VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgavi-590014

CG Mini Project Report on

"CONGESTION CONTROL"

Submitted to

Visvesvaraya Technological University

In the partial fulfilment of requirements for the award of degree

Bachelor of Engineering

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

ABHISHEK D M	**4RA21CS001**
DARSHAN B R	**4RA21CS015**

Under The Guidance of

Mr. SANJAY M BE, M.Tech

Assistant Professor, Dept. of CSE

Rajeev Institute of Technology, Hassan

RAJEEV INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

HASSAN-573201

2023-2024

RAJEEV INSTITUTE OF TECHNOLOGY, HASSAN

(Approved by AICTE, New Delhi and Affiliated to VTU, Belagavi.)

Plot #1-D, Growth Centre, Industrial Area, B-M Bypass Road, Hassan – 573201Ph:(08172)-243180/83/84 Fax:(08172)-243183

Department of Computer Science and Engineering

CERTIFICATE

Certified that the **CG Mini Project** entitled **“CONGESTION CONTROL”** is carried out by **Mr. ABHISHEK D M[4RA21CS001], Mr. DARSHAN B R [4RA21CS015]** and respectively, a bonafide students of **Rajeev Institute Of Technology,** Hassan in partial fulfillment of the requirements forthe **6th Semester** of **Bachelor Of Engineering** in **Computer Science And Engineering** of the **Visvesvaraya Technological University**, Belagavi during the year 2023-2024. The Project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

Mr. Sanjay M	**Dr. Shrishail math**
Assistant Professor	Head of the Department
Dept. of CSE	Computer Science Engineering
RIT, HASSAN	**RIT, HASSAN**

Name of the examiners	Signature with date
1.	……………………
2. ……………….....	……………………

DECLARATION

We **ABHISHEK D M [4RA21CS001], DARSHAN B R**, [**4RA21CS015],** students of 6th Semester B.E in **Computer Science and Engineering**, **Rajeev Institute of Technology, Hassan**, hereby declare that the work being presented in the dissertation entitled " **CONGESTION CONTROL** " has been carried out by us under the supervision of guide **Mr. Sanjay M**, Assistant Professor, Computer Science and Engineering, **Rajeev Institute of Technology, Hassan**, as partial fulfillment of requirement for the award of B.E Degree of **Bachelor of Engineering in Computer Science and Engineering** at **Visvesvaraya Technological University, Belagavi** is and authentic record of my own carried out by us during the academic year 2023-2024.

ABHISHEK D M	DARSHAN B R
[4RA21CS001]	[4RA21CS015]

PLACE: HASSAN

DATE: ....

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success.

We would like to profoundly thank our **Management of Rajeev Institute of Technology** & our President **Dr. Rachana Rajeev** for providing such a healthy environment.

We would like to express our sincere thanks to our principal **Dr. Mahesh P K**, Rajeev Institute of Technology for his encouragement that motivated us for successful completion of project work.

We wish to express our gratitude to **Dr. Shrishail math**, Head of the Department of Computer Science & Engineering for providing a good working environment and for his constant support and encouragement.

It gives us great pleasure to express our gratitude to **Mr. Sanjay M** Assistant Professor, Department of Computer Science & Engineering for her expert guidance, initiative and encouragement that led us to complete this project.

We would also like to thank all our staffs of Computer Science and Engineering department who have directly or indirectly helped us in the successful completion of this project and also, we would like to thank our parents.

ABHISHEK D M	DARSHAN B R
[4RA21CS001]	[4RA21CS015]

ABSTRACT

Main aim of this Mini Project is to illustrate the concepts and usage of Congestion Control in OpenGL. In data network and queuing theory, network congestion occurs when a link or node is carrying so much data that its quality of service deteriorates When more packets were sent than could be handled by intermediate routers, the intermediate routers discarded many packets, expecting the end points of the network to retransmit the information. When this packet loss occurred, the end points sent *extra* packets that repeated the information lost, doubling the data rate sent, exactly the opposite of what should be done during congestion. This pushed the entire network into a 'congestion collapse' where most packets were lost and the resultant throughput was negligible. We have used input devices like mouse and key board to interact with program

CONTENTS

LIST OF FIGURES

LIST OF SNAPSHORTS

Chapter 1

INTRODUCTION

1.1 Computer Graphics

- Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D Or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly.
- Computers have become a powerful medium for the rapid and economical production of pictures.
- There is virtually no area in which graphical displays cannot be used to some advantage.
- Graphics provide a so natural means of communicating with the computer that they have become widespread.
- Interactive graphics is the most important means of producing pictures since the invention of photography and television.
- We can make pictures of not only the real world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry.
- A computer graphics system is a computer system with all the components of the general purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.

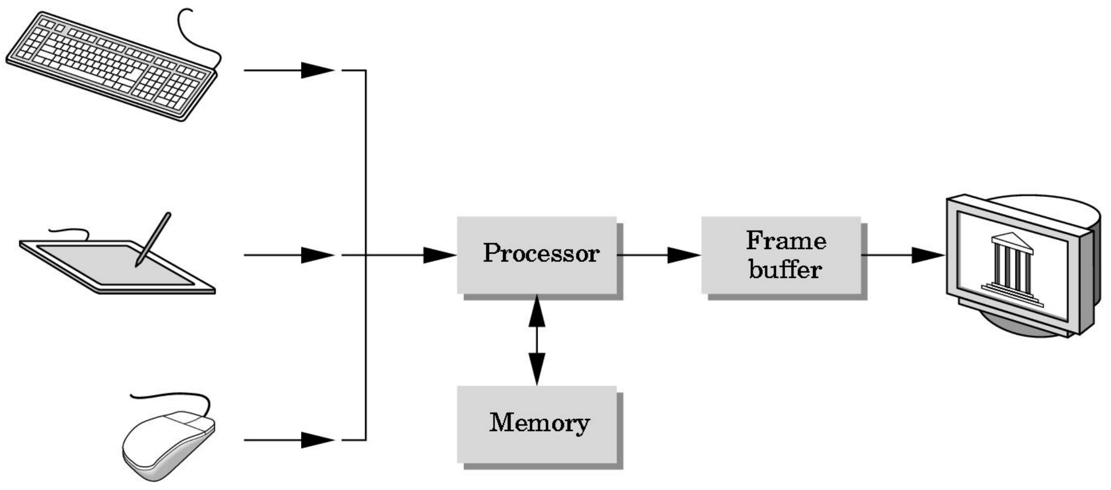


Fig 1.1: CG Model

1.2 OpenGL Technology

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms.

OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

OpenGL Available Everywhere: Supported on all UNIX® workstations, and shipped standard with every Windows 95/98/2000/NT and MacOS PC, no other graphics API operates on a wider range of hardware platforms and software environments.

OpenGL runs on every major operating system including Mac OS, OS/2, UNIX, Windows 95/98, Windows 2000, Windows NT, Linux, OPEN Step, and BeOS; it also works with every major windowing system, including Win32, MacOS, Presentation Manager, and X-Window System. OpenGL is callable from Ada, C, C++, Fortran, Python, Perl and Java and offers complete independence from network protocols and topologies.

The OpenGL interface Our application will be designed to access OpenGL directly through functions in three libraries namely: gl, glu, glut

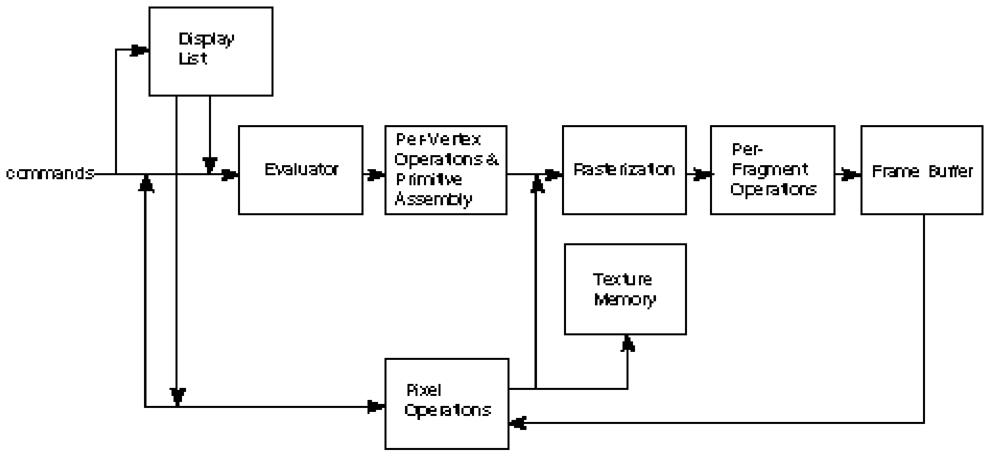


Fig 1.2: OpenGL block diagram

Chapter 2

REQUIREMENT SPECIFICATION

2.1 Hardware Requirements

- The standard output device is assumed to be a **Color Monitor**. It is quite essential for any graphics package to have this, as provision of color options to the user is a must. The **mouse**, the main input device, has to be functional i.e. used to move the car left orright in the game. A **keyboard** is used for controlling and inputting data in the form of characters, numbers i.e. to enter the player name in Need For Speed 2012, etc. Apart from these hardware requirements there should be sufficient hard disk space and primary memory available for proper working of the package to execute the program. Pentium III orhigher processor, 128MB or more RAM, VGA monitor. A functional display card.

Minimum Requirements Expected are cursor movement, creating objects like lines, squares, rectangles, polygons, etc. Transformations on objects/selected area should be possible. Filling of area with the specified color should be possible

2.2 Software Requirements

- The editor has been implemented on the OpenGL platform and mainly requires an appropriate version of eclipse compiler to be installed and functional in Ubuntu. Thought itis implemented in Open GL, it is pretty much performed and independent with the restriction that there is support for the execution of C and C++ files. Text Modes is recommended.
- Operating System : Windows
- GLUT libraries, GLUT utility toolkit must be available
- Language : C++
- Developed platform: VS code

Chapter 3

SYSTEM DESIGN

Existing System

Existing system for a graphics is the TC++ . This system will support only the 2D graphics. 2D graphics package being designed should be easy to use and understand. It should provide various option such as free hand drawing , line drawing , polygon drawing , filled polygons, flood fill, translation , rotation , scaling , clipping etc. Even though these properties were supported, it was difficult to render 2D graphics cannot be . Very difficult to get a 3 Dimensional object. Even the effects like lighting , shading cannot be provided. So we go for Microsoft Visual Studio software.

Proposed System

To achieve three dimensional effects, open GL software is proposed . It is software which provides a graphical interface. It is a interface between application program and graphics hardware. the advantages are:

- Open GL is designed as a streamlined.
- It's a hardware independent interface i.e. it can be implemented on many different hardware platforms.
- With OpenGL, we can draw a small set of geometric primitives such as points, lines and polygons etc.
- Its provides double buffering which is vital in providing transformations.
- It is event driven software.
- It provides call back function.
- It's a hardware independent interface i.e. it can be implemented on many different hardware platforms.
- With OpenGL, we can draw a small set of geometric primitives such as points, lines and polygons etc.

