by the client 1.

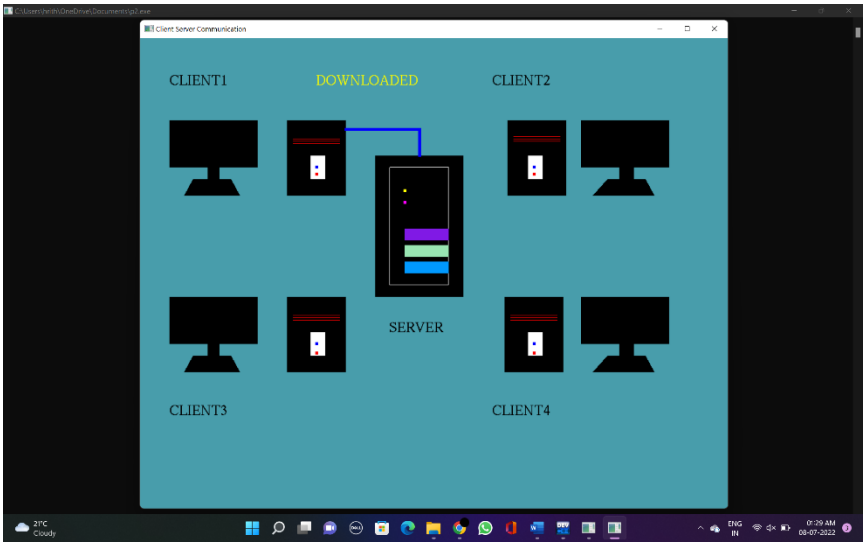


Figure 5.16 Screenshot Of Client 1- download data from Client 2

Figure 5.16 indicates the download of the data by client 1 that was sent by client 2.

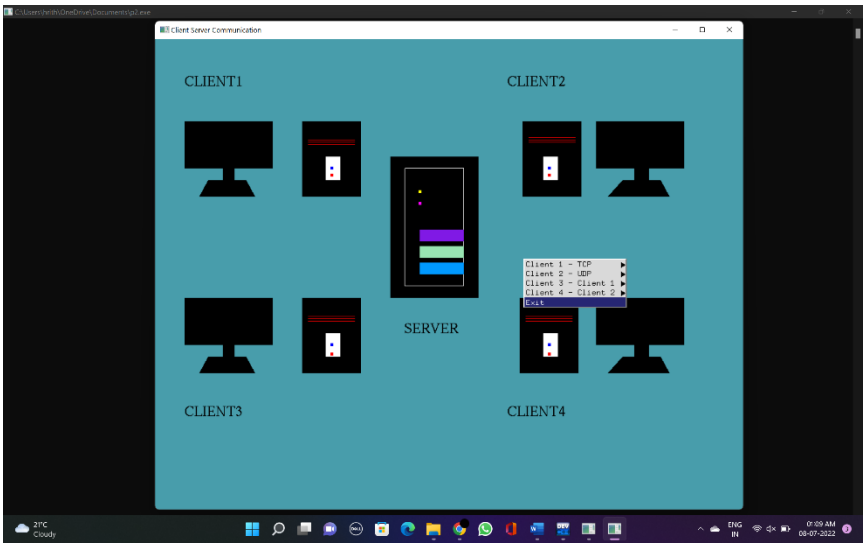


Figure 5.17 Screenshot Of Menu- Exit option

Figure 5.17 indicates the final option of the main menu that is the exit option.

Chapter 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

The project simulates the Client-server communication process in OpenGL. This project gives an insight into demonstration of how the communication takes place when it is initiated over the internet by the client and how the data is transferred between the client and server using TCP and also UDP architecture to establish distributed systems and also the communication between two clients with the help of the server using various translation and scaling related techniques. The overall implementation of the code is done in Dev C++ and programming language used is OpenGL which is mainly used to design the computer graphic projects.

6.2 Future Enhancement

This project shows a certain simulation process that is based on the demonstration of the communication. A better visualization can be given using the OpenGL graphics library. The proposed project uses OpenGL's graphic functions in 2D in most of the work. As an improvement one can make use of 3D graphics for the same demonstration. We can apply lighting and shading effect and can be used in schools for better understanding of Client-server communication.

REFERENCES

[1] Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd/ 4th Edition, Pearson Education, 2011

[2] Edward Angel: Interactive Computer Graphics- A Top-Down approach with OpenGL, 5th Edition, Pearson Education, 2008

[3] https://www.lighthouse3d.com/tutorials/glut-tutorial/modifying-a-menu/