CHAPTER 1

INTRODUCTION

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. It is the creation and manipulation of picture with the aid of computers. It is divided into two broad classes. It is also called passive graphics. Here the user has no control over the pictures produced on the screen. Interactive graphics provides extensive user-computer interaction. It provides a two-way communication between computer and user. It provides a tool called "motion dynamics" using which the user can move objects. It is also able to produce audio feedback to make the simulated environment more realistic. C language helps to implement different graphics objects which are interactive and non-interactive.

Computer Graphics is one of the most powerful and interesting facets of computers. There is a lot we can do in graphics apart from drawing figures of various shapes. All video games, animation, multimedia predominantly works using computer graphics, which make it significant. Computer graphics is concerned with all aspects of producing pictures or images using a computer.

Applications of Computer Graphics

1. Display of information
2. Design
3. Simulation and animation
4. User interfaces

The Graphics Architecture

Graphics Architecture can be made up of seven components:

1. Display processors
2. Pipeline architectures
3. The graphics pipeline
4. Vertex processing

5. Clipping and primitive assembly
6. Rasterization
7. Fragment processing

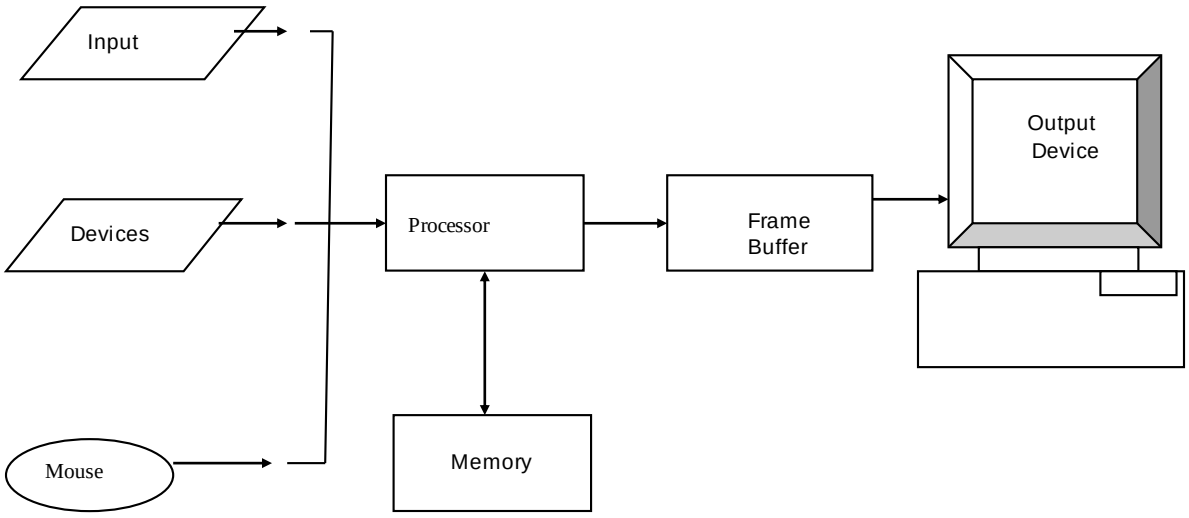

Figure 1: Components of Graphics Architecture and their working

1.1 Overview of project

This project is on simulation of 3D car .This project is done using open GL .The simulation is a menu driven simulation which provides different modes of working. There are several modes on the menu namely The driving mode, The car model mode, The fog mode, The night mode, Car color (contains sub options),Wheel movement mode. Using all these options we choose the modes of simulation. It also provides keyboard functions which are used in the navigation of the car.

1.2 Problem statement

OpenGL software consists of distinct commands which can be used to specify the object and operations needed to produce interactive three-dimensional applications.

Simulation of 3D CAR is a project developed to demonstrate different commands and primitives in OpenGL.

1.3 OpenGL

OpenGL (Open Graphics) 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 da=raw complex three-dimensional scenes for simple primitives. It is also used in video games, where it competes with Direct3Don Microsoft Windows platforms.

OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, user must work through whatever windowing system controls the particular hardware that the user is using. Similarly, OpenGL doesn't provide high-level commands for describing models of three-dimensional objects. Such commands might allow the user to specify relatively complicated shapes such as automobiles, parts of the body, airplanes, or molecules. A sophisticated library that provides these features could certainly be built on top of OpenGL.

OpenGL serves two main purposes:

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

CHAPTER 1

REQUIREMENTS ANALYSIS

The requirement analysis specifies the requirements needed to develop a graphics project.
In this phase we collect the requirements needed for designing the project. The requirements collected are then analyzed and carried to the next phase.

2.1Software Requirements

- Operating system – Windows XP/Windows 7/Ubuntu
- Code::Blocks 16.01/Microsoft Visual Studio 2012
- OPENGL library files – GL, GLU, GLUT
- Language used is C/C++

2.2Hardware Requirements

There are no rigorous restrictions on the machine configuration. The model was made to work on Windows Operating System, Code::Blocks 16.01/Microsoft Visual Studio software with the following specs

- Processor: Intel® CORE™ i5 CPU
- Memory – 4GB RAM
- 1TB Hard Disk Drive
- Mouse or other pointing device
- Keyboard: Standard
- Display device

2.3Functional Requirements

A **functional requirement** defines a function of a system and its components. A function is described as a set of inputs, the behavior, and its outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish.

