# CHAPTER 1

# INTRODUCTION

Computer graphics is branch of computer science that deals with generating images with the aid of computers. Today, computer graphics is a core technology in digital photography, film, video games, cell phone and computer display, and many specialized applications. A great deal of specialized hardware and software has been developed, with the displays of most devices being driven by computer hardware graphics. Our project uses the OpenGL library which is a cross-language, cross-platform application programming interface for rendering 2D and 3D vector graphics. This API interacts with the GPU to achieve accelerated rendering. The project “Client Server Architect” shows the communication process between a client and a server in a computer network.

In computer networks server is someone who runs a server programs which share their resources with client. A client does not share any of its resources, but it requests content or service form a server. Client, therefore, initiate communication sessions with servers, which await incoming requests. Client’s and servers are core of world wide web. Client and server exchange messages in a request-response messaging pattern.The client sends a request, and the server returns a response. This exchange of messages is an example of inter-process communication.Communication between nodes in network is contolled by various protocols.Client,server can communicate either over tcp (transmission control protocol) or udp (user datagram protocol).udp is a stateless protocol and dosen’t guarantee the delivery of packets to their destination.Where as tcp guarantees the delivery of packets.Tcp connections are initiated with a three way handshake.On the whole Client/server systems have become the computing and application architecture for business organizations across the globe. Speaking in technical terms, a client/server system puts application processing close to the user; helping improve the performance.

# CHAPTER 2

# DESCRIPTION

## 2.1 DESCRIPTION OF THE PROJECT

Our project uses free-GLUT to achieve the desired graphics.The project illustrates the client server communication process using 2-D animation. The first scene is the home page which displays the institute name,name of the project and student details.This page is displayed for short period of time dictated by a timer. Then there's a change in scene after the set period and the client server scene is displayed. In the main scene there are four client and a centralized server.The interaction is initiated by the mouse.Right click of the mouse displays the five options with each option having a sub option.Each option control the interaction of a particular client.Each client has a upload and download functionality,with client one having an additional tcp connection initiation which starts a three way handshake with the server.Uploads are denoted by blue line from the client to the server and downloads are denoted by a red line from the server to the client.

## 2.2 DESCRIPTION OF THE FUNCTION USED

### 2.2.1 INTERACTING WITH WINDOWS

**glutInit(&argc,argv)**

glutInit will initialize the GLUT library and negotiate a session with the window system. glutInit also processes command line options, but the specific options parse are window system dependent.

A pointer to the program's unmodified argc variable from main. Upon return, the value pointed to by argc will be updated, because glutInit extracts any command line options intended for the GLUT library.

The program's unmodified argv variable from main. Like argc, the data for argv will be updated because glutInit extracts any command line options understood by the GLUT library.

**glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)**

The above statement specifies a double refresh buffer to be used and to use red,green and bluecomponents to select color values.Glut_DEPTH gives a bit mask to select a window with a depth buffer.

**glutInitWindowPosition(x,y)**

The above statement specifies location that is x pixels to the right of the left edge of screen and y pixels down from top edge.The origin is at upper-left corner of the screen.

**glutInitWindowSize(x,y)**

The above statement specifies the size of display window with a width of x pixels and height of y pixels.

**glutCreateWindow(" Client server architecture")**

The above function accepts a string which will be the title of display-window.

**glutDisplayFunc(frontscreen)**

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.

**glutTimerFunc(2000,mytimer,0)**

glutTimerFunc registers the timer callback func to be triggered in at least msecs milliseconds.The last parameter to the timer callback will be the value of the value parameter to glutTimerFunc.

**glutCreateMenu(void (*func)(int value))**

It creates a new pop-up menu and returns a unique small integer identifier.The range of allocated identifiers starts at one.

**glutAddMenuEntry(char *name,int value)**

It adds a menu entry to the bottom of the current menu.Name is the ascii character string to display in the menu entry.value to return to the menu’s callback function if the menu entry is selected.

**glutMainLoop()**

It displays the Initial graphics and puts the program into an infinite loop that checks for input from devices such as mouse or keyboard.

### 2.2.2 INTERACTION

**glutAttachMenuEntry(int button)**

glutAttachMenu attaches a mouse button for the *current window* to the identifier of the current menu. By attaching a menu identifier to a button, the named menu will be popped up when the user presses the specified button.Button should be one of GLUT_LEFT_BUTTON,GLUT_MIDDLE_BUTTON,GLUT_RIGHT_BUTTON.

### 2.2.3 TRANSFORMATIONS

**glClearColor(0.0,0.0,0.0,0.0)**

The statement sets the background color to black.The first three parameter are for color black where 0.0 is black and 1.0 is white. The fourth is called alpha value where 0.0 indicates transparent object.

**glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)**

glClear sets the bitplane area of the window to values previously selected by glClearColor.GL_COLOR_BUFFER_BIT  indicates the buffer currently enabled for color writing.GL_DEPTH_BUFFER_BIT indicates the depth buffer.

**void glMatrixMode(GL_PROJECTION)**

Specifies which matrix stack is the target for subsequent matrix operations.GL_PROJECTION applies subsequent matrix operations to the projection martrix stack.

