VISVESVARAYA TECHNOLOGICAL UNIVERSITY BELGAUM

PROJECT REPORT
on
"AMUSEMENT PARK"

Submitted in partial fulfilment of the requirements for the award of

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE & ENGINEERING

Name	USN
PAWAN BHARADWAJ N P	4VP18CS059
SAMPATH KUMAR P L	4VP18CS074

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VIVEKANANDA COLLEGE OF ENGINEERING & TECHNOLOGY

NEHRU NAGARA, PUTTUR (D.K.) - 574 203

2020-21

VIVEKANANDA COLLEGE OF ENGINEERING & TECHNOLOGY

NEHRU NAGARA, PUTTUR-574 203

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

Certified that the project work entitled 'AMUSEMENT PARK' carried out by SAMPATH KUMAR P L, **4VP18CS074**, a Bonafede student of **VIVEKANANDA COLLEGE OF ENGINEERING AND TECHNOLOGY, in partial fulfillment for the award of Bachelor of** Engineering in **COMPUTER SCIENCE & ENGINEERING** of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**, Belgaum during the year 2020-21. The project has been approved as it satisfies the academic requirements in respect of Project work prescribed for the Bachelor of Engineering Degree

Prof. Shwetha C H	Prof. Krishna Mohana A J	Prof. Krishna Mohana A J
(Project Guide)	(Project Guide)	(Head of the Department)

Name of the Examiners: **Signature with date**

1. _____ _____

2. _____ _____

ACKNOWLEDGEMENT

We wish to express our sincere gratitude to our guide **Prof.Shwetha C H**, **Prof. Krishna Mohana A J**, Assistant Professors of CSE Department, for their valuable guidance and suggestions.

We are grateful to **Prof. Krishna Mohana A J** Head of the Department, CSE Department for his support and encouragement.

We are indebted to our beloved Principal **Dr. Mahesh Prasanna K** and the management of V.C.E.T, Puttur, for having provided all facilities and helped us in completing our project work.

We thank all teaching and non-teaching staff of Department of Computer Science and Engineering for their support and help.

DECLARATION

We hereby declare that, the project entitled **"AMUSEMENT PARK"** is executed only by us, under the guidance of **Prof. Shwetha C H** and **Prof. Krishna MohanaA J,** Asst. Professors, Department of Computer Science and Engineering, **Vivekananda College of Engineering and Technology**, for the partial fulfillment of the requirements for the award of the Bachelor degree in Computer Science and Engineering prescribed by **Visvesvaraya Technological University,** Belagavi.

4VP18CS059 PAWAN BHARADWAJ N P

4VP18CS074 SAMPATH KUMAR P L

ABSTRACT

An Amusement Park or theme park is a group of entertainment attractions, rides, and garden in a location for the enjoyment of large numbers of people.

The aim of this CG project is to create a 3-D/VIRTUAL CAR AMUSEMENT PARK. The viewer is allowed to roam around in the Track and see the trees closely and to drive a car. we can view the movement of the car in helicopter, side, back, front view. These views can be chosen via menu bar defined in the program.

Here we use predefined functions which perform the viewing transformations. all these functions are defined in our code which are triggered when a specified keyboard keys are pressed.