2.4 Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-functional requirements define how a system is supposed to be. It essentially specifies how the system should behave and that is a constraint upon the system's behavior.

Reliability describes the ability of a system or component to function under stated conditions for a specified period of time. Reliability is defined as the probability of success as the frequency of failures.

Maintainability is the ease with which a product can be maintained in order to isolate defects, correct defects or their cause, maximize a product's useful life, and maximize efficiency, reliability and safety.

Availability is the probability that a system, at a point in time, will be operational and able to deliver the requested services. It is typically measured as a factor of its reliability – as reliability increases, so does availability.

2.5 Miscellaneous Requirements

- All the required library files and the header files should be available in the include directory.
- The library files are GL, GLU, and GLUT.

.

Chapter 3

DESIGN PHASE

3.1 Algorithm

The entire design process can be explained using a simple algorithm. The algorithm gives a detailed description of the design process of 'Projections'

Algorithm for SIMULATION OF 3D CAR is as follows:

Step 1: Start the program

Step 2: Set initial Display mode

Step 3: Set the initial Window Size to (1350, 800)

Step 4: Set the initial Window Position to (0, 0)

Step 5: Create window "3D CAR"

Step 6: Call the display function to render the Car

Step 7: Call the keyboard function

Step 8: Call the mouse function

Step 9: Exit.

3.2 Data flow diagram

The data flow diagram represents how the control is passed throughout the project.

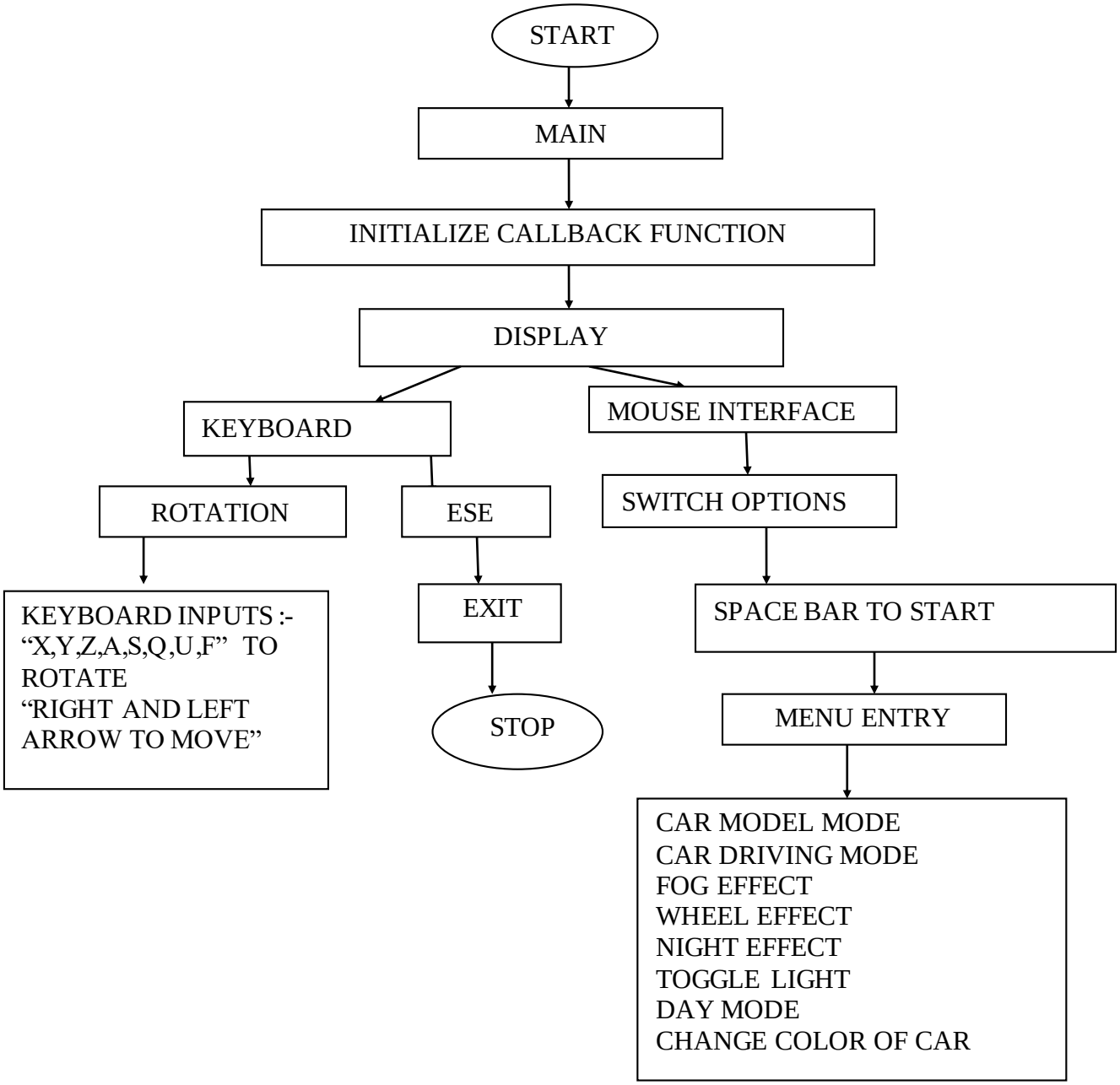

Figure 3.2: Data and Control Flow Diagram

Chapter 4

IMPLEMENTATION

The implementation stage of this model involves the following phases.

