**SRI SIDDHARTHA INSTITUTE OF TECHNOLOGY**

**(A Constituent college of Sri Siddhartha Academy of Higher Education)**

**Department of Computer Science & Engineering**

**Report on “Mini Project on Pulvama Attack”**

***Submitted by :***

**Surabhi S (17CS127)**

**Vishala G (17CS116)**

**UNDER THE GUIDANCE OF:**

**Madumala G Channabasavaraju T P**

Assistant Professor Assistant Professor

Dept. of C.S.E . Dept. of C.S.E.

S.S.I.T,Tumkur S.S.I.T,Tumkur

***Certificate***

This is to certify that the course ***CG mini project on “Pulvama Attack” (CS6L01)*** has been completed successfully in sixth semester of Bachelor of Engineering in Computer Science and Engineering of Sri Siddhartha Academy of Higher Education during the academic year 2019-20.

**UNDER THE GUIDANCE OF:**

**Madumala G Channabasavaraju T P**

Assistant Professor Assistant Professor

Dept. of C.S.E . Dept. of C.S.E.

S.S.I.T,Tumkur S.S.I.T,Tumkur

**Dr.M.Siddappa Ph.D.**

Professor & Head,

Dept. of C.S.E.

S.S.I.T., Tumkur

Examiner 1 Name: Examiner 2 Name:

Signature: Signature:

**MINI PROJECT ON**

**‘’ PULVAMA ATTACK”**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SL.NO** | **particulars** | **PAGE.NO** |
| **1** | Abstract | 5 |
| **2** | System Specifications | 6 |
| **3** | Introduction to openGL | 7 |
| **5** | Implementation | 9 |
| **6** | Interaction | 11 |
| **7** | Source Code | 12 |
| **8** | Output | 42 |
| **9** | Conclusion | 44 |
| **10** | Bibliography | 45 |
|  |  |  |

**Abstract**

**System specifications**

* **SOFTWARE REQUIREMENTS :**
* MICROSOFT VISUAL C++
* OPENGL
* **HARDWARE REQUIREMENT :**
* GRAPHICS SYSTEM,
* Pentium P4 with 256 of Ram(Min)

**Introduction to openGL**

As a software interface for graphics hardware, OpenGL's main purpose is to render two- and three-dimensional objects into a frame buffer.

These objects are described as sequences of vertices or pixels.

OpenGL performs several processing steps on this data to convert it to pixels to form the final desired image in the frame buffer.

## OpenGL Fundamentals

This section explains some of the concepts inherent in OpenGL.

### Primitives and Commands

OpenGL draws primitives—points, line segments, or polygons—subject to several selectable modes.

You can control modes independently of each other; that is, setting one mode doesn't affect whether other modes are set .Primitives are specified, modes are set, and other OpenGL operations are described by issuing commands in the form of function calls.

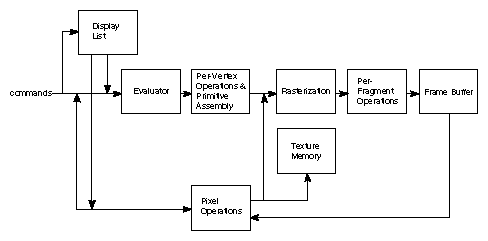
Primitives are defined by a group of one or more vertices. A vertex defines a point, an endpoint of a line, or a corner of a polygon where two edges meet. Data is associated with a vertex, and each vertex and its associated data are processed independently, in order, and in the same way. The type of clipping depends on which primitive the group of vertices represents.

Commands are always processed in the order in which they are received, although there may be an indeterminate delay before a command takes effect. This means that each primitive is drawn completely before any subsequent command takes effect. It also means that state-querying commands return data that's consistent with complete execution of all previously issued OpenGL commands.

## Basic OpenGL Operation

The figure shown below gives an abstract, high-level block diagram of how OpenGL processes data. In the diagram, commands enter from the left and proceed through what can be thought of as a processing pipeline. Some commands specify geometric objects to be drawn, and others control how the objects are handled during the various processing stages.

**Figure . OpenGL Block Diagram**



As shown by the first block in the diagram, rather than having all commands proceed immediately through the pipeline, you can choose to accumulate some of them in a display list for processing at a later time.

Rasterization produces a series of frame buffer addresses and associated values using a two-dimensional description of a point, line segment, or polygon.

Each fragment so produced is fed into the last stage,

per-fragment operations, which performs the final operations on the data before it's stored as pixels in the frame buffer. These operations include conditional updates to the frame buffer based on incoming and previously stored z-value s (for z-buffering) and blending of incoming pixel colors with stored colors, as well as masking and other logical operations on pixel values.

All elements of OpenGL state, including the contents of the texture memory and even of the frame buffer, can be obtained by an OpenGL application.

