ACKNOWLEDGEMENT

The knowledge & satisfaction that accompany the successful completion of any task would be incomplete without mention of people who made it possible, whose guidance and encouragement crowned my effort with success. I would like to thank all and acknowledge the help I have received to carry out this Mini Project.

I would like to convey my thanks to Head of Department **Dr. J. Girija., Professor and Head** for being kind enough to provide the necessary support to carry out the mini project.

I am most humbled to mention the enthusiastic influence provided by the lab in-charge **Dr Maya B S, Assistant Professor**, on the project for their ideas, time to time suggestions for being a constant guide and co-operation showed during the venture and making this project a great success.

I would also take this opportunity to thank my friends and family for their constant support and help. I'm very much pleasured to express my sincere gratitude to the friendly co-operation showed by all the **staff members** of Computer Science Department, BIT.

Machupalli SreePragna
1BI19CS083

Table of Contents

1. Introduction……..1

1.1 Computer Graphics…….....1

1.2 Application of Computer Graphics……..……1

1.3 OpenGL……3

1.4 Problem Statement……..…….3

1.5 Objective of The Project ……..……..……...4

1.6 Organization of The Project..……4

2. System Specification ……..……..……5

2.1 Hardware Requirements……..……..……5

2.2 Software Requirements……...……5

3. Design……...……6

3.1 Flow Diagram…….……...6

3.2 Description of Flow Diagram……..……..6

4. Implementation…….……8

4.1 Built In Functions……....……8

4.2 User Defined Functions With Modules…….....……..10

4.3 Pseudo code…….…12

5. Snapshots……..……...28

6. Conclusion……....33

Future Enhancement……33

Bibliography……34

Chapter - 1

INTRODUCTION

1.1 Computer Graphics

Computer graphics is an art of drawing pictures, lines, charts, using computers with the help of programming. Computer graphics are made up of a number of pixels. Pixel is the smallest graphical picture or unit represented on the computer screen. Basically, there are 2 types of computer graphics namely,

Interactive Computer Graphics involves a two-way communication between computer and user. The observer is given some control over the image by providing him with an input device. This helps him to signal his request to the computer.

Non-Interactive Computer Graphics otherwise known as passive computer graphics is computer graphics in which the user does not have any kind of control over the image. Image is merely the product of a static stored program and will work according to the instructions given in the program linearly. The image is totally under the control of program instructions not under the user. Example: Screensavers.

1.2 Application of Computer Graphics

1. Scientific Visualization: Scientific visualization is a branch of science, concerned with the visualization of three-dimensional phenomena, such as architectural, meteorological, medical, biological systems.
2. Graphic Design: The term graphic design can refer to a number of artistic and professional disciplines which focus on visual communication and presentation
3. Computer-Aided Design: Computer-Aided design (CAD) is the use of computer technology for the design of objects, real or virtual. The design of geometric models for object shapes, in particular, is often called computer-aided geometric

design (CAGD). The manufacturing process is tied in to the computer description of the designed objects so that the fabrication of a product can be automated using methods that are referred to as CAM, computer-aided manufacturing

4. Web Design: Web design is the skill of designing presentations of content usually hypertext or hypermedia that is delivered to an end-user through the World Wide Web, by way of a Web browser.
5. Digital Art: Digital art most commonly refers to art created on a computer in digital form.
6. Video Games: A video game is an electronic game that involves interaction with a user interface to generate visual feedback on a raster display device.
7. Virtual Reality: Virtual reality (VR) is a technology which allows a user to interact with a computer simulated environment. The simulated environment can be similar to the real world. This allows the designer to explore various positions of an object. Animations in virtual reality environments are used to train heavy equipment operators or to analyze the effectiveness of various cabin configurations and control placements.
8. Computer Simulation: A computer simulation, a computer model or a computational model is a computer program, or network of computers, that attempts to simulate an abstract model of a particular system.
9. Education and Training: Computer simulations have become a useful part of mathematical modeling of many natural systems in physics, chemistry and biology, human systems in economics, psychology, and social science and in the process of engineering new technology, to gain insight into the operation of those systems, or to observe their behavior. Most simulators provide screens for visual display of the external environment with multiple panels mounted in front of the simulator.
10. Image Processing: The modification or interpretation of existing pictures such as photographs and TV scans, is called image processing. In computer graphics, a computer is used to create a picture. Image processing techniques, on the other

hand, are used to improve picture quality, analyze images, or recognize visual patterns for robotics applications

1.3 OpenGL

OpenGL has become a widely accepted standard for developing graphics applications. Most of our applications will be designed to access OpenGL directly through functions in the three libraries. Functions in main GL libraries have names that begin with the letters gl and are stored in a library usually referred to as GL.

The second is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects and simplifying viewing. All functions in GLU can be created from the core GL library. The GLU library is available in all OpenGL implementations. Functions in the GLU library start with the letters glu.

The third is the OpenGL Utility Toolkit (GLUT). It provides the minimum functionality.

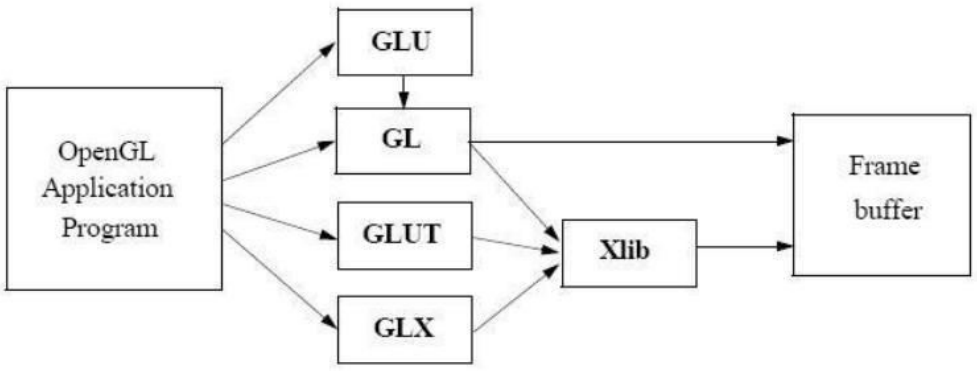


Figure 1: Basic block diagram of OpenGL

1.4 Problem Statement

"To provide a graphical interface for a user to show exactly how client-server simulation in a network happens, to allow the user to see how the commands are sent from client to server and how the responses happen"

