Visvesvaraya Technological University, Belagavi – 590010

CG MINI PROJECT REPORT
ON
Line Art and Functions

Submitted by

Darshan Gouda *4SO19CS181*

Sukith S *4SO19CS182*

Under the guidance of

Ms Jaishma Kumari B

(Assistant Professor, CSE Department)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ST JOSEPH ENGINEERING COLLEGE

Vamanjoor, Mangaluru -575028, Karnataka

2021-2022

Visvesvaraya Technological University, Belagavi – 590010

CG MINI PROJECT REPORT
ON
Line Art and Functions

Submitted by

Darshan Gouda *4SO19CS181*

Sukith S *4SO19CS182*

Under the guidance of

Ms Jaishma Kumari B

(Assistant Professor, CSE Department)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ST JOSEPH ENGINEERING COLLEGE

Vamanjoor, Mangaluru -575028, Karnataka

2021-2022

ST JOSEPH ENGINEERING COLLEGE

Vamanjoor, Mangaluru- 575 028

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

*This is to certify that the Mini project entitled **"Line Art and Functions"** is a bonafide work carried out by*

Darshan Gouda	*4SO19CS181*
Sukith S	*4SO19CS182*

Students of sixth semester B.E. Computer Science & Engineering and submitted as a part of the course Computer Graphics Laboratory with Mini Project (18CSL67), during the academic year 2021-2022.

----------	----------
Ms Jaishma Kumari B	**Dr Sridevi Saralaya**
Project Guide	**Head of the Department**

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

ABSTRACT

Line Art - An interactive computer graphics project using OpenGL libraries in C programming to create three-line functions. This project is menu driven where the menu is added to the mouse right button.

The first function is to generate running lines that bounces at each edge of the window creating pattern along its way. The second function is to generate a skeletal cube in 3D space and can be viewed from different angles using mouse buttons to move viewing angle. The third function implements free line drawing to create beautiful art of users' choice. The user can delete lines and update as required.

ACKNOWLEDGEMENT

We dedicate this page to acknowledge and thank those responsible for the shaping of the project. Without their guidance and help, the experience while constructing the dissertation would not have been so smooth and efficient.

We sincerely thank **Ms Jaishma Kumari B, Assistant Professor**, Department of Computer Science and Engineering for his guidance and valuable suggestions which helped us to complete the project.

We owe our profound gratitude to **Dr Sridevi Saralaya, Head of the Department**, Computer Science and Engineering, whose kind consent and guidance helped us to complete this work successfully.

We are extremely thankful to our **Director, Rev. Fr Wilfred Prakash D'Souza**, our **Principal, Dr Rio D'Souza**, and **Assistant Director, Rev. Fr Alwyn Richard D'Souza** for their support and encouragement.

We would like to thank all our Computer Science and Engineering staff members who have always been with us extending their support, precious suggestions, guidance, and encouragement throughout in all possible ways.

We also extend our gratitude to our friends and family members for their continuous support.

CONTENTS

CHAPTER 1 – INTRODUCTION

1.1 - Problem Definition

To implement Line functions with OpenGL in c program and its implications along with various user controls and viewing properties in order to make it more interactive and fun with its application.

1.2 - Scope and importance

This project involves line functions implemented in 2D and 3D perspective enabling us to understand different functionality using OpenGL commands in order to learn, design and implement in different fields of graphic application. With advancement of graphic designs and modelling , OpenGL has become one better option for creating visual arts , video games and other interactive applications. This project allows to work with lines, generate different patterns, view line models in suitable angles and even draw freely to express their idea of picture art.

CHAPTER 2 – SOFTWARE REQUIREMENT SPECIFICATION

2.1 Functional Requirements

This project implements several OpenGL functions along with C-libraries. Some features used in this project are as follows.

Random Lines : Used to run line generation from any point on screen selected with mouse. The line rebounds at every surface or the window edge it collides to making beautiful pattern on the screen.

Shapes -> Cube : This is to run a 3D model cube allowing the user to change camera position in order to have different viewing angles.

Draw : Used to run a simple draw program that allows users to draw anything with lines on the screen. The drawn lines can be deleted one at a time or all at once.

Undo : To deletes one line drawn last on the screen

Clear : To clear whole drawing space and create a blank draw page

Home : Returns the user to main page

Exit : Closes the application

2.2 Software Requirements

Operating System: Linux , Windows (with OpenGL library)

Language: C-language

Tools: VisualStudioCode

2.3 Hardware Requirements

Installed Memory (RAM): 1GB or Higher

Processor: 1GHz or Higher

Hard disk space: 100Mb Availability

CHAPTER 3 – IMPLEMENTATION

This project involves four C-program files to implement different output screens. The front page, bouncing line and free drawing involves 2D mode while the cube rotation and viewing involves 3D space.

