LIST OF CONTENTS

		Page No.
i.	**Acknowledgement**	**i**
ii.	**Abstract**	**ii**
iii.	**List of Contents**	**iii**
iv.	**List of Figures**	**iv**

Chapter 1: INTRODUCTION	**1-2**
1.1: About Computer Graphics	
1.2: About OpenGL	
1.3: Features of OpenGL	
1.4: Aim	
1.5: Overview	
1.6 : Outcome	
Chapter 2: REQUIREMENT SPECIFICATION	**3**
2.1: Software Requirements	
2.2: Hardware Requirements	
Chapter 3: DESIGN	**4-6**
3.1: User Defined Functions	
3.2: Flowchart	
3.3: OpenGL API's Used with Description	
Chapter 4: IMPLEMENTATION	**7-12**
4.1: Main Functions	
4.2: Pseudo code	

LIST OF FIGURES

Page No.

Chapter 5 : RESULT AND SNAPSHOTS **13-17**

Figure 5.1: The Help Window

Figure 5.2: The car in model mode

Figure 5.3: Car in driving mode

Figure 5.4: The display with menu

Figure 5.5: Car with fog effects

Figure 5.6: Car with wheel effects

Figure 5.7: Car in the night mode

CONCLUSION & FUTURE ENHANCEMENT **18**

REFERENCES **19**

CHAPTER 1
INTRODUCTION

1.1 About Computer Graphics

The Computer Graphics is one of the most effective and commonly used methods to communicate the processed information to the user. It displays the information in the form of graphics objects such as pictures, charts, graphs and diagram instead of simple text.In computer graphics, pictures or graphics objects are presented as a collection of discrete picture elements called pixels. The pixel is the smallest addressable screen elementComputer graphics today is largely interactive: The user controls the contents structure, and appearance of objects and their displayed images by using input devices, such as a keyboard, mouse, or touch-sensitive panel on the screen.Computer Graphics relies on an internal model of the scene, that is, mathematical representation suitable for graphical computations. The model describes the 3D shapes, layout andmaterials of the scene. This 3D representation then has to be projected to compute a 2D image from a given viewpoint, this is rendering step. Rendering involves projecting the objects, handling visibility (which parts of objects are hidden) and computing their appearance and lighting interactions. Finally, for animated sequence, the motion of objects has to be specified.

1.2 About OpenGL

OpenGL (Open Graphics Library) is an open specification for an applications program interface for defining 2D and 3D objects. The specification is cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. It renders 3D objects to the screen, providing the same set of instructions on different computers and graphics adapters. Thus it allows us to write an application that can create the same effects in any operating system using any OpenGL-adhering graphics adapter.In Computer graphics, a 3-dimensional primitive can be anything from a single point to an ‘n’ sided polygon. From the software standpoint, primitives utilize the basic 3-dimensional rasterization algorithms such as Bresenham's line drawing algorithm, polygon scan line fill, texture mapping and so forth.OpenGL is a low-level, procedural API, requiring the programmer to dictate the exact steps requiredto render a scene.

OpenGL's low-level design requires programmers to have a good knowledge of the graphics pipeline, but also gives a certain amount of freedom to implement novel rendering algorithms.

1.3 Features of OpenGL

- Geometric Primitives allow you to construct mathematical descriptions of objects.
- Viewing and Modeling permits arranging objects in a 3-dimensional scene, move our cameraaround space and select the desired vantage point for viewing the scene to be rendered.

1.4 Aim

The aim of the project is to illustrate the concepts and usage of functions in OpenGL by creating a3D Car Simulation in different geometrical figures.

1.5 Overview

A car is a useful means of transportation from one place to another place. Development in the field of science and technology led to the discovery of car which helped the mankind in various ways. Since many years the automobiles have been subjected to continuous evolution and hence a variety of car models with each one having its own features, shape and size exist today. This is an approach to simulate the working of one such model of a car. The car used here is a 3D object with simple features. One can either view the car model or enter into the driving mode. In the former case, the car is placed against a plain background whereas in the latter case, it is placed on a road having greenery in the background. The car can be moved to and forth, rotated about the three principle axes and can be changed in terms of size and color. Along with the above features the fog and wheel effects can also be imposed on the car. The car can be viewed either in the day mode or in the night mode. A uniform lighting effect is also included, however it is left to the user to enable or disable lighting. Also, the car can be zoomed in or zoomed out. All these actions can be performed by using the menu provided.