1.5 Objective of the Project

1. To show the working of the orthographic projections in appearance of the objects used in the scene.
2. To show the implementation of Textures for a better appearance of the real-world objects.
3. To show the implementation of the OpenGL transformation functions.
4. To show the user and programme interaction using input devices

1.6 Organization of the Report

The project was organized in a systematic way. First, we analyzed what are the basic features to be included in the project to make it acceptable. We first decided the theme, story and functionalities to be included in the project so as to have an idea like how our output must look like. After all these, the source code was formulated as a paper work. All the required software was downloaded. Finally, the successful implementation of the project.

Chapter - 2

SYSTEM SPECIFICATION

2.1 Software Requirements

- Operating System: Windows 10 or Linux (Fedora) or macOS
- Hypervisor used: Docker
- Compiler used: GNU / GCC Compiler.
- Language used: C language
- Editor: Visual Studio Code
- Toolkit: GLUT Toolkit

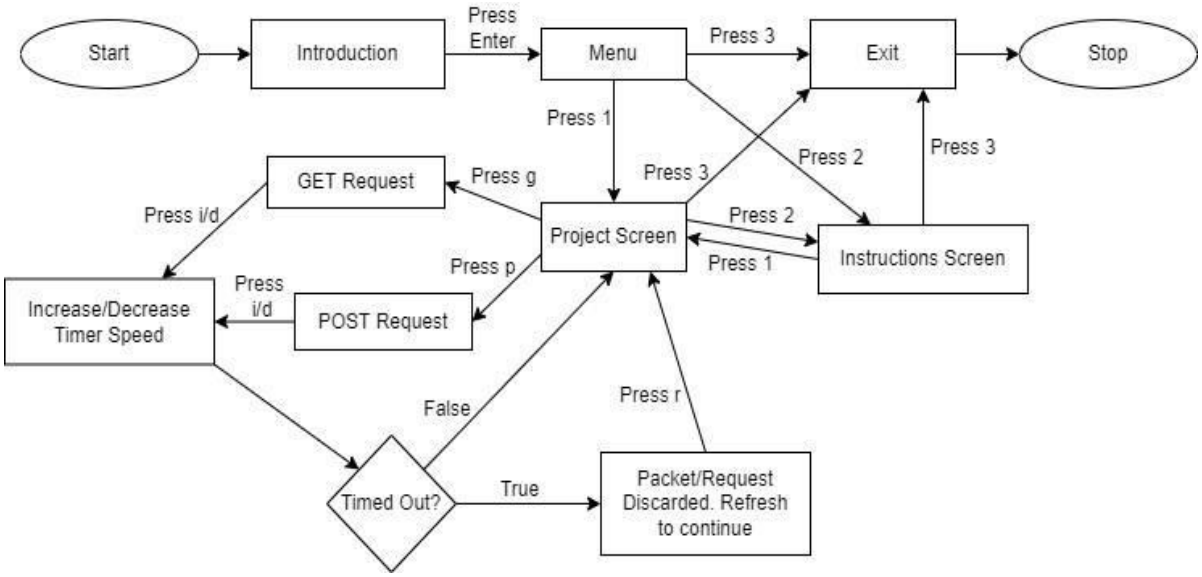
2.2 Hardware Requirements

- Main Processor : PENTIUM III
- Processor Speed: 800 MHz
- RAM Size : 128 MB DDR
- Keyboard : Standard qwerty serial or PS/2 keyboard
- Mouse : Standard serial or PS/2 mouse
- Compatibility : AT/T Compatible
- Cache memory : 256 KB
- Diskette drive : 1.44MB, 3.5 inches

Chapter - 3

DESIGN

3.1 Flow Diagram



3.2 Description of Flow Diagram

The description of the flow diagram is as follows:

Step 1: Start

Step 2: Introduction Screen appears

Step 3: Press Enter to enter the Menu Screen. Menu Screen contains options to enter different screens - Project Screen (Press '1'), Instructions Screen (Press '2'), and Exit (Press '3')

Step 4: If user presses '1', Project Screen appears

Step 5: Now, if user presses 'g', GET request is executed and server and client; If user presses 'p', POST request is executed between the server and client

Step 6: Timer Speed can be increased by pressing ‘i’ or decreased by pressing ‘d’

Step 7: If the timer doesn’t expire and the GET / POST request is successfully completed, the screen returns back to the initial Project Screen

Step 8: If the timer expires, the request / packet is discarded. To continue with the next request, user must press ‘r’ to refresh the screen

Step 9: Instead of pressing ‘1’, the user presses ‘2’, then the Instructions Screen appears. It contains the details, instructions and commands to operate the Project Screen

Step 10: If the user presses neither ‘1’ nor ‘2’ and presses ‘3’, the user can exit the screen to stop the execution of the program

Step 11: The user can navigate from Project Screen to Instructions Screen by pressing ‘2’. The user can also navigate from Instructions Screen to Project Screen by pressing ‘1’. The user can exit the screen to stop the execution of the program by pressing ‘3’ from both Project Screen and Instructions Screen

Step 12: Stop

Chapter - 4

IMPLEMENTATION

4.1 Built In Functions

1. **glutInit()**: glutInit is used to initialize the GLUT library.

   Usage: void glutInit (int *argc, char **argv);

   Description: glutInit will initialize the GLUT library and negotiate a session with the window system.

2. **glutInitDisplayMode()**: glutInitDisplayMode sets the initial display mode.

