VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAUM-590014

A Computer Graphics and Visualization

Mini-Project Report

On

"3D CAR MODEL SIMULATION USING OPENGL"

A Mini-project report submitted in partial fulfilment of the requirements for the award of the degree of ***Bachelor of Engineering in Computer Science and Engineering*** *of Visvesvaraya Technological University, Belgaum.*

Submitted by:
THEJAS RAJU G (1DT14CS104)
RANJITH V S (1DT14CS075)
Under the Guidance of:
Mr. RAGHU.M.T

(Asst. Prof. Dept of CSE)

Department of Computer Science and Engineering
DAYANADA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT
Kanakpura Road,Udayapura, Bangalore

2016-2017

DAYANADA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT

Kanakpura Road,Udayapura, Bangalore

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Mini-Project on Computer Graphics and Visualization work entitled **"3D CAR MODEL SIMULATION USING OPENGL"** has been successfully carried out by **THEJAS RAJU G (1DT14CS104) and RANJITH VS (1DT14CS075)** who are bonafide students of **Dayananda Sagar Academy of Technology and Management** in partial fulfilment of the requirements for the award of degree in **Bachelor of Engineering in Computer Science and Engineering** of **Visvesvaraya Technological University, Belgaum** during academic year 2016-2017. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

GUIDE:

Mr. MANJUNATH D.R	**Dr. C. NANDINI**
(Asst. Professor, Dept. Of CSE)	***(Vice Principal and HoD, DSATM)***

Examiners:

1:

2:

Signature with Date

ABSTRACT

This project makes use of functions available in OpenGL to render and simulate a three dimensional object. Primitives offered by OpenGL are used to render a three dimensional car structure as the output. The project consists of various functions which are used to load, initialize and display the 3-D model.

Using keyboard and mouse functions, a user can navigate through the application. The application mainly consists of a home screen which displays the details of the project, an instruction window which enlists the functionality of the application and various sub windows which consist of different three dimensional car models.

The project makes use of perspective viewing which allows the user to make changes in viewing angles which are incorporated as keyboard and mouse functions. This allows a user to view the model from virtually any angle possible. This gives the user a very detailed understanding of the structure of the model. Keyboard functions can be used to make changes in the colour of the model.

ACKNOWLEDGEMENT

It gives us immense pleasure to present before you our project titled **'3D CAR MODEL SIMULATION USING OPENGL'.** The satisfaction and euphoria that accompany the successful completion of any task would be but incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success.

We are glad to express our gratitude towards our prestigious institution **DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT** for providing us with utmost knowledge, encouragement and maximum facilities in undertaking this project.

We wish to express a sincere thanks to our respected principal **Dr. B. R. Lakshmikantha** for his constant support.

We express our deepest gratitude and special thanks to **Dr.C.Nandini, Vice Principal H.O.D, Dept. Of Computer Science Engineering** for all her guidance and encouragement.

We sincerely acknowledge the guidance and constant encouragement of our mini- project guide, **Asst. Prof. Mr. Manjunath D.R** who helped us greatly in developing and delivering this project.

We would also like to acknowledge the invaluable inputs and assistance provided by our Computer Graphics and Visualization faculty, **Asst. Prof. Mr. Raghu M.T**.

THEJAS RAJU G (1DT14CS104)

RANJITH V S (1DT14CS075)

TABLE OF CONTENT

Chapter-1

INTRODUCTION

1.1 Computer Graphics

Computer graphics are graphics created using computers and more generally, the representation and manipulation of image data by a computer. "Almost everything on computers that is not text or sound".

Now, we can create image using computer that are indistinguishable from photographs from the real objects

The various applications of computer graphics are:

- Display of information
- Design
- Simulation and animation
- User interfaces

The phrase “Computer Graphics” was coined in 1960 by William Fetter, a graphic designer for Boeing

Today, we find computer graphics used in various areas that include science, medicine, business, industry, art, entertainment etc.

The main reason for effectiveness of the interactive computer graphics is the speed with which the user can understand the displayed information

The current trend of computer graphics is to incorporate more physics principles into 3D graphics algorithm to better simulate the complex interactions between objects and lighting environment.

1.2 OpenGL Technology

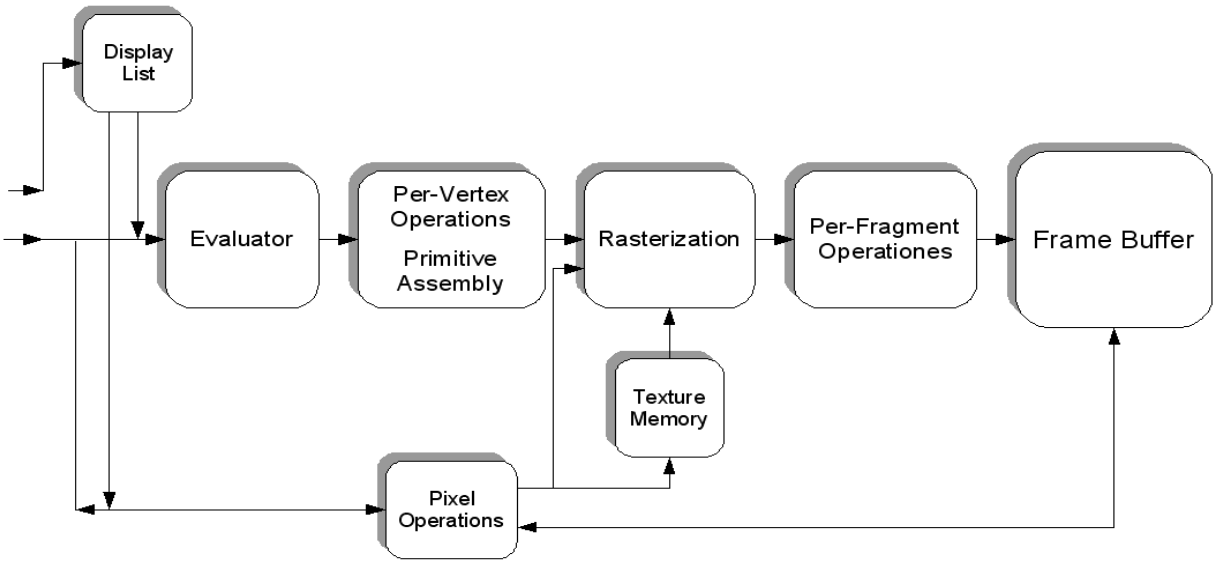
OpenGL (**Open G**raphics **L**ibrary) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, flight simulation, and video games.