**Implementation**

**Interaction with program**

**Source Code**

#include <stdio.h>

#include <GL/glut.h>

#include<math.h>

int scene = 0,x=0,y=-195,opt=0, clouds =0; ;

int wx1[] ={-670,-150,418};

int wy1=-232;

float cx[]={-540,-150,520,-30};

float cy[]={210,130,250,250};

float i =-245;

float j= 400;

int hasCrashed=0, flag2 = 0;

float expFactor;

float explodeTx = 1.0;

float explodeTy = 1.0;

void options(int id){ //menu option

if(id == 1 && scene == 4) opt = 1;

else if(id == 2) exit(0);

}

typedef struct stColor {

double red, green, blue;

double alpha;

} RgbColor;

RgbColor createColor(int red, int green, int blue, double alpha) {

RgbColor color;

color.red = (double) red / 256;

color.green = (double) green / 256;

color.blue = (double) blue / 256;

color.alpha = (double) alpha;

return color;

}

RgbColor color;

void DrawCircle(float cx, float cy, float cz, float r, int num\_segments)

{

int ii;

float theta, x, y, i , j;

if(clouds == 1) j=num\_segments/2; //half circle or full circle

else j=num\_segments;

glBegin(GL\_POLYGON);

for(ii = 0; ii < j; ii++)

{

i = (float)ii/(float)num\_segments;

theta = 2.0f\*3.1415926f\*i;

x = r \* cosf(theta);

y = r \* sinf(theta);

glVertex3f(x + cx, y + cy, cz);

}

glEnd();

}

void drawText(char text[], int x, int y, int z)

{

// The color

glColor3f(1, 1, 1);

// Position of the text to be printer

glRasterPos3f(x, y, z);

for(int i = 0; text[i] != '\0'; i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, text[i]);

}

void printLines(char \*\*s, int offsetY, int n, int x, int y) {

int i = 0;

for (; i < n; i++) {

drawText(s[i],x, y + offsetY,0 );

offsetY -= 35;

}

}

void drawMainScene()

{

glBegin(GL\_QUADS);

color = createColor(166, 73, 154, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f(-670.0,350.0);

glVertex2f(670.0,350.0);

color = createColor(19, 13, 86, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f(670.0,-350.0);

glVertex2f(-670.0,-350.0);

glEnd();

glClearColor(1.0,1.0, 1.0, 1.0);

char h1[] = "Computer Graphics Project on";

char h2[] = "PULVAMA ATTACK";

char h3[] = " ";

char name1[] = "Surabhi S";

char usn1[] = "17CS127";

char name2[] = "Vishala G";

char usn2[] = "17CS116";

char message[] = "Press 1 to start";

drawText(h1, -110, 200, 0);

drawText(h2, -90, 170, 0);

drawText(h3, -5, 150, 0);

drawText(name1, -230, 90, 0);

drawText(usn1, -237, 70, 0);

drawText(name2, 100, 90, 0);

drawText(usn2, 150, 70, 0);

drawText(message, -70, -200, 0);

}

void reset()

{

i =-245;

j= 400;

hasCrashed=0, flag2 = 0;

expFactor=0;

explodeTx = 1.0;

explodeTy = 1.0;

}

void explode() {

color = createColor(244, 163, 44, 0);

glColor3f(color.red, color.green, color.blue);

DrawCircle(-15, -150,0, 1.0f \* expFactor,200);

color = createColor(237, 105, 74, 0);

glColor3f(color.red, color.green, color.blue);

DrawCircle(-25, -130,0, 1.0f \* expFactor,200);

color = createColor(216, 85, 58, 0);

glColor3f(color.red, color.green, color.blue);

DrawCircle(-5, -135,0, 1.0f \* expFactor,200);

}

void draw\_clouds(int r){

int i = 0;

clouds=1;

for(;i<4;i++){

color = createColor(255, 255, 255, 0);

glColor3f(color.red, color.green, color.blue);

//clouds half circle

DrawCircle(cx[i],cy[i],0,r,200);

DrawCircle(cx[i]+40,cy[i]+20,0,r,200);

DrawCircle(cx[i]+70,cy[i],0,r,200);

DrawCircle(cx[i]+100,cy[i],0,r,200);

glBegin(GL\_POLYGON); //fill polygon

glColor3f(color.red, color.green, color.blue);

glVertex2f(cx[i]-5,cy[i]);

glVertex2f(cx[i]+40,cy[i]+45);

glVertex2f(cx[i]+70,cy[i]);

glVertex2f(cx[i]+100,cy[i]);

glEnd();

if(scene == 1|| scene == 3 && !hasCrashed){

if(cx[i]+300>-670) cx[i]=cx[i]-0.03;

else cx[i]=670;

}

if(scene == 2 ){

if(cx[i]<670) cx[i]=cx[i]+0.06;

else cx[i]=-970;

}

}

clouds=0;

}

void bus()

{

if(hasCrashed == 0){

if(scene == 3)

i+=0.6;

else i+=0.1;

}

glBegin(GL\_POLYGON); //body

color = createColor(74, 147, 138, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-600+i,-245,0);

glVertex3f(-600+i,-75,0);

glVertex3f(-10+i,-75,0);

glVertex3f(15+i,-170,0);

glVertex3f(15+i,-245,0);

glEnd();

glBegin(GL\_POLYGON); //

color = createColor(76, 157, 150, 0); //body 2

glColor3f(color.red, color.green, color.blue);

glVertex3f(-590+i,-235,0);

glVertex3f(-590+i,-85,0);

glVertex3f(-20+i,-85,0);

glVertex3f(5+i,-180,0);

glVertex3f(5+i,-235,0);

glEnd();

glBegin(GL\_POLYGON);

color = createColor(168, 225, 232, 0); //window1

glColor3f(color.red, color.green, color.blue);

glVertex3f(-550+i,-150,0);