   Usage: void glutInitDisplayMode (unsigned int mode);

   Mode: Display mode, normally bitwise OR-ing GLUT display mode bit masks

   Description: The initial display mode is used when creating top-level windows, sub- windows, and overlays to determine the OpenGL display mode for the to-be created window or overlay.

3. **glutCreateWindow()**: glutCreateWindow creates a top-level window.

   Usage: intglutCreateWindow (char *name); Name-ASCII character string for use as window name.

   Description: glutCreateWindow creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name. Implicitly, the current window is set to the newly created window. Each created window has a unique associated OpenGL context.

4. **glutDisplayFunc()**: glutDisplayFunc sets display callback for current window.

   Usage: void glutDisplayFunc (void(*func)(void));

   Func: The new display callback function.

   Description: glutDisplayFunc sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the

current window is set to the window needing to be redisplayed and the layer in use is set to the normal plane. The display callback is called with no parameters. The entire normal plane region should be redisplayed in response to the callback.

5. **glutMainLoop**(): glutMainLoop enters the GLUT event processing loop.

   Usage: void glutMainLoop(void);

   Description: glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

6. **glMatrixMode(): ** The two most important matrices are the model-view and projection matrix. At many times, the state includes values for both of these matrices, which are initially set to identity matrices. There is only a single set of functions that can be applied to any type of matrix. Select the matrix to which the operations apply by first set in the matrix mode, a variable that is set to one type of matrix and is also part of the state.
7. **glTranslate(GLfloat X, GLfloat Y, GLfloat Z)**: glTranslate produces a translation by x y z. If the matrix mode is either GL_MODEL_VIEW or GL_PROJECTION, all objects drawn after a call to glTranslate are translated.
8. **glRotatef(GLdouble angle, GLdouble X, GLdouble Y, GLdouble Z)**: glRotatef produces a rotation of angle degrees around the vector x y z. If the matrix mode is either GL_MODEL_VIEW or GL_PROJECTION, all objects drawn after glRotatef is called are rotated. Use glPushMatrix() and glPopmatrix() to save and restore the unrotated coordinate system.
9. **glPushMatrix()**: There is a stack of matrices for each of the matrix mode. In GL_MODELVIEW mode, the stack depth is atleast 32. In other modes, GL_COLOR, GL_PROJECTION, and GL_TEXTURE, the depth is atleast 2. The current matrix in any mode is the matrix on the top of the stack for that mode.
10. **glPopMatrix()**: glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack. Initially, each of the stack contains one matrix, an identity matrix. It is an error to push a full matrix stack or pop a matrix

stack that contains only a single matrix. In either case, the error flag is set and no other change is made to GL state.

11. **glutSwapBuffers()**:

    Usage: void glutSwapBuffers(void);

    Description: Performs a buffer swap on the layer in use for the current window. Specifically, glutSwapBuffers promotes the contents of the front buffer. The contents of the back buffer then become undefined.

12. **glPointSize(GLfloat size)**: glPointSize specifies the rasterized diameter of points. This value will be used rasterize points. Otherwise, the value written to the shading language built-in variable gl- PointSize will be used. The point size specified by glPointSize is always returned when GL_POINT_SIZE is queried.

13. **glutKeyboardFunc()**:

    Usage: void glutKeyboardFunc(void(*func)(unsigned char key, int x, int y) Func: The new keyboard callback function.

    Description: glutKeyboardFunc sets keyboard callback for current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character.

14. **glLineWidth(GLfloat width)**:

    Parameters: width- Specifies the width of rasterized lines. The initial value is 1.

    Description: glLineWidth specifies the rasterized width of lines. The actual width is determined by rounding the supplied width to the nearest integer. i pixels are filled in each column that is rasterized, where I is the rounded value of width.

15. **glLoadIdentity(void)**: glLoadIdentity replaces current matrix with identity matrix. It is semantically equivalent to calling glLoadMatrix with identity matrix.

4.2 User Defined Functions with Modules

1. **void instructionsscreen()**: Function to display the instruction screen
2. **void instructionsscreen_caller()**: Function to call the instruction screen

3. **void initGL()**: Function to initially specify the red, green, blue, and alpha values to clear the color buffers
4. **void idle()**: Function for progress bar handling as the packets are transferred between the server and the client
5. **void display()**: Function to display the introduction screen
6. **void reshape(GLsizei width, GLsizei height)**: Function called when the screen reshapes to display different screens as selected by the user
7. **void keyboard(unsigned char key, int x, int y)**: Function for user to interact with the system to specify the required option using keyboard
8. **void drawText(char [],float ,float)**: Function to display text having a 24-point proportional spaced Times Roman font.
9. **void drawCurrentText(char [],float ,float)**: Function to display text having a 18-point proportional spaced Helvetica font
10. **void drawSmallText(char [],float ,float)**: Function to display fixed width text with every character fitting in an 9 by 15 pixel rectangle
11. **void menuscreen_caller()**: Function to call the menu screen to display the choices of screen - Project Screen, Instructions and Exit, for the user
12. **void menuscreen()**: Function to display the screen containing choices of screen - Project Screen, Instructions and Exit, to display on specifying the correct option
13. **void projectscreen_caller()**: Function to call the Project Screen when user presses '1' key in the keyboard
14. **void Setting()**: Function to specify the red, green, blue, and alpha values to clear the color buffers and also specify implementation-specific hints
15. **void projectscreen()**: Function to draw the Project Screen, showing progress bar, GET and POST requests between the server and the client, along with the corresponding acknowledgements
16. **void drawclient()**: Function to display the client in the Project Screen
17. **void drawserver()**: Function to display the server in the Project Screen
18. **void drawvisuals()**: Function to draw visuals for command section, progress bar, main body and instruction section