OpenGL is a software API to graphics hardware, designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms, Intuitive, procedural interface with c binding, No windowing commands! And No high-level commands for describing models of three-dimensional objects.

OpenGL serves two main purposes:

1. To hide the complexities of interfacing with different 3D accelerators, by presenting the programmer with a single, uniform API.
2. To hide the differing capabilities of hardware platforms, by requiring that all implementations support the full OpenGL feature set (using software emulation if necessary).

OpenGL has historically been influential on the development of 3D accelerators, promoting a base level of functionality that is now common in consumer-level hardware: Rasterised points, lines and polygons as basic primitives.



Simplified version of the Graphics Pipeline Process; excludes a number of features like blending, VBOs and logic ops

- A transform and lighting pipeline
- Z-buffering
- Texture mapping
- Alpha blending

A brief description of the process in the graphics pipeline could be:

- Evaluation, if necessary, of the polynomial functions which define certain inputs, like NURBS surfaces, approximating curves and the surface geometry.
- Vertex operations, transforming and lighting them depending on their material. Also clipping non visible parts of the scene in order to produce the viewing volume.
- Rasterization or conversion of the previous information into pixels. The polygons are represented by the appropriate colour by means of interpolation algorithms.
- Per-fragment operations, like updating values depending on incoming an game d previously stored depth values, or colour combinations, among others.
- Lastly, fragments are inserted into the Frame buffer.

Many modern 3D accelerators provide functionality far above this baseline, but these new features are generally enhancements of this basic pipeline rather than radical revisions of it.

As OpenGL is system independent, there are no functions to create windows etc., but there are helper functions for each platform. A very useful thing is GLUT.

1.3 PROJECT DESCRIPTION:

This project makes use of functions available in OpenGL to render and simulate a three dimensional object. Primitives offered by OpenGL are used to render a three dimensional car structure as the output. The project consists of various functions which are used to load, initialize and display the 3-D model.

Using keyboard and mouse functions, a user can navigate through the application. The application mainly consists of a home screen which displays the details of the project, an instruction window which enlists the functionality of the application and various sub windows which consist of different three dimensional car models.

The project makes use of perspective viewing which allows the user to make changes in viewing angles which are incorporated as keyboard and mouse functions. This allows a user to view the model from virtually any angle possible. This gives the user a very detailed understanding of the structure of the model. Keyboard functions can be used to make changes in the colour of the model.

1.4 FUNCTIONS USED

This project is developed using CodeBlocks and this project is implemented by making extensive use of library functions offered by graphics package of OpenGL, a summary of those functions follows:

glBegin() :

Specifies the primitives that will be created from vertices presented between glBegin and subsequent glEnd.. GL_POLYGON, GL_LINE_LOOP etc.

glEnd() :

Syntax: *void **glEnd**(void);*

It ends the list of vertices.

glPushMatrix() :

Syntax: *void **glPushMatrix**(void);*

glPushMatrix pushes the current matrix stack down by one level, duplicating the current matrix.

glPopMatrix() :

Syntax: *void **glPopMatrix(** void);*

glPopMatrix pops the top matrix off the stack, destroying the contents of the popped matrix. Initially, each of the stacks contains one matrix, an identity matrix.

glTranslate() :

Syntax: *void **glTranslate**(GLdouble x, GLdouble y,GLdouble z);*

Translation is an operation that displaces points by a fixed distance in a given direction. *Parameters x, y, z* specify the *x, y,* and *z* coordinates of a translation vector. Multiplies current matrix by a matrix that translates an object by the given x, y and z-values.

glClear() :

Syntax: *void **glClear**(GLbitfield mask);*

glClear takes a single argument that is the bitwise *or* of several values indicating which buffer is to be cleared.
GL_COLOR_BUFFER_BIT, GL_DEPTH_BUFFER_BIT, GL_ACCUM_ BUFFER_BIT, and GL_STENCIL_BUFFER_BIT. Clears the specified buffers to their current clearing values.

glClearColor() :

Syntax: *void **glClearColor**(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha);*

Sets the current clearing color for use in clearing color buffers in RGBA mode. The red, green, blue, and alpha values are clamped if necessary to the range [0,1]. The default clearing color is (0, 0, 0, 0), which is black.

glMatrixMode() :

Syntax : *void **glMatrixMode** (GLenum mode) ;*

It accepts three values GL_MODELVIEW, GL_PROJECTION and GL_TEXTURE. It specifies which matrix is the current matrix. Subsequent transformation commands affect the specified matrix.

glutInitWindowPosition() :

Syntax: *void **glutInitWindowPosition**(int x, int y);*

This api will request the windows created to have an initial position. The arguments x, y indicate the location of a corner of the window, relative to the entire display.

glLoadIdentity() :