Program:

main.c

// Creating home screen with menu functions

```
#include <stdio.h>
#include <GL/glut.h>
#define pi 3.142
static GLfloat angle = 0;
int x=100,y=0,x11=450,y11=300,x22=600,y22=300;
int r=0,op,a,a1,a2,b,i,j,width=650,height=700,first=0;
int mouseX = 0, mouseY = 0;
static int submenu;
static int mainmenu;
#define MAX_PTS 1000
int ptListX[MAX_PTS];
int ptListY[MAX_PTS];
int noOfPts = 0;
int closed = 0;

void init(){
   gluOrtho2D(-1000, 1000, -1000, 1000);
}
void drawhead(float x, float y, float z, char *string){
   glColor3f(1, 1, 1);
   glRasterPos3f(x, y, z);

   for (char *c = string; *c != '\0'; c++){
       glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, *c);
   }
}
void drawsubhead(float x, float y, float z, char *string){
   glColor3f(1, 1, 1);
   glRasterPos3f(x, y, z);
```

```
    for (char *c = string; *c != '\0'; c++){
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, *c);
    }
}

void mouse(int button, int state, int x, int y){
    mouseX = x;
    mouseY = y;

    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN){
        if (closed || noOfPts >= MAX_PTS - 1)
            noOfPts = 0;
        closed = 0;
        ptListX[noOfPts] = mouseX;
        ptListY[noOfPts] = mouseY;
        noOfPts ++;
    }
    if ( button == GLUT_MIDDLE_BUTTON && state == GLUT_DOWN
)
        closed = 1;
}
void front(){
    char cn[] = "ST JOSEPH ENGINEERING COLLEGE";
        drawhead(-445, 800, 0, cn);
        char pn[] = "Vamanjoor, Mangaluru - 575028";
        drawsubhead(-275, 750, 0, pn);
        char dn[] = "DEPARTMENT OF COMPUTER SCIENCE &
ENGINEERING";
        drawhead(-675, 550, 0, dn);
        char prn[] = "A Mini Project On";
        drawsubhead(-175, 350, 0, prn);
        char pro[] = "Line Art using OpenGL";
        drawhead(-275, 250, 0, pro);
        char pb[] = "PROJECT BY: ";
        drawhead(-715, -200, 0, pb);
        char p1[] = "Darshan Gouda";
        drawhead(-625, -350, 0, p1);
        char p1u[] = "4SO19CS181";
        drawsubhead(-625, -400, 0, p1u);
        char p2[] = "Sukith S";
        drawhead(-625, -500, 0, p2);
        char p2u[] = "4SO19CS182";
        drawsubhead(-625, -550, 0, p2u);
        char in[] = "Click right mouse button for menu";
        drawhead(-375, -800, 0, in);
}
void display(){
    glClear(GL_COLOR_BUFFER_BIT);
    if (r == 0){
```

```
        front();
        glutSwapBuffers();
    }
    if(r==1){
        system("./animate.out");
        r=0;
    }
    if(r==11){
        system("./cube.out");
        r=0;
    }
    if(r==3){
        system("./draw.out");
        r=0;
    }
    glutPostRedisplay();
    glutSwapBuffers();
}


void menu(int op){
    if(op==1)
        r=1;
    if(op==11)
        r=11;
    if(op==3)
        r=3;
    if(op==4)
        exit(0);
}
void menuFunction(){
    char subMenu = glutCreateMenu(menu);
    glutAddMenuEntry("Cube", 11);
    glutCreateMenu(menu);
    glutAddMenuEntry("Random Lines",1);
    glutAddSubMenu("Shapes",subMenu);
    glutAddMenuEntry("Draw",3);
    glutAddMenuEntry("Exit",4);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
}


int main(int argc, char **argv){
    glutInit(&argc, argv);
    glutInitWindowPosition(300,10);
    glutInitWindowSize(650, 700);
    glutCreateWindow("CG Mini Project : Line Art");
    init();
    glutDisplayFunc(display);
    menuFunction();
```

```
    glutMainLoop();
    return 0;
}
```