1.6 Outcome

Through the OpenGL function we were able to demonstrate the working of 3D Car Simulation which demonstrates the use of keyboard interface and the mouse interface, when mouse is pressed the menuopens for selecting different shapes respectively.

CHAPTER 2

REQUIREMENT SPECIFICATION

A requirements specification is a description of a software system to be developed, laying out functional and non-functional requirements, and may include a set of use cases that describe interactions the users will have with the software.

2.1 Software Requirements

Operating System: Windows XP, Windows 2002
Language Tools: OpenGL
Compiler: GNU GCC compiler/C++ compiler
Documentation Tool: Visual C++ with OpenGL
functions

2.2 Hardware Requirements :

Processor: Pentium 4 and aboveRAM: 1GB with 256MHz
Hard disk: 80GB
Keyboard: Standard
Mouse: Standard

CHAPTER 3

DESIGN

Design of any software depends on the architecture of the machine on which that software runs, for which the designer needs to know the system architecture. Design process involves design of suitable algorithms, modules, subsystems, interfaces etc. This project has been created using the OpenGL interface along with the GLUT(OpenGL Library Tool), using the platform Visual C++ 6.0 as a compiling tool. This project has been designed in a simple and lucid manner so that further developments can be made, and run on many platforms with a few changes in the code.

3.1 User Defined Functions

1. **Void display(void):** In this function first we should print the instructions that we would bedisplayed on the pop-up window.
2. **int main(int argc, char** argv):** Here we call all the function we defined previously in the program and this function creates an output window.
3. **void NormalKey(GLubyte key, GLint x, GLint y):** The function called whenever a "normal" key is pressed.
4. **void myMenu(int id) :** The function used to create menu.
5. **void myreshape(int w,int h) :** This function is used to reshape object.
6. **void display_string(int x, int y, char *string, int font) :** This is the function hich will be called when a "redisplay" is requested.

3.2 Flowchart

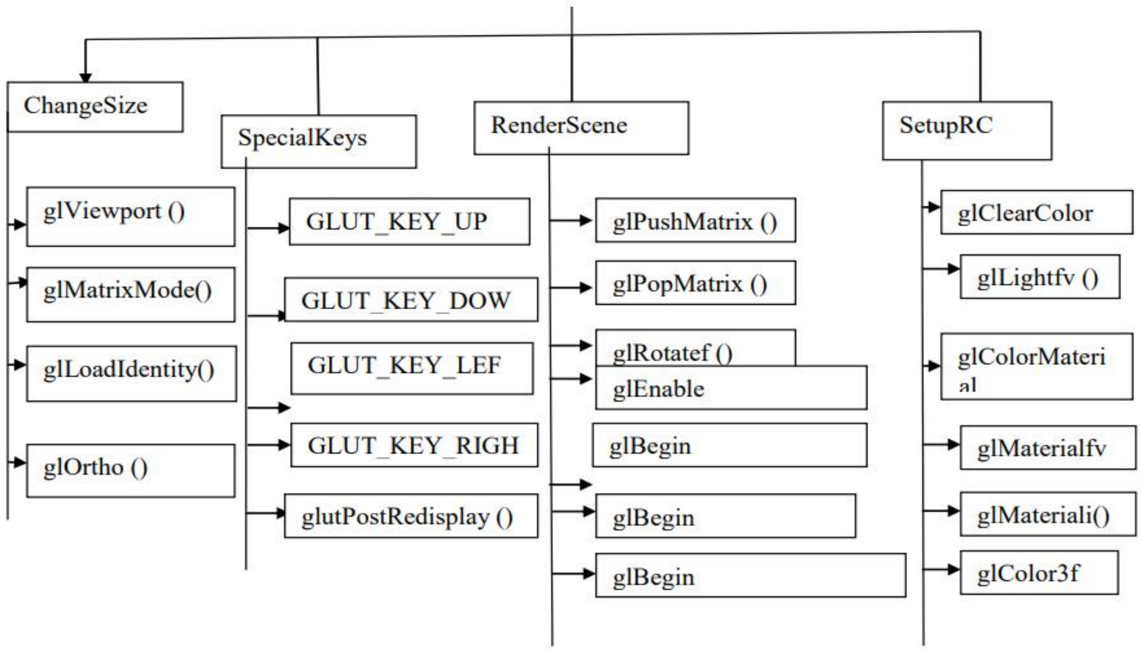


Figure 3.2.1 : Data flow Diagram

3.3 OpenGL API's Used with Description

This program is implemented using various openGL functions which are shown below

1. glClear Function

The glClear function clears buffers to preset values.SYNTAX:glClear(GLbitfield mask);

Where 'mask' is Bitwise OR operators of masks that indicate the buffers to be cleared.The masks used are as follows.