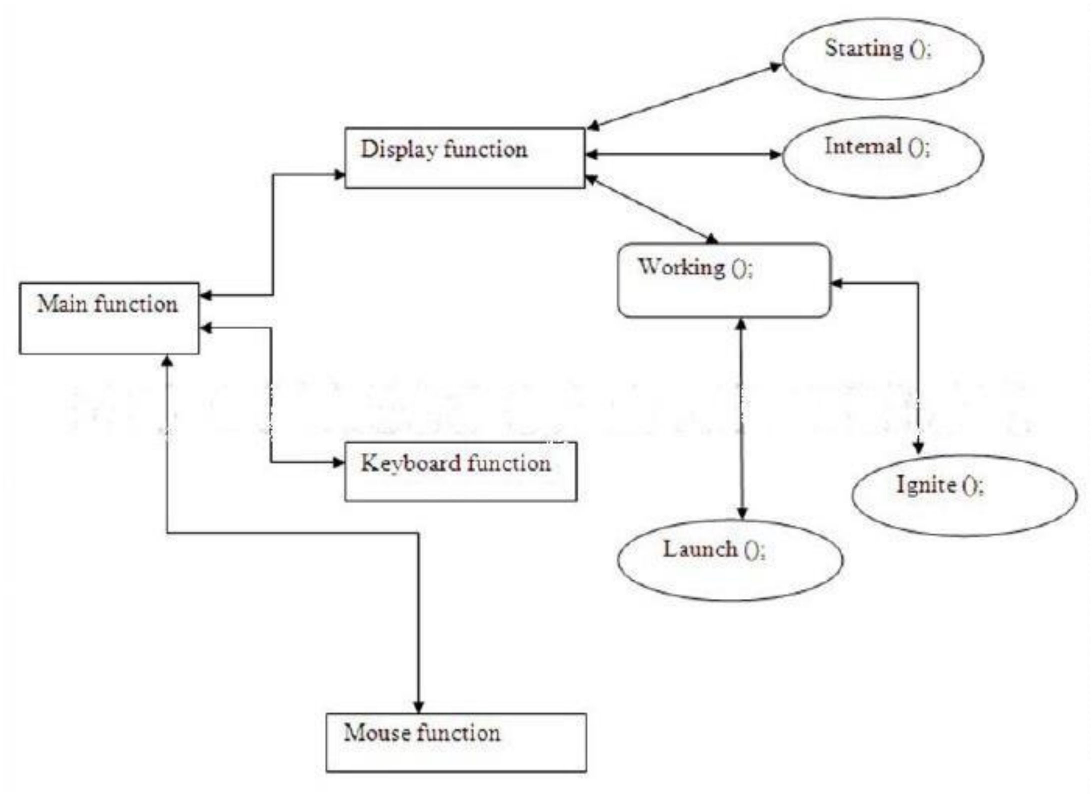


Fig 3.1: System Design

3.2 DETAILED DESIGN

Transformation Functions

Matrices allow arbitrary linear transformations to be represented in a consistent format, suitable for computation. This also allows transformations to be concatenated easily (by multiplying their matrices).

Linear transformations are not the only ones that can be represented by matrices. Using homogenous coordinates, both affine transformation and perspective projection on $\mathbf{R}^n$ can be represented as linear transformations on $\mathbf{RP}^{n+1}$ (that is, n+1- dimensional real projective space). For this reason, 4x4 transformation matrices are widely used in 3D computer graphics.

3-by-3 or 4-by-4 transformation matrices containing homogeneous coordinates are often called, somewhat improperly, "*homogeneous transformation matrices*". However, the transformations they represent are, in most cases, definitely non-homogeneous and non-linear (like translation, roto-translation or perspective projection). And even the matrices themselves look rather heterogeneous, i.e. composed of different kinds of elements (see below). Because they are multi-purpose transformation matrices, capable of representing both affine and projective transformations, they might be called "*general transformation matrices*", or, depending on the application, "*affine transformation*" or "*perspective projection*" matrices. Moreover, since the homogeneous coordinates describe a projective vector space, they can also be called "*projective space transformation matrices*".

Finding the matrix of a transformation

If one has a linear transformation $T(x)$ in functional form, it is easy to determine the transformation matrix **A** by simply transforming each of the vectors of the standard basis by T and then inserting the results into the columns of a matrix. In other words,

$$\mathbf{A} = \begin{bmatrix} T(\vec{e}_1) & T(\vec{e}_2) & \cdots & T(\vec{e}_n) \end{bmatrix}$$

For example, the function $T(x) = 5x$ is a linear transformation. Applying the above process (suppose that $n = 2$ in this case) reveals that

$$T(\vec{x}) = 5\vec{x} = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \vec{x}$$

Examples in 2D graphics

Most common geometric transformations that keep the origin fixed are linear, including rotation, scaling, shearing, reflection, and orthogonal projection; if an affine transformation is not a pure translation it keeps some point fixed, and that point can be chosen as origin to make the transformation linear. linear transformation $T(x)$ in functional form, it is easy to determine the transformation matrix **A** by simply transforming each of the vectors of the standard basis by T and then inserting the results into the columns of a matrix. In two dimensions, linear transformations can be represented using a 2×2 transformation matrix.

Rotation:

For rotation by an angle θ anticlockwise about the origin, the functional form is $x' = x\cos\theta - y\sin\theta$ and $y' = x\sin\theta + y\cos\theta$. Written in matrix form, this becomes:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Similarly, for a rotation clockwise about the origin, the functional form is $x' = x\cos\theta + y\sin\theta$ and $y' = -x\sin\theta + y\cos\theta$ and the matrix form is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scaling:

For scaling (that is, enlarging or shrinking), we have $x' = s_x \cdot x$ and $y' = s_y \cdot y$. The matrix form is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

When $s_x s_y = 1$, then the matrix is a squeeze mapping and preserves areas in the plane.