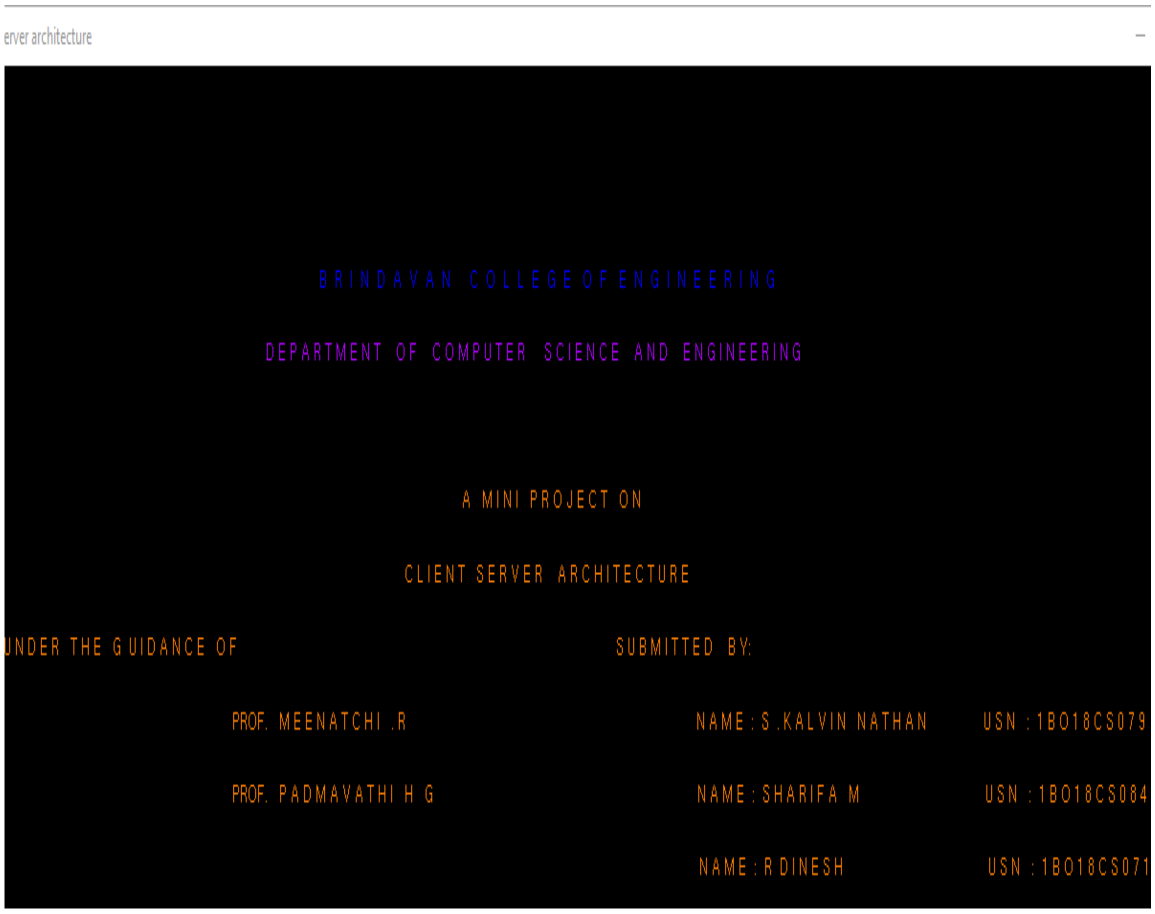
**gl_Ortho2D(0.0,400.0,0.0,400.0)**

This defines the coordinate reference frame within the display window to be (0.0,0.0) at lower-left corner of the display window and (400.0,400.0) at the upper-right window corner.

## 2.3 MAIN OBJECTS IN THE PROJECT

### 2.3.1 The Home screen

The home screen is created using the function frontscreen() which uses the glutBitmapCharacter() function is used to display the characters on the screen


erver architecture

BRINDAVAN COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A MINI PROJECT ON

CLIENT SERVER ARCHITECTURE

UNDER THE GUIDANCE OF SUBMITTED BY:

PROF. MEENATCHI .R NAME : S .KALVIN NATHAN USN : 1BO18CS079

PROF. PADMAVATHI H G NAME : SHARIFA M USN : 1BO18CS084

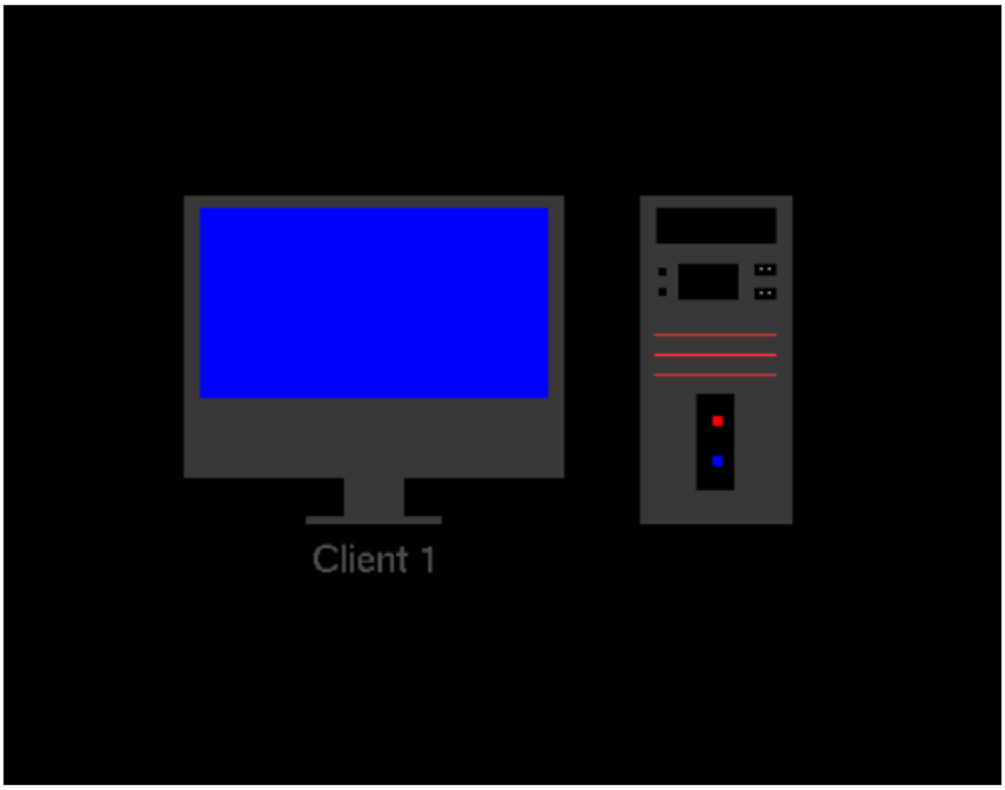
NAME : R DINESH USN : 1BO18CS071

<<pseudocode for home screen>>

```
void frontscreen()
{
int i;
 glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
 //glClearColor(1,1,1,0);
 glColor3f(0,0,1);
// drawstring(20.0,90.0,0.0,"NAME OF THE COLLEGE ");
 ptr6="B R I N D A V A N    C O L L E G E    O F    E N G I N E E R I N G";
 glRasterPos3f(100.0,340.0,0.0);
 len6=strlen(ptr6);
 for (i = 0; i<len6; i++)
            glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, ptr6[i]);
……    //similar code for other characters to be displayed
glutSwapBuffer();
glutFlus();
}
```

### 2.3.2 Client

The client is drawn using completely using GL_Lines.