Value	Meaning
GL_COLOR_BUFFER_BIT	The buffers currently enabled for color writing.
GL_DEPTH_BUFFER_BIT	The depth buffer.

2. glMatrixMode Function

The glMatrixMode function specifies which matrix is the current matrix.

SYNTAX:

voidglMatrixMode(GLenum mode);

where the 'mode' indicates the matrix stack that is the target for subsequent matrix operations.

The mode parameter can assume values like GL_PROJECTION which applies subsequent matrix operations to the projection matrix stack or GL_MODELVIEWetc.

3. glClearColor function

Sets the present RGBA clear color used when clearing the color buffer. Variables of typeGLclampf are floating-point numbers between 0.0 and 1.0.

SYNTAX: void glClearColor(GLclampf r, GLclampf g, GLclampf b, GLclampf a);

4. glPointSize function

sets the size attribute in pixel

SYNTAX: void glPointSize(GLfloat size);

5. glFlush function

Forces any buffered OpenGL command to execute. It ensures that points rendered the screen as soon as possible.

SYNTAX: void glFlush( );

6. glutSwapBuffers function

Swaps the front and back buffers.

SYNTAX: void glutSwapBuffers( );

7. gluOrtho2D function

Defines a two dimensional viewing rectangle in the plane z=0.

SYNTAX: void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom,GLdouble top);

CHAPTER 4

IMPLEMENTATION

Implementation is the stage where all planned activities are put into action. Before the implementation of a project ,the implementors (spear headed by the project committee or executive)should identify their strength and weaknesses (internal forces), opportunities and threats(external forces)

4.1 Main Functions:

1. Transform,InitGL, Function

to initialize and manipulate the OpenGL environment

2. ReSizeGLScene Function

handle window resizing

3. glLineWidth Function

The glLineWidth function specifies the width of rasterized lines.

4. DrawGLScene Function

draw various geometric shapes and objects.

5. glutInit Function

Initializes GLUT, if arguments are passed then they are passed from the main and can be used by the application.

6. glutBitmapCharacter

glutBitmapCharacter renders a bitmap character using OpenGL**.**

7. gluNewQuadric

The gluNewQuadric function creates and returns a pointer to a new quadric object.

8. gluPerspective Functions

The gluPerspective function sets up a perspective projection matrix.

9. gluQuadricDrawStyle Function

The **gluQuadricDrawStyle** function specifies the draw style desired for quadrics.

10. gluOrtho2D function

Defines a two dimensional viewing rectangle in the plane.

11. glMatrixMode Function

The glMatrixMode function specifies which matrix is the current matrix

12. glRasterPos2f Function

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

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

4.2 Pseudo code:

//Implementation also includes a pseudo code

```
#include<stdio.h>
#include<stdlib.h>
#include<GL/glut.h>
#include<math.h>
#include<string.h>
#include<windows.h>
```

The execution of program starts from main

```
int main( argc, * argv[] )
{
        Handle following functions
        using glut call back functions
        DrawGLScene ( )
          //Display call back
        ResizeGLScene ( )
        //Reshape call back
        NormalKey ( )
        //Keyboard call back
        SpecialKeyFunc ( )
        //Special Key board call
        backCreate_menu ( )
        Start event processing
        loop.
}
```

```
Create_menu ( ) // creating menu
    {
            Create the submenu using glutCreateMenu(ColorMenu)
              Each menu entry is created using glutAddMenuEntry ()
              Create the Main menu using glutCreateMenu(myMenu)
    glutAttachMenu (GLUT_RIGHT_BUTTON).
    }
```

The function called when window is resized.

```
ResizeGLscene ( )
{
Perform Transformations
    Transform ( )
}
```

Transformation Routine.

```
    Transform ( )
{

}
```

The general OpenGL initialization function to set initial parameters.

```
    InitGL( width, Height)
{

}
```

The display function to create the Help screen.

```
    display1( )
{

}
```

The main display function handled by display call back.

```
            View=0 // to view help screen
    DrawGLScene ( )
{
    display1( )
   {
             InitGL( )
           Enable rotation by along the three axes using
    glRotatef( xangle,1.0 ,0.0 ,0.0 )
    glRotatef(yangle,0.0, 1.0, 0.0)
    glRotatef(zangle, 0.0, 0.0, 1.0)
    glBegin (GL_QUADS)
    glEnd ()
    }
    glutSwapBuffers( );
 } //end of DrawGLScene( )
```

The action taken when a Normal key is pressed.

```
    NormalKey (key, x, y)
      {
      }
```

The action taken when a special key is pressed.