- Implementation of OpenGL built-in functions.
- User defined function implementation.

4.1 IMPLEMENTATION OF OPENGL BUILT-IN FUNCTIONS

1. glutInit()

glutInit() is used to initialize the GLUT library.

Usage: void glutInit(int argc, char **argv);

Description: glutInit will initialize the GLUT library and negotiate a session with the window system.

2. glutInitDisplayMode()

glutInitDisplayMode sets the initial display mode.

Usage: void glutInitDisplayMode(unsigned int mode);

Mode – Display mode, normally the bitwise OR-ing of GLUT display mode bit masks.

See values below:

GLUT_RGB- An alias for GLUT_RGBA.

GLUT_SINGLE- Bit mask to select a single buffered window.

GLUT_DOUBLE or GLUT_SINGLE are specified.

GLUT_DOUBLE- Bit mask to select a double buffered window. This overrides GLUT_SINGLE if it is also specified.

GLUT_DEPTH- Bit mask to select a window with a depth buffer.

Description: The initial display mode is used when creating a top-level windows, sub windows, and overlays to determine the OpenGL display mode for the to-be-created window or overlay.

3. glutCreateWindow()

glutCreateWindow creates a top-level window.

Usage: int glutCreateWindow(char *name);

Name- ASCII character string for use as window name.

Description: glutCreateWindow creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name. Implicitly, the current window is set to the newly created window. Each created window has a unique associated OpenGL context.

4. glutDisplayFunc()

glutDisplayFunc sets the display callback for the current window.

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

Func- The new display callback function.

Description: glutDisplayFunc sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the current window is set to the window needing to be redisplayed and the layer in use is set to the normal plane. The display callback is called with no parameters. The entire normal plane region should be redisplayed in response to the callback.

5. glutKeyboardFunc()

glutKeyboardFunc sets the keyboard callback for the current window.

Usage: void glutKeyboardFunc (void(*func)(unsigned char key,int x,,int y));

Func - The new keyboard callback function.

Description: glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character. The x and y callback

parameters indicate the mouse location in window relative coordinates when the key was pressed.

Passing NULL to glutKeyboardFunc disables the generation of keyboard callbacks.

6. glutMouseFunc()

glutMouseFunc sets the mouse callback for the current window.

Usage: void glutMouseFunc(void (*func)(int button, int state,int x, int y));

Description: glutMouseFunc sets the mouse callback for the current window. When a user presses and releases mouse buttons in the window, each press and each release generates a mouse callback.The button parameter is one of GLUT_LEFT_BUTTON or GLUT_RIGHT_BUTTON or GLUT_MIDDLE_BUTTON.

7. glMatrixMode()

The two most important matrices are the model-view and projection matrix. At any time, the state includes values for both of these matrices, which are initially set to identity matrices. There is only a single set of functions that can be applied to any type of matrix. Select the matrix to which the operations apply by first set in the matrix mode, a variable that is set to one type of matrix and is also part of the state.

8. glutMainLoop()

glutMainLoop enters the GLUT event processing loop.

Usage: void glutMainLoop(void);

Description: glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

9. glClear()

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

4.2 IMPLEMENTATION OF USER DEFINED FUNCTIONS

- **GLvoid Transform():**Set the viewport. Select the projection matrix. Reset the projection Matrix .Calculate the aspect ratio of the window. Switch back to the model view matrix.

- **GLvoid InitGL():**This function add the line width and performs the transformation. Create light components. Assign created components to GL_LIGHT0.

- **GLvoid ReSizeGLScene():**The function called when our window is resized and performs the sanity checks and performs the transformation.

- **void display_string():**The main drawing function In here we put all the OpenGL and calls to routines which manipulate the OpenGL state and environment. This is the function which will be called when a "redisplay " is requested.

- **GLvoid DrawGLScene():**This function Clears The Screen And sets The Depth Buffer which Defines the fog color. How dense exponential decay .Where we start fogging compute per vertex. The following sets are defined in this function OBJECT MODULE FRONT BODY: Set The Color To Blue for background MIDDLE BODY ENTER WINDOW:quad front window, quad back window, first separation, second separation,3d Separation, line strip, road and surrounding development, a green surroundings.

- **void NormalKey():**The function called whenever a "normal " key is pressed. x- Rotate the car in 'x' direction. y- Rotate the car in 'y' direction. z- Rotate the car in 'z' direction. a- Increase the size of car in'x' direction. s- Increase the size of car in 'y' direction. q- Increase the size of car in 'z' direction. u- Camera top view. f- Camera side view. Left arrow key – Move car in forward direction. Right arrow key - Move car in backward direction. Esc - Exit from the program. Spacebar - Enter the main screen from start screen.

- **static void SpecialKeyFunc():**Press right mouse button to get the mouse. Following is the menu:

Car model mode - This is default mode of car display which will display the only car.

Car driving mode - This will display the driving mode, which includes the long road and green field.

Fog effect - This will apply the fog around the environment of the car.

Wheel effect - This is one of the finest effects, it will animate the car while it moves

Toggle light - This will apply the light effect on/off when selected.