glVertex3f(-550+i,-100,0);

glVertex3f(-490+i,-100,0);

glVertex3f(-490+i,-150,0);

glEnd();

glBegin(GL\_POLYGON);

color = createColor(168, 225, 232, 0); //window 2

glColor3f(color.red, color.green, color.blue);

glVertex3f(-470+i,-150,0);

glVertex3f(-470+i,-100,0);

glVertex3f(-410+i,-100,0);

glVertex3f(-410+i,-150,0);

glEnd();

glBegin(GL\_POLYGON);

color = createColor(168, 225, 232, 0); //window 3

glColor3f(color.red, color.green, color.blue);

glVertex3f(-390+i,-150,0);

glVertex3f(-390+i,-100,0);

glVertex3f(-330+i,-100,0);

glVertex3f(-330+i,-150,0);

glEnd();

glBegin(GL\_POLYGON);

color = createColor(168, 225, 232, 0); //window5

glColor3f(color.red, color.green, color.blue);

glVertex3f(-310+i,-150,0);

glVertex3f(-310+i,-100,0);

glVertex3f(-250+i,-100,0);

glVertex3f(-250+i,-150,0);

glEnd();

glBegin(GL\_POLYGON);

color = createColor(132, 193, 186, 0); //door

glColor3f(color.red, color.green, color.blue);

glVertex3f(-230+i,-235,0);

glVertex3f(-230+i,-100,0);

glVertex3f(-170+i,-100,0);

glVertex3f(-170+i,-235,0);

glEnd();

glLineWidth(15);

glBegin(GL\_LINES);//door window 1

color = createColor(168, 225, 232, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-215+i,-200,0);

glVertex3f(-215+i,-110,0);

glEnd();

glBegin(GL\_LINES);//door window 2

color = createColor(168, 225, 232, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-185+i,-200,0);

glVertex3f(-185+i,-110,0);

glEnd();

glLineWidth(7);

glBegin(GL\_LINES); //top1

color = createColor(74, 147, 138, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-520+i,-72,0);

glVertex3f(-440+i,-72,0);

glEnd();

glBegin(GL\_LINES);//top2

color = createColor(74, 147, 138, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-200+i,-72,0);

glVertex3f(-120+i,-72,0);

glEnd();

glBegin(GL\_POLYGON); //window 5

color = createColor(168, 225, 232, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-150+i,-150,0);

glVertex3f(-150+i,-100,0);

glVertex3f(-90+i,-100,0);

glVertex3f(-90+i,-150,0);

glEnd();

glBegin(GL\_POLYGON); //window 5

color = createColor(168, 225, 232, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-70+i,-150,0);

glVertex3f(-70+i,-100,0);

glVertex3f(-20+i,-100,0);

glVertex3f(-10+i,-150,0);

glEnd();

//headlight, tailLight, mirror, bumper

glBegin(GL\_POLYGON);

color = createColor(137, 137, 137, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-610+i,-245,0);

glVertex3f(-610+i,-220,0);

glVertex3f(-530+i,-220,0);

glVertex3f(-530+i,-245,0);

glEnd();

glBegin(GL\_POLYGON);

color = createColor(137, 137, 137, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-45+i,-245,0);

glVertex3f(-45+i,-220,0);

glVertex3f(25+i,-220,0);

glVertex3f(25+i,-245,0);

glEnd();

glBegin(GL\_POLYGON);

color = createColor(250, 197, 29, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-15+i,-220,0);

glVertex3f(-15+i,-195,0);

glVertex3f(15+i,-195,0);

glVertex3f(15+i,-220,0);

glEnd();

glBegin(GL\_POLYGON);

color = createColor(225, 74, 57, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-600+i,-220,0);

glVertex3f(-600+i,-195,0);

glVertex3f(-575+i,-195,0);

glVertex3f(-575+i,-220,0);

glEnd();

glLineWidth(3);

glBegin(GL\_LINES);

color = createColor(0, 0, 0, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-5+i,-100,0);

glVertex3f(30+i,-102,0);

glEnd();

glLineWidth(7);

glBegin(GL\_LINES);

color = createColor(0, 0, 0, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(30+i,-102,0);

glVertex3f(30+i,-130,0);

glEnd();

glLineWidth(1);

//wheels

color = createColor(0, 0, 0, 0);

glColor3f(color.red, color.green, color.blue);

DrawCircle(-500+i,-235,0,30,200);

color = createColor(255, 255, 255, 0);

glColor3f(color.red, color.green, color.blue);

DrawCircle(-500+i,-235,0,15,200);

color = createColor(0, 0, 0, 0);

glColor3f(color.red, color.green, color.blue);

DrawCircle(-75+i,-235,0,30,200);

color = createColor(255, 255, 255, 0);

glColor3f(color.red, color.green, color.blue);

DrawCircle(-75+i,-235,0,15,200);

}

void car()

{

if(hasCrashed == 0){

if(scene == 3)

j-=4;

else j-=0.4;