Shearing:

For shear mapping (visually similar to slanting), there are two possibilities. For a shear parallel to the *x* axis has $x' = x + ky$ and $y' = y$; the shear matrix, appliedto column vectors, is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

A shear parallel to the *y* axis has $x' = x$ and $y' = y + kx$, which has matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Chapter 4

IMPLIMENTATION

4.1 Functions Used

1. **glClear**

   Takes a single argument that is the bitwise OR of several values indicating which buffer is to be cleared.

2. **glClearColor**

   Specifies the red, green, blue, and alpha values used by glClear to clear the color buffers.

3. **glLoadIdentity**

   Replaces the current matrix with the identity matrix.

4. **glMatrixMode**

   Sets the current matrix mode , *mode* can be GL _ MODELVIEW , GL _PROJECTION or GL_TEXTURE**.**

5. **glutCreateWindow**

   Creates a top-level window.

6. **glutDisplayFunc**

   Sets the display callback for the *current window*

7. **glutInit**

   Initializes the GLUT library and negotiates a session with the window system.

8. **glutInitDisplayMode**

   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.

9. **glutInitWindowPosition**

   Sets the initial window position.

10. **glutInitWindowSize**

It specifies the size of the window to be created.

11. glvertex()

It is used to set the x, y, z vertex values.

12. glutBitmapCharacter

Displays an character on the screen.

13. glutSwapBuffer

Swaps between the front and back buffers.

14. glFlush()

Forces any buffered OpenGL commands to exec.

15. glEnd()

Termiinates a list of vertices.

4.2 Architecture

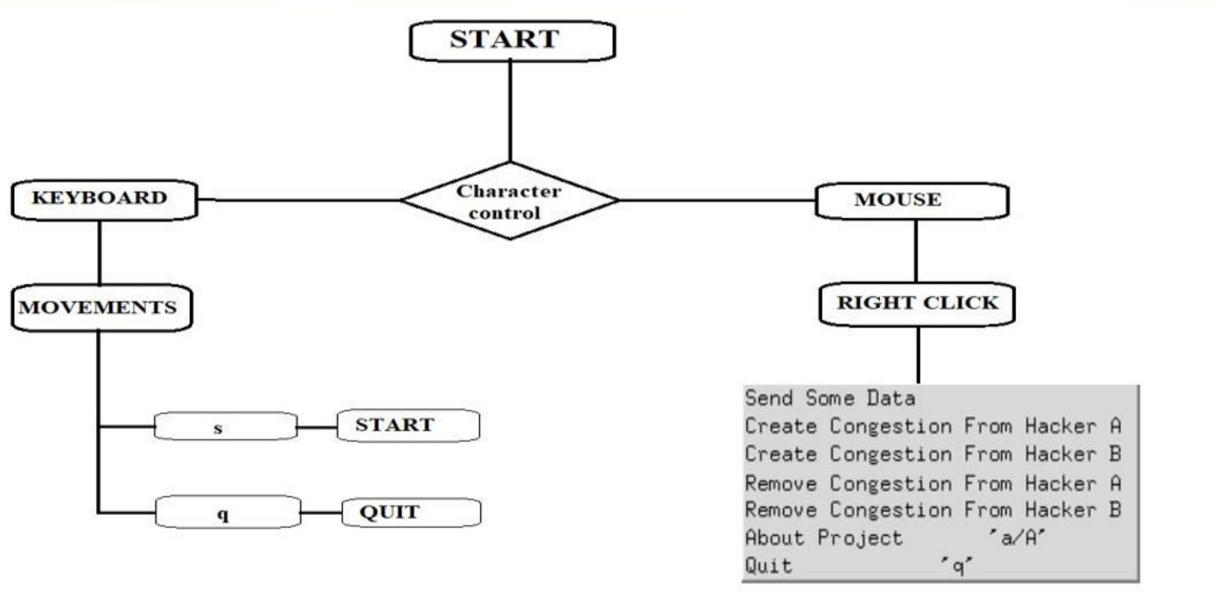

START
Character control
KEYBOARD
MOUSE
MOVEMENTS
RIGHT CLICK
s
START
q
QUIT
Send Some Data
Create Congestion From Hacker A
Create Congestion From Hacker B
Remove Congestion From Hacker A
Remove Congestion From Hacker B
About Project 'a/A'
Quit 'q'