Syntax: *void **glLoadIdentity**(void);*

It replaces the current matrix with the identity matrix

glutInitWindowSize() :

Syntax: *void **glutInitWindowSize**(int width, int height);*

The api requests windows created to have an initial size. The arguments width and height indicate the window's size (in pixels). The initial window size and position are hints and may be overridden by other requests.

glutInitDisplayMode

Syntax: *void **glutInitDisplayMode** (unsigned int mode);*

Specifies the display mode, normally the bitwise OR-ing of GLUT display mode bit *masks.* This api specifies a display mode (such as RGBA or color-index, or single or double-buffered) for windows.

glFlush() :

Syntax: *void **glFlush**(void);*

The glFlush function forces execution of OpenGL functions in finite time.

glutCreateWindow() :

Syntax: *int **glutCreateWindow**(char *name);*

The parameter *name* specifies any name for window and is enclosed in double quotes. This opens a window with the set characteristics like display mode, width, height, and so on. The string name will appear in the title bar of the window system . The value returned is a unique integer identifier for the window. This identifier can be used for controlling and rendering to multiple windows from the same application.

glutDisplayFunc() :

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

Specifies the new display callback function. The api specifies the function that's called whenever the contents of the window need to be redrawn. All the routines need to be redraw the scene are put in display callback function.

glVertex2f():

Syntax: *void **glVertex2f**(GLfloat x,GLfloat y);*

x Specifies the x-coordinate of a vertex.

y Specifies the y-coordinate of a vertex.

The glVertex function commands are used within glBegin/glEnd pairs to specify point, line, and polygon vertices. The current color, normal, and texture coordinates are associated with the vertex when glVertex is called. When only x and y are specified, z defaults to 0.0 and w defaults to 1.0. When x, y, and z are specified, w defaults to 1.0.

glutBitMapCharacter() :

Syntax: *void **glutBitmapCharacter**(void *font, int character);*

Without using any display lists, glutBitmapCharacter renders the character in the named bitmap font.

glRasterPos2f() :

Syntax: *void **glRasterPos2f**(GLfloat x,GLfloat y);*

x Specifies the x-coordinate for the current raster position.

y Specifies the y-coordinate for the current raster position.

OpenGL maintains a 3-D position in window coordinates. This position, called the raster position, is maintained with sub pixel accuracy. It is used to position pixel and bitmap write operations. The current raster position consists of three window coordinates (x, y, z), a clip coordinate w value, an eye coordinate distance, a valid bit, and associated color data and texture coordinates. The w coordinate is a clip coordinate, because w is not projected to window coordinates.

glutMainLoop(void) :

glutMainLoop() enters the GLUT event processing loop. These routines should be called at most once in the GLUT program. Once called, these routine will never exit. It will call all the necessary call back functions that have been registered.

glOrtho():

Syntax: *void* ***glOrtho****(GLdouble left, Gldouble right, Gldouble bottom, Gldouble top, Gldouble nearVal, Gldouble farVal);*

left, right: Specify the coordinates for the left and right vertical clipping planes.

bottom, *top*: Specify the coordinates for the bottom and top horizontal clipping planes.

nearVal, *farVal*: Specify the distances to the nearer and farther depth clipping planes. These values are negative if the plane is to be behind the viewer.

glOrtho describes a transformation that produces a parallel projection. The current matrix is multiplied by this matrix and the result replaces the current matrix.

glEnable() & glDisable():

Syntax: *void* **glEnable***(GLenum cap);*
void ***glDisable****(GLenum cap);*

cap: Specifies a symbolic constant indicating a GL capability

glEnable() and **glDisable()** enable and disable various capabilities.

GluLookAt():

Syntax: *void* ***gluLookAt****(GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ, GLdouble centerX, , GLdouble centerY, GLdouble centerZ, GLdouble upX, , GLdouble upY, , GLdouble upZ)*

eyeX, *eyeY*, *eyeZ:* Specifies the position of the eye point.

centerX, *centerY*, *centerZ:* Specifies the position of the reference point.

upX, *upY*, *upZ*: Specifies the direction of the *up* vector.

gluLookAt() creates a viewing matrix derived from an eye point, a reference point indicating the center of the scene, and an *UP* vector.

Chapter-2

REQUIREMENTS SPECIFICATION

2.1 Hardware requirements:

- Pentium 4 or Equivalent Processor.
- 256 MB RAM
- 5 MB Disk Space
- Intel GMA 850 or better display adaptor
- Mouse and Keyboard input devices

2.2 Software requirements:

- Microsoft Windows XP / Linux 2.6.27 or later
- OpenGL and GLUT libraries
- CodeBlocks IDE

Platform and Tools:

1. The project has been carried out in Linux platform.
2. Built using Code Blocks IDE with glut libraries.
3. Keyboard interface is required for this project.

Chapter-3

INTERFACE AND ARCHITECTURE

KEYBOARD FUNCTIONS:

KEY (‘x’|’X’): used to move camera in x-direction.

KEY (‘y’|’Y’): used to move camera in y-direction.

KEY (‘z’|’Z’): used to move camera in z-direction.

KEY (<): used to rotate model by ten degrees in x-direction.

KEY (∧): used to rotate model by ten degrees in z-direction.

KEY (>): used to rotate model by ten degrees in y-direction.

KEY (1): used to display first car model.

KEY (2): used to display second car model.

KEY (3): used to display third car model..

KEY (4): used to display fourth car model..

KEY (5): used to display fifth car model.

KEY (6): used to display sixth car model.

KEY (‘c’|’C’): used to change the color of the model.

KEY (‘h’|’H’): used to navigate to the home screen.

KEY (‘i’|’I’): used to navigate to the instructions screen.

KEY (‘q’|’Q’|ESC): used to exit the application.