}

glBegin(GL\_POLYGON);//body

color = createColor(233, 76, 61, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(550+j,-245,0);

glVertex3f(275+j,-245,0);

glVertex3f(290+j,-210,0);

glVertex3f(295+j,-190,0);

glVertex3f(370+j,-183,0);

glVertex3f(400+j,-140,0);

glVertex3f(545+j,-135,0);

glEnd();

glLineWidth(12.5);

glBegin(GL\_LINES); //bumper

glColor3f(0, 0, 0);

glVertex3f(555+j,-245,0);

glVertex3f(270+j,-245,0);

glEnd();

glLineWidth(1);

glBegin(GL\_POLYGON);

color = createColor(197, 224, 240, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(378+j,-183,0);

glVertex3f(402+j,-145,0);

glVertex3f(450+j,-142,0);

glVertex3f(450+j,-183,0);

glEnd();

glBegin(GL\_POLYGON);

color = createColor(197, 224, 240, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(468+j,-183,0);

glVertex3f(468+j,-145,0);

glVertex3f(535+j,-142,0);

glVertex3f(537+j,-183,0);

glEnd();

glBegin(GL\_POLYGON);

color = createColor(0, 0, 0, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(550.1+j,-220,0);

glVertex3f(570+j,-220,0);

glVertex3f(570+j,-175,0);

glVertex3f(549.5+j,-175,0);

glEnd();

glBegin(GL\_POLYGON);

color = createColor(184, 49, 46, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(550+j,-235,0);

glVertex3f(530+j,-235,0);

glVertex3f(530+j,-225,0);

glVertex3f(550+j,-225,0);

glEnd();

glBegin(GL\_POLYGON);

color = createColor(250, 197, 29, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(280+j,-235,0);

glVertex3f(300+j,-235,0);

glVertex3f(300+j,-225,0);

glVertex3f(280+j,-225,0);

glEnd();

glLineWidth(1);

glBegin(GL\_LINES);

color = createColor(169, 64, 47, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(459+j,-142,0);

glVertex3f(460+j,-238,0);

glEnd();

glBegin(GL\_LINES);

color = createColor(169, 64, 47, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(368+j,-188,0);

glVertex3f(368+j,-238,0);

glEnd();

glLineWidth(5);

glBegin(GL\_LINES);

glColor3f(0, 0, 0);

glVertex3f(450+j,-190,0);

glVertex3f(430+j,-190,0);

glEnd();

glBegin(GL\_LINES);

color = createColor(247, 146, 36, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(365+j,-195,0);

glVertex3f(350+j,-195,0);

glEnd();

glLineWidth(1);

color = createColor(0,0,0, 0); //wheel

glColor3f(color.red, color.green, color.blue);

DrawCircle(330+j,-245,0,30,480);

color = createColor(147,135,139, 0); //wheel

glColor3f(color.red, color.green, color.blue);

DrawCircle(330+j,-245,0,20,480);

color = createColor(0,0,0, 0); //wheel

glColor3f(color.red, color.green, color.blue);

DrawCircle(500+j,-245,0,30,480);

color = createColor(147,135,139, 0); //wheel

glColor3f(color.red, color.green, color.blue);

DrawCircle(500+j,-245,0,20,480);

glLineWidth(1);

}

void drawWS(){

int i = 0;

for(;i<3;i++){

color = createColor(250, 250, 255, 0);

glBegin(GL\_POLYGON);

glColor3f(color.red, color.green, color.blue);

glVertex3f(wx1[i], wy1, 0);

glVertex3f(wx1[i]+252, wy1, 0);

glVertex3f(wx1[i]+240, wy1-10, 0);

glVertex3f(wx1[i]-12, wy1-10, 0);

glEnd();

if(scene == 1|| scene == 3 && !hasCrashed){

if(wx1[i]+252>-670) wx1[i]=wx1[i]-4;

else wx1[i]=670;}

if(scene == 2){

if(wx1[i]<670) wx1[i]=wx1[i]+8;

else wx1[i]=-992;}

}

}

void drawStreetLight(){

color = createColor(201, 200, 197, 0);

glBegin(GL\_POLYGON);

glColor3f(color.red, color.green, color.blue);

glVertex3f(x,y, -2);

color = createColor(0,0,0,0);;

glColor3f(color.red, color.green, color.blue);

glVertex3f(x-8, y, -2);

color = createColor(0,0,0, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(x-8, y+255, -2);

color = createColor(201, 200, 197, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(x, y+255, -2);

glEnd();

color = createColor(201, 200, 197, 0);

glBegin(GL\_POLYGON);

glColor3f(color.red, color.green, color.blue);

glVertex3f(x,y+257, -2);

color = createColor(0,0,0,0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(x-8, y+255, -2);

color = createColor(0,0,0, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(x-70, y+275, -2);

color = createColor(201, 200, 197, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(x-62, y+277, -2);

glEnd();

color = createColor(254, 242, 178, 0);

glBegin(GL\_POLYGON);

glColor3f(color.red, color.green, color.blue);

glVertex3f(x-70,y+275, 0);

glVertex3f(x-20,y+260, 0);

color = createColor(255, 255, 255, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(x-40,y+245, 0);

glVertex3f(x-60,y+255, 0);

glEnd();

if(scene == 1|| scene == 3 && !hasCrashed){

if(x>-670) x=x-4;

else x=670;}

if(scene == 2){

if(x<670) x=x+8;

else x=-670;}

}

void drawCandles(){

glBegin(GL\_QUADS); //candle1

color = createColor(227, 242, 249, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f(400.0,-100.0);

glVertex2f(450.0,-100.0);

glVertex2f(450.0,-350.0);