Fig 4.1: Architecture

4.3 Appendix

```
#include <windows.h>
#include<string.h>
#include<stdarg.h>
#include<stdio.h>

#include <GL/glut.h>

void *font;
void *currentfont;
static double x=0.0,x1=-3.5,x2=3.5,y1=-1.4,y2=-1.4,x3=-3.5,y3=1.3;
static double move=-60;
static bool goDown=false,goup=false,down=false,congested=false,remote=false;

void setFont(void *font)
{
   currentfont=font;
}

void drawstring(float x,float y,float z,char *string)
{
   char *c;
   glRasterPos3f(x,y,z);

   for(c=string;*c!='\0';c++)
   {   glColor3f(0.0,1.0,1.0);
      glutBitmapCharacter(currentfont,*c);
   }
}

void
stroke_output(GLfloat x, GLfloat y, char *format,...)
{
   va_list args;
   char buffer[200], *p;
   va_start(args, format);
   vsprintf(buffer, format, args);
   va_end(args);
   glPushMatrix();
   glTranslatef(x, y,2);
   glScaled(0.003, 0.005, 0.005);
   for (p = buffer; *p; p++)
   glutStrokeCharacter(GLUT_STROKE_ROMAN, *p);
   glPopMatrix();
}

void cloud(){

   //vertical eclipse
   glPushMatrix();
   glTranslatef(0.0,0.0,0.0);
   glScaled(1,3.5,0.1);
```

```
    glColor3f(1,1,1);
    glutSolidSphere(1,100,100);
    glPopMatrix();

    //horizotal eclipse
    glPushMatrix();
    glTranslatef(0.0,0.0,0.0);
    glScaled(3,2.65,0.1);
    glColor3f(1,1,1);
    glutSolidSphere(1,100,100);
    glPopMatrix();


glPushMatrix();
    glScaled(3.5,1.8,0.1);
    glRotatef(60,0.0f,1.0f,1.0f);
    glTranslatef(0.0,0.0,0.0);
    glColor3f(1,1,1);
    glutSolidSphere(1,100,100);
    glPopMatrix();

    glPushMatrix();
    glScaled(1.5,3.5,0.1);
    glRotatef(50,0.0f,1.0f,1.0f);
    glTranslatef(0.0,0.0,0.0);
    glColor3f(1,1,1);
    glutSolidSphere(1,100,100);
    glPopMatrix();
}

void router(float x6, float y6,float z6)
{

    glColor3f(0.2,0.3,1);
    glPushMatrix();
    glScaled(0.2,1.0,0.1);
    glRotatef(91,1.0f,0.0f,0.0f);
    glTranslatef(x6,y6,z6);
    glutSolidTorus(0.2,1.5,100,100);
    glPopMatrix();
}

void dte(float x1, float x2)
{
    glPushMatrix();
    glTranslatef(x1,x2,0.1);
    glScaled(1,0.1,0.1);
    glColor3f(1,1,1);
    glRotatef(0,0.0f,1.0f,0.0f);
    glutWireCube(1.5);
    glPopMatrix();
}

void line()
{
```

```
glPushMatrix();
    glBegin(GL_LINE_LOOP);
    glColor3f(1,0,0);
    glVertex3f(0.1,-0.2,1);
    glVertex3f(0.0,0,1);
    glVertex3f(2,1.3,1);
    glVertex3f(2.1,1.1,1);
    glEnd();
glPopMatrix();
}

void line1()
{
glPushMatrix();
    glBegin(GL_LINE_LOOP);
    glColor3f(1,0,0);
    glVertex3f(0.2,2.5,1);
    glVertex3f(0.1,2.3,1);
    glVertex3f(2.1,1.1,1);
    glVertex3f(2.3,1.2,1);
    glEnd();
glPopMatrix();
}

void rack(float x1, float x2)
{
glPushMatrix();
    glScaled(0.35,.05,1.0);
    glTranslatef(x1,x2,0.1);
    glColor3f(0,0,0.3);
    glutSolidCube(1.1);
    glPopMatrix();
}

void rack1(float x1, float x2)
{
glPushMatrix();
    glScaled(1.75,.1,1.0);
    glTranslatef(x1,x2,0.3);
    glColor3f(0,0,1.3);
    glutSolidCube(0.5);
    glPopMatrix();
}

void window(float x1, float x2)
{
glPushMatrix();
    glScaled(0.5,.3,1.0);
    glTranslatef(x1,x2,0.4);
    glColor3f(0.1,0.1,0.1);
    glutSolidCube(0.5);
    glPopMatrix();
}

void top()
```

```
{
glBegin(GL_TRIANGLES);

   /* Front */
   glColor3f(1.0f,1.0f,1.0f);
   glVertex3f( 0.0f, 1.0f, 0.0f);
   glColor3f(1.0f,1.0f,1.0f);
   glVertex3f(-0.5f,-0.5f, 0.5f);
   glColor3f(1.0f,1.0f,1.0f);
   glVertex3f( 0.5f,-0.5f, 0.5f);

   /* Left */
   glColor3f(1.0f,1.0f,1.0f);
   glVertex3f(0.0f,1.0f,0.0f);
   glColor3f(1.0f,1.0f,1.0f);
   glVertex3f(-0.5f,-0.5f,0.5f);
   glColor3f(1.0f,1.0f,1.0f);
   glVertex3f(-0.5f,-0.5,-0.5f);

   /* Right */
   glColor3f(1.0f,1.0f,1.0f);
   glVertex3f(0.0f,1.0f,0.0f);
   glColor3f(1.0f,1.0f,1.0f);
   glVertex3f(0.5f,-0.5f,0.5f);
   glColor3f(1.0f,1.0f,1.0f);
   glVertex3f(0.5f,-0.5,-0.5f);

 /* Back */
   glColor3f(1.0f,1.0f,1.0f);
   glVertex3f( 0.0f, 1.0f, 0.0f);
   glColor3f(1.0f,1.0f,1.0f);
   glVertex3f(-0.5f,-0.5f, -0.5f);
   glColor3f(1.0f,1.0f,1.0f);
   glVertex3f( 0.5f,-0.5f, -0.5f);
   glEnd();
}

void animate();

void network()
{
  glClearColor(0.0,0,0,0.0);
 //glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   glTranslatef(0.0f,0.0f,-13.0f);

   // glRotatef(ang,1.0f,1.0f,0.0f);

   //Server
   glPushMatrix();
   glScaled(0.8,2.0,0.8);
   glTranslatef(5,0.8,0.0);
   glColor3f(0,1.5,1.5);
   glutSolidCube(1);
```

```
    glPushMatrix();
    glScaled(0.5,.1,1.0);
    glTranslatef(.0,3.5,0.01);
    glColor3f(0.3,0.3,0.3);
    glutSolidCube(1.5);
    glPopMatrix();

    rack1(.0,1);
    rack1(.0,2);
    rack1(.0,0);

    rack(-1,-4);
    rack(1,-4);
    rack(-1,-6);
    rack(1,-6);
    rack(-1,-8);
    rack(1,-8);

    glPopMatrix();

    //Home
    glPushMatrix();
    glScaled(0.8,1.0,0.8);
    glTranslatef(4.8,-2.5,0.0);
    glColor3f(1,1,0);
    glutSolidCube(1);


    glPushMatrix();
    glScaled(.7,.1,1.0);
    glTranslatef(0.0,4.5,0.2);
    glColor3f(1.3,1.3,1.3);
    glutSolidCube(1.5);
    glPopMatrix();


    glPushMatrix();
    glScaled(0.12,.3,.75);
    glTranslatef(0,-.75,0.02);
    glColor3f(0.3,0.3,0.3);
    glutSolidCube(1.5);
    glPopMatrix();

//home2
    glPushMatrix();
    glScaled(0.8,1.0,0.8);
    glTranslatef(4.8,-2.5,0.0);
    glColor3f(1,1,0);
    glutSolidCube(1);
```