MOUSE FUNCTIONS:

Left Mouse Button: Displays a menu using which

- Car model to be displayed can be selected.
- Application can be exited.

Right Mouse Button (Available only in car display windows): Displays a menu using which

- Color of the car model can be changed.
- Application can be exited.

Chapter-4

IMPLEMENTATION

Implementation: Implementation of the project is the most important part of the project. The implement phase deals with the quality, performance, accuracy and debugging. The end derivable is the simulation.
This simulation is implemented using keyboard and mouse interfaces which are as follows:

Keyboard Function :

void keys(unsigned char, int x, int y);

This function is used to make responds to the keys that are pressed from the keyboard. In this project, these keyboard keys are used to switch on the kit, to choose different gates, etc…

Mouse Function :

void mouse(int btn, int state, int x,int y);

This function is used to make responds to the buttons that are pressed from the mouse. In this project, these buttons are used for rotating purpose.

Display Function :

void display();

In this function, rotation,translation and scaling is done by pushing and popping the matrix and using the inbuilt glutSolid function.We draw the helicopter,human and clouds.

PrintText Function:

void printtext(int, int, string);

In this function, string that has to be printed is passed as a parameter which is then displayed in the co-ordinates which are also passed as parameters.

LoadObj Function:

void loadObj(char *);

In this function the source .obj file name is passed as the parameter. The fuction parses the file and converts it into a list containing vertex values.

DrawCar Function:

void drawCar(void);

In this function the model is translated, its color is defined and the structure is scaled accordingly.

Initialize Function:

void init(void);

In this function the application functions such as reshape, keyboard and mouse functions are initialized.

Menu Handler Function:

void handler_name(int choice);

In this function the choice returned when the user chooses a menu option is evaluated to perform a specific operation such as selecting a car, changing color or quiting the application.

Main Function:

int main(int argc, char *argv[]);

In this function, we are creating the window, enabling the mouse and keyboard functions, initializing the display mode and also enables the 3D view.