glVertex2f(400.0,-350.0);

glEnd();

glBegin(GL\_QUADS); //candle1 shadow

color = createColor(173, 232, 255, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f(405.0,-102.0);

glVertex2f(415.0,-102.0);

glVertex2f(415.0,-350.0);

glVertex2f(405.0,-350.0);

glEnd();

glBegin(GL\_QUADS);//candle2

color = createColor(227, 242, 249, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f(460.0,-130.0);

glVertex2f(510.0,-130.0);

glVertex2f(510.0,-350.0);

glVertex2f(460.0,-350.0);

glEnd();

glBegin(GL\_QUADS);//candle2 shadow

color = createColor(173, 232, 255, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f(465.0,-132.0);

glVertex2f(475.0,-132.0);

glVertex2f(475.0,-350.0);

glVertex2f(465.0,-350.0);

glEnd();

glBegin(GL\_QUADS);//candle3

color = createColor(227, 242, 249, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f(340.0,-150.0);

glVertex2f(390.0,-150.0);

glVertex2f(390.0,-350.0);

glVertex2f(340.0,-350.0);

glEnd();

glBegin(GL\_QUADS);//candle3 shadow

color = createColor(173, 232, 255, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f(345.0,-152.0);

glVertex2f(355.0,-152.0);

glVertex2f(355.0,-350.0);

glVertex2f(345.0,-350.0);

glEnd();

//drawWick

glLineWidth(3.0);

glColor3f(0.0,0.0,0.0); //wick1

glBegin(GL\_LINES);

glVertex2f(425.0,-85.0);

glVertex2f(425.0,-100.0);

glEnd();

glColor3f(0.0,0.0,0.0); //wick2

glBegin(GL\_LINES);

glVertex2f(485.0,-115.0);

glVertex2f(485.0,-130.0);

glEnd();

glColor3f(0.0,0.0,0.0); //wick3

glBegin(GL\_LINES);

glVertex2f(365.0,-135.0);

glVertex2f(365.0,-150.0);

glEnd();

}

void drawFire(int xfire, int yfire){

glBegin(GL\_POLYGON);

color = createColor(230, 78, 58, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f(xfire,yfire+20);

color = createColor(241, 119, 57, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f((xfire+15.0),(yfire-12.0));

color = createColor(246, 166, 63, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f((xfire+18.5),(yfire-26.0));

color = createColor(241, 196, 101, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f((xfire+18.0),(yfire-45.0));

color = createColor(6, 206, 255, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f((xfire),(yfire-77));

color = createColor(241, 196, 101, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f((xfire-18.0),(yfire-45.0));

color = createColor(246, 166, 63, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f((xfire-18.5),(yfire-26.0));

color = createColor(241, 119, 57, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f((xfire-15.0),(yfire-12.0));

glEnd();

}

void drawBG(){

glPushMatrix();

//road

color = createColor(81, 81, 81, 0);

glBegin(GL\_POLYGON);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-670,-210, -2);

glVertex3f(670, -210, -2);

glVertex3f(670, -300, -2);

glVertex3f(-670, -300, -2);

glEnd();

//grass

color = createColor(86, 121, 70, 0);

glBegin(GL\_POLYGON);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-670,-300, -2);

glVertex3f(670, -300, -2);

glVertex3f(670, -350, -2);

glVertex3f(-670, -350, -2);

glEnd();

//gradient BG

glBegin(GL\_POLYGON);

color = createColor(255, 255, 255, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-670,-210, -2);

glVertex3f(670, -210, -2);

color = createColor(109, 209, 241, 0);

glColor3f(color.red, color.green, color.blue);

glVertex3f(670, 350, -2);

glVertex3f(-670, 350, -2);

glEnd();

//drawing clouds

draw\_clouds(25);

draw\_clouds(25);

draw\_clouds(25);

draw\_clouds(25);

//drawing white strips

drawWS();

//divider

color = createColor(152, 151, 147, 0);

glBegin(GL\_POLYGON);

glColor3f(color.red, color.green, color.blue);

glVertex3f(-670,-195, -2);

glVertex3f(670, -195, -2);

glVertex3f(670, -210, -2);

glVertex3f(-670, -210, -2);

glEnd();

//streetLight

drawStreetLight();

glPopMatrix();

}

void StoryScene(){

glPushMatrix();

glBegin(GL\_QUADS);

color = createColor(166, 73, 154, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f(-670.0,350.0);

glVertex2f(670.0,350.0);

color = createColor(19, 13, 86, 0);

glColor3f(color.red, color.green, color.blue);

glVertex2f(670.0,-350.0);

glVertex2f(-670.0,-350.0);

glEnd();

char \*story[] = {"On 14 February 2019, a convoy of vehicles carrying security personnel on the Jammu Srinagar National Highway was attacked",

"by a vehicle-borne suicide bomber at Lethpora (near Awantipora) in the Pulwama district, Jammu and Kashmir, India.",

"The attack resulted in the deaths of 40 Central Reserve Police Force (CRPF) personnel and the attacker.",

"The responsibility for the attack was claimed by the Pakistan-based Islamist militant group Jaish-e-Mohammed.",

"The attacker was Adil Ahmad Dar, a local from Pulwama district, and a member of Jaish-e-Mohammed.",

"India has blamed Pakistan for the attack. Pakistan condemned the attack and denied any connection to it."