<<pseudocode for client>>

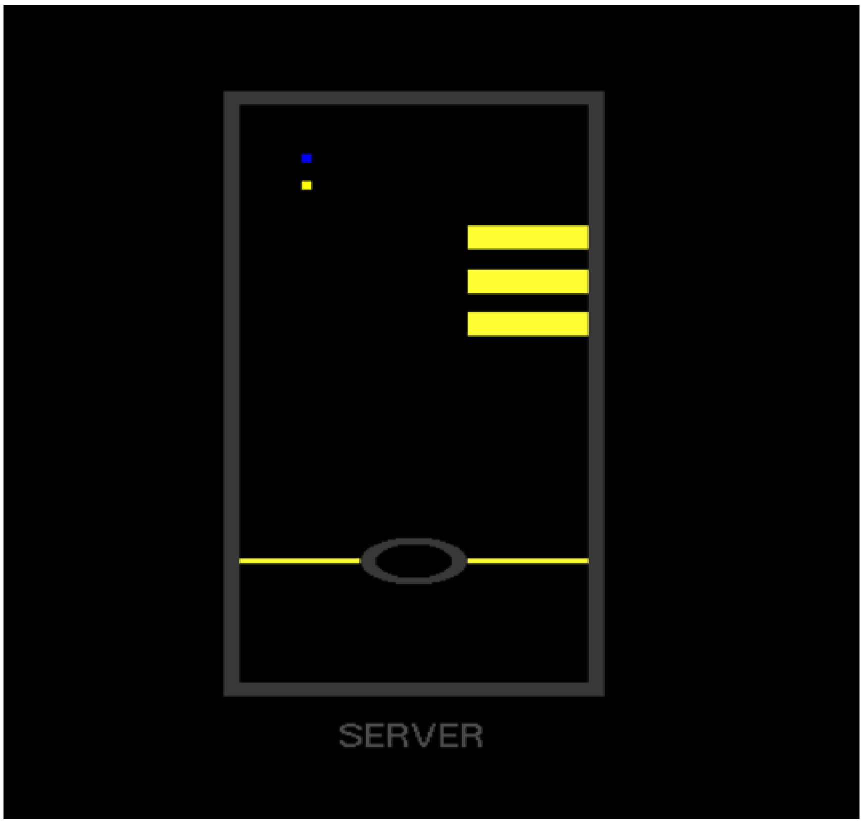
```
void client()     //this function is to design the system(monitor and CPU)
{
        glColor3f(0.22, 0.22, 0.22);
        glBegin(GL_POLYGON);
        glVertex2i(30, 60);
        glVertex2i(80, 60);
        glVertex2i(80, 115);
        glVertex2i(30, 115);
        glEnd();
```

```
glColor3f(0.0, 0.0, 1.0);

glBegin(GL_POLYGON);

glVertex2f(32.0, 64.0);

glVertex2f(78.0, 64.0);

glVertex2f(78.0, 112.0);

glVertex2f(32.0, 112.0);

glEnd();


glColor3f(0.22, 0.22, 0.22);

glBegin(GL_POLYGON);

glVertex2f(30.0, 44.0);

glVertex2f(80.1, 44.0);

glVertex2f(80.1, 60.0);

glVertex2f(30.0, 60.0);

glEnd();

glColor3f(0.22, 0.22, 0.22);

glBegin(GL_POLYGON);

glVertex2f(51.0, 34.0);

glVertex2f(51.0, 44.0);

glVertex2f(59.0, 44.0);

glVertex2f(59.0, 34.0);
```

```
  glVertex2f(64.0, 34.0);

  glVertex2f(64.0, 32.0);

  glVertex2f(46.0, 32.0);

  glVertex2f(46.0, 34.0);

  glVertex2f(51.0, 34.0);

  glEnd();

…..

}
```

## 2.3.3 The Server

The Server is drawn by using GL_Lines entirely.