Car colors - This menu has a submenu which allow us to select the color of car. The submenu has the following options - blue, red, green, black, yellow and grey.

Day mode - By default we have Day mode on in this project. So while in Night mode you can select this to toggle to Daylight mode.

Night mode - This menu will let you switch to the Night mode, by showing darkness around.

- **void display1(void):** In this function first we should print the instructions that would be displayed on the pop-up window.

- **void colorMenu(int id):** GLUT supports pop-up menus. A menu can have sub menus. This function defines a sub-menu to choose between different colors.

- **void myreshape(int w,int h):** The reshape function is a call back function which is called whenever size or shape of the application window changes. Reshape function takes 2 arguments; they are width and height of reshaped window. Mainly these parameters are used to set a new viewport.

- **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.

Chapter 5

TESTING AND SNAPSHOTS

5.1 Testing

Testing is the process of executing a program to find errors. A good test has the high probability of finding a yet undiscovered error. A test is vital to the success of the system. System test makes a logical assumption that if all parts of the system are correct, then the goal will be successfully achieved.

5.2 Types of Testing

- Unit Testing
- Integration Testing
- System Testing

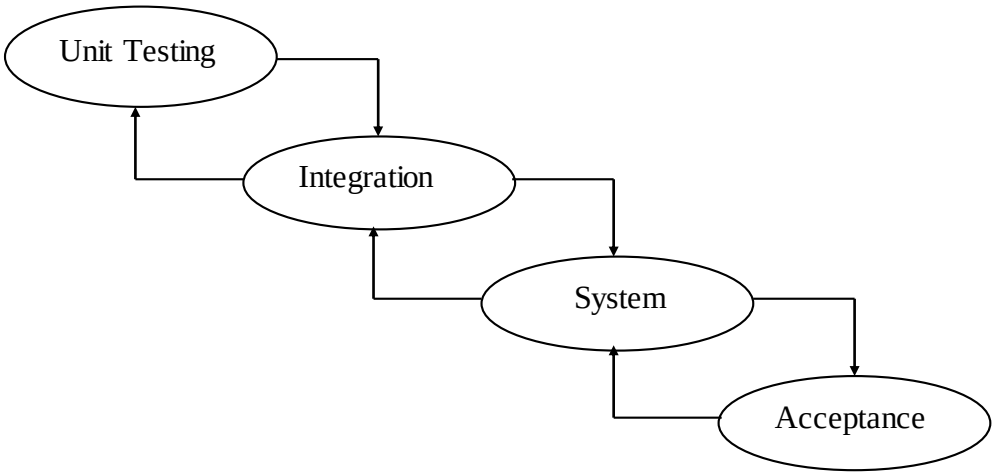

Fig 5.2 : Testing Process

5.2.1 Unit Testing

Individual components are tested to ensure that they operate correctly. Each component is tested independently, without other system components.

5.2.2 Integration Testing

All the patterns are called together and we add keyboard functions and check whether it is executing successfully, check whether the keyboard input can be taken successfully and menu functions are displayed correctly.

Black box testing takes an external perspective of the test object to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid inputs and determines the correct output. There is no knowledge of the test object's internal structure.

This method of test design is applicable to all levels of software testing: unit, integration, functional, system and acceptance testing. The higher the level, and hence the bigger and more complex the box, the more one is forced to use black box testing to simplify. While this method can uncover unimplemented parts of the specification, one cannot be sure that all existent paths are tested.

Bottom-Up Testing is an approach to integration testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested.

This project has been tested for its working and is found the requirements as mentioned.

5.2.3 System Testing

The testing process is concerned with finding errors, which result from unanticipated interactions between the different components. It is also concerned with validating that the system meets its functional and non-functional requirements.

5.2.4 Acceptance Testing

Acceptance testing, a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it is has met the required criteria for delivery to end users.

5.3 Snapshots

In computer file systems, a snapshot is a copy of set of files and directories as they were at a particular point in the past. The term was coined as an analogy to that in photography.

One approach to safely backing up live data is to temporarily disable write access to data during the backup, either by topping the accessing applications or by using the locking API provided by the operating system to enforce exclusive read access. This is tolerable for low-availability systems (on desktop computers and small workgroup servers, on which regular downtime is acceptable). High availability 24/7 systems, however, cannot bear service stoppages.

To avoid downtime, high-availability systems may instead perform the backup on a snapshot – read-only copies of data set frozen at a point in time – and allow applications to continue writing to their data. Most snapshot implementation systems are efficient and can create snapshots in O (1). In other words, the time and I/O needed to create the snapshot does not increase with the size of the data set, whereas the same for a direct backup is proportional to the size of the data set.

Snapshots are the pictures representing different phases of the program execution.

The below given figure is the initial screen which displays the name of the institution, subject name, project title and the name of the student along with the USN.