19. **void drawpackets(int ,int)**: Function to display the packets while it is being transferred between the server and the client
20. **void drawprogressbar()**: Function to implement progress bar according to current status of packets
21. **void drawcontrollines(int ,float,float)**: Function to draw control lines during GET or POST request from client to server
22. **void refreshtimer()**: Function to refresh the timer incase of timeout, when the timer expires while the server and client exchange packets
23. **void drawinstructions()**: Function to display instructions in the Project Screen
24. **void drawcurrentdetails()**: Function to display current details according to current status of packets
25. **void drawcurrentstatus()**: Function to show current status of the packets
26. **void Refresh()**: Function to refresh the screen to the starting state
27. **void RefreshColor()**: Function to specify the red, green, blue, and alpha values to refresh the color buffers
28. **void menu(int num)**: Function to handle the menus created by createMenu()
29. **void createMenu()**: Function to create menu - background, control lines, packets
30. **void setBackground()**: Function to set background based on user's menu choice

4.3 Pseudo code

```
#include <GL/glut.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int WINDOW_WIDTH ,WINDOW_HEIGHT;
static int window;
int controllineflag1 = 0, postpacketflag = 0, aflag = 0, getpacketflag = 0, ack = 0, timerstopflag = 0, packetcountenableflag = 0, intrupt_color = 0, packet_colors = 0, back_color = 0, proc = 0, packetcountingangle = 30, i;
static int submenu_id, bmenu_id, menu_id,
nu_id; char noofpackets = '-';
```

```
float dir = 0.0, packet = 0.0, timer = 0, angle = 0.0, scalepackets = 0.0, controllinepos1 = 0.001, controllinepos2 = 0.001, packetspeed = 0.001, timerspeed = 0.0, angleofrotation = 3.0, controllinespeed = 0.001;
GLfloat colorslite[][3] = {{1.000, 0.980, 0.804},{0.902, 0.902, 0.980},{0.867, 0.627, 0.867},{0.565, 0.933, 0.565}};
GLfloat colorsdeep[][3] = {{0.804, 0.361, 0.361},{1.000, 0.271, 0.000},{1.000, 0.271, 0.000},{0.000, 0.392, 0.000},{0.000, 1.000, 1.000},{0.855, 0.647, 0.125}}; void setBackground()
{switch(back_color)
        {       case 1: glClearColor(0.1, 0.0, 0.1, 0.1);       break;
                case 3: glClearColor(0.1,0.2,0.3,1.0);          break;
                case 4: glClearColor(0.2,0.1,0.3,1.0);          break;
                default: glClearColor(0.0,0.0,0.0,1.0);         break;
        }
}
void RefreshColor()  {          glColor3f(0.0,1.0,0.0);         }
void projectscreen()
{glClear(GL_COLOR_BUFFER_BIT);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity(); setBackground();

        drawvisuals();
        glColor3f(0.43,0.72,0.52);
        drawCurrentText("Client",-0.65,0.8);
        drawCurrentText("Server",0.7,0.8);
        RefreshColor();
        glColor3f(1.0,1.0,1.0);
        glBegin(GL_LINES);
                glVertex2d(-0.95,-0.95);glVertex2d(0.25,-0.95);
                glVertex2d(0.25,-0.47); glVertex2d(-0.95,-0.47);
        glEnd();
        RefreshColor();
        drawclient();
        drawserver();
        drawinstructions();
        drawcurrentstatus();
        glColor3f(0.0,1.0,0.0);
        drawcurrentdetails();
        glPushMatrix();
```

```
glBegin(GL_POLYGON);
        glVertex2f(-0.95,0.05);                 glVertex2f(-0.93 + timer,0.05);
        glVertex2f(-0.93 + timer,0.0);          glVertex2f(-0.95,0.0);
glEnd();
glPopMatrix();
if(proc == 1)
{       glPushMatrix();
        if(controllineflag1 == 1)       glTranslated(100,0,0);
        glTranslatef(controllinepos1 - 0.08,0.0,0.0);
        drawcontrollines(1,-0.25,-0.3);
        glPopMatrix();
        if(controllineflag1 == 1)
        {       glPushMatrix();
                glTranslatef(-0.6 + packet,0.515f,0.0f);
                glScaled(1.5 + scalepackets,1.5 + scalepackets,1.5 + scalepackets);
                if(packetcountenableflag == 1)
                        angleofrotation = packetcountingangle;
                glRotatef(angle, 0.0f, 0.0f, 1.0f);
                glTranslatef(0.4f,-0.5f,0.0f);
                drawpackets(1,1);
                glPopMatrix();
                if(postpacketflag == 1)
                {       glPushMatrix();
                        glTranslatef(controllinepos2,0.0,0.0);
                        drawcontrollines(3,0.65,0.6);
                        glPopMatrix();          }
        }
}
else if(proc == 2)
{glPushMatrix();
        if(aflag == 1)          glTranslated(100,0,0);
        glTranslatef(-0.08 + controllinepos1,0.0,0.0);
        drawcontrollines(1,-0.4,-0.35);
        glPopMatrix();
        if(aflag == 1)
        {glPushMatrix(); glTranslatef(0.9 -
                packet,0.55f,0.0f);
                glScaled(1.5 + scalepackets,1.5 + scalepackets,1.5 + scalepackets);
                if(packetcountenableflag == 1)
```

```
                        angleofrotation = packetcountingangle;
                    glRotatef(angle,0.0f,0.0f,1.0f);
                    glTranslatef(-0.55,-0.55,0.0f);
                    drawpackets(2,1);
                    glPopMatrix();
                    if(getpacketflag == 1)
                    {       glPushMatrix();         glTranslatef(controllinepos2,0.0,0.0);
                        drawcontrollines(3,-0.4,-0.35);         glPopMatrix();
                    }
                }
        }
        glutSwapBuffers();
        if(!timerstopflag)      timer += timerspeed;
        angle += angleofrotation;
}
void idle()
{       if(angleofrotation < 30.0)      noofpackets = '1';
        if(timer >= 1.88)
        {       timerstopflag = 1;      timerspeed = 0;         timer = 0;      Refresh();      }