};

printLines(story, 0, 6,-600, 200);

drawCandles();

if(opt == 1) {drawFire(425.0,-23); drawFire(485.0,-53); drawFire(365.0,-73); }

glPopMatrix();

}

void keyboardDown(unsigned char key, int x, int y) {

switch(key) {

case 49: //enter 1

scene += 1;

if (scene >4) exit(0);

break;

case 50: //enter 2

scene -= 1;

if (scene<0) scene=0;

break;

case 'Q':

case 'q':

case 27: // ESC

exit(0);

}

}

void reshape(int width, int height) {

GLfloat fieldOfView = 90.0f;

glViewport (0, 0, (GLsizei) width, (GLsizei) height);

glMatrixMode (GL\_PROJECTION);

glLoadIdentity();

glOrtho(-662, 662, -350, 350, -450, 450);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void scene1(){

glPushMatrix();

drawBG();

bus();

glPopMatrix();

}

void scene2(){

glPushMatrix();

drawBG();

car();

glPopMatrix();

}

void scene3(){

int k;

glPushMatrix();

if(flag2 == 0){

i =-690;

j= 4200;

flag2=1;

}

drawBG();

bus();

car();

if(int(j) == int(i-276)) hasCrashed = 1;

printf("%f %f\n",j,i-276);

if (hasCrashed){

glPushMatrix();

glLoadIdentity();

glScalef(explodeTx, explodeTy, 0);

glTranslatef(0.05, 0, 0);

explode();

glPopMatrix();

glPushMatrix();

glLoadIdentity();

glScalef(explodeTx, explodeTy, 0);

glTranslatef(0.1, 0, 0);

explode();

glPopMatrix();

if (explodeTx <= 1.3) {

explodeTx = explodeTy += 0.001;

}

expFactor = (float) (expFactor >= 1000 ? expFactor : expFactor + 3); //0.97

if (expFactor>900){reset(); scene=4;}

}

glPopMatrix();

}

void scene4(){

glPushMatrix();

StoryScene();

glPopMatrix();

}

void draw() {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

/\* render the scene here \*/

switch(scene)

{

case 0:

drawMainScene();

break;

case 1:

scene1();

break;

case 2:

scene2();

break;

case 3:

scene3();

break;

case 4:

scene4();

break;

}

glFlush();

glutSwapBuffers();

}

void idle() {

glutPostRedisplay();

}

void initGL(int width, int height) {

reshape(width, height);

glClearColor(0.2f, 0.8f, 1.0f, 1.0f );

glClearDepth(1.0f);

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LEQUAL);

}

int main(int argc, char\*\* argv) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(1244, 700);

glutInitWindowPosition(100, 0);

glutCreateWindow("PULWAMA ATTACK");

// register glut call backs

glutKeyboardFunc(keyboardDown);

glutReshapeFunc(reshape);

glutDisplayFunc(draw);

glutIdleFunc(idle);

glutIgnoreKeyRepeat(1); // ignore keys held down

int submenu = glutCreateMenu(options); //sub menu

glutAddMenuEntry("Really quit",2);

glutAddMenuEntry("No",0);

glutCreateMenu(options);//menu

glutAddMenuEntry("Light candles for the Soldiers",1);

glutAddSubMenu("Quit",submenu);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

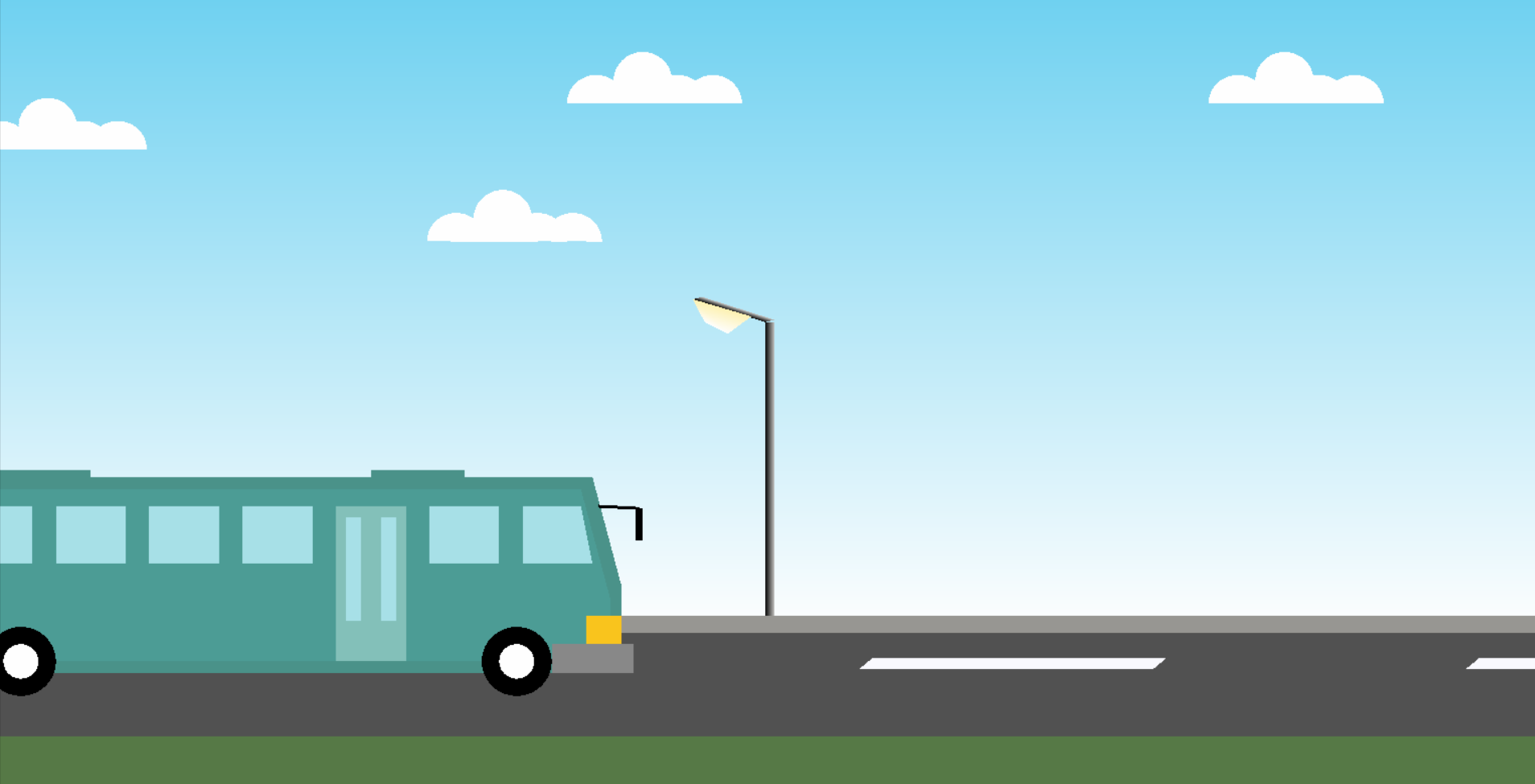
initGL(1244, 700);