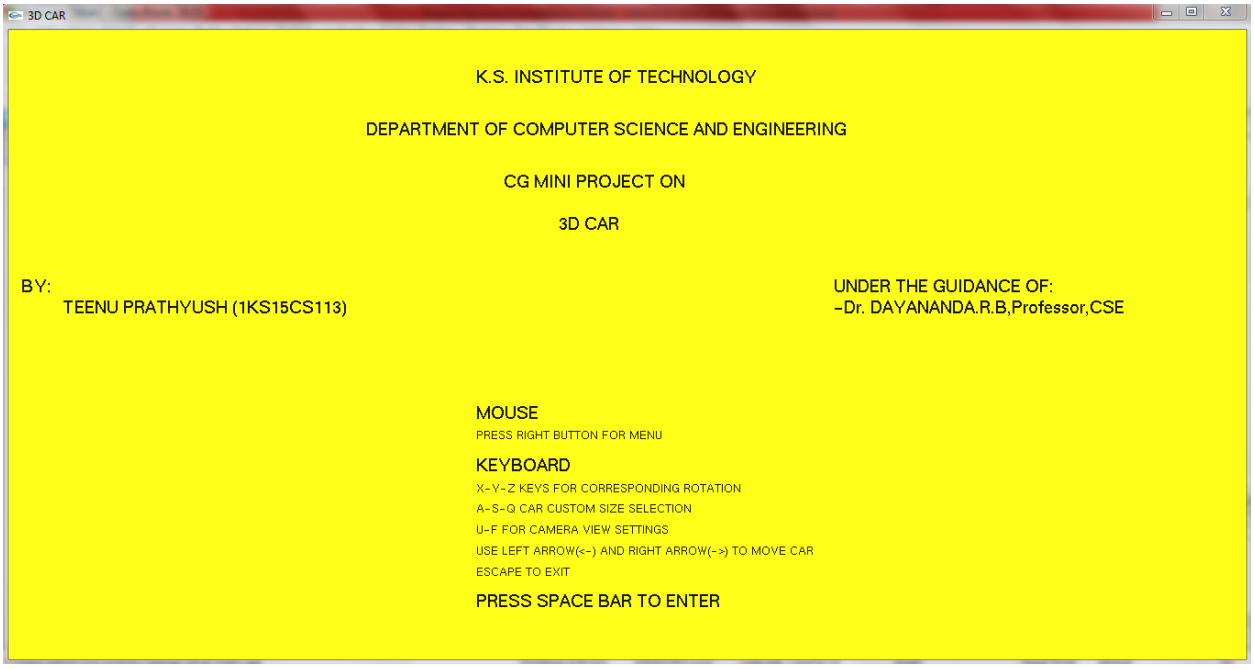

Figure 5.3.1: Information screen

The following figure is the screen shot of the display after selecting Car model view. This can be obtained by pressing the right mouse button and selecting the car model view option from the menu.

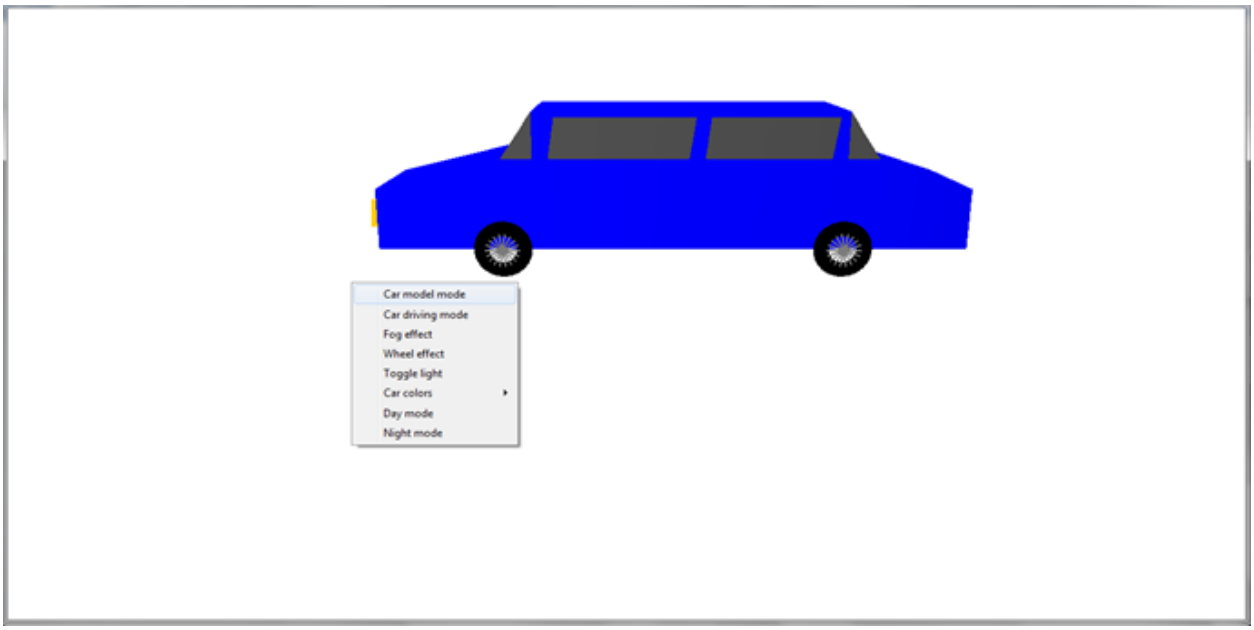

Figure 5.3.2: Snapshot showing car model mode view.

On click of car driving mode option from the menu, It gives an animated view of car driving.