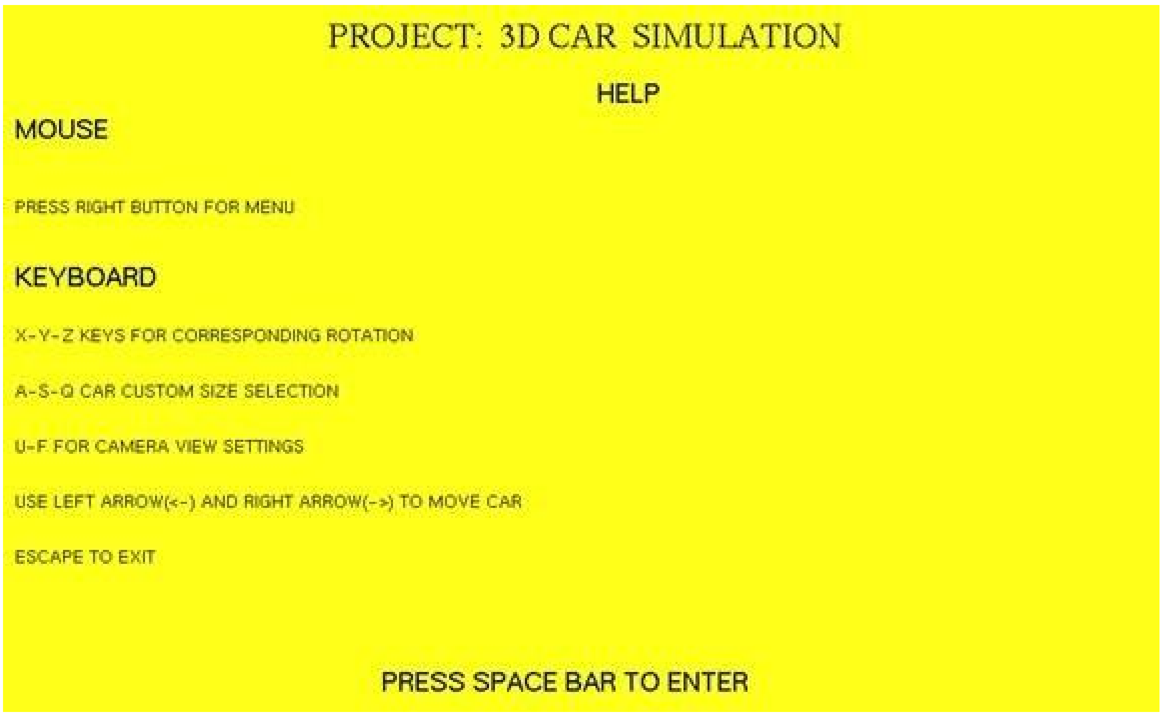
```
SpecialkeyFunc(key, x, y)
{

}
```

The actions performed by each menu entry identified by

```
id.myMenu( id )
    {
```

CHAPTER 5

RESULT & SNAPSHOTS


PROJECT: 3D CAR SIMULATION

HELP

MOUSE

PRESS RIGHT BUTTON FOR MENU

KEYBOARD

X-Y-Z KEYS FOR CORRESPONDING ROTATION

A-S-Q CAR CUSTOM SIZE SELECTION

U-F FOR CAMERA VIEW SETTINGS

USE LEFT ARROW(<-) AND RIGHT ARROW(->) TO MOVE CAR

ESCAPE TO EXIT

PRESS SPACE BAR TO ENTER

Figure 5.1 The Help Window.

The Help window displays the details about the usage of various keys and mouse button in this project. It is immediately displayed when the program begins itsexecution.