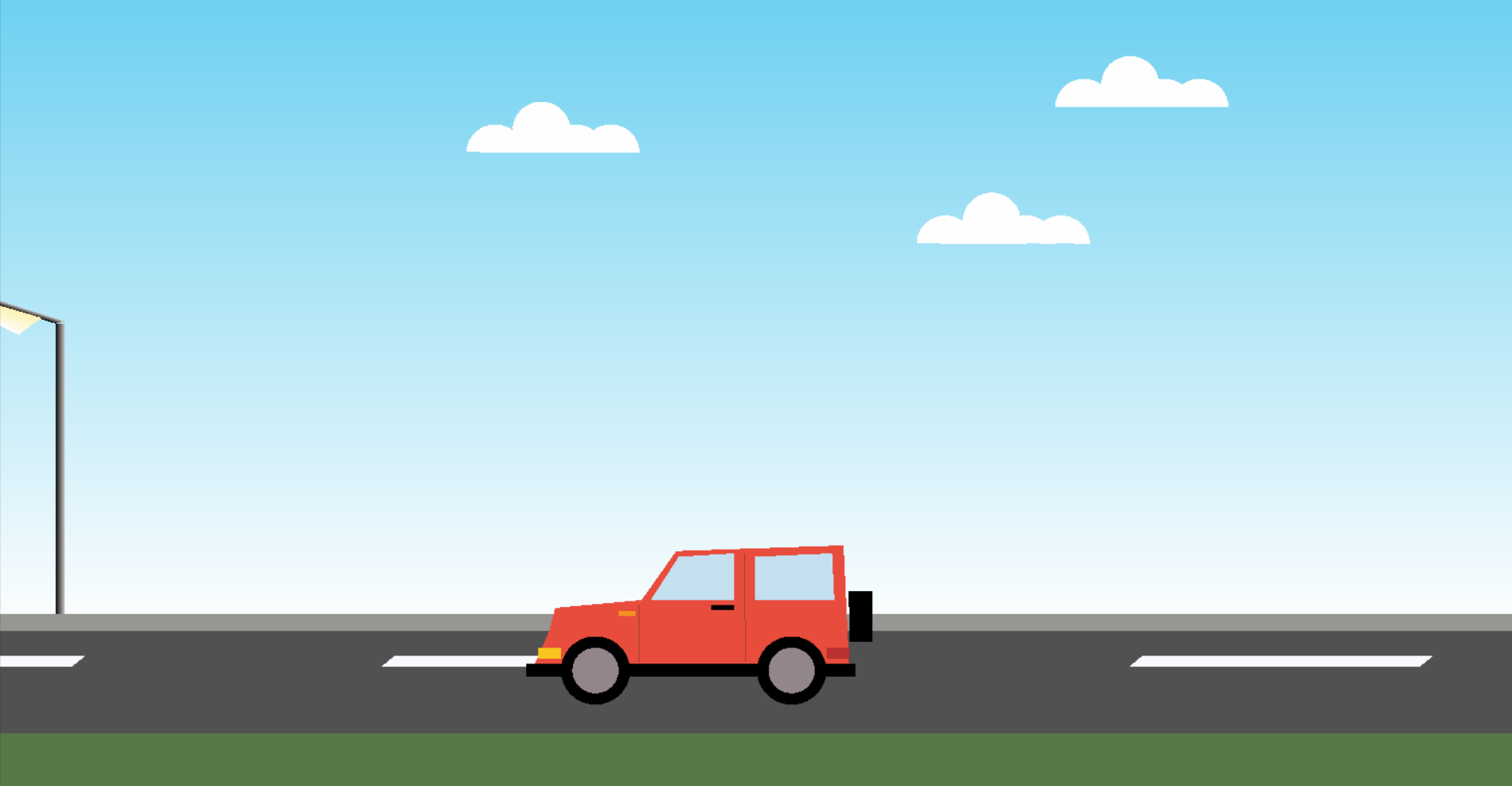
glutMainLoop();