Chapter 5

TESTING AND RESULT

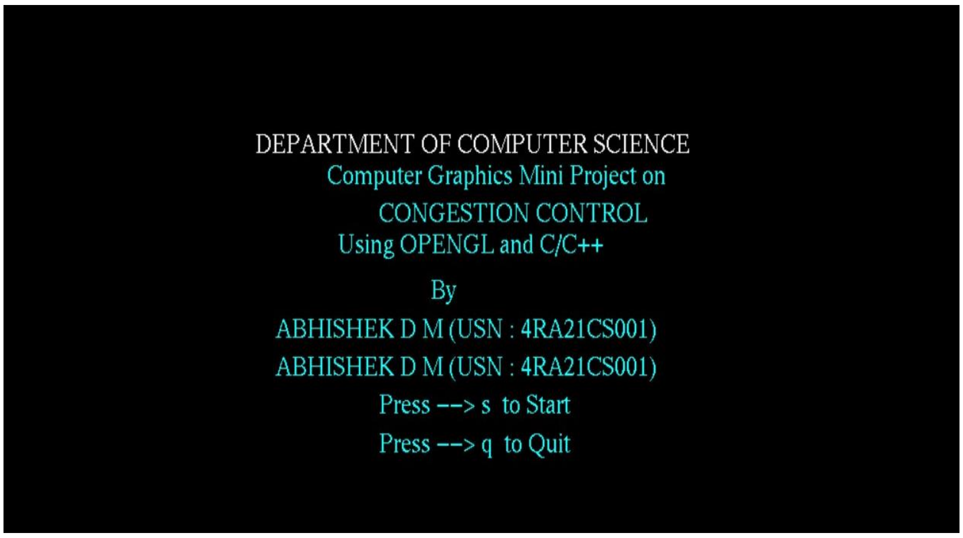
The full designing and creating of Need For Speed 2012 has been executed under windows operating system using VS code, this platform provides a and satisfies the basic need of a good compiler. Using GL/glut.h library and built in functions make it easy to design good graphics package such as this racing game.

This game allows the gamer to overtake obstacles, and accomplish 5 levels of intense racing. It tells the user what will happen if they collide with an obstacle i.e. setting an example in real life. The levels automatically increment unless the vehicle has not crashed.Testing involves unit testing, module testing and system testing. The testing phase of the project is critical for verifying that all components function correctly and meet the specified requirements. It ensures that the project is robust, performs well under various conditions, and is secure from potential vulnerabilities. This phase also helps in identifying and fixing any issues before the project is deployed, thereby enhancing the quality and reliability of the final product. The primary objectives of the testing phase were to ensure that the system operates as intended, performs efficiently under expected workloads, is secure from threats, and provides a user-friendly experience. Specific goals included validating the functionality of individual components, verifying the integration between different modules, assessing the system's performance, and identifying any security vulnerabilities. To achieve the testing objectives, various testing methodologies were employed. Unit testing was conducted to verify the functionality of individual components using JUnit. Integration testing, utilizing Selenium, was performed to ensure that different modules interact correctly. System testing was carried out to validate the overall functionality of the entire application. Performance testing, using Apache JMeter, assessed how the system performs under load, while security testing with OWASP ZAP identified potential vulnerabilities. uring the testing phase, several bugs were identified and reported. One notable issue was the failure of the login functionality when special characters were included in the username. This issue was resolved in the second iteration of testing.

Chapter 6

SNAPSHOTS

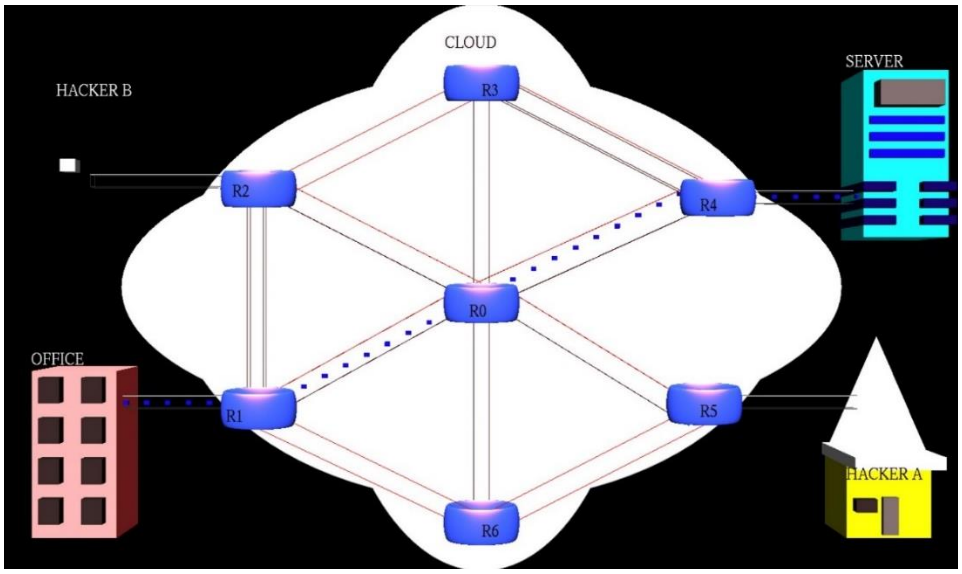
- Introduction page, press s to Start and q to Quit