Chapter-5

SOURCE CODE

```
//Header Files
#include<windows.h>
#include<GL/glut.h>
#include<stdio.h>
#include <iostream>
#include <cstdio>
#include <string>
#include <cstdlib>
using namespace std;

//Function Declaration
void printtext(int, int, string );
void loadObj(char *);
void drawCar(void);
void init(void);
void init1(void);
void display(void);
void display1(void);
void display2(void);
void myReshape(int , int);
void keys(unsigned char, int , int );
void spec_keys(int,int,int );
void color1(int choice);
void menu2(int choice);
void menu1(int choice);
void carHandler(int choice);

//Global Variables
GLuint car;
int home,color=0;
static GLfloat theta[] = {0.0,0.0,0.0};
static GLint axis = 2;
static GLdouble viewer[]= {0.0, 0.0, 5.0}; /* initial viewer location */

//Main Function
int  main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
	home=glutCreateWindow("3D CAR MODEL SIMULATION USING OpenGL");
	glutInitWindowSize(1000, 1000);
	init1();
	glutDisplayFunc(display1);
	glutMainLoop();
}
```

```
//PrintText Function

void printtext(int x, int y, string String)
{
//(x,y) is from the bottom left of the window

   int WindowHeight = 1000;
   int WindowWidth = 1000;
   glMatrixMode(GL_PROJECTION);
   glPushMatrix();
   glLoadIdentity();
   glOrtho(0, WindowWidth, 0, WindowHeight, -1.0f, 1.0f);
   glMatrixMode(GL_MODELVIEW);
   glPushMatrix();
   glLoadIdentity();
   glPushAttrib(GL_DEPTH_TEST);
   glDisable(GL_DEPTH_TEST);
   glRasterPos2i(x,y);
   for (int i=0; i<String.size(); i++)
   {
      glutBitmapCharacter(GLUT_BITMAP_9_BY_15, String[i]);
   }
   glPopAttrib();
   glMatrixMode(GL_PROJECTION);
   glPopMatrix();
   glMatrixMode(GL_MODELVIEW);
   glPopMatrix();
}

//Object Loading Fuction

void loadObj(char *fname)
{
   FILE *fp;
   int read;
   int i=0;
   GLfloat x, y, z;
   char ch;
   car=glGenLists(1);
   fp=fopen(fname,"r");
   if (!fp)
   {
      printf("can't open file %s\n", fname);
      exit(1);
   }

   glPointSize(20.0);
   glNewList(car, GL_COMPILE);
   {
      glPushMatrix();
```

```
        glBegin(GL_LINE_STRIP);

        while(!(feof(fp)))
        {
           read=fscanf(fp,"%c %f %f %f",&ch,&x,&y,&z);
           if(read==4&&ch=='v')
           {
              glVertex3f(x,y,z);
           }

        }
        glEnd();
     }
     glPopMatrix();
     glEndList();
     fclose(fp);
}

//Function to draw the car
void drawCar()
{
     glPushMatrix();
     glTranslatef(0,0,0);
     int a = color%21;
          if(a==0)       glColor3f(.97, .97, .97);//white
          else if(a==1)  glColor3f(.9, .9, .9);
          else if(a==2)  glColor3f(.275, .510, .706);   //blue
          else if(a==3)  glColor3f(.824, .412, .118);   //chocolate
          else if(a==4)  glColor3f(.941, .502, .502);   //Coral
          else if(a==5)  glColor3f(.0, .502, .0);       //green
          else if(a==6)  glColor3f(.502, .502, .502);   //grey
          else if(a==7)  glColor3f(.372549, .623529, .623529);      //cadet blue
          else if(a==8)  glColor3f(.6, .196078, .8);    //dark orchid
          else if(a==9)  glColor3f(.858824, .858824, .239216);      //golden yellow
          else if(a==10) glColor3f(0.502,0, 0.502);     //purple
          else if(a==11) glColor3f(0.184314,.184314, .309804);      //midnight blue
          else if(a==12) glColor3f(0.137255,.137255, .556863);      //NavyBlue
          else if(a==13) glColor3f(0.753 ,.753 , .753); //Silver
          else if(a==14) glColor3f(1.00,.078, 0.576);   //Neon pink
          else if(a==15) glColor3f(1.00,.855, .725);    //Peach
          else if(a==16) glColor3f(0.55,.09, .09);      //Scarlet
          else if(a==17) glColor3f(0.13,.37, .31);      //Dusty Green
          else if(a==18) glColor3f(0.42,.26, .15);      //browny
          else if(a==19) glColor3f(0.85,.85, .95);      //Quartz
          else if(a==20) glColor3f(0.85,.85, .10);      //light gold
          else if(a==21) glColor3f(1.00, 0.00, 1.00);   //Magenta
     glScalef(1,1,1);
     glCallList(car);
     glPopMatrix();
}
```

```
//Initialize Function
void init()
{
   glutReshapeFunc(myReshape);
   glutDisplayFunc(display);
   int nColorMenu;
   int nMainMenu;
   int carMainMenu;
   int nCarMenu;
   nCarMenu = glutCreateMenu(carHandler);
   glutAddMenuEntry("Lamborghini Aventador",1);
   glutAddMenuEntry("Porsche Cayman",2);
   glutAddMenuEntry("Nissan GTR",3);
   glutAddMenuEntry("Ferrari F1 racecar",4);
   glutAddMenuEntry("Ford Mustang Shelby",5);
   glutAddMenuEntry("Audi R8",6);
   carMainMenu = glutCreateMenu(menu1);
   glutAddSubMenu("Select Car",nCarMenu);
   glutAddMenuEntry("Quit",1);
   glutAttachMenu(GLUT_LEFT_BUTTON);

   nColorMenu=glutCreateMenu(color1);
         glutAddMenuEntry("White",1);
         glutAddMenuEntry("Blue",2);
         glutAddMenuEntry("Chocolate",3);
         glutAddMenuEntry("Coral",4);
         glutAddMenuEntry("Green",5);
         glutAddMenuEntry("Grey",6);
         glutAddMenuEntry("Cadet Blue",7);
         glutAddMenuEntry("Dark Orchid",8);
         glutAddMenuEntry("Golden Yellow",9);
         glutAddMenuEntry("Purple",10);
         glutAddMenuEntry("Midnight Blue",11);
         glutAddMenuEntry("Navy Blue",12);
         glutAddMenuEntry("Silver",13);
         glutAddMenuEntry("Neon Pink",14);
         glutAddMenuEntry("Peach",15);
         glutAddMenuEntry("Scarlet",16);
         glutAddMenuEntry("Dusty Green",17);
         glutAddMenuEntry("Brown",18);
         glutAddMenuEntry("Quartz",19);
         glutAddMenuEntry("Light Gold",20);
         nMainMenu = glutCreateMenu(menu2);
         glutAddSubMenu("Change Color", nColorMenu);
         glutAddMenuEntry("Quit",1);
         glutAttachMenu(GLUT_RIGHT_BUTTON);
   glutSpecialFunc(spec_keys);
   glutKeyboardFunc(keys);
   glEnable(GL_DEPTH_TEST);
}
```

```
void init1(void)
{
   glutReshapeFunc(myReshape);
   glMatrixMode(GL_PROJECTION);
         glLoadIdentity();
         gluPerspective(70, 1, 1, 100);
         glMatrixMode(GL_MODELVIEW);
         glLoadIdentity();
         gluLookAt(2, 2, 10, 2, 0, 0, 0, 1, 0);
   glutKeyboardFunc(keys);
   int carMainMenu;
   int nCarMenu;
   nCarMenu = glutCreateMenu(carHandler);
   glutAddMenuEntry("Lamborghini Aventador",1);
   glutAddMenuEntry("Porsche Cayman",2);
   glutAddMenuEntry("Nissan GTR",3);
   glutAddMenuEntry("Ferrari F1 racecar",4);
   glutAddMenuEntry("Ford Mustang Shelby",5);
   glutAddMenuEntry("Audi R8",6);
   carMainMenu = glutCreateMenu(menu1);
   glutAddSubMenu("Select Car",nCarMenu);