animate.c

// Program to create bouncing line

```
#include<stdio.h>
#include<stdlib.h>
#include<GL/glut.h>
int width=650, height=700,X,Y,i=0,j=0,A,B;
void Init(){
    gluOrtho2D(0,650,0,700);
}
void menu(int op){
  if(op==1)
     exit(0);
}

void time();
void menuFunction(){
    glutCreateMenu(menu);
    glutAddMenuEntry("Home",1);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
}
void mouse(int button, int state, int x, int y){
    int mouseX = x;
    int mouseY = 700- y;

    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN){
        glClear(GL_COLOR_BUFFER_BIT);
        glPointSize(3);
        glColor3f(1,1,1);
        glBegin(GL_POINTS);
        glVertex2i(mouseX,mouseY);
        glEnd();
        X=mouseX;
        Y=mouseY;
        A=mouseX;
        B=mouseY;
        glutTimerFunc(0,time,0);
    }
}

void display(){
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1,1,1);
}
```

```
void time(){
   if(A>B){
       if(X==650 && Y<700){
           i=1;
       }else if(X<650 && Y==700){
           i=2;
       }else if(X==0 && Y<700){
           i=3;
       }else if(X<650 && Y==0){
           i=0;
       }
       if(i==0){
           X++;
           Y++;
       }
       else if(i==1){
           X--;
           Y++;
       }
       else if(i==2){
           X--;
           Y--;
       }
       else if(i==3){
           X++;
           Y--;
       }
       glBegin(GL_POINTS);
       glVertex2i(X,Y);
       glEnd();
   }else if(A<B){
       if(X==650 && Y<700){
           j=1;
       }else if(X<650 && Y==700){
           j=2;
       }else if(X==0 && Y<700){
           j=0;
       }else if(X<650 && Y==0){
           j=3;
       }
       if(j==0){
           X++;
           Y++;
       }else if(j==1){
           X--;
           Y--;
       }else if(j==2){
           X++;
           Y--;
       }else if(j==3){
```

```
            X--;
            Y++;
        }
        glBegin(GL_POINTS);
        glVertex2i(X,Y);
        glEnd();
    }else{
        if(X==650 && Y==700){
            X++;
            Y++;
        }
        else if(X==0 && Y==0){
            X++;
            Y++;
        }
        glBegin(GL_POINTS);
        glVertex2i(X,Y);
        glEnd();
    }

    glutTimerFunc(1000/360,time,0);
    glutSwapBuffers();
}
int main(int argc, char **argv){
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
    glutInitWindowSize(650,700);
    glutInitWindowPosition(300,10);
    glutCreateWindow("CG Mini Project : Line Art");
    Init();
    glutDisplayFunc(display);
    glutMouseFunc(mouse);
    menuFunction();
    glutMainLoop();
}
```

cube.c

// Program to create 3D cube

```
#include<stdio.h>
#include<GL/glut.h>
static int t,u,v,w;
int i=0;
GLfloat vertices[][3] ={
   {-1.0,-1.0,-1.0},
   {1.0,-1.0,-1.0},
   {1.0,1.0,-1.0},
   {-1.0,1.0,-1.0},
   {-1.0,-1.0,1.0},
   {1.0,-1.0,1.0},
   {1.0,1.0,1.0},
   {-1.0,1.0,1.0}};
GLfloat colors[][3]= {
   {0.0,0.0,0.0},
   {1.0,0.0,0.0},
   {1.0,1.0,0.0},
   {0.0,1.0,0.0},
   {0.0,0.0,1.0},
   {1.0,0.0,1.0},
   {1.0,1.0,1.0},
   {0.0,1.0,1.0}
};
void draw_px(int p,int q){
   glColor3f(0,0,0);
   glBegin(GL_POINTS);
   glVertex2i(p,q);
   glEnd();
}
void draw(int a,int b, int c, int d){
   t=a;
   u=b;
   v=c;
   w=d;
   glBegin(GL_LINE_LOOP);
   glVertex3fv(vertices[a]);
   glVertex3fv(vertices[b]);
   glVertex3fv(vertices[c]);
   glVertex3fv(vertices[d]);
   glEnd();
   glFlush();
}
void colorcube(){
   draw(0,3,2,1);
   draw(2,3,7,6);
   draw(0,4,7,3);
```

```
   draw(1,2,6,5);
   draw(4,5,6,7);
   draw(0,1,5,4);
}
static GLfloat theta[]={0.0,0.0,0.0};
static GLint axis=2;
static GLdouble viewer[]={0.0,0.0,5.0};

void display(void)
{
   glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   gluLookAt(viewer[0],viewer[1],viewer[2],0.0,0.0,0.0,0.0,
1.0,0.0);
   glRotatef(theta[0],1.0,0.0,0.0);
   glRotatef(theta[1],0.0,1.0,0.0);
   glRotatef(theta[2],0.0,0.0,1.0);
   colorcube();
   glFlush();
   glutSwapBuffers();
}
void menu(int op){
  if(op==1)
     exit(0);
}

void menuFunction(){
   glutCreateMenu(menu);
   glutAddMenuEntry("Home",1);
   glutAttachMenu(GLUT_RIGHT_BUTTON);
}
void mouse(int btn,int state, int x ,int y){
   if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
       axis=0;
   if(btn==GLUT_MIDDLE_BUTTON && state==GLUT_DOWN)
       axis=1;
   if(btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
       axis=2;
    theta[axis]+=2.0;
    if(theta[axis]>360.0)
        theta[axis]-=360.0;
    display();
}
void keys(unsigned char key, int x, int y){
   if(key=='x')
       viewer[0]-=1.0;
   if(key=='X')
       viewer[0]+=1.0;
    if(key=='y')
               viewer[0]-=1.0;
```

```
        if(key=='Y')
                viewer[0]+=1.0;
     if(key=='z')
                viewer[0]-=1.0;
        if(key=='Z')
                viewer[0]+=1.0;
    display();
}
void myReshape(int w ,int h){
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if(w<=h)
        glFrustum(-2.0,2.0,-2.0*(GLfloat)h/(GLfloat)w,
2.0*(GLfloat)h/(GLfloat)w,2.0,20.0);
    else
        glFrustum(-2.0,2.0,-2.0*(GLfloat)w/(GLfloat)h,
2.0*(GLfloat)w/(GLfloat)h,2.0,20.0);
    glMatrixMode(GL_MODELVIEW);
}
void main(int argc,char**argv){
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowPosition(300,10);
    glutInitWindowSize(650, 700);
    glutCreateWindow("CG Mini Project : Line Art");
    glutReshapeFunc(myReshape);
    glutDisplayFunc(display);
    glutMouseFunc(mouse);
    menuFunction();
    glutKeyboardFunc(keys);
    glEnable(GL_DEPTH_TEST);
    glutMainLoop();
}
```