        if(proc == 1)
        {if(controllinepos1 >= 1.0 && controllineflag1 != 1)
                {       controllineflag1 = 1;   timer = 0;      timerspeed = 0.0001;
                }
                if(controllineflag1 == 1)
                {if(packet >= 1.2 && postpacketflag != 1)
                        {       packetspeed = 100.0;            postpacketflag = 1;
                                timer = 0;      timerspeed = 0.0001;
                        }
                        else
                        {       if(scalepackets + 1 < 1)        scalepackets = 0.0;
                                else if(packet - 0.4 >= 0.0)    scalepackets -= 0.002;
                                else    scalepackets += 0.002;
                        }
                        if(postpacketflag == 1)         controllinepos2 -= controllinespeed;
                        if(controllinepos2 <= -1.0)
                        {       controllinepos2 = 100;          Refresh();      }
                }
        }
        else if(proc == 2)
```

```
        {       if(controllinepos1 >= 1.0 && controllineflag1 != 1)         aflag = 1;
                if(aflag == 1)
                {       if(packet >= 1.25 && postpacketflag != 1)
                        {       packetspeed = 100.0;            getpacketflag = 1;      }
                        else
                        {       if(scalepackets + 1 < 1)        scalepackets = 0.0;
                                else if(packet - 0.9 >= 0.0)    scalepackets -= 0.002;
                                else    scalepackets += 0.002;          }
                        if(getpacketflag == 1)          controllinepos2 += controllinespeed;
                        if(controllinepos2 >= 1.0)
                        {       controllinepos2 = 100;          Refresh();      }
                }       }
        controllinepos1 += controllinespeed;
        packet += packetspeed;
        if(!timerstopflag)      timer += timerspeed;
        glutPostRedisplay();
}
void drawcontrollines(int color,float a,float b)
{switch(intrupt_color)
        {       case 4: glColor3f(1.0,0.0,0.0);         break;
                case 5: glColor3f(0.0,1.0,0.0);         break;
                case 6: glColor3f(0.0,0.0,1.0);         break;
                default: glColor3f(1.0,0.0,0.0);        break;
        }
        glLineWidth(3);
        glBegin(GL_LINES);
                glVertex2d(a,0.51);    glVertex2f(b,0.51);
        glEnd();
        glLineWidth(1);
}
void drawpackets(int flag,int no)
{switch(packet_colors)
        {       case 7: glColor3d(1,0,0);       break;
                case 8: glColor3d(0,1,1);       break;
                case 9: glColor3d(0,0,1);       break;
                default: glColor3d(1,0,0);      break;
        }
        if(flag == 1)
        {glBegin(GL_POLYGON);
```

```
                    glVertex2f(-0.4,0.407);     glVertex2f(-0.35,0.407);
                    glVertex2f(-0.35,0.442);    glVertex2f(-0.4,0.442);
                glEnd();
                RefreshColor();
        }
        else
        {glBegin(GL_POLYGON);
                    glVertex2f(0.6,0.55);       glVertex2f(0.65,0.55);
                    glVertex2f(0.65,0.585);     glVertex2f(0.6,0.585);
                glEnd();
                RefreshColor();
        }
}
void drawprogressbar()
{glColor3f(0.0,1.0,0.0);
        glBegin(GL_POLYGON);
                glVertex2f(-0.95,0.05);glVertex2f(-0.93,0.05);
                glVertex2f(-0.93,0.0);glVertex2f(-0.95,0.0);
        glEnd();
}
void drawclient()
{glBegin(GL_LINE_LOOP);
        glColor3f(2.0, 0.5, 1.0);
                glVertex2f(-0.85,0.45);glVertex2f(-0.6,0.45);
                glVertex2f(-0.6,0.7); glVertex2f(-0.85,0.7);
        glEnd();
        glColor3f(0.14,0.64,0.75);
        glBegin(GL_POLYGON);
                glVertex2f(-0.83,0.47);glVertex2f(-0.62,0.47);
                glVertex2f(-0.62,0.68); glVertex2f(-0.83,0.68);
        glEnd();
        RefreshColor();
        glColor3f(2.0, 0.5, 1.0);
        glBegin(GL_LINE_LOOP);
                glVertex2f(-0.9,0.38);glVertex2f(-0.55,0.38);
                glVertex2f(-0.6,0.43); glVertex2f(-0.85,0.43);
        glEnd();
        glBegin(GL_LINE_LOOP);
                glVertex2f(-0.5,0.38);glVertex2f(-0.4,0.38);
```

```
                glVertex2f(-0.4,0.65); glVertex2f(-0.5,0.65);
        glEnd();
        glColor3f(0.14,0.64,0.75);
        glBegin(GL_LINE_LOOP);
                glVertex2f(-0.49,0.63);         glVertex2f(-0.41,0.63);
                glVertex2f(-0.41,0.61);         glVertex2f(-0.49,0.61);
        glEnd();
        RefreshColor();
}
void drawvisuals()
{glColor3d(1,1,1);
        glBegin(GL_LINE_LOOP);
                glVertex2f(-0.95,0.05);glVertex2f(0.95,0.05);
                glVertex2f(0.95,0.0); glVertex2f(-0.95,0.0);
        glEnd();
        RefreshColor();
}
void refreshtimer()     {         timer = 0.0;                }
void drawserver()
{glColor3f(0.8, 0.8, 0.8);
        glLineWidth(1.3);
        glColor3fv(colorsdeep[4]);
        if(packetcountenableflag == 1) angleofrotation = packetcountingangle;
        glColor3f(0.8, 0.8, 0.8);
        glBegin(GL_POLYGON);
                glVertex2f(0.65,0.35);glVertex2f(0.9,0.35);
                glVertex2f(0.9,0.45); glVertex2f(0.65,0.45);
        glEnd();
        glColor3f(0.8, 0.8, 0.8);
        glBegin(GL_LINE_LOOP);
                glVertex2f(0.65,0.45);glVertex2f(0.9,0.45);
                glVertex2f(0.9,0.55); glVertex2f(0.65,0.55);
        glEnd();
        glColor3f(0.8, 0.8, 0.8);
        glBegin(GL_POLYGON);
                glVertex2f(0.65,0.55);glVertex2f(0.9,0.55);
                glVertex2f(0.9,0.65); glVertex2f(0.65,0.65);