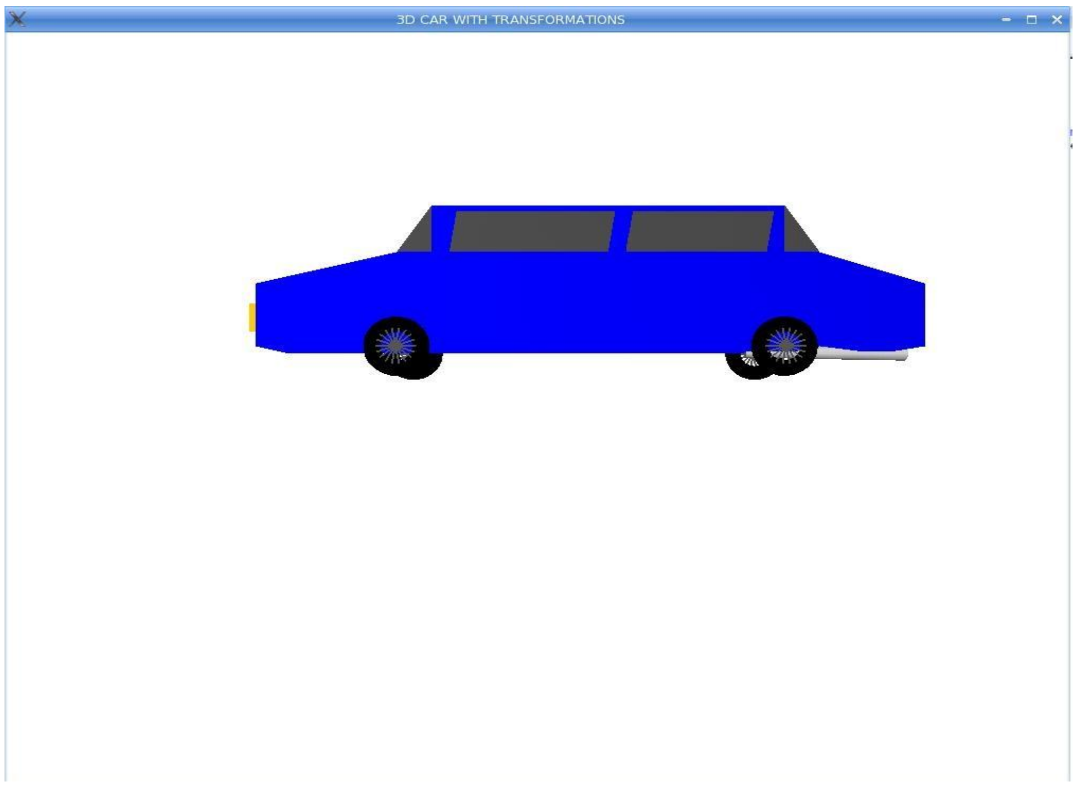


Figure 5.2 The car in model mode.

This type of display is obtained when the car model mode is chosen from the menu. By default it will be in model mode.The car can be transformed and several effectscan be added to it with the use of keyboard and mouse.

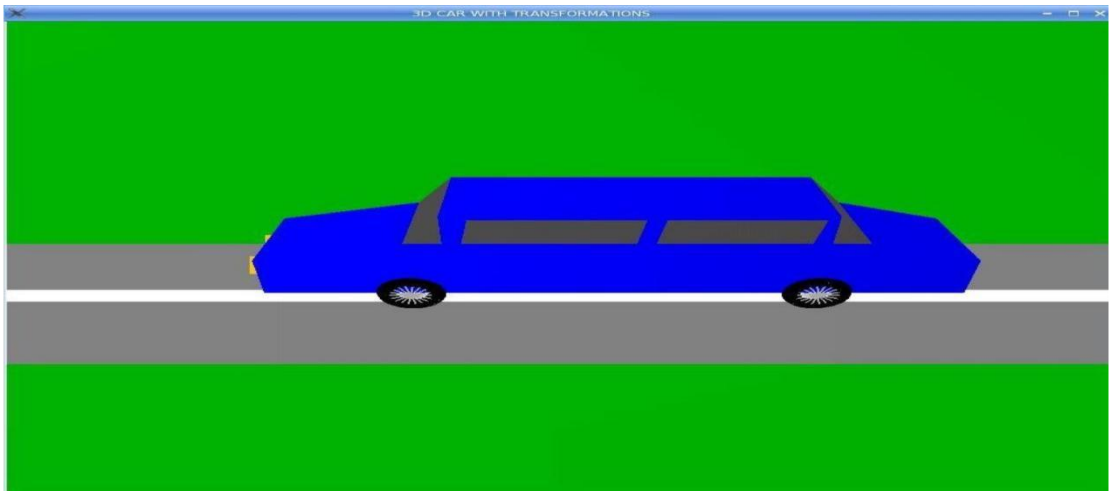


Figure 5.3 Car in driving mode.