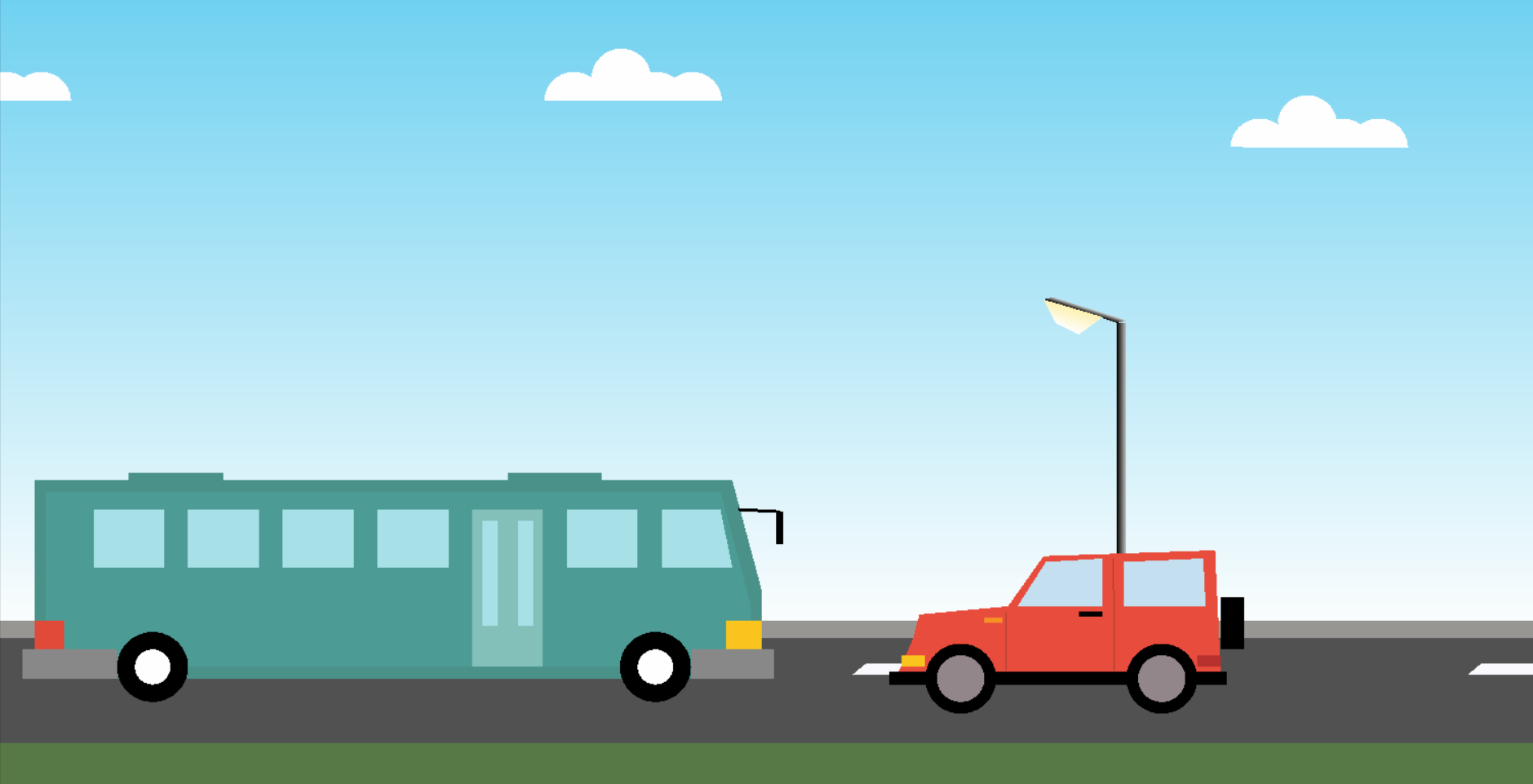
return 0;

}

**OUTPUT OF THE PROGRAM**









## *Conclusions*

**Bibliography**

WE HAVE OBTAINED INFORMATION FROM MANY RESOURCES TO DESIGN AND IMPLEMENT OUR PROJECT SUCCESSIVELY. WE HAVE ACQUIRED MOST OF THE KNOWLEDGE FROM RELATED WEBSITES. THE FOLLOWING ARE SOME OF THE RESOURCES :

* TEXT BOOKS :

INTERACTIVE COMPUTER GRAPHICS A TOP-DOWN APPROACH

-By Edward Angel.

* COMPUTER GRAPHICS,PRINCIPLES & PRACTICES

- Foley van dam

- Feiner hughes

* WEB REFERENCES: <http://jerome.jouvie.free.fr/OpenGl/Lessons/Lesson3.php>

<http://google.com>

<http://opengl.org>