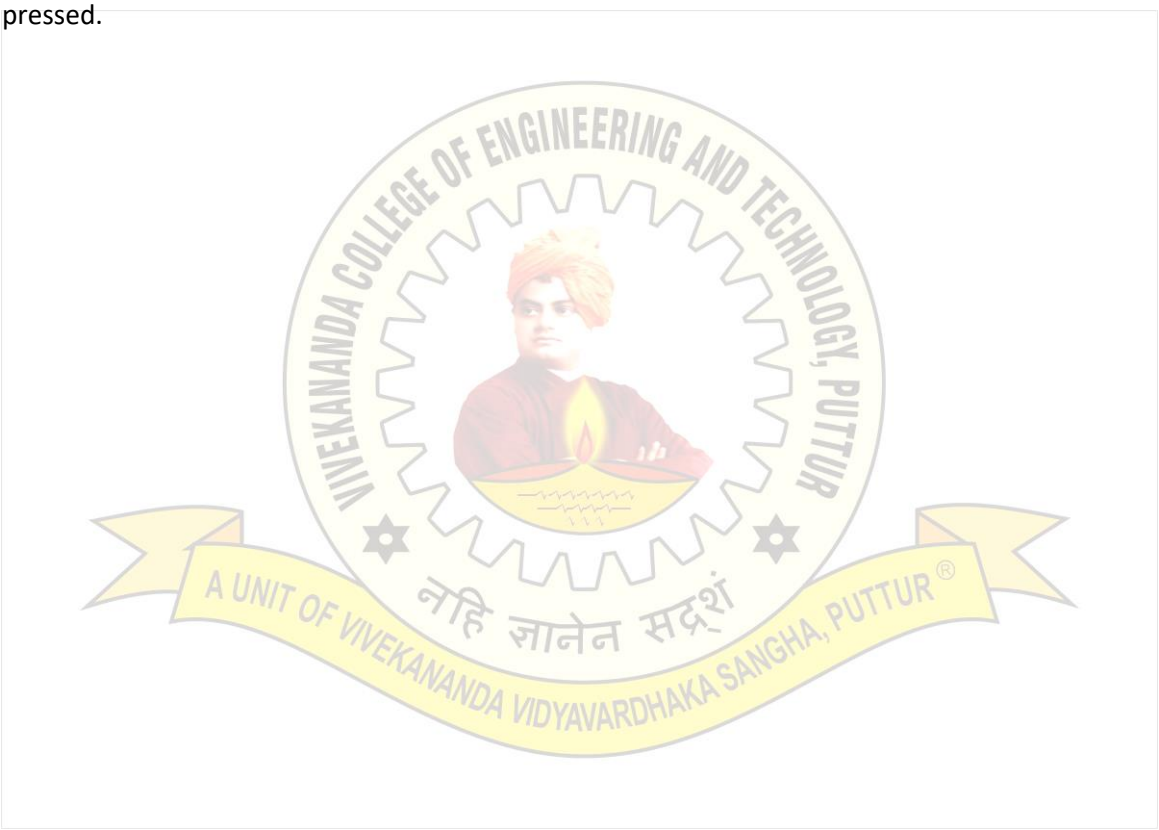


Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

1.1 About Computer Graphics

Computer Graphics in today's world is one of the most widely used powerful and interesting features of the computer which gives us the power to handle the graphical data very efficiently and also process them rapidly and effectively. It is concerned with all the aspects of producing pictures or images using a computer.

Computer graphics started with a display of data on hard copy plotters and cathode ray tube screens soon after the introduction of computers themselves. The development of computer graphics has been driven both by the needs of the user community and by advances in hardware and software. Computer graphics today is largely interactive. The user controls the contents, structures and appearances of the object and of their displayed images by using input devices, such as keyboard, mouse or touch-screen. Due to the close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics.

1.2 About OpenGL

OpenGL is a software interface to graphics hardware. OpenGL is designed to work efficiently even if the computer that displays the graphics you create isn't the computer that runs your graphics program. OpenGL is designed as a streamlined, hardware- independent interface to be implemented on many different hardware platforms. OpenGL doesn't provide high-level commands for describing models of three-dimensional objects.

1.3 OpenGL Architecture

. There are two pipelines of data flow. The upper pipeline is for geometric, vertex-based primitives. The lower pipeline is for pixel-based, image primitives. Texturing combines the two types of primitives together. There are several APIs related to OpenGL:

- AGL, WGL, GLX : OpenGL & Windowing Systems Interfaces
- GLU : 2D & 3D Geometry, Tessellations, etc.
- GLUT : Portable Windowing API.