This is the display seen when driving mode is selected from the menu. All the effects seen in car model mode can also be viewed in this mode by pressing appropriate keys and options in menu.

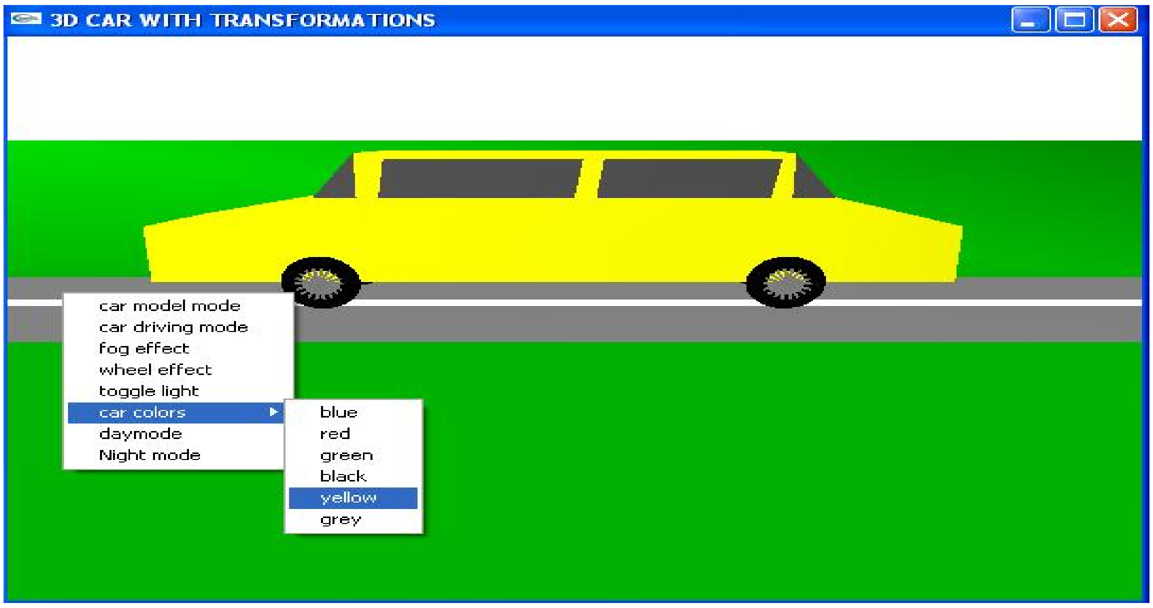


Figure 5.4 The display with menu.

This is the display obtained when the right mouse button is pressed and menus are viewed. Here the car colors menu is chosen and a submenu called yellow is chosen. Hence the car turned to yellow. Various options in the menu can be viewed in the above figure.

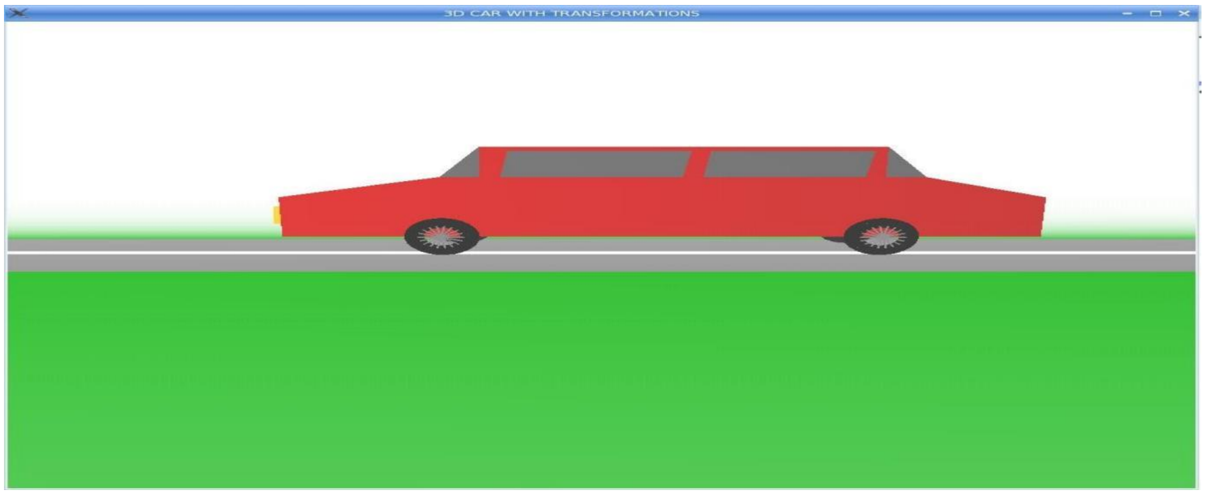


Figure 5.5 Car with fog effects.