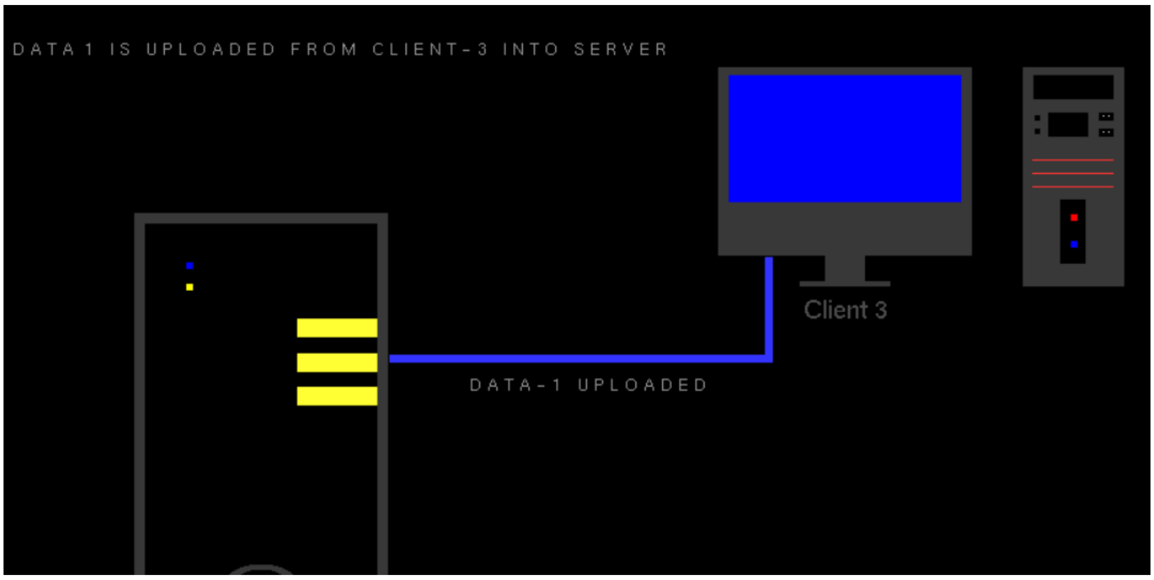


<<pseudocode for server>>

```
void server() // this function is to design the central server.
{
        glColor3f(0.22, 0.22, 0.22);
        glBegin(GL_POLYGON);
        glVertex2f(175.0, 100.0);
        glVertex2f(225.0, 100.0);
        glVertex2f(225.0, 280.0);
        glVertex2f(175.0, 280.0);
        glEnd();
       glColor3f(0.0, 0.0, 0.0);
        glBegin(GL_POLYGON);
        glVertex2f(177.2, 104.0);
        glVertex2f(222.8, 104.0);
        glVertex2f(222.8, 276.0);
        glVertex2f(177.2, 276.0);
        glEnd();
       ……
}
```

### 2.3.4 Upload

The upload line is used to depict upload functionality using a blue line.It uses matrix fuctions to make changes in the scene using Pushmatrix() and PopMatrix functions.



<<pseudocode for upload>>

```
void threethreeU() //upload fuction for client 3.
{       upflag=0;
        downflag=0;
        glPushMatrix();
        glColor3f(0.0,0.0,0.0);
        alreadyupload();
        glPopMatrix();
        glPushMatrix();
        glColor3f(0.0,0.0,0.0);
        alreadydownload();
```

```
glPopMatrix();

upc3flag=0;

downc3flag=0;

 upc4flag=0;

downc4flag=0;

glPushMatrix();

glColor3f(0.0,0.0,0.0);

alreadyupload1();

glPopMatrix();

glPushMatrix();

glColor3f(0.0,0.0,0.0);

alreadydownload1();

glPopMatrix();

glPushMatrix();

glColor3f(0.0,0.0,0.0);

alreadyupload2();

glPopMatrix();

glPushMatrix();

glColor3f(0.0,0.0,0.0);

alreadydownload2();

glPopMatrix();

glPushMatrix();
```

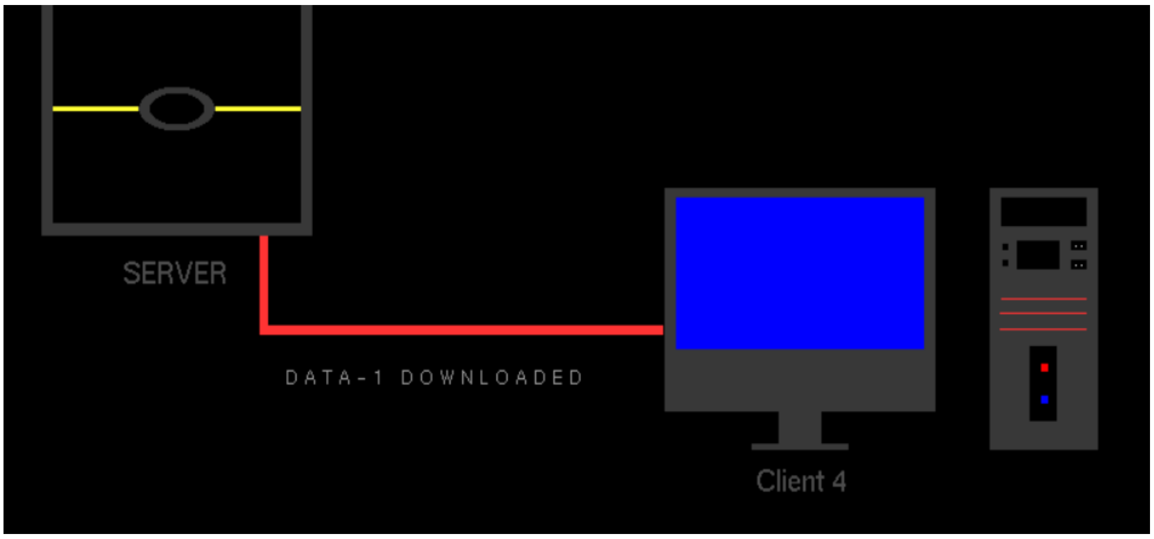
```
                glColor3f(0.0,0.0,0.0);

                datatwostatus();

                glPopMatrix();

                glPushMatrix();

                glColor3f(0.0,0.0,0.0);

                dataonestatus();

                glPopMatrix();

if (p13 == 300.0 && p14 <= 262.0 && p14 > 225.0)

{

        p14 = p14 - 0.08;



        glutPostRedisplay();

}

else

{

        if (p13 <= 300.0 && p13>226.0)

        {

                p13 = p13 - 0.08;

                glutPostRedisplay();

        }

}

 if (p13 > 229.0)
```

```
{       glPushMatrix();

        glColor3f(0.6, 0.6, 0.6);

        uploading3();

        glPopMatrix();

}

else

{

        glPushMatrix();

        glColor3f(0.0, 0.0, 0.0);

        uploading3();

        glPopMatrix();

        upload3();

                upc3flag=1;

                        if(upc3flag==1 )

                        {       glPushMatrix();

                                glColor3f(0.0,0.0,0.0);

                                alreadydownload1();

                                glPopMatrix();

                                glPushMatrix();

                                glColor3f(0.6,0.6,0.6);

                                alreadyupload1();

                                glPopMatrix();
```

```
                                        }
}
}
```

### 2.3.5 Download

The download line depicts the download functionality using the red line.