1.4 OpenGL in our Project

In our project, we use a particular graphics software system, OpenGL. The applications are designed to access OpenGL directly through functions in three libraries. The first is the GL library in which the function name begins with gl. The second is the OpenGL utility library (GLU) which uses only GL functions but contains code for creating common objects and simplify viewing and these functions that begin with GLU. To interface with the windows system and to get input from the external devices into our programs we use the third library, the OpenGL Utility Toolkit (GLUT) whose functions are prefixed as glut.

1.5 OpenGL Functions

1.5.1 GLUT Callbacks

OpenGL has a wide variety of functions in GL, GLU, GLUT, and GLE.GLUT uses a callback mechanism to do its event processing. Callbacks simplify event processing for the application developer. Given below are some of the functions used in this project: glutDisplayFunc()- called when pixels in the window need to be refreshed. glutMouseFunc()- called when the user presses a mouse button on the mouse. glutKeyboardFunc() - called when user enters a value (input) though Keyboard. glutIdleFunc() - called when nothing else is going on. glutReshapeFunc()- sets the reshape callback for the current window.

1.5.2. OpenGL Command

The OpenGL API calls are designed to accept almost any basic data type, which is reflected in the calls name. Knowing how the calls are structured makes it easy to determine which call should be used for a particular data format and size. For instance, vertices from most commercial models are stored as three component floating point vectors. As such, the appropriate OpenGL command to use is glVertex3fv (co-ordinates).
For glVertex*() calls which don't specify all the coordinates i.e., glvertex2f (), OpenGL will take default z=0.0 and w=1.0.

1.5.3 material functions

The glMaterialfv function assigns values to material parameters. There are two matched sets of material parameters. One, the front-facing set, is used to shade points, lines, bitmaps, and all polygons (when two-sided lighting is disabled), or just front-facing polygons (when two-sided lighting is enabled). Syntax: -glMaterialfv(GL_FRONT, GL_AMBIENT, ambient);

CHAPTER 2

System Requirements

The package is designed such that users with a computer having minimum configuration can also use it, which does not require complex graphics packages.

The package requires simple in-built functions found in the header file along with a few users defined functions.

Software Requirements

- Windows/Mac Operating System
- MICROSOFT VISUAL C++
- OpenGL

Hardware Requirements

Processor	-	Intel Pentium onwards Compatible Hardware
RAM	-	256Mb RAM (minimum)
Hard Disk	-	3 GB (minimum)
Monitor	-	VGA Compatible
Keyboard	-	Standard 101 keys Keyboard

CHAPTER 3

Design

we have created a circular track using a circle and also by defining the vertices for the chassis of a car we have created a car model. Inside the car we have placed a character representing the driver of the car. We have created spherical and cone shaped trees and have placed those around the circular track. We have implemented different views by using glLookat functions. We have helicopter view, side, front and back views in this project.

The project is mainly divided into two transformations. They are as follows:

1. Translation
2. Rotation

- **Translation:**

Translation is done by adding the required amount of translation quantities to each of the points of the objects in the selected area. If P(x,y) be the a point and (tx,ty) translation quantities then the translated point is given by glTranlate(dx,dy,dz);

void glTranslate[fd](TYPE x,TYPEy,TYPE z);

Rotation:

The rotation of an object by an angle 'a' is accomplished by rotating each of the points of the object. The rotated points can be obtained using the OpenGL functions glRotate (angle, vx,vy,vz);

void glRotate[fd](TYPE angle,TYPEdx,TYPEdy,TYPEdz);

CHAPTER 4

Implementation

4.1 OpenGL In-built Functions

4.1.1 glClearColor

This function call sets the present RGBA clear color used when clearing the color buffer.

Variables of type GLclampf are floating type number between 0.0 and 1.0.

General syntax:

void glClearColor(GLclampfR,GLclampfG,GLclampfB,GLclampfA);

Eg: glClearColor(1.0,0.0,0.0,1.0);

4.1.2 glutInit

This function call initializes glut library.

The arguments from main are passed in and can be used by an application.