        glEnd();
        glColor3f(0.8, 0.8, 0.8);
```

```
        glBegin(GL_LINE_LOOP);
                glVertex2f(0.65,0.65);          glVertex2f(0.9,0.65);
                glVertex2f(0.9,0.75);           glVertex2f(0.65,0.75);
        glEnd();
        RefreshColor();
        glLineWidth(1.0);
}
void keyboard(unsigned char key, int x, int y)
{switch (key)
        {       case 13: Refresh();             glClearColor(0.0,1.0,0.5,0.0);
                        menuscreen_caller();        break;
                case '1': Refresh();    projectscreen_caller();         break;
                case '2': Refresh();    instructionsscreen_caller();  break;
                case 'g': Refresh();    proc = 2;       timerspeed = 0.0001;
                        glutIdleFunc(idle);   break;
                case 'p': Refresh();    proc = 1;       timerspeed = 0.0001;
                        glutIdleFunc(idle);    break;
                case 'i': if(timerspeed < 0.005)        timerspeed += 0.0001;           break;
                case 'd': if(timerspeed >= 0.0001)      timerspeed -= 0.0001;           break;
                case 't': refreshtimer();       break;
                case '9': if(packetspeed <= 0.003)      packetspeed += 0.001;           break;
                case '8': if(packetspeed > 0.000)       packetspeed -= 0.001;
                        else            packetspeed = 0.0;
                        break;
                case 'z': if(controllinespeed > 0.0)    controllinespeed -= 0.001;      break;
                case 'x': if(controllinespeed <= 0.005)         controllinespeed += 0.001;
                        break;
                case 'a': if(angleofrotation == 0.0)            angleofrotation = 360.0;
                        else            angleofrotation -= 1.0;
                        packetcountenableflag = 0;          break;
                case 's': if(angleofrotation == 360.0)          angleofrotation = 0.0;
                        else            angleofrotation += 1.0;
                        packetcountenableflag = 0;          break;
                case 'r': Refresh();    timerstopflag = 0;      break;
                case ';': packetcountenableflag = 1;
                        switch(packetcountingangle)
                        {       case 30: packetcountingangle = 184;
                                        noofpackets = '2';      break;
                                case 184: packetcountingangle = 122;
```

```
                                        noofpackets = '3';    break;
                                case 122: packetcountingangle = 92;
                                        noofpackets = '4';    break;
                                case 92: packetcountingangle = 73;
                                        noofpackets = '5';       break;
                                case 73: packetcountingangle = 61;
                                        noofpackets = '6';       break;
                                case 61: packetcountingangle = 46;
                                        noofpackets = '8';       break;
                                default: packetcountingangle = 30;
                                        packetcountenableflag = 0; noofpackets = '-';
                        }
                        break;
                case 'q': exit(0);
                case '3': exit(0);
        }
}
int main(int argc, char** argv)
{glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE);
        glutInitWindowSize(640, 480);
        glutInitWindowPosition(50, 50);
        window = glutCreateWindow("Project Info");
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutIdleFunc(idle);
        glutKeyboardFunc(keyboard);
        initGL();
        glutMainLoop();
        return 0;
}
void display()
{glClear(GL_COLOR_BUFFER_BIT);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        glColor3f(0.5,1.0,0.0);
        drawText("Bangalore Institute of Technology",-
        0.6,0.8); glColor3f(0.0,1.0,0.9);
        drawText("Computer Graphics and Visualization : 18CS62",-0.8,0.6);
```

```
        glColor3f(1.0,0.9,0.0);
        glColor3f(0.64,0.0,3.55);
        glColor3f(1.0,0.31,0.352);
        drawCurrentText("Client - Server Model",-0.35,0.4);
        glColor3f(0.32,0.123,0.84);
        drawSmallText("Press Enter to Continue",-0.35,-0.4);
        glColor3f(0.82,0.31,0.321);
        drawSmallText("By Pragna & Pranjal",0.3,-0.8);
        drawSmallText("1BI19CS083 & 1BI19CS108",0.2,-0.9);
        RefreshColor();
        glutSwapBuffers();
}
void reshape(GLsizei width, GLsizei height)
{       if (height == 0)        height = 1;
        WINDOW_WIDTH = width;
        WINDOW_HEIGHT = height;
        GLfloat aspect = (GLfloat)width / (GLfloat)height;
        glViewport(0, 0, width, height);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        if (width >= height)    gluOrtho2D(-1.0, 1.0, -1.0, 1.0);
        Else    gluOrtho2D(-1.0, 1.0, -1.0 / aspect, 1.0 / aspect);
}
void initGL()  {        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);    }
void drawText(char string[],float x, float y)
{glRasterPos2f(x,y);
        int len=(int) strlen(string);
        for(int i = 0; i < len; i++)
                glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,string[i]);
}
void drawSmallText(char string[],float x, float y)
{glRasterPos2f(x,y);
        int len=(int) strlen(string);
        for(int i = 0; i < len; i++)
                glutBitmapCharacter(GLUT_BITMAP_9_BY_15,string[i]);
}
void drawCurrentText(char string[],float x, float y)
{glRasterPos2f(x,y);
        int len=(int) strlen(string);
```

```
        for(int i = 0; i < len; i++)
                glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18 ,string[i]);
}
void Setting(double r,double g,double b,double alpha)
{glClearColor (r, g, b, alpha);
        glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
}
void instructionsscreen()
{glClear(GL_COLOR_BUFFER_BIT );
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        glColor3f(1.0,1.0,0.0);
        drawText("Instructions",-0.2,0.7);
        glColor3fv(colorsdeep[4]);
        drawSmallText("GET request: Retrieve data from a server.",-0.75,0.5);
        drawSmallText("POST request: Store some data into server for future use",-
0.75,0.4);
        glColor3f(0.32,0.123,0.84);
        drawSmallText("Press '1' for Project Screen or Press '3' to Quit",-0.55,-
        0.8); glColor3f(0.0,1.0,0.0);
        drawSmallText("Project Screen Commands:",-0.75,0.2);
        drawSmallText("Post request - p",-0.75,0.1);
        drawSmallText("Get request - g",0.0,0.1);
        drawSmallText("Inc Timer Speed - i",-0.75,0.0);
        drawSmallText("Dec Timer Speed - d",0.0,0.0);
        drawSmallText("Inc Packet Speed - 9",-0.75,-0.11);
        drawSmallText("Dec Packet Speed - 8",0.0,-0.11);
        drawSmallText("Refresh Timer - t",-0.75,-0.21);
        drawSmallText("Inc Angle - s",-0.75,-0.31);
        drawSmallText("Dec Angle - a", 0.0,-0.31);
        drawSmallText("Change No.of Packets - ;",-0.75,-0.41);
        drawSmallText("Menu Screen - m",-0.75,-0.51);
        drawSmallText("Project Screen - 1",0.0,-0.51);
        drawSmallText("Instructions Screen - 2",-0.75,-0.61);
        drawSmallText("Quit - q",0.0,-0.61);
        glutSwapBuffers();
        RefreshColor();
}
void instructionsscreen_caller()
```

```
{glutDestroyWindow(window); glutInitDisplayMode (
        GLUT_DOUBLE | GLUT_RGB); glutInitWindowSize
        (800, 600); glutInitWindowPosition (250,70);
        window=glutCreateWindow("Instructions");
        glutDisplayFunc(instructionsscreen);
        glutIdleFunc(instructionsscreen);
        glutReshapeFunc(reshape);
        glutKeyboardFunc(keyboard);

        Setting  (0,0,0,0);
        glutMainLoop();
}
void projectscreen_caller()
{glutDestroyWindow(window);
        glutInitDisplayMode ( GLUT_DOUBLE);
        WINDOW_WIDTH = 900;
        WINDOW_HEIGHT = 650;
        glutInitWindowSize (WINDOW_WIDTH,WINDOW_HEIGHT);
        glutInitWindowPosition(0,0); window=glutCreateWindow("Client-
        Server Model"); glutDisplayFunc(projectscreen);

        glutReshapeFunc(reshape);
        glutIdleFunc(idle);
        glutKeyboardFunc(keyboard);
        Setting (0,0,0,0);
        initGL();
        createMenu();
        glutMainLoop();         }
void menuscreen_caller()
{glutDestroyWindow(window); glutInitDisplayMode (
        GLUT_DOUBLE | GLUT_RGB); glutInitWindowSize
        (800, 600); glutInitWindowPosition (250, 70);
        window=glutCreateWindow("Menu");
        glutDisplayFunc(menuscreen);
        glutIdleFunc(menuscreen); glutReshapeFunc(reshape);
        glutKeyboardFunc(keyboard);


        gluOrtho2D (0.0, 200.0, 0.0, 150.0);
```

```
        Setting (0,0,0,0);
        glutMainLoop();
}
void menuscreen()
{glClear(GL_COLOR_BUFFER_BIT );
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity(); glColor3f(0.23,0.452,0.5653);
        drawText("Client-Server Architecture",-
        0.5,0.7); glColor3f(0,1,0);

        drawText("1.Project Screen",-
        0.3,0.3); glColor3f(1,1,0);
        drawText("2.Instructions",-0.3,0.15);
        glColor3f(1,0.5,0);
        drawText("3.Quit",-0.3,0.0);
        glColor3f(0,0,0);
        glutSwapBuffers();
}
void drawinstructions()
{glColor3f(1.0,0.0,0.0);
        drawSmallText("Timer",-
        0.9,0.07); RefreshColor();
        drawSmallText("Post request - p",0.4,-0.15);
        drawSmallText("Get request - g",0.4,-0.20);
        drawSmallText("Inc Timer Speed - i",0.4,-0.25);
        drawSmallText("Dec Timer Speed - d",0.4,-0.3);
        drawSmallText("Inc Packet Speed - 9",0.4,-0.35);
        drawSmallText("Dec packet Speed - 8",0.4,-0.4);
        drawSmallText("Refresh Timer - t",0.4,-0.45);
        drawSmallText("Inc Angle - s",0.4,-0.5);
        drawSmallText("Dec Angle - a", 0.4,-0.55);
        drawSmallText("No.of Packets - ;",0.4,-0.6);
        drawSmallText("Menu Screen - m",0.4,-0.65);
        drawSmallText("Project Screen - 1",0.4,-0.7);
        drawSmallText("Instructions Screen",0.4,-0.75);
        drawSmallText("    - 2",0.4,-0.8);
        drawSmallText("Quit - q",0.4,-0.85);
}
void drawcurrentdetails()
```

```
{char str[1000];
        if(!timerstopflag)
        {if(proc == 1)
                {       if(postpacketflag == 1)         sprintf(str,"Request:  POST     Status:
Acknowledgement from Server");
                        else if(controllineflag1 == 1) sprintf(str,"Request: POST       Status:
Client Posting Packets");
                        else     sprintf(str,"Request: POST       Status: Post Control Request
        from Client");
                }
                else if(proc == 2)
                {       if(getpacketflag)       sprintf(str,"Request:   GET             Status:
Acknowledgement from Client");
                        else if(aflag)  sprintf(str,"Request: GET       Status: Getting Packets
from Server");
                        else     sprintf(str,"Request: GET          Status: Get Control Request
from Client"); }
                 else   {       glColor3f(0.0,0.0,1.0);
                        sprintf(str,"Make  a  Get(g)  Request  or  a Post(p) Request to
Continue");     }
                drawCurrentText(str,-0.95,-0.25);
                RefreshColor();
        }
        else
        {glColor3f(1.0,0.0,0.0);
                drawCurrentText("Timed  out!  Packet/Request  Discarded,  Hit  r  to