draw.c

// Program to implement free line drawing

```
#include <GL/glut.h>
#include <stdio.h>
#include <math.h>
#include <stdbool.h>

const int MAX_ITEMS = 1000;
struct ItemData {
  double x1, y1;
  double x2, y2;
};
```

```
struct ItemData items[MAX_ITEMS];
int itemCt = 0;
int width;
int height;
bool dragging = false;
int dragModifiers;
void handleStartDraw(double x, double y, int modifiers) {
  if (itemCt == MAX_ITEMS) {
     printf("Line draw limit has been reached..!\n");
     return;
  }
  dragging = true;
  dragModifiers = modifiers;
  items[itemCt].x1 = x;
  items[itemCt].y1 = y;
  items[itemCt].x2 = x;
  items[itemCt].y2 = y;
  itemCt++;
}

void handleContinueDraw(double x, double y) {
  if (!dragging)
     return;
  int current = itemCt - 1;
  bool shifted = ((dragModifiers & GLUT_ACTIVE_SHIFT) !=
0);
  if (shifted) {
     if (abs(x - items[current].x1) > abs(y -
items[current].y1))
        y = items[current].y1;
     else
        x = items[current].x1;
  }
  items[current].x2 = x;
  items[current].y2 = y;
  glutPostRedisplay();
}

void handleFinishDraw(double x, double y) {
  if (!dragging)
     return;
  dragging = false;
  int current = itemCt - 1;
  if (items[current].x1 == items[current].x2 &&
     items[current].y1 == items[current].y2) {
        itemCt--;
  }
  glutPostRedisplay();
}
```

```
void menu(int op){
  if(op==1){
     itemCt--;
     glutPostRedisplay();
  }
  if(op==2){
     itemCt=0;
     glutPostRedisplay();
  }
  if(op==3)
     exit(0);
}

void menuFunction(){
   glutCreateMenu(menu);
   glutAddMenuEntry("Undo",1);
   glutAddMenuEntry("Clear",2);
   glutAddMenuEntry("Home",3);
   glutAttachMenu(GLUT_RIGHT_BUTTON);
}

void init(){
}

void drawItems(){
  glColor3f(1,1,1);
  for (int i = 0; i < itemCt; i++) {
     glBegin(GL_LINES);
        glVertex2f(items[i].x1, items[i].y1);
        glVertex2f(items[i].x2, items[i].y2);
     glEnd();
  }
}

void initTransformation(double x1, double x2, double y1,
double y2){
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluOrtho2D(x1,x2,y1,y2);
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}


void display(){
  glClearColor(0,0,0,1);
  glClear(GL_COLOR_BUFFER_BIT);
  glViewport(0,0,width-0,height);
  initTransformation(0,1,0,1);
```

```
  drawItems();
  glViewport(0,0,0,height);
  initTransformation(0,0,0,height);
  glutSwapBuffers();
}

void reshape(int new_width, int new_height){
  height = new_height;
  width = new_width;
  glViewport(0,0,width,height);
}

void mouse(int button, int state, int x, int y) {
  if (button == GLUT_LEFT_BUTTON) {
     double wx, wy;
     wx = (double)(x-0)/(width-0);
     wy = (double)(height-y)/height;
     if (state == GLUT_DOWN)
        handleStartDraw(wx,wy,glutGetModifiers());
     else
        handleFinishDraw(wx,wy);
  }
}

void motion(int x, int y){
  if (dragging) {
     double wx, wy;
     wx = (double)(x-0)/(width-0);
     wy = (double)(height-y)/height;
     handleContinueDraw(wx,wy);
  }
}

int main(int argc, char **argv){
  glutInit(&argc,argv);
  glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
  glutInitWindowSize(650,700);
  glutInitWindowPosition(300,10);
  glutCreateWindow("CG Mini Project : Line Art");
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutMouseFunc(mouse);
  glutMotionFunc(motion);
  menuFunction();
  init();
  glutMainLoop();
}
```