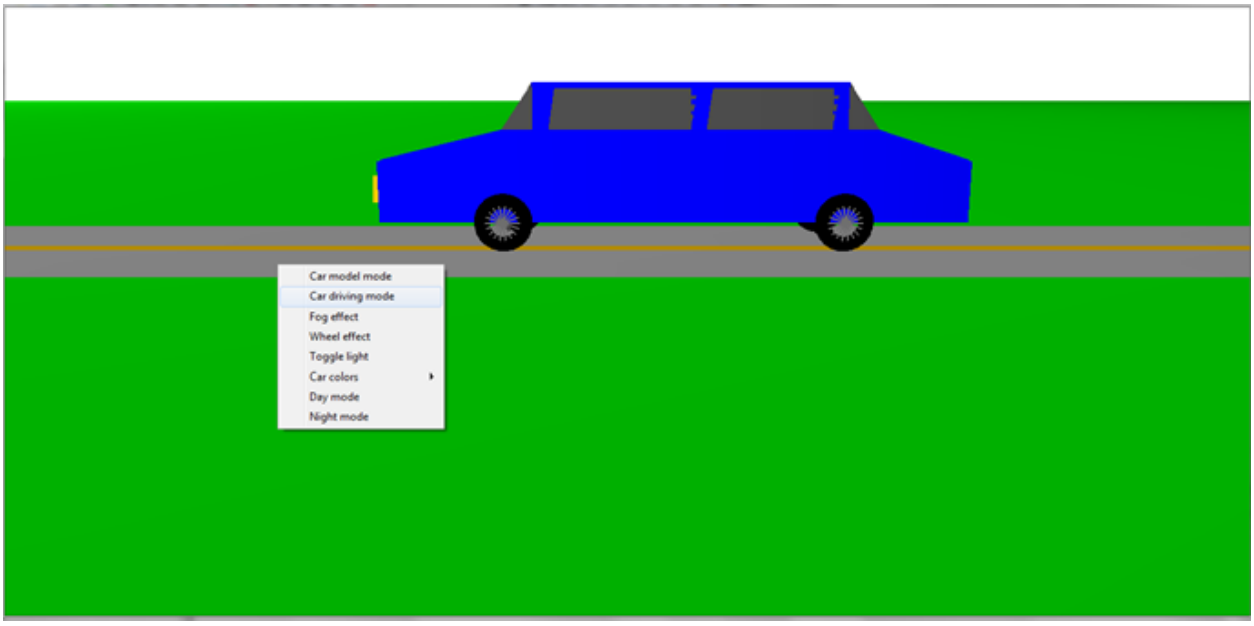

Figure 5.3.3: Snapshot showing car driving mode view.

On click of wheel effect option from the menu, It gives an animated example of wheel effect.

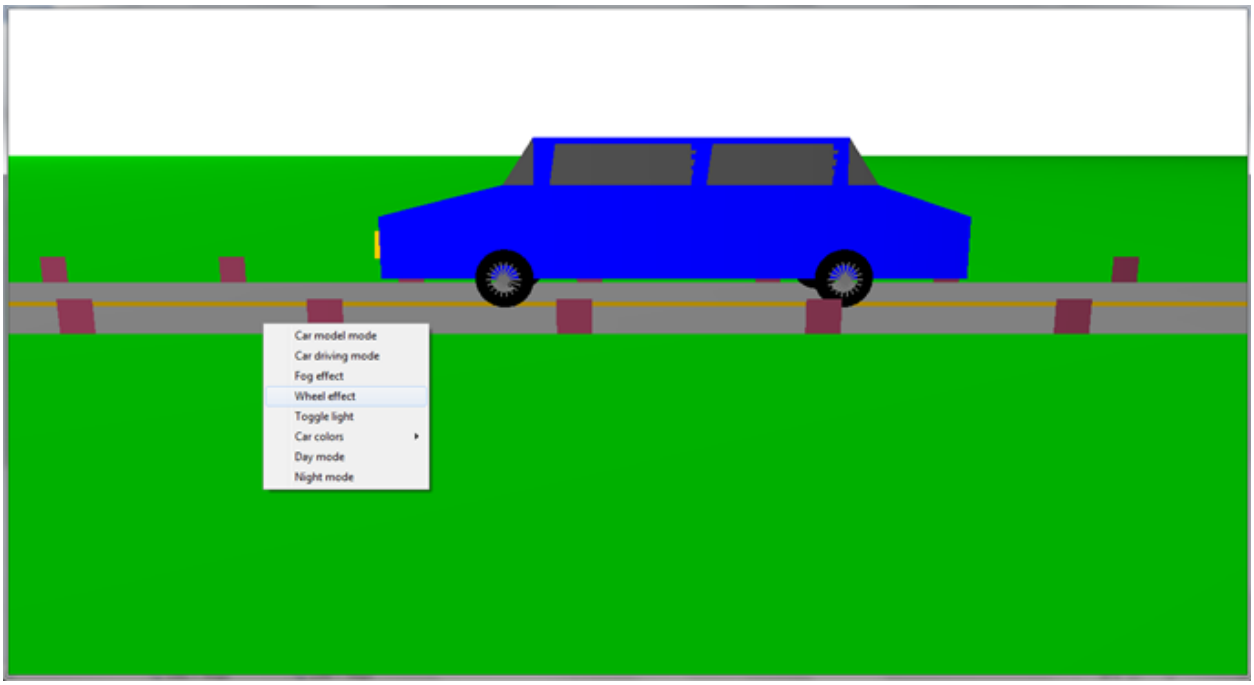

Figure 5.3.4: Snapshot showing wheel effect view

On click of fog effect option from the menu, It gives an animated view of fog around the car.