refresh",-0.95,-0.25);
                RefreshColor();         }
        RefreshColor();
}
void drawcurrentstatus()
{char str[100]; glColor3fv(colorslite[5]);
        sprintf(str,"Timer speed :
        %.4lf",timerspeed); drawCurrentText(str,-
        0.93,-0.55);
        sprintf(str,"Packet Speed : %.3lf", (packetspeed <= 50) ? packetspeed :
        0.000); drawCurrentText(str,-0.23,-0.55);
        sprintf(str,"Intrupt Speed : %.3lf",
        controllinespeed); drawCurrentText(str,-0.93,-0.65);
```

```
        sprintf(str,"Angle of Rotation : %.3lf",
        angleofrotation); drawCurrentText(str,-0.93,-0.75);
        sprintf(str,"Number of Packets : %c", noofpackets);
        drawCurrentText(str,-0.23,-0.65); switch(back_color)

        {       case 1: sprintf(str,"Background - Gray\n");             break;
                case 3: sprintf(str,"Background - Dark Green\n");       break;
                case 4: sprintf(str,"Background - Purple\n");           break;
                default: sprintf(str,"Background - Black\n");           break;
        }
        drawCurrentText(str,-0.93,-0.85);
        switch(packet_colors)
        {       case 7: glColor3f(1.0,0.0,0.0);         break;
                case 9: glColor3f(0.0,0.0,1.0);         break;
                case 8: glColor3f(0.0,1.0,1.0);         break;
                default: glColor3f(1.0,0.0,0.0);        break;
        }
        drawCurrentText("Packet Color",-0.23,-0.75);
        switch(intrupt_color)
        {       case 4: glColor3f(1.0,0.0,0.0);         break;
                case 5: glColor3f(0.0,1.0,0.0);         break;
                case 6: glColor3f(0.0,0.0,1.0);         break;
                default: glColor3f(1.0,0.0,0.0);        break;
        }
        drawCurrentText("Control Lines Color",-0.23,-0.85);
        RefreshColor();
}
void Refresh()
{       controllineflag1 = 0;   postpacketflag = 0;     aflag = 0;      getpacketflag = 0;
        ack = 0;        proc = 0;       dir = 0.0;      packetcountenableflag = 0;
        packet = 0.0;   timer = 0;      angle = 0.0;    scalepackets = 0.0;
        controllinepos1 = 0.001;        controllinepos2 = 0.001;
        intrupt_color = 0;      packet_colors = 0;      back_color = 0;     submenu_id = 0;
        bmenu_id = 0;       menu_id = 0;        nu_id = 0;      packetspeed = 0.001;
        timerspeed = 0.0;       angleofrotation = 3.0;      controllinespeed = 0.001;
        glColor3f(0.0,1.0,0.0);
        drawcurrentdetails();
}
void createMenu(void)
```

```
{submenu_id = glutCreateMenu(menu);
        glutAddMenuEntry("Grey",1);        glutAddMenuEntry("Dark Green",2);
        glutAddMenuEntry("Black",3);       glutAddMenuEntry("Purple",10);
        bmenu_id = glutCreateMenu(menu);
        glutAddMenuEntry("Red",4);         glutAddMenuEntry("Green",5);
        glutAddMenuEntry("Blue",6);
        menu_id = glutCreateMenu(menu);
        glutAddMenuEntry("Red",7);         glutAddMenuEntry("cyan",8);
        glutAddMenuEntry("Blue",9);
        nu_id = glutCreateMenu(menu);
        glutAddSubMenu("Background", submenu_id);
        glutAddSubMenu("Control Lines", bmenu_id);
        glutAddSubMenu("Packets", menu_id);
        glutAttachMenu(GLUT_RIGHT_BUTTON);
}
void menu(int num)
{       if(num == 0)            exit(0);
        else
        {switch(num)
                {       case 1: back_color = 1;         break;
                        case 2: back_color = 3;         break;
                        case 3: back_color = 2;         break;
                        case 4: intrupt_color = 4;      break;
                        case 5: intrupt_color = 5       break;
                        case 6: intrupt_color = 6;      break;
                        case 7: packet_colors = 7;      break;
                        case 8: packet_colors = 8;      break;
                        case 9: packet_colors = 9;      break;
                        case 10: back_color = 4;        break;
                }glutPostRedisplay();
        }
}
```

Chapter - 6

CONCLUSION

Users of this client-server simulation will be able to see the data packets as they move between the client and server within a given client – server network. Testing of the application is done to ensure its integrity and reliability. Users can learn about what exactly happens in TCP communication.

A graphical representation with different colors is used to explain the various aspects of the components and their working. This helps the user understand networks and how the client connects to the server in general.

We thus would like to emphasize the importance of this project to many other perspectives of Technical, mathematical, graphical and software concepts which we were unaware of.

6.1 Future Enhancement

- In future, the same project can be enhanced in such a way that we can interact more with the project
- A vast amount of future work can be possible by following investigations and strategies
- More features can be included and can be modified in a more versatile way
- Visualization of content inside the data packets.
- Explanation of routing through the network using different network devices such as router, hub, gateway, etc.
- Connectivity in the network.
- Visualization of further concepts such as peer – to – peer.
- Visualization of protocols used in networks.

BIBLIOGRAPHY

Reference Books

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, 5 th Edition, Pearson Education
3. James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer Graphics with OpenGL, Pearson Education

Websites

1. https://www.opengl.org/
2. https://learnopengl.com/
3. https://www.britannica.com/technology/client-server-architecture