<<pseudocode for download>>

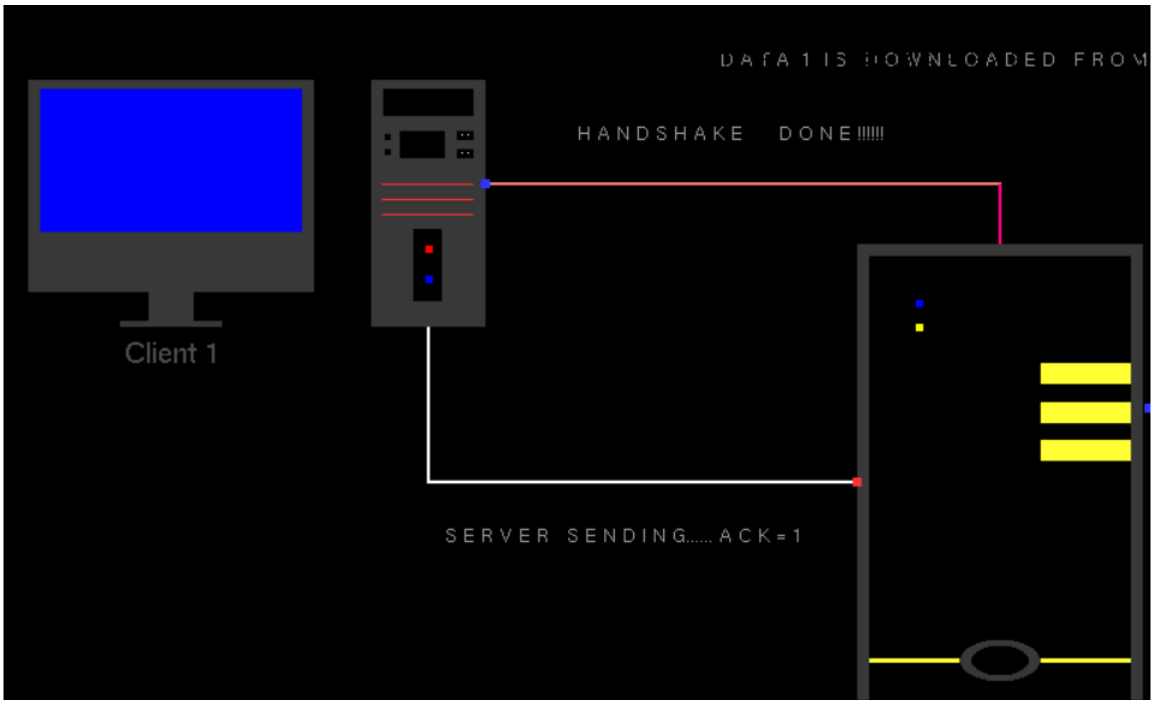
```
void fourfourD() //download function for client 4.
{
    upflag=0;
    downflag=0;
    glPushMatrix();
    glColor3f(0.0,0.0,0.0);
    alreadyupload();
    glPopMatrix();
```

```
glPushMatrix();
 glColor3f(0.0,0.0,0.0);
 alreadydownload();
 glPopMatrix();
 upc3flag=0;
 downc3flag=0;
 upc4flag=0;
 downc4flag=0;
 glPushMatrix();
 glColor3f(0.0,0.0,0.0);
 alreadyupload1();
 glPopMatrix();
glPushMatrix();
 glColor3f(0.0,0.0,0.0);
 alreadydownload1();
 glPopMatrix();
 glPushMatrix();
 glColor3f(0.0,0.0,0.0);
 alreadyupload2();
 glPopMatrix();
glPushMatrix();
 glColor3f(0.0,0.0,0.0);
 alreadydownload2();

 glPopMatrix();

 glPushMatrix();

 glColor3f(0.0,0.0,0.0);

datatwostatus();

 glPopMatrix();

glPushMatrix();

 glColor3f(0.0,0.0,0.0);

 dataonestatus();

 glPopMatrix();

 if (p17 == 216.0 && p18 <= 99.0 && p18 > 70.0)

 {p18 = p18 - 0.08;


 glutPostRedisplay();

 }

 else

 {

 if (p17 >= 216.0 && p17<289.0)

 {

 p17 = p17 + 0.08;

 glutPostRedisplay();

 }
```

```
     }
    if (p17 < 287.0)
     {
     glPushMatrix();
     glColor3f(0.6, 0.6, 0.6);
     downloading4();
     glPopMatrix();
   }
     else
     {
     glPushMatrix();
     glColor3f(0.0, 0.0, 0.0);
     downloading4();
     glPopMatrix();
     download4();
     downc4flag=1;
    if(downc4flag==1 )
                                                                      {
 glPushMatrix();
glColor3f(0.0,0.0,0.0);
alreadydownload2();
glPopMatrix();
glPushMatrix();

glColor3f(0.6,0.6,0.6);

alreadydownload2();

glPopMatrix();

}

}

}
```

### 2.3.6 TCP handshake.

TCP is dipicted using the handshake1H() function.