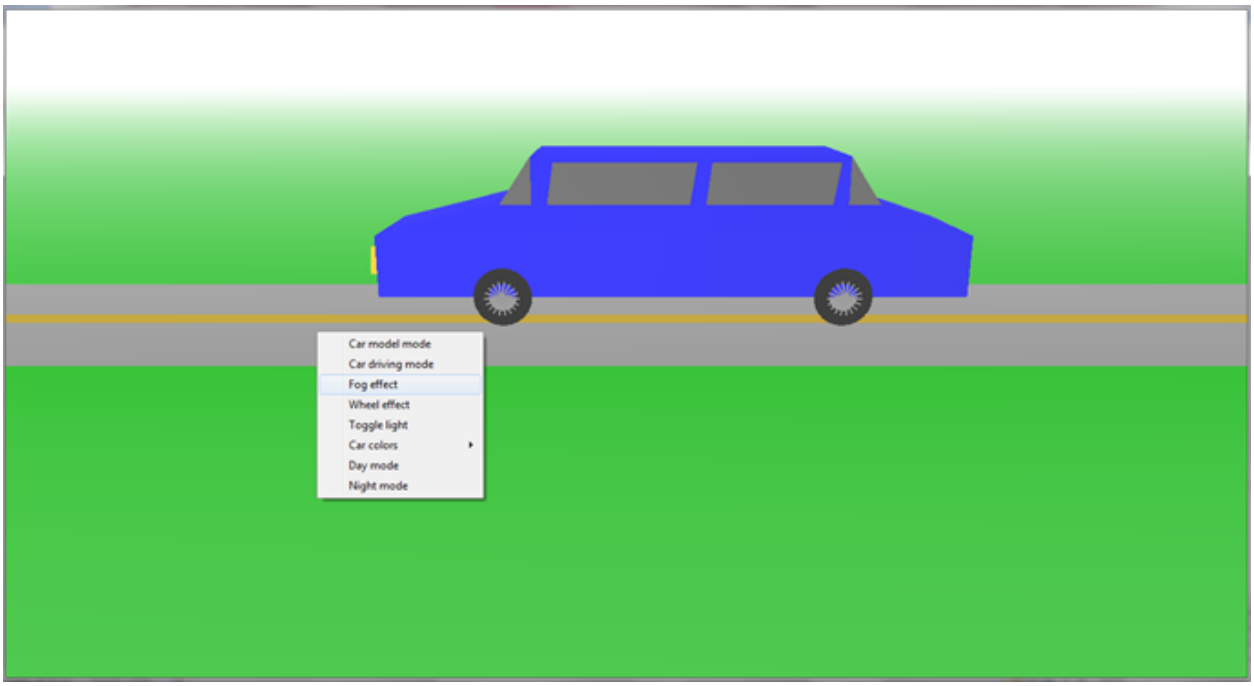

Figure 5.3.5: Snapshot showing fog effect view.

The car colors option has a sub – menu from which we can change the color of the car.

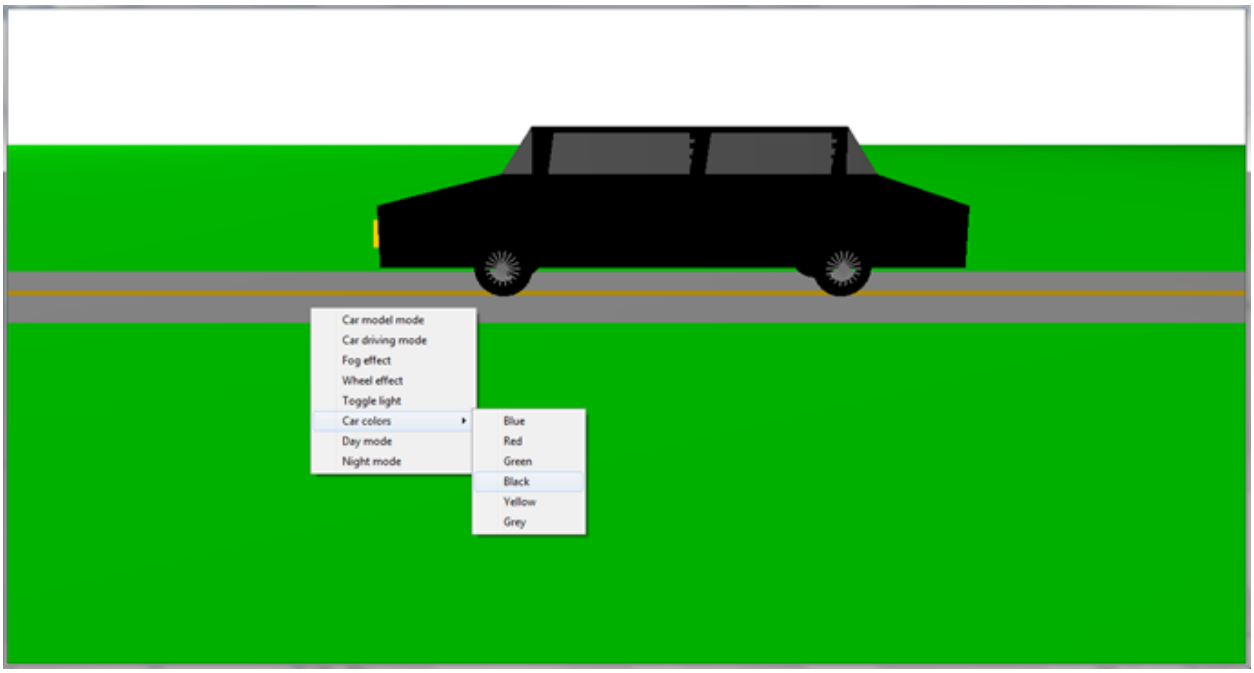

Figure 5.3.6: Options to change color of car

On click of night mode option from the menu, It shows an animated view of Night effect.