The car is amidst white fog. This effect comes into picture when the fog effects menu is chosen.The entire screen is covered with white fog.

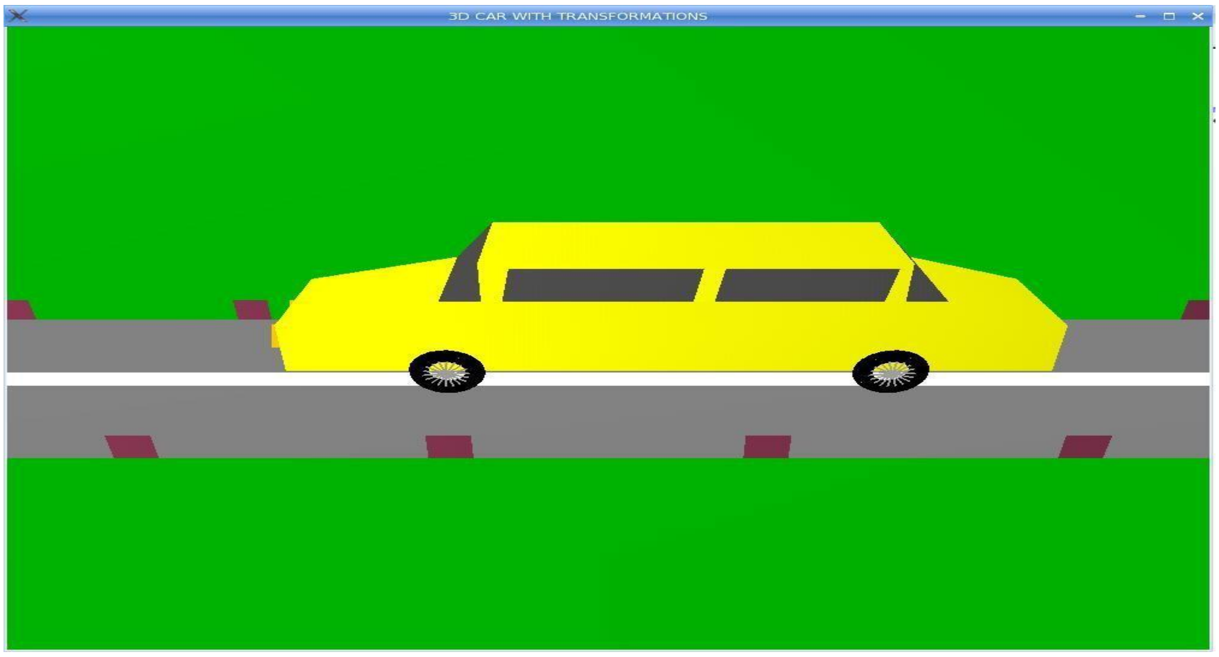


Figure 5.6 Car with wheel effects.

When the wheel effect is chosen from the menu, objects resembling milestones are added to the display. When the car is tried to move with this effect, only the wheels of the car moves and the milestones move, giving the effect of a car moving with a high speed.