General Syntax:

Void glutInit(int*argc,char **argv);

Eg: glutInit(&argc,argv);

4.1.3 glutCreateWindow

This function call creates a window on the display. The string title can be used to label the window. General Syntax:

void glutCreateWindow(char *title);

Eg: glutCreateWindow("2D and 3D Objects Rotation");

4.1.4 glutInitWindowSize

This function call specifies the initial height and width of window in pixels.

General syntax:

Void glutInitWindowSize(intwidth,int height);

Eg: glutInitWindowSize(500,500);

4.1.5 glutInitWindowPosition

This function call specifies the initial position of top left corner of the window in pixels.

General syntax:

void glutInitWindowPosition(intx,int y);

Eg: glutInitWindowPosition(100,100);

4.1.6 glutDisplayFunc

This function call registers the display function *func i.e executed when the window needs to be redrawn.

General syntax: **void glutDisplayFunc(void (*func) (void))**

Eg: glutDisplayFunc(display);

4.1.7 glutMainLoop

This function call causes the program to enter an event processing loop.It should be the last statement in main.

General Syntax:

void glutMainLoop();

Eg: glutMainLoop();

4.1.8 glutMouseFunc

The function call registers the mouse call back function *f. The call back function returns the function (GL_LEFT_BUTTON, GLUT_MIDDLE_BUTTON,

GLUT_RIGHT_BUTTON), the state of the button after the event will be either up or down (GLUT_UP,GLUT_DOWN) and position of the mouse relative to top left corner of the window.
General syntax: **void glutMouseFunc(void*f(intbutton,intstate,intx,int y);**

Eg: glutMouseFunc(mousefunc);

4.1.9 glutSwapBuffers

This function call swaps the front and back buffers.

General syntax:

void glutSwapBuffers();

Eg: glutSwapBuffers();

4.1.10 glClear

This function is used to clear the window. As the algorithm stores information in the depth buffer, it is necessary to clear the buffer whenever we wish to redraw the display.

Eg: glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

4.1.11 glColor

This function will set the current drawing color.

Eg: glColor3f(1.0,0.0,0.0);

This function will set the current drawing color to red. It conveys that we are using three-color (RGB) model and that the values of the components are given as float.

4.1.12 glMatrixMode

This function call specifies which matrix will be affected by subsequent transformation.

General syntax:

void glMatrixMode(Glenum mode);

Eg:

glMatrixMode(GL_PROJECTION);

4.1.13 glLoadIdentity

This function call sets current transformation matrix to an identity matrix.
General syntax:

void glLoadIdentiy();

Eg: glLoadIdentity();

4.1.14 glutPostRedisplay

. `It` marks the *current window* as needing to be redisplayed.

General syntax:

void glutPostRedisplay(void);

4.1.15 glMaterialfv

The glLightfv function sets the value or values of individual light source parameters. The light parameter names the light and is a symbolic name of the form GL_LIGHTi, where 0 = i< GL_MAX_LIGHTS. The pname parameter specifies one of the light source parameters, again by symbolic name

4.1.16 glLightModel

glLightModel sets the lighting model parameter. pname names a parameter and params gives the new value. There are three lighting model parameters: GL_LIGHT_MODEL_AMBIENT. params contains four integer or floating-point values that specify the ambient RGBA intensity of the entire scene

4.1.17 gluPerspective

gluPerspective specifies a viewing frustum into the world coordinate system. In general, the aspect ratio in gluPerspective should match the aspect ratio of the associated viewport

4.2 Standard header and library functions

4.2.1 GL/glut.h

Library utility in OpenGL program.

4.2.2 stdio.h

This header file is used for standard input and output and stream manipulation function.

4.2.3 stdlib.h

stdlib.h is the header of the general-purpose standard library of C programming language which include functions involving memory allocation, process control, conversions and others.