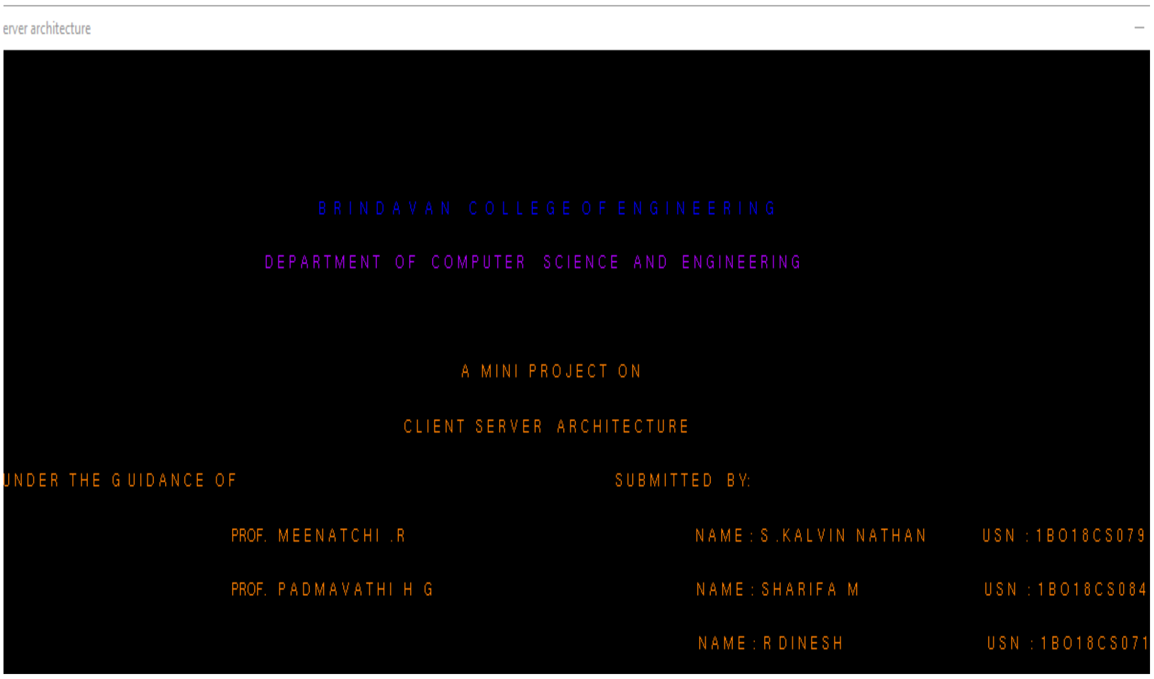
<<pseudocode for TCP >>

```
void handshake1H()//handshake process function

{
```

```
if (p1 >= 107.0 && p2 == 300.0 && p1 < 200.0)
        {    p1 = p1 + 0.08;
                glPushMatrix();
                glPointSize(2.0);
                glColor3f(1.0, 1.0, 1.0);
                glBegin(GL_POINTS);
                glVertex2f(p1, p2);
                glEnd();
                glPopMatrix();
                glutPostRedisplay();
        }
        else
        { if (p1 >= 200.0 && p2>278.0)
                {
                        p2 = p2 - 0.08;
                        glPushMatrix();
                        glColor3f(1.0, 1.0, 1.0);
                        glPointSize(2.0);
                        glBegin(GL_POINTS);
                        glVertex2f(p1, p2);
                        glEnd();
                        glPopMatrix();
                        glutPostRedisplay();

                }

        }

        glPushMatrix();

        glColor3f(0.6, 0.6, 0.6);

        request();

        glPopMatrix();

        if (p1 >= 200.0 & p2 <= 278.0)

        {

                handshake1V();

        }

        glFlush();

}
```

# CHAPTER 3

# SCREENSHOTS



erver architecture

B R I N D A V A N  C O L L E G E O F E N G I N E E R I N G

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A MINI PROJECT ON

CLIENT SERVER ARCHITECTURE

UNDER THE GUIDANCE OF

PROF. MEENATCHI .R

PROF. PADMAVATHI H G

SUBMITTED BY:

NAME : S .KALVIN NATHAN    USN : 1BO18CS079

NAME : SHARIFA M    USN : 1BO18CS084

NAME : R DINESH    USN : 1BO18CS071

Fig 3.1 Home Screen

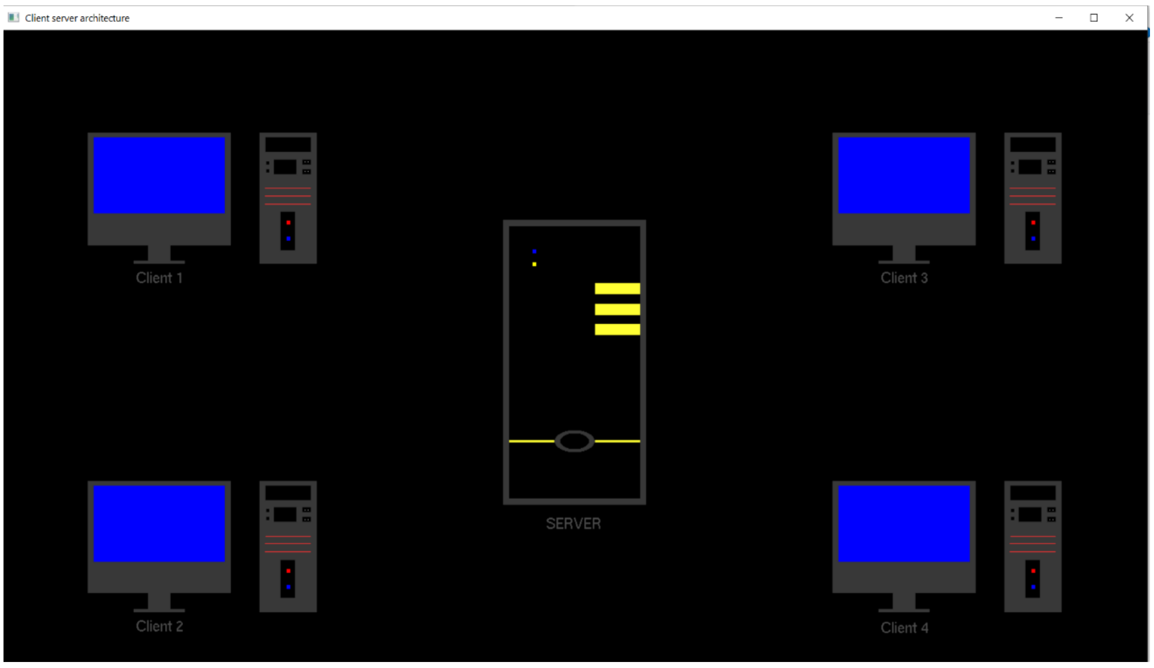


Fig 3.2 Client and Server

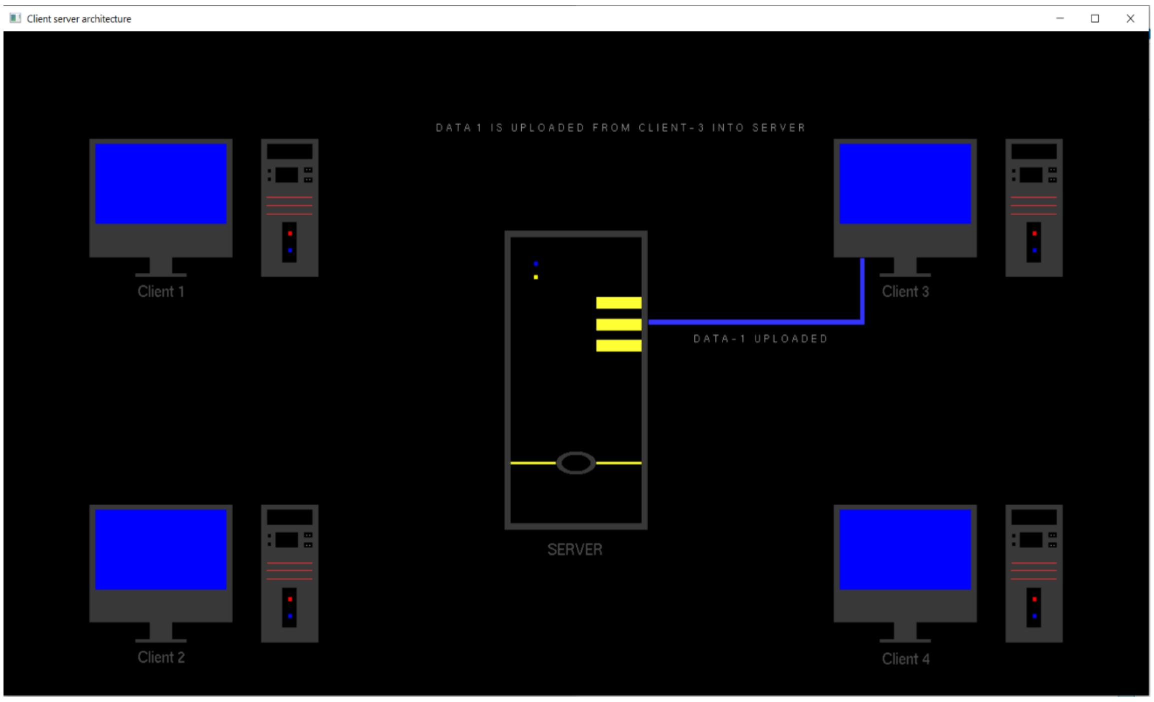

Client server architecture
DATA 1 IS UPLOADED FROM CLIENT-3 INTO SERVER
Client 1
Client 3
DATA-1 UPLOADED
SERVER
Client 2
Client 4

**Fig 3.3 Uploading data from client 3**

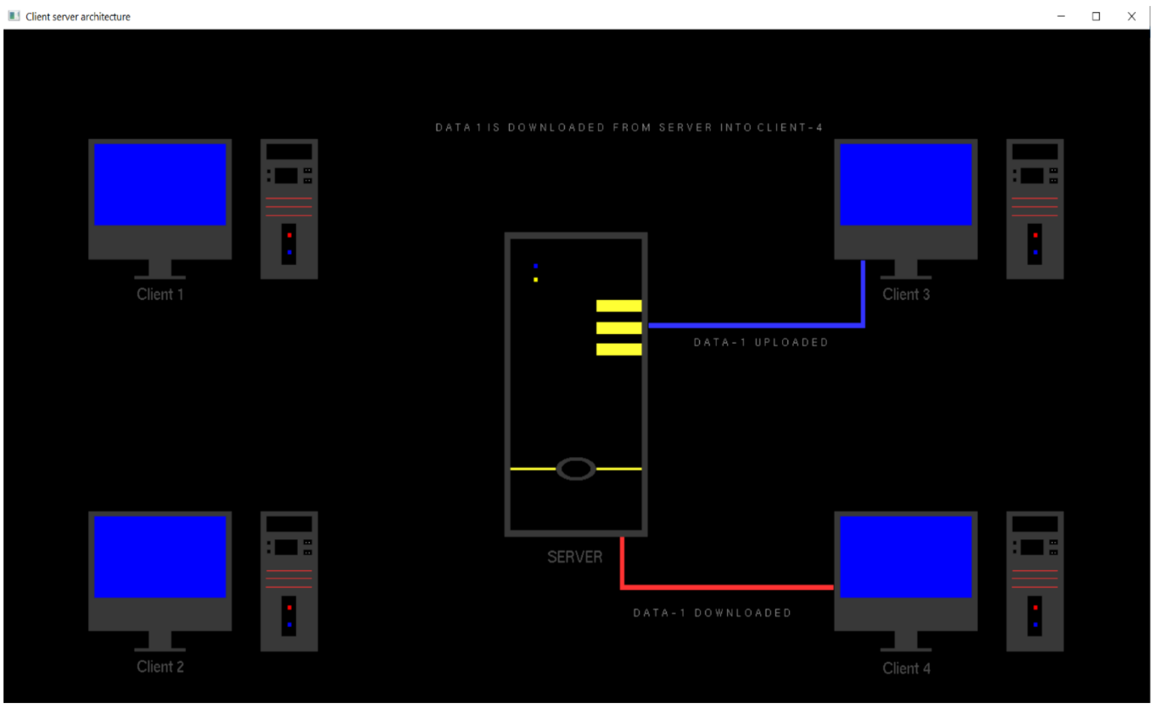

Client server architecture
DATA 1 IS DOWNLOADED FROM SERVER INTO CLIENT-4
Client 1
Client 3
DATA-1 UPLOADED
SERVER
DATA-1 DOWNLOADED
Client 2
Client 4