DEPARTMENT OF COMPUTER SCIENCE
Computer Graphics Mini Project on
CONGESTION CONTROL
Using OPENGL and C/C++
By
ABHISHEK D M (USN : 4RA21CS001)
ABHISHEK D M (USN : 4RA21CS001)
Press --> s to Start
Press --> q to Quit

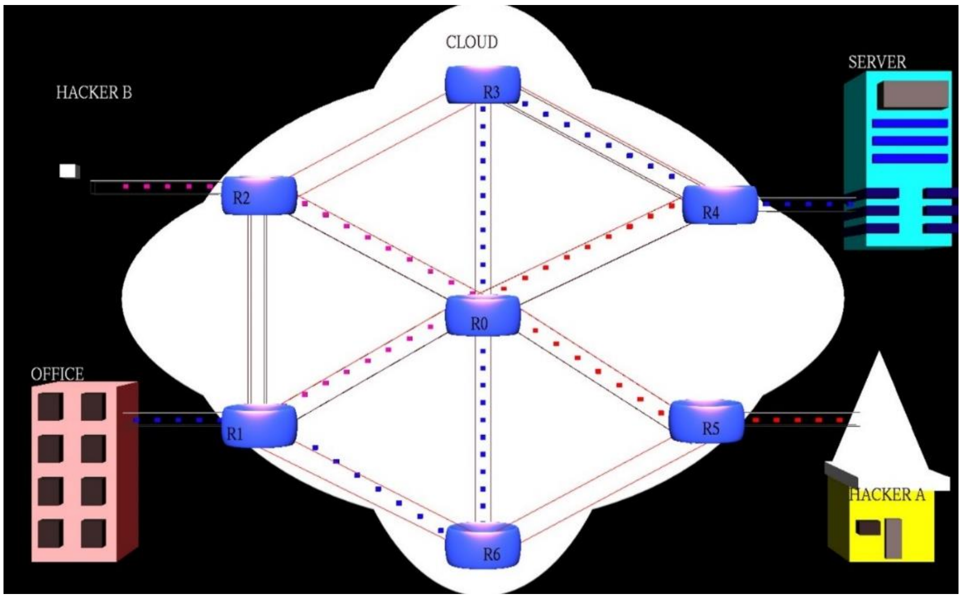
Snapshort 6.1 : Introduction page.

- Sending data from office to server through R1,R0,R4.



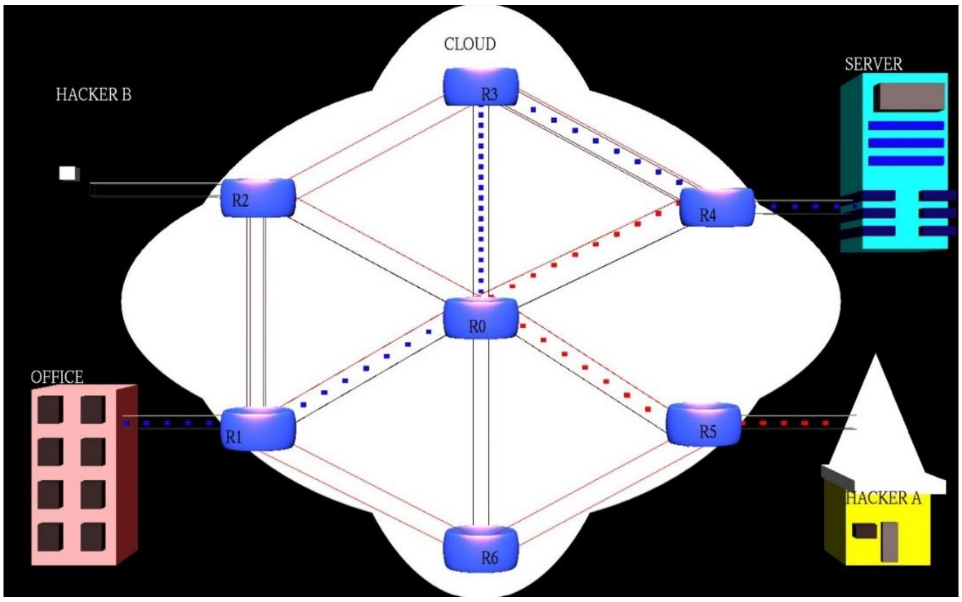
Snapshort 6.2 : Sending Data

➢ Creating congestion from both Hacker A and B.