CHAPTER 5

Result

5.1 Screenshots

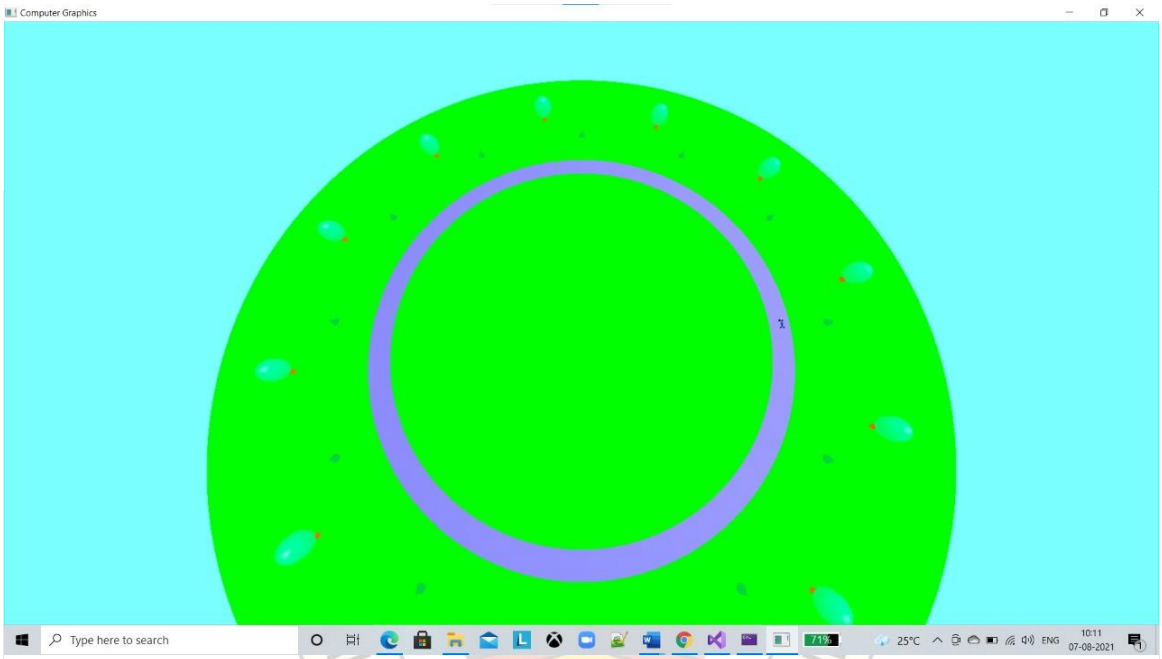


Fig.5.1.1 HELICOPTER VIEW

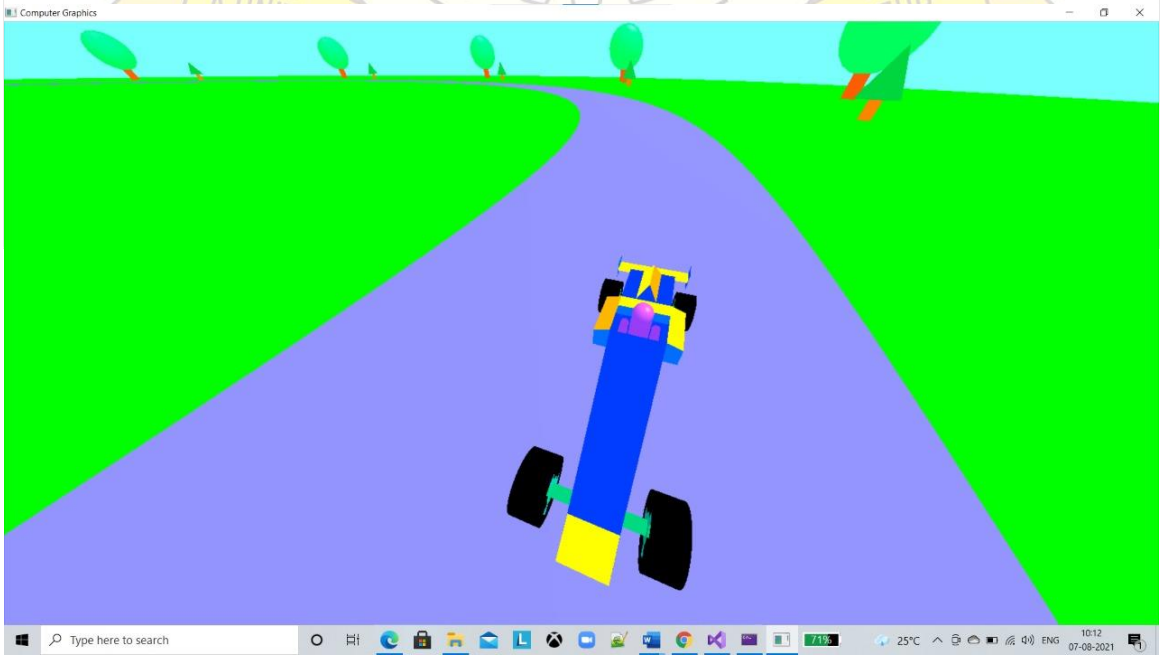


Fig.5.1.2 FRONT VEW

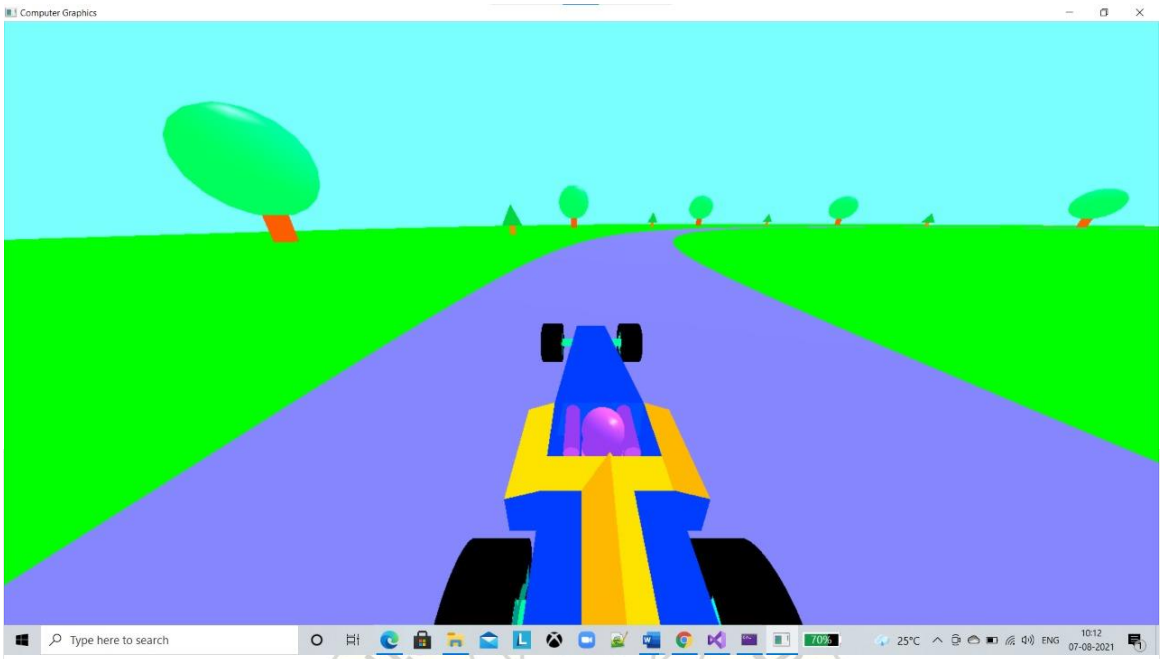


Fig.5.1.3 BACK VIEW

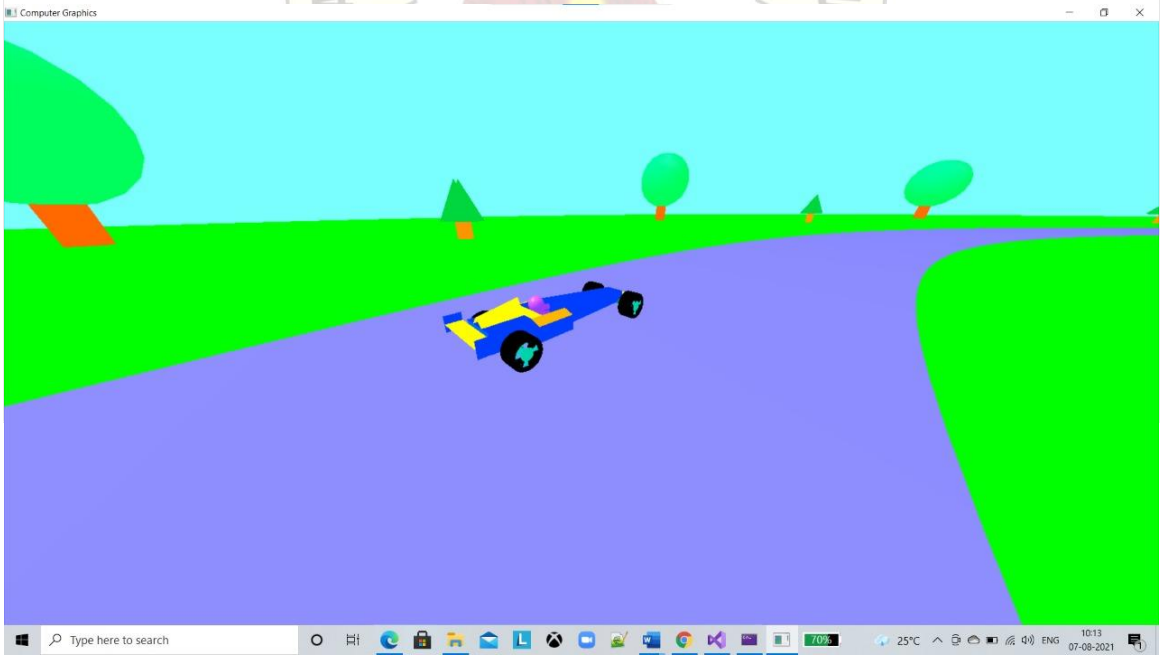


Fig.5.1.4 SIDE VIEW

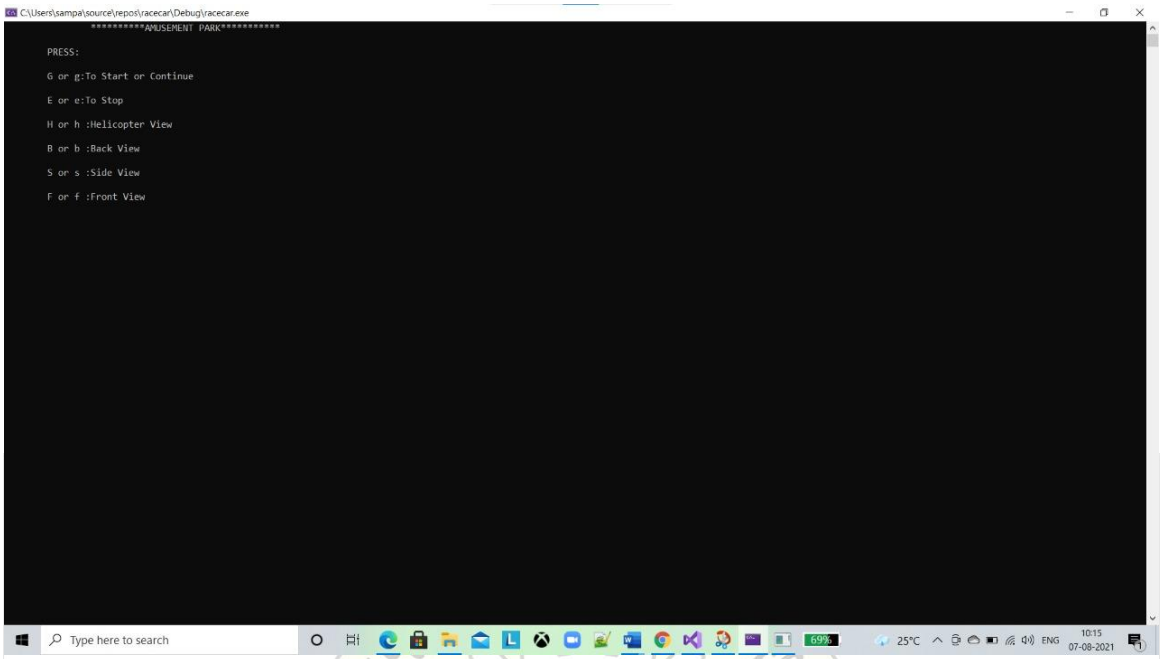