**Fig 3.4 Downloading data from client 4**

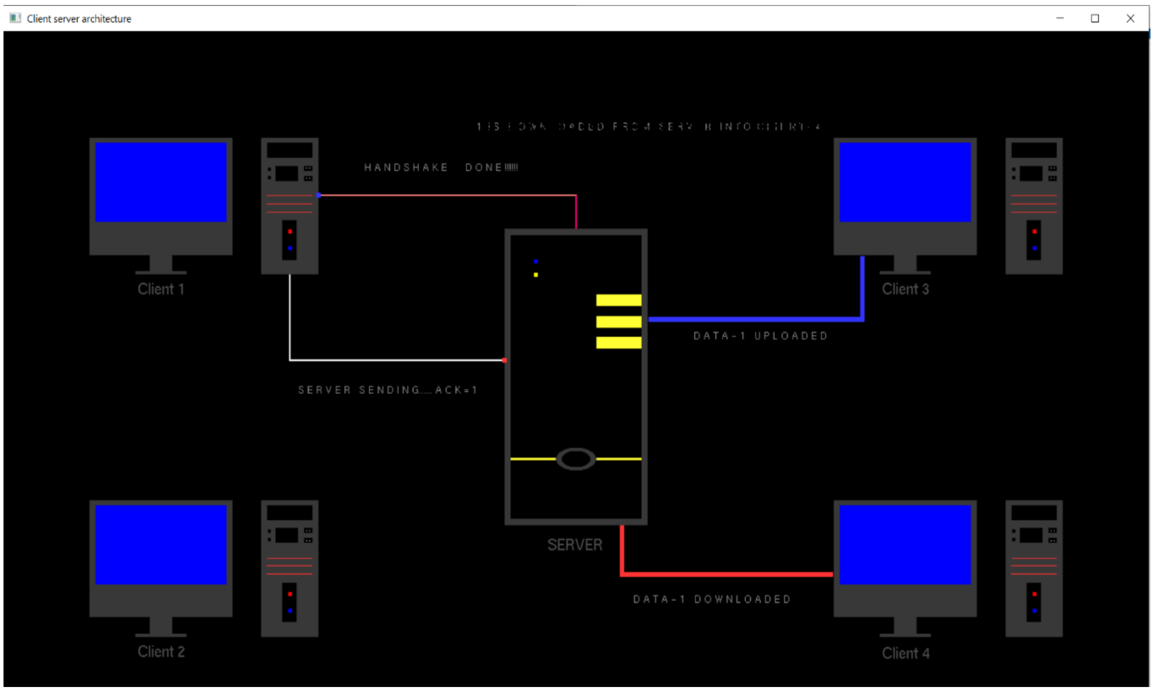


**Fig 3.5 TCP Handshake**

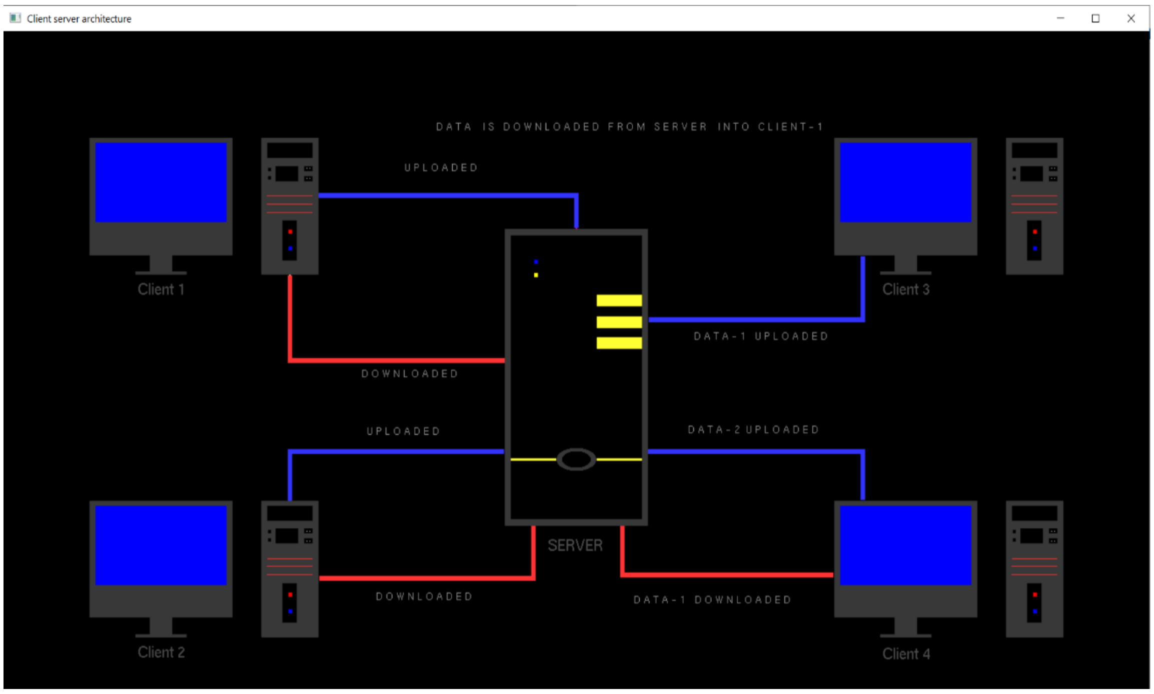


**Fig 3.6 Completion of all process**

# CHAPTER 4

# CONCLUSION and FUTURE WORK

The primary goal of this project is to show how a client communicates with a server. As visual memories are more easily remembered or recalled  than text based reading this can help people to visualize the concepts of computer network.This can help in educational institutes to introduce the student to networking fundamentals and basics help them in understanding the subject matter visually.

This project can be upgraded to a 3-D environment and additional protocols can be implemented such as ping,smtp,ftp etc.And in addition to that we can also upgrade it to simulate communication in a mobile client.

# CHAPTER 5

# REFERENCES

- http://www.google.com/
- https://www.khronos.org/
- http://www.opengl.org/documentation/
- https://github.com