CHAPTER 4 – SCREENSHOTS

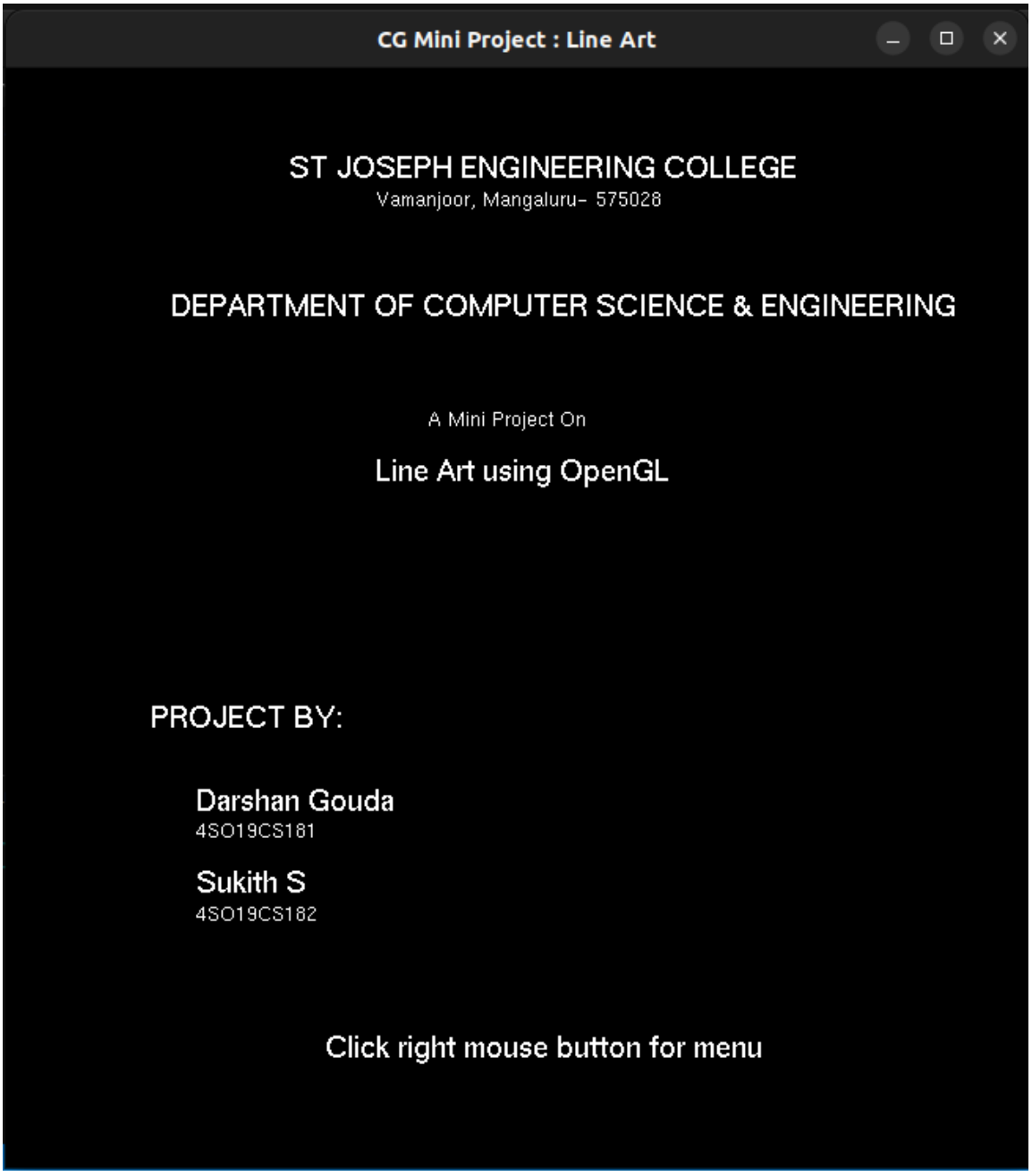

Figure 1 : Home Screen

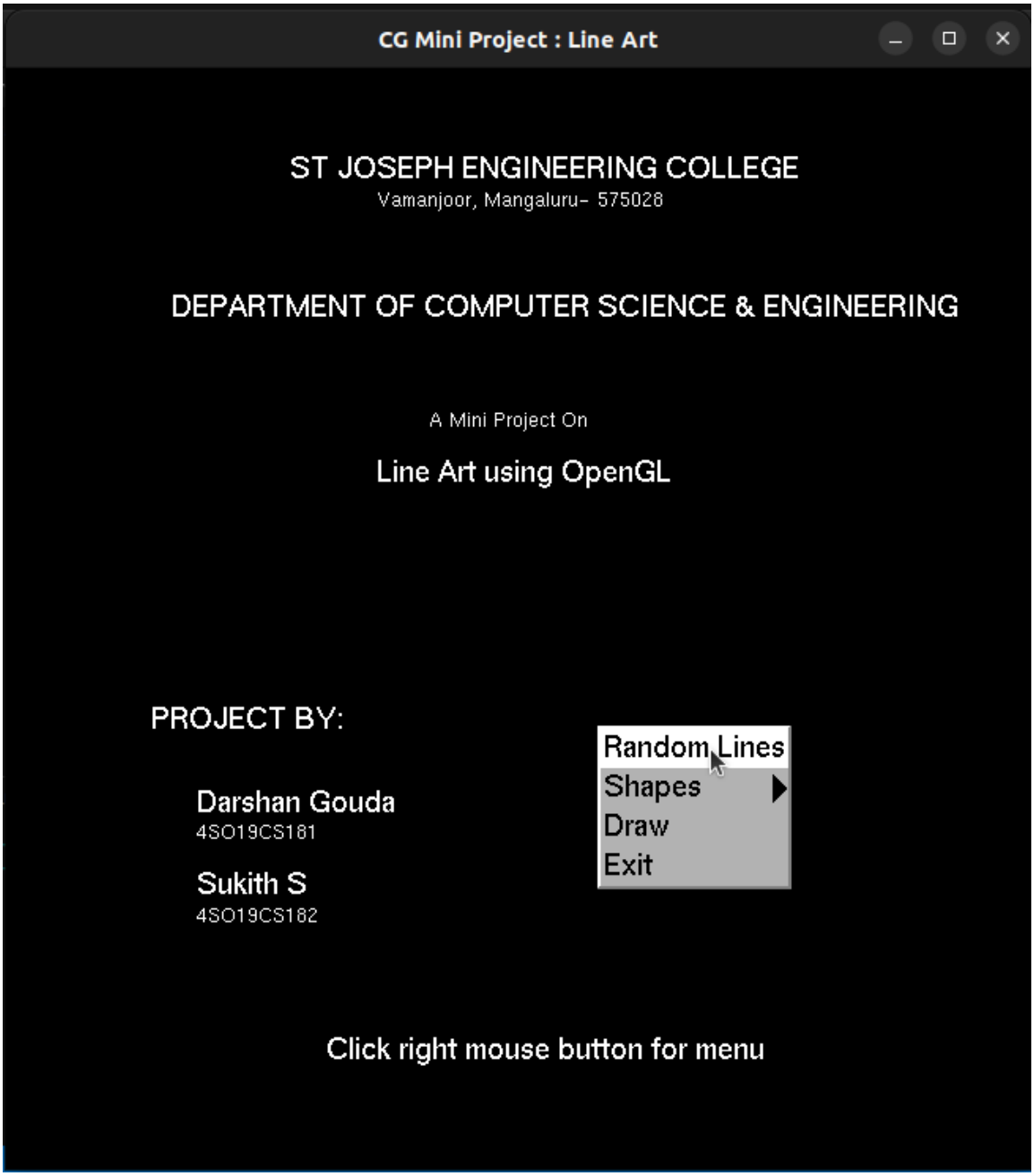

Figure 2 : Menu attached to Mouse right button

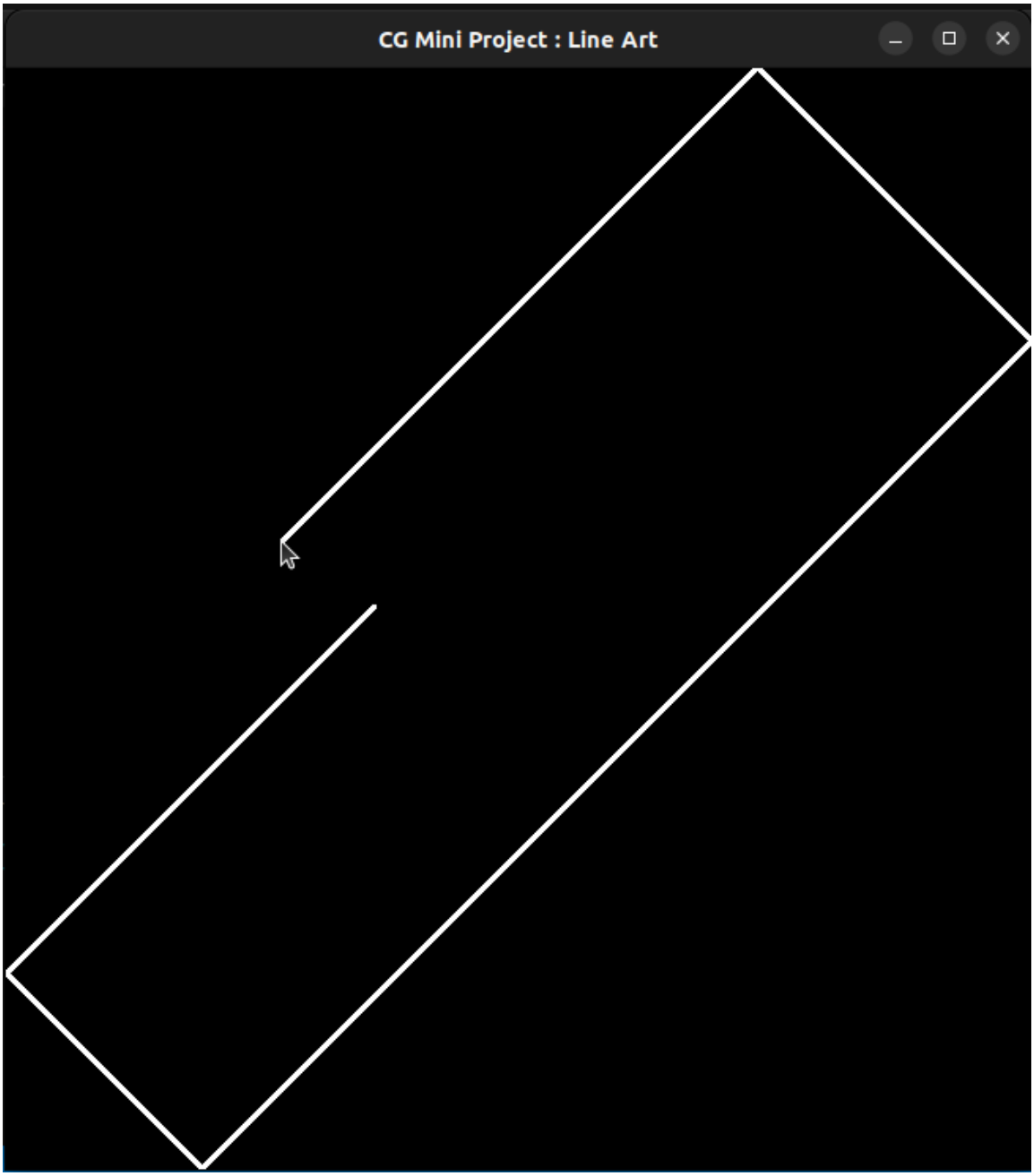

Figure 2.1 : Bouncing line animation screen

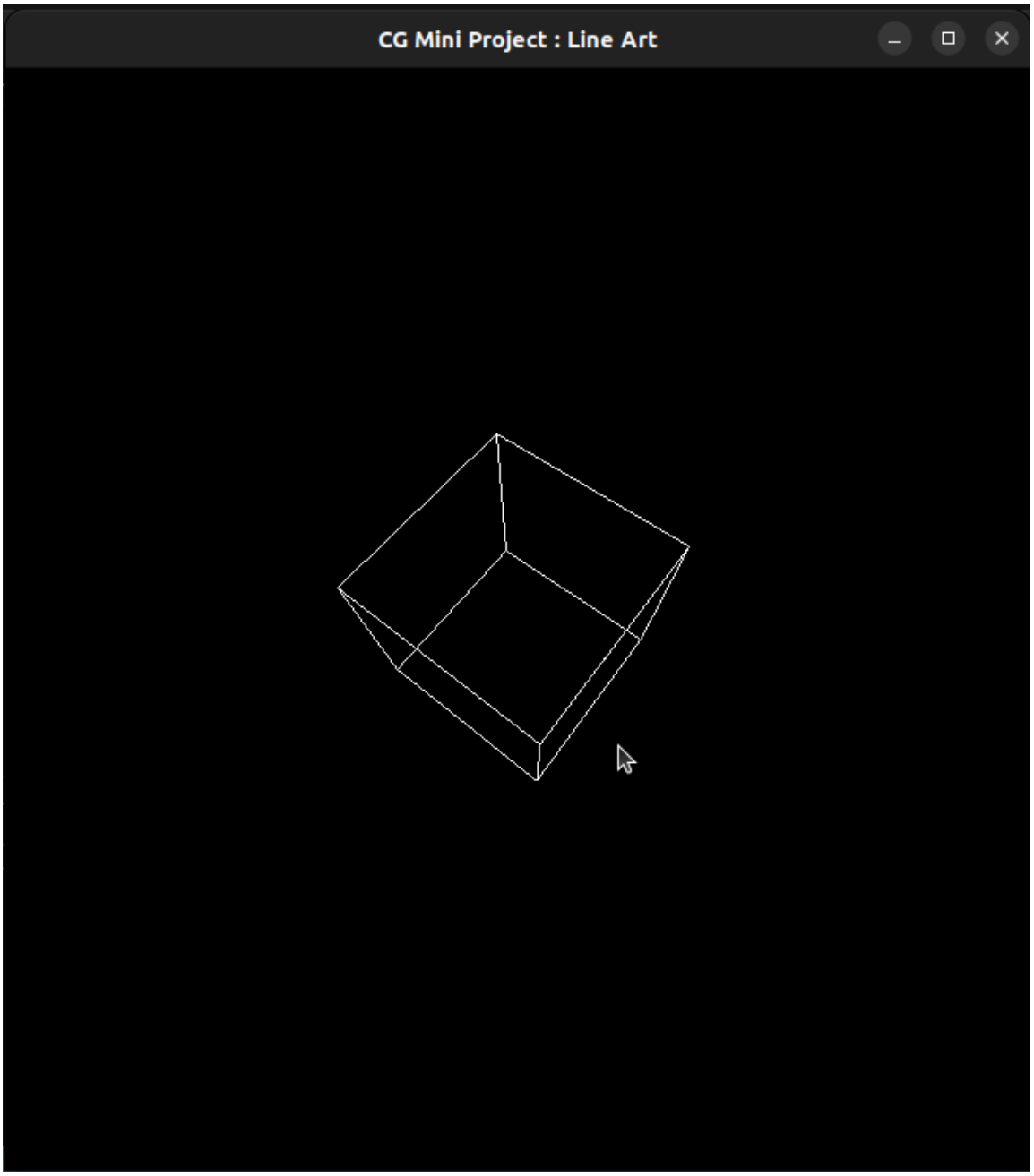

Figure 2.2 : 3D Cube viewing