Figure 5.3.7: Snapshot showing night effect view.

The car can be rotated using X, Y, Z keys. The car size can be changed using A, S, Q keys. The keys U, F are used for camera view settings.

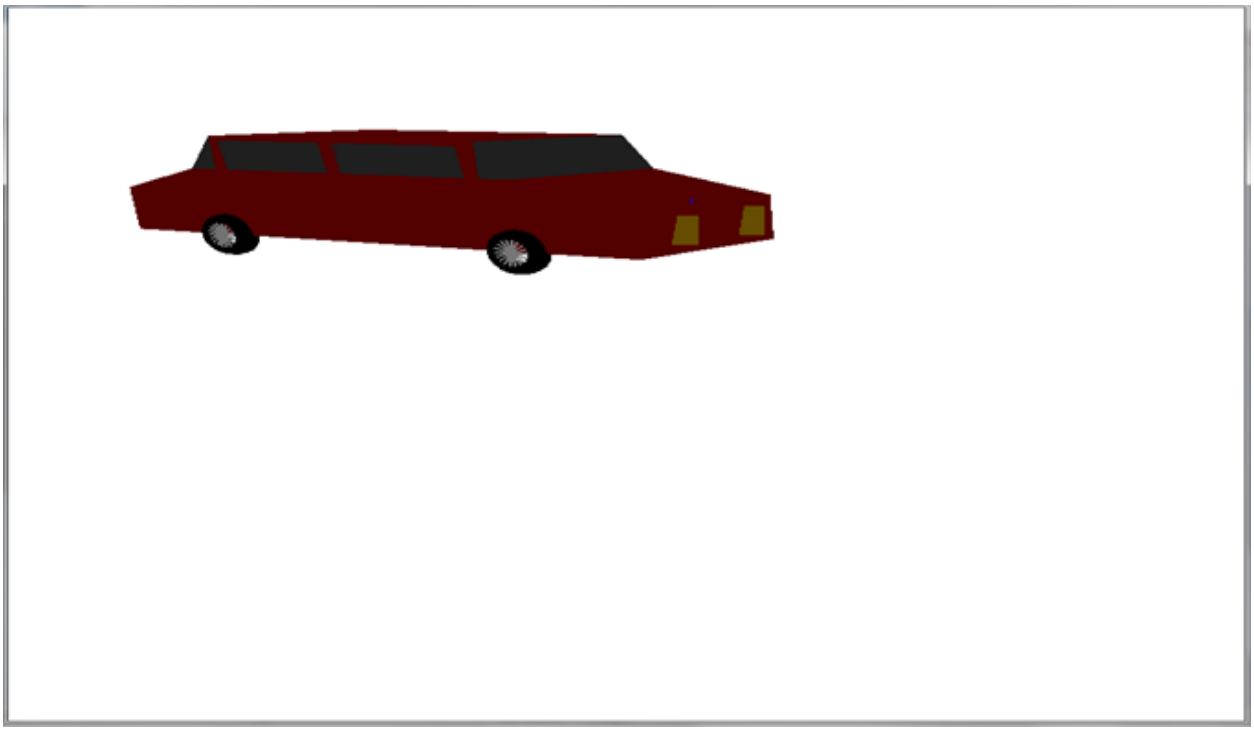

Figure 5.3.8: Snapshot showing rotation, custom car size selection, and camera settings.

CONCLUSION

This project is done using Open GL to show the simulation of a 3D CAR which is designed to move, rotate and change color of the car by using either a mouse interface or keyboard interface This project also different modes in which the car will simulate .The arrow keys are used for the movement of the car by selecting the driving mode in the menu .Thus this project allows to show the simulation of a 3D car in different modes defined by GL functions. The development of this project was very helpful to develop the programming skills and to know OpenGL better. Designing and developing this project were interesting and a good learning experience.

FUTURE ENHANCEMENT

The currently developed project can improve in many areas. Time constraint and deficient knowledge has been a major factor in limiting the features offered by this display model. The following features were thought, but not implemented.

- Adding some more suitable texture in the background.
- Adding virtual images.
- More interactive feature addition
- Hardware interface inclusion.

REFERENCES

Books

[1] Edward Angel:

Interactive Computer Graphics: A Top-Down Approach with OpenGL

[2] Shalini Govil-Pai:

Principles of Computer Graphics: Theory and Practice Using OpenGL

Websites

[3] http://www.daniweb.com/software-development/cpp/threads/195853/undefined-reference-to-opengl-functions

[4] http://www.khronos.org/opengles/sdk/docs/man

[5] http://www.opengl.org/sdk/docs/man4/xhtml/glLinkProgram.xml