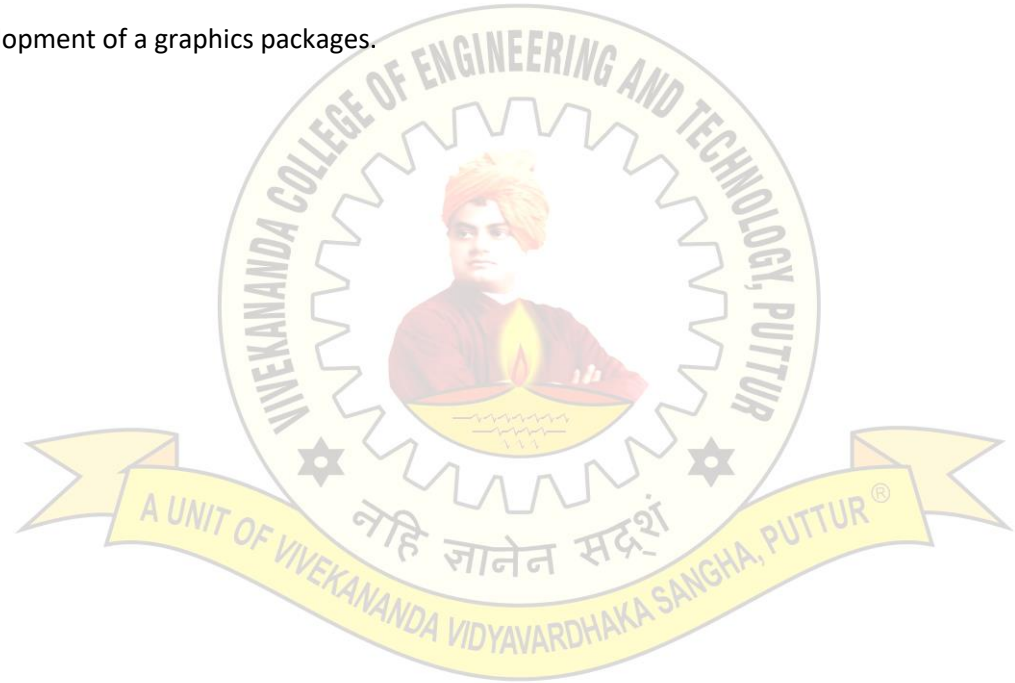


Fig.5.1.5 MENU IN COMMAND PROMPT

Conclusion

We had a wonderful experience developing this project. By working on this project we gained hands-on experience of using the OpenGL software using which we experimented this project. We have made use of hardware devices such as mouse driven interface, thus to reduce the complexity and make it user friendly. The project helped in understanding the working of computer graphics using OpenGL and various concepts, functions and methodologies for the development of a graphics packages.



REFERENCES

[1] Edward Angel, Interactive Computer Graphics A Top-Down Approach Using OpenGL, 5th edition, Dorling Kindersley India Pvt. Ltd., UP, India, 2009.

[2] Computer Graphics Using OpenGL– F.S. Hill, Jr. 2nd Edition, Pearson 1. Education, 2001.

[3] Computer Graphics– James D Foley, Andries Van Dam, Steven K Feiner, John F Hughes, Addison-wesley 1997.

[4] W3 SCHOOLS