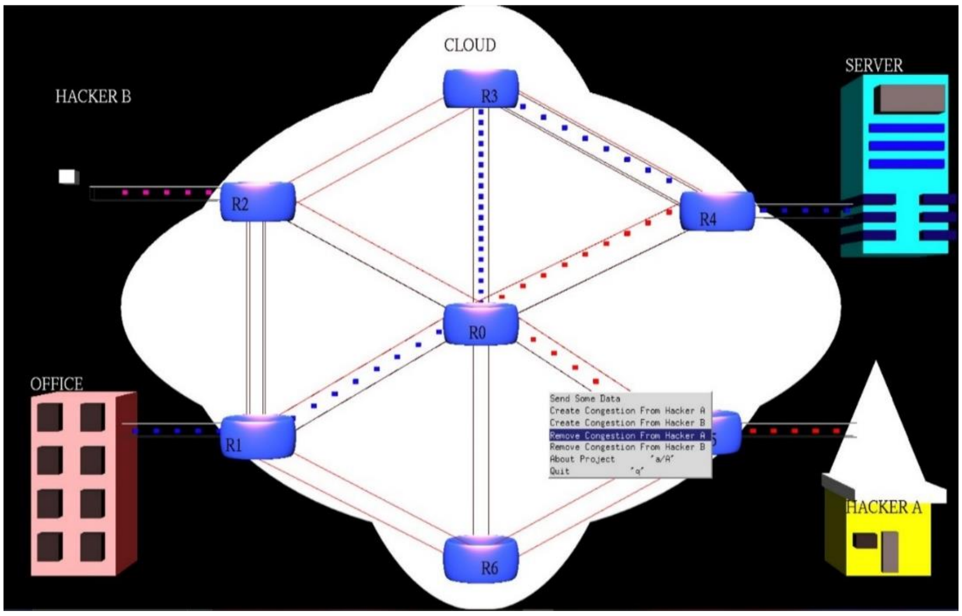
Snapshort 6.3 : Creating congestion

➢ Creating congestion from both Hacker A and sending Data from office to the server.



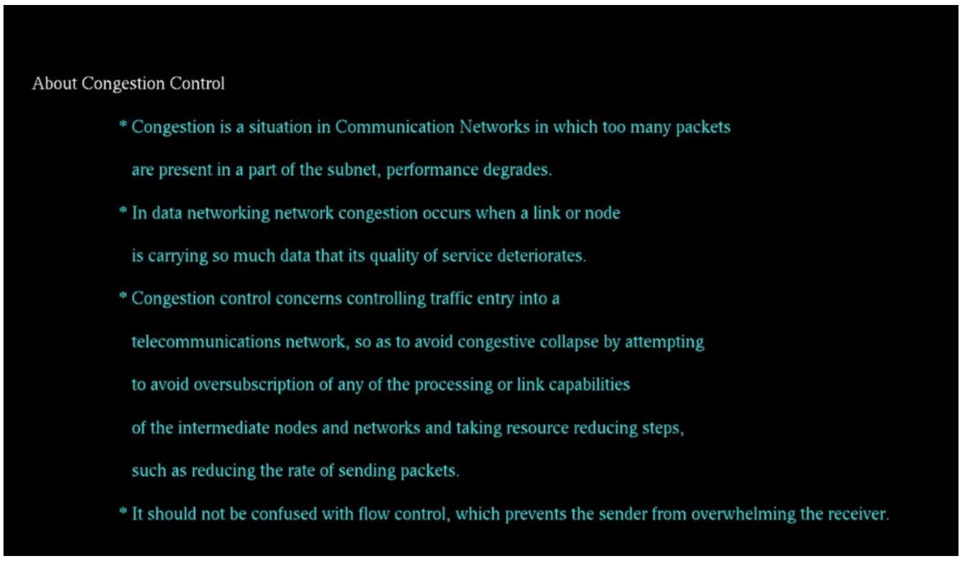
Snapshort 6.4 : Creating congestion from hackerA

- Removing congestion from Hacker B and sending the data.


CLOUD
SERVER
HACKER B
R3
R2
R4
R0
OFFICE
R1
R6
HACKER A
Send Some Data
Create Congestion From Hacker A
Create Congestion From Hacker B
Remove Congestion From Hacker A
Remove Congestion From Hacker B
About Project "a/A"
Quit "q"

Snapshort 6.5 : Removing congestion

- Brief description about Congestion control.



About Congestion Control

* Congestion is a situation in Communication Networks in which too many packets are present in a part of the subnet, performance degrades.
* In data networking network congestion occurs when a link or node is carrying so much data that its quality of service deteriorates.
* Congestion control concerns controlling traffic entry into a telecommunications network, so as to avoid congestive collapse by attempting to avoid oversubscription of any of the processing or link capabilities of the intermediate nodes and networks and taking resource reducing steps, such as reducing the rate of sending packets.
* It should not be confused with flow control, which prevents the sender from overwhelming the receiver.

Snapshort 6.6 : About Congestion control

Chapter 7

FUTURE ENHANCEMENT

- User Interface Improvements.
- Enhanced Visualization Techniques.
- Real-time Traffic Simulation.
- Algorithm Optimization.
- Advanced Collision Detection.

CONCLUSION

congestion control in computer graphics is a crucial aspect of network management that aims to optimize data transmission, enhance user experience, and ensure efficient utilization of network resources. By effectively managing data flow and mitigating congestion, this technology supports smoother rendering, reduced latency, and improved quality of service in graphics-intensive applications.

However, the implementation of congestion control comes with challenges, such as increased complexity, potential delays, and the need for adaptive algorithms to handle dynamic network conditions. Balancing these considerations is essential to achieve optimal performance while maintaining responsiveness in real-time applications.

Ultimately, the benefits of congestion control—such as smoother playback, reduced latency, and enhanced scalability—outweigh its drawbacks when carefully implemented and tailored to specific application requirements. As technology evolves, ongoing advancements in congestion control algorithms and network infrastructure will continue to play a critical role in advancing the field of computer graphics and improving the overall user experience.

REFERENCES

Books referred during this project work:

- Interactive Computer Graphics A top –down approach with OpenGL – Edward Angel, 5th Edition, Addition-Wesley 2008.
- Computer Graphics Using OpenGL – F S Hill.Jr. 2nd Edition, Pearson Education,2001.
- 1.Edward Angel-”Interactive computer Graphics A Top-Down Approach with OpenGL”,5th edition,Addison-Wesley,2008.
- 2.“Computer Graphics”: Donald Hearn, M Pauline Baker”.

Websites referred:

http://github.com
http://javapoint.com/xampp
http:// javapoint.com/opengl
http:// javapoint.com/html
www.google.com
www.opengl.org
www.w3schools.com
www.google.co.in
www.htmlguddies.com
www.beginneres.com
www.tutorialspoint.com