   glutAddMenuEntry("Quit",1);
   glutAttachMenu(GLUT_LEFT_BUTTON);
}

//Display Functions

void display(void)
{
         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
/* Update viewer position in modelview matrix */
         glLoadIdentity();
         glutReshapeFunc(myReshape);
         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);
         drawCar();
         glutSwapBuffers();
}


void display1(void)
{
         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

         glEnable(GL_DEPTH_TEST);

   char string[200];
   sprintf(string, "3-D CAR MODEL SIMULATION USING OPENGL");
```

```
   printtext(375,600,string);

   sprintf(string, "A VTU 6th SEMESTER COMPUTER GRAPHICS AND
VISUALIZATION PROJECT");
   printtext(285,550,string);

   sprintf(string, "BY:");
   printtext(600,450,string);

   sprintf(string,"THEJAS RAJU G  (1DT14CS104)");
   printtext(600,400,string);

   sprintf(string,"RANJITH VS     (1DT14CS075)");
   printtext(600,350,string);

   sprintf(string,"PRESS SPACE TO CONTINUE");
   printtext(425,200,string);

   glutSwapBuffers();
}

void display2(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        glEnable(GL_DEPTH_TEST);

   char string[200];
   sprintf(string, "3-D CAR MODEL SIMULATION USING OPENGL");
   printtext(375,800,string);

   sprintf(string, "INSTRUCTIONS:");
   printtext(275,675,string);

   sprintf(string, "---PERSPECTIVE VIEWING CHANGES:     USE ARROW and x|X y|Y
z|Z keys");
   printtext(275,600,string);

   sprintf(string, "---CHANGE CAR MODELS:              USE NUMBER keys 1-6 or LEFT
MOUSE BUTTON");
   printtext(275,525,string);

   sprintf(string, "---CHANGE CAR COLOUR:              USE c|C keys or RIGHT MOUSE
BUTTON ");
   printtext(275,450,string);

   sprintf(string, "---NAVIGATION TO HOME PAGE:        h|H key");
   printtext(275,375,string);

   sprintf(string, "---NAVIGATION TO INSTRUCTIONS PAGE: i|I key");
```

```
   printtext(275,300,string);

   sprintf(string, "---QUIT APPLICATION:              q|Q key or ESC key");
   printtext(275,225,string);

   glutSwapBuffers();
}

//Reshape Function

void myReshape(int w, int h)
{
       glViewport(0,0,w,h);
       glMatrixMode(GL_PROJECTION);
       glLoadIdentity();
       gluPerspective (60, (GLfloat)w / (GLfloat)h, 0.1, 1000.0);
       glMatrixMode(GL_MODELVIEW);
}

//Function to incorporate Arrow keys

void spec_keys(int key,int x,int y)
{
          if(key == GLUT_KEY_UP)
          axis = 2;
           if(key == GLUT_KEY_LEFT)
          axis = 0;
           if(key == GLUT_KEY_RIGHT)
          axis = 1;
   theta[axis] += 10.0;
   if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
   display();
}

//Function to incorporate other keys

void keys(unsigned char key, int x, int y)
{
/* Use x, X, y, Y, z, and Z keys to move viewer */
  if(key == 'x')
  {
     viewer[0]-= 1.0;
     display();
  }
  if(key == 'X')
  {
     viewer[0]+= 1.0;
     display();
  }
  if(key == 'y')
```

```
{
   viewer[1]-= 1.0;
   display();
}
if(key == 'Y')
{
   viewer[1]+= 1.0;
   display();
}
if(key == 'z')
{
   viewer[2]-= 1.0;
   display();
}
if(key == 'Z')
{
   viewer[2]+= 1.0;
   display();
}
if(key == 'c' | key == 'C')
{
   color++;
   init();
   display();
}
if(key == 'h'| key == 'H')
{
    glutDestroyWindow(home);
    home=glutCreateWindow("3D CAR MODEL SIMULATION USING OpenGL");
    glutInitWindowSize(1000, 1000);
    glutDisplayFunc(display1);
    init1();

}
if(key == 'i'| key == 'I')
{
    glutCreateSubWindow(home,0,0,500,500);
    glutDisplayFunc(display2);
    glutFullScreen();
    init1();

}
if(key == 32)
{
    glutCreateSubWindow(home,0,0,500,500);
    glutDisplayFunc(display2);
    glutFullScreen();
    init1();
}
if(key == 49)
```

```
{
    glutCreateSubWindow(home,200,300,500,500);
    glutDisplayFunc(display);
    loadObj("F://opengl//car obj//Avent.obj");
    glutFullScreen();
    init();
}
 if(key == 50)
{
    glutCreateSubWindow(home,200,300,500,500);
    glutDisplayFunc(display);
    loadObj("F://opengl//car obj//porsche-cayman.obj");
    glutFullScreen();
    init();
}
if(key == 51)
{
    glutCreateSubWindow(home,200,300,500,500);
    glutDisplayFunc(display);
    loadObj("F://opengl//car obj//nissan-gt-r-nismo.obj");
    glutFullScreen();
    init();
}
if(key == 52)
{
    glutCreateSubWindow(home,200,300,500,500);
    glutDisplayFunc(display);
    loadObj("F://opengl//car obj//ferrari-f1-race-car.obj");
    glutFullScreen();
    init();
}
  if(key == 53)
{
    glutCreateSubWindow(home,200,300,500,500);
    glutDisplayFunc(display);
    loadObj("F://opengl//car obj//1967-shelby-ford-mustang.obj");
    glutFullScreen();
    init();
}
  if(key == 54)
{
    glutCreateSubWindow(home,200,300,500,500);
    glutDisplayFunc(display);
    loadObj("F://opengl//car obj//audi-r8-red.obj");
    glutFullScreen();
    init();
}
if(key == 'q' | key == 'Q')
{
    exit(0);
```

```
  }
  if(key == 27)
  {
      exit(1);
  }
}

//Menu Handler Functions

void color1(int choice)
{
   color=choice;
   glutPostRedisplay();
}


void menu2(int choice)
{
   if(choice=1)
   {
      exit(0);
   }
}


void menu1(int choice)
{
   if(choice==1)
   {
      exit(1);
   }
}


void carHandler(int choice)
{
   if(choice == 1)
  {

      glutCreateSubWindow(home,200,300,500,500);
      glutDisplayFunc(display);
      loadObj("F://opengl//car obj//Avent.obj");
      glutFullScreen();
      init();
  }
   if(choice == 2)
  {
      glutCreateSubWindow(home,200,300,500,500);
      glutDisplayFunc(display);
      loadObj("F://opengl//car obj//porsche-cayman.obj");
```

```
        glutFullScreen();
        init();
    }
    if(choice == 3)
    {
        glutCreateSubWindow(home,200,300,500,500);
        glutDisplayFunc(display);
        loadObj("F://opengl//car obj//nissan-gt-r-nismo.obj");
        glutFullScreen();
        init();
    }
    if(choice == 4)
    {
        glutCreateSubWindow(home,200,300,500,500);
        glutDisplayFunc(display);
        loadObj("F://opengl//car obj//ferrari-f1-race-car.obj");
        glutFullScreen();
        init();
    }
     if(choice == 5)
    {
        glutCreateSubWindow(home,200,300,500,500);
        glutDisplayFunc(display);
        loadObj("F://opengl//car obj//1967-shelby-ford-mustang.obj");
        glutFullScreen();
        init();
    }
     if(choice == 6)
    {
        glutCreateSubWindow(home,200,300,500,500);
        glutDisplayFunc(display);
        loadObj("F://opengl//car obj//audi-r8-red.obj");
        glutFullScreen();
        init();
    }
}
```