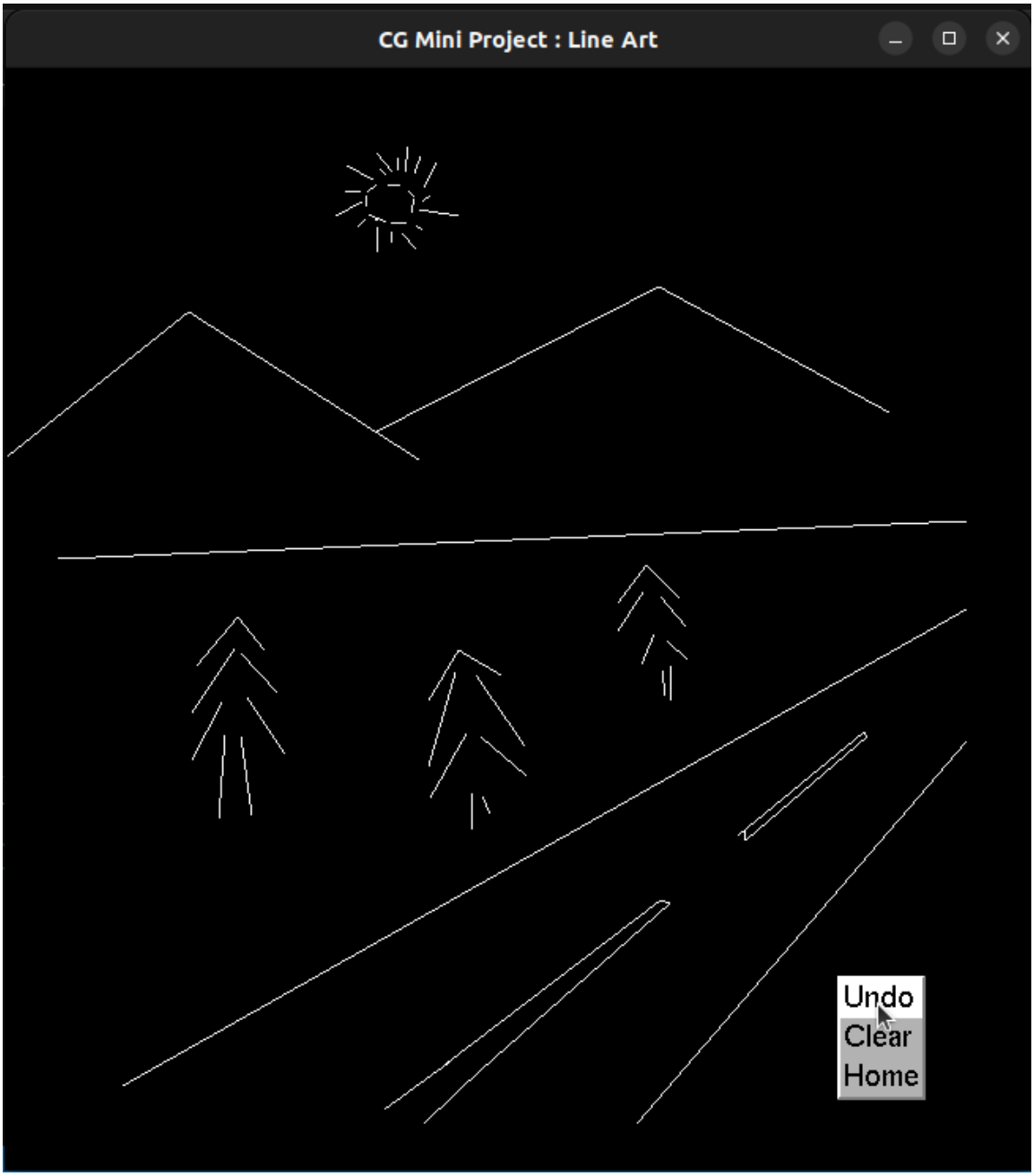

Figure 2.3 : Free line drawing with Menu

CHAPTER 5 – CONCLUSION

This Computer Graphic Project allows users to have an interactive experience in using line art along with model viewing implemented using OpenGL functionalities in C programming language. Computer graphics will continue to get more sophisticated. Their 3-D photorealistic capabilities and ability to predict changes over time have revolutionized product development and marketing, as well as scientific research and education.

REFERENCES

1. Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3rd / 4th Edition, *Pearson Education*,2011
2. Edward Angel: Interactive Computer Graphics- A Top-Down approach with OpenGL, 5th edition. *Pearson Education*, 2008
3. LearnOpenGL (*https://learnopengl.com*)
4. GeeksForGeekst (*https://www.geeksforgeeks.org/getting-started-with-opengl/*)
5. StackOverflow (*www.stackoverflow.com*)
6. YouTube (*https://www.youtube.com/*)
7. Google (*https://www.google.com/*)