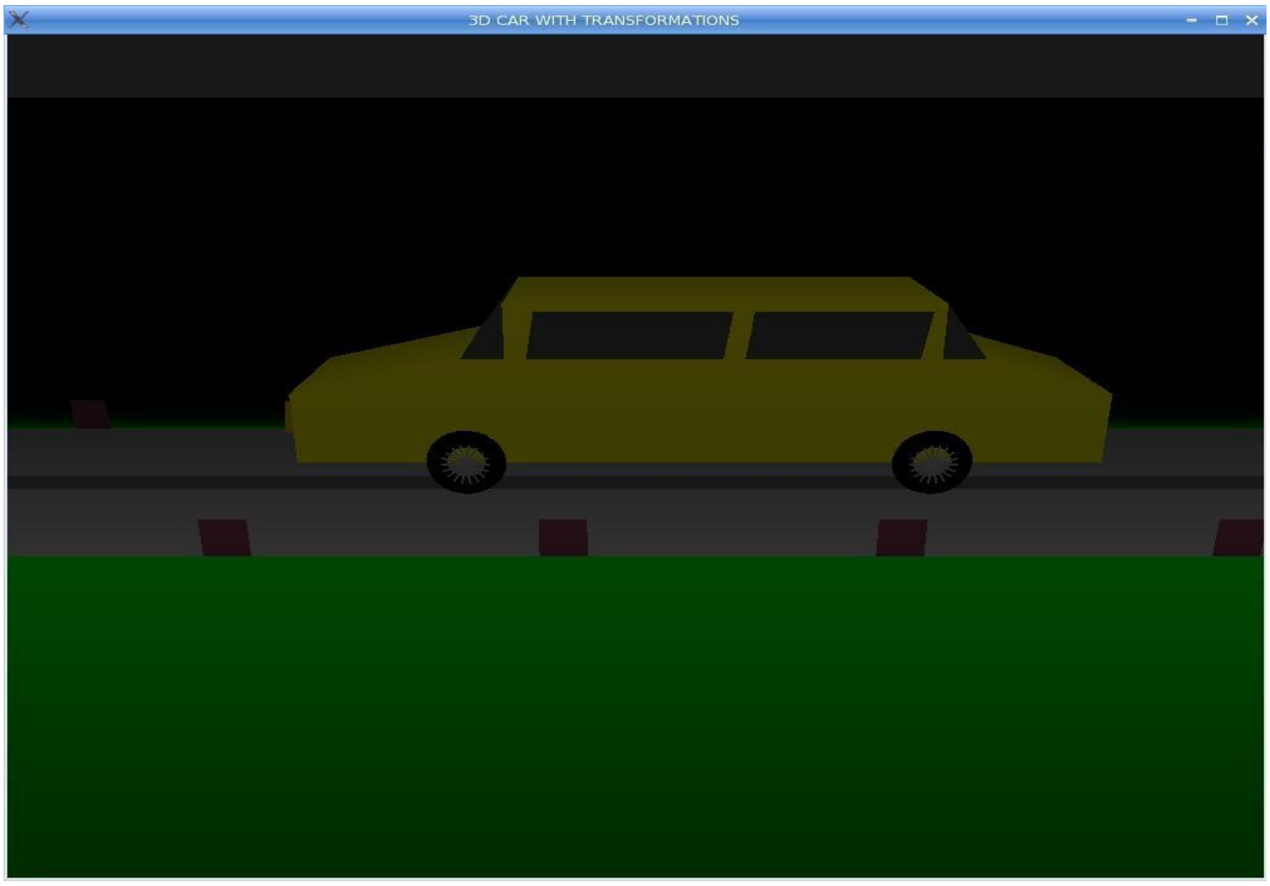


Figure 5.7 Car in the night mode.

The car is chosen to be displayed in the dawn through the menu provided. It can beswitched back to day mode.

CONCLUSION AND FUTURE ENHANCEMENTS

Conclusion

After the completion of this project, we came to know how we can implement a project using an Open-source OpenGL tool kit. By implementing a project using Open GL we came to know how to use the functions like menu ‘s, rotation, translation and scaling. Using OpenGL functions and user defined functions a 3D Car Simulation is being done which is runs at different speed and no. of cabins can also be change. This is basic graphic animation made using only some built-in function and APIs. There are many functions and APIs available in OpenGL that makes animation effective and realistic.The user-friendly interface allows the user to interact with it very effectively. So, I conclude on note that this project has given me a great exposure to the OpenGL and computer graphics. This is very reliable graphics package supporting various primitive objects like polygon, line loops, ambient light, triangle fan, quad strip etc. Also, this project is designed in such a way that one canview it from any directions using keyboard keys. Transformations like translation, rotation is alsoprovided

Future Enhancements

In future many other 3D geometrical shapes can be added to improve the epilogue of educationproject.More buttons are implemented to move the objects. For example opening and closing the doorsof cars using left and right arrows of the keyboard

REFERENCES

Reference Books:

[1] The Red Book –OpenGL Programming Guide,6th edition.

[2] Rost , Randi J. : OpenGL Shading Language, Addison-Wesley

[3] Interactive Computer Graphics-A Top-Down Approach Using OpenGL, Edward Angel, Pearson-5th edition.

Internet:

[4] www.stackoverflow.com

[5] www.opengl.org

[6] www.khronos.org

[7] www.wikipedia.org