Chapter-6

SNAPSHOTS

SAMPLE OUTPUT

Fig1. Screenshot of the Home Page

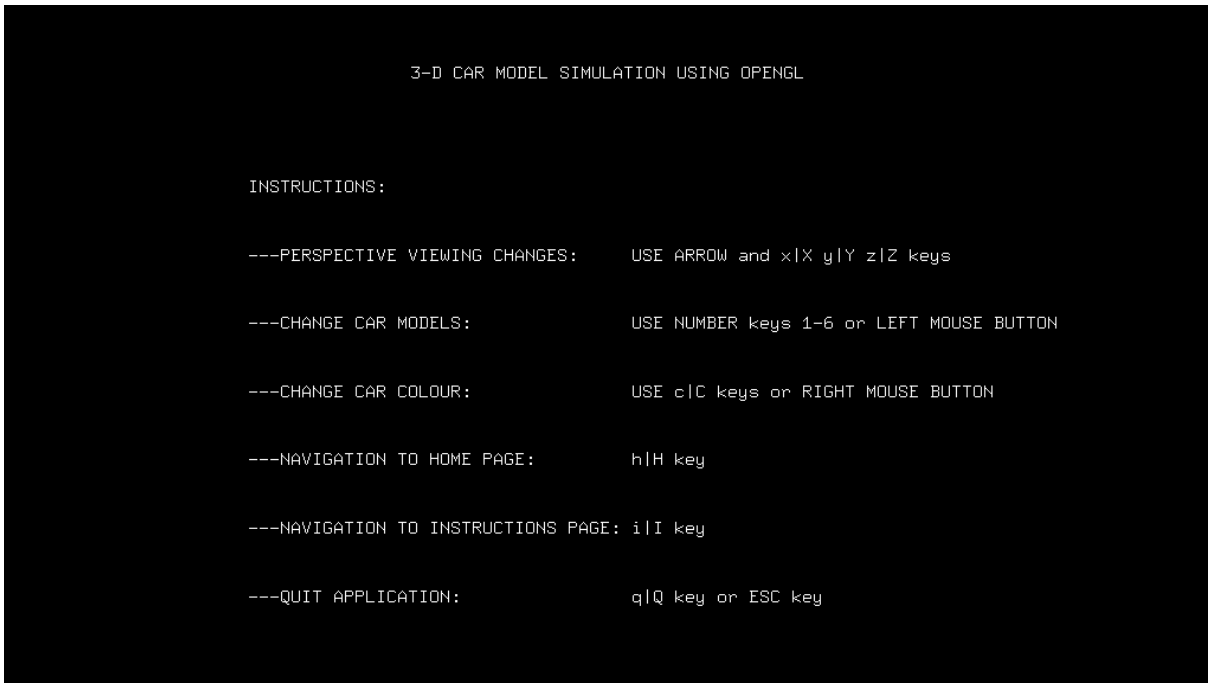


Fig2. Screenshot of the Instructions Page

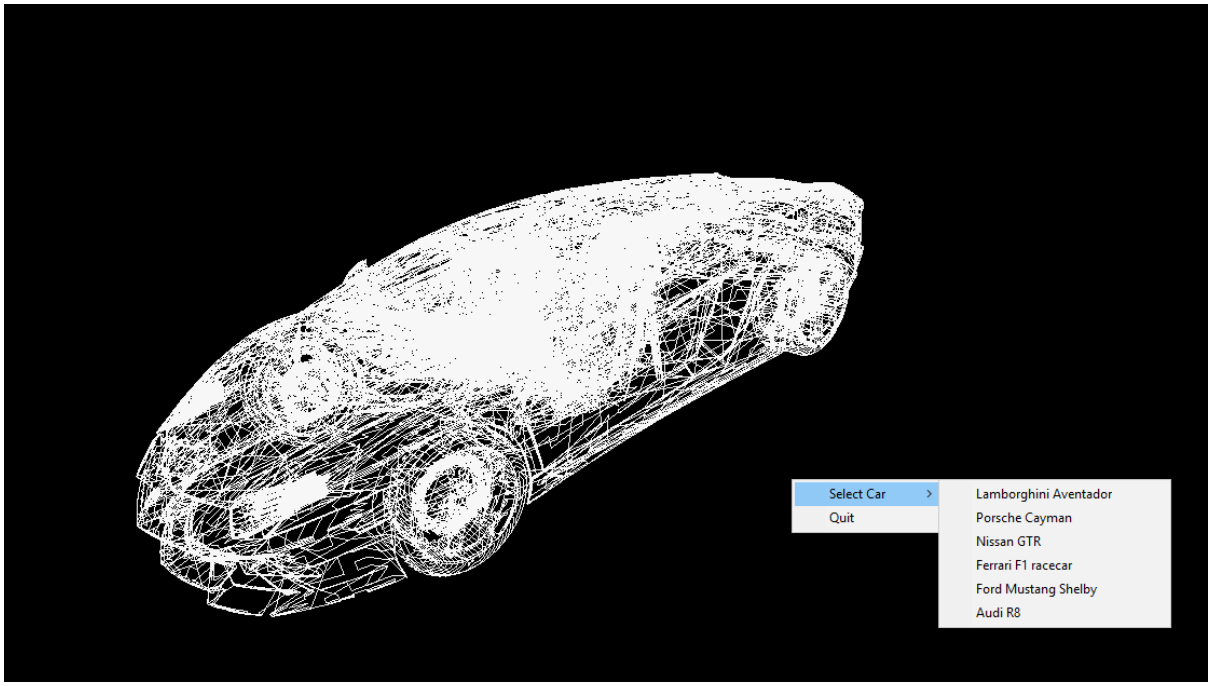


Fig 3. Screenshot of a 3D Car Model

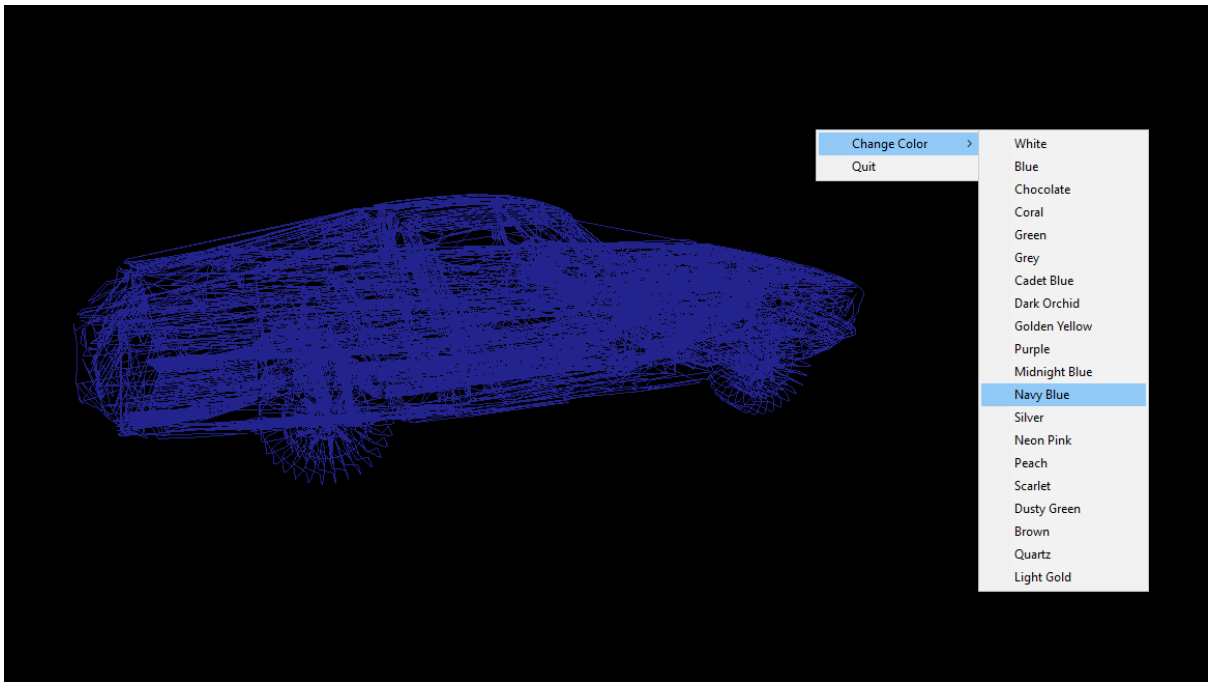


Fig 4. Screenshot depicting change in colour and viewing angle of another model

Chapter-7

FUTURE ENHANCEMENT

The project can be enhanced by adding more models which can be rendered using the same application.

The application can be tweaked such that the car modelling is performed accurately using polygons and incorporate lighting to enhance the detailing of the model.

Texture mapping can be performed to render the model as realistically as possible.

More functionality can be added to the car such as opening the doors, operating headlamps, sounding horn etc.

Chapter-8

CONCLUSION

We have been successful in our attempt at implementing “3D CAR MODEL SIMULATION USING OPENGL”.

This project was developed in Windows 10 operating system. It was also tested on Windows XP, Windows 7, Linux and Mac OS X making it a truly cross-platform project.

The basic OpenGL primitives when coupled with functions to load and parse an object file can be used to render a basic three dimensional model.

Keyboard and mouse functions enable the user to navigate through the application and perform various other operations such as moving to another viewing angle and changing the colour of models.

Chapter-9

REFERENCES

Books:

[1] Interactive Computer Graphics A Top-Down Approach with OpenGL -Edward Angel

[2] Computer Graphics using OpenGL by F.S. Hill Jr. & Stephen M. Kelley Jr

Websites:

[1] http://www.opengl.org

[2] http://pyopengl.sourceforge.net/

[3] http://stackoverflow.com/

[4] http://programmingexamples.net/

[5] http://codeproject.com/

[6] http://www.khronos.org/

[7] http://netization.blogspot.in/

Chapter-10

APPENDIX

10.1 User Manual

- Ensure that header file 'windows.h' is included if the system uses Windows OS.
- The file path of obj files must be defined according to where the .obj files are available on the system.

10.2 Personal Details

NAME: THEJAS RAJU G **USN:** 1DT14CS104

RANJITH V S 1DT14CS075

SEMESTER AND SECTION: 6TH SEMESTER AND SECTION 'B'

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

COLLEGE: DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT

EMAIL ID: thejasraju96@gmail.com

ranjith.raj007@gmail.com