“VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI”

B.L.D.E.A's V.P. DR. P.G. HALAKATTI COLLEGE OF ENGINEERING AND TECHNOLOGY, VIJAYAPUR -- 586103

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE & ENGINEERING

COMPUTER GRAPHICS MINI PROJECT (18CSL67)

MINI PROJECT REPORT ON:

“BIKE SIMULATION”

UNDER THE GUIDANCE OF

Mr.Santoshkumar Dewar

Mr.Kiran Patil

SUBMITTED BY:

Nikhil Gugawad (2BL19CS057)

Shreehari Hulyalkar (2BL19CS088)

Rohit Kotyal(2BL19CS076)

B.L.D.E.A.'s V.P. Dr. P.G. HALAKATTI COLLEGE OF ENGINEERING AND TECHNOLOGY, VIJAYAPURA – 586 103

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that mini project work of **Computer Graphics Laboratory (18CSL67)** entitled **"Bike Simulation"** is a bonafide work carried out in the sixth semester by **Nikhil Gugawad** [2BL19CS057], **Shreehari Hulyalkar** [2BL19CS088], **Rohit Kotyal** [2BL19CS076] in partial fulfilment for the award of Bachelor of Engineering in Computer Science and Engineering from **Visvesvaraya Technological University, Belagavi** during the academic year 2021-2022.

GUIDE	HOD	PRINCIPAL
Mr.Santoshkumar Dewar	**Dr.Pushpa B. Patil**	**Dr. V.G.Sangam**
Mr.Kiran Patil		

Examiner 1:

Examiner 2:

ACKNOWLEDGEMENT

We would also like to express our heartfelt gratitude to our beloved Principal Dr. V. G. Sangam and Dr. Pushpa. B. Patil, HOD, Computer Science and Engineering Sir BLDEA's CET whose guidance and support was truly invaluable.

We express our gratitude to our guides Mr.Kiran Patil and Mr.Santoshkumar Dewar, Asst. Prof., BLDEA's CET for their valuable guidance, instance motivation and constant supervision at all places of study for making this a success.

We are extremely thankful to our parents and friends for their unfailing enthusiasm, moral boosting and encouragement for us in completion of this.

Finally, a vote of thanks to the Department of Computer Science Engineering, both teaching and non-teaching staff for their co-operation extended.

Project Associates

1.Nikhil Gugawad

2.Shreehari Hulyalkar

3.Rohit Kotyal

ABSTRACT

This Project is on “3D BIKE SIMULATION” Computer Graphics using OpenGL Functions. It is a User interactive program where in the User can view the required display by making use of the input devices like Keyboard and Mouse. This project mainly consists of a bike and a robot. The Robot is made to sit on the bike, so that it looks like a man riding it. The bike can be viewed in any direction and in any angle. The bike is accelerated and its movements are controlled using the keyboard. For viewing, we make use of the mouse. The bike is ridden on a polygonal surface, which looks like a giant rectangular mesh. Lighting has been incorporated on only one side-Viewer side.

CONTENTS

Chapter No.	Name	Page No.
1.	INTRODUCTION	1
2.	REQUIREMENTS	2
3.	ABOUT THE PROJECT	3
4.	IMPLEMENTATION	4-22
5.	RESULTS	23-25
6.	CONCLUSION	
	REFERENCES	

1.INTRODUCTION

This report contains implementation of '3D BIKE SIMULATION 'using a set of OpenGL functions. The project consists of different views for 3D bike. We are mainly using keyboard and mouse as interface to view the bike, to accelerate, control it's movements. The objects are drawn by using GLUT functions. This project has been developed using CodeBlocks with OpenGL package.

1.1 Computer Graphics

Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

1.2 OpenGL Interface

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects. Most of our applications will be designed to access OpenGL directly through functions in three libraries. They are:

- Main GL: Library has names that begin with the letter gl and are stored in a library usually referred to as GL.
- OpenGL Utility Library (GLU): This library uses only GL functions but contains code for creating common objects and simplifying viewing.
- OpenGL Utility Toolkit(GLUT): This provides the minimum functionality that should be accepted in any modern windowing system.

1.3 OpenGL Overview

OpenGL (Open Graphics Library) is the interface between a graphic program and graphics hardware. It is streamlined. In other words, it provides low-level functionality. For example, all objects are built from points, lines and convex polygons. Higher level objects like cubes are implemented as six four-sided polygons. OpenGL supports features like 3-dimensions, lighting, anti-aliasing, shadows, textures, depth effects, etc. It is system-independent. It does not assume anything about hardware or operating system and is only concerned with efficiently rendering mathematically described scenes. As a result, it does not provide any windowing capabilities.

2.REQUIREMENTS & SPECIFICATIONS

2.1 Hardware Requirements

- Microprocessor: 1.0 GHz and above CPU based on either AMD or INTEL Microprocessor Architecture
- Main memory : 2 GB RAM
- Hard Disk : 40 GB
- Hard disk speed in RPM:5400 RPM
- Keyboard: QWERTY Keyboard
- Mouse :2 or 3 Button mouse
- Monitor : 1024 x 768 display resolution

2.2 Software Requirements

- Programming language – C/C++ using OpenGL
- Operating system – Linux operating system/Windows OS with Codeblocks application
- Compiler – C Compiler
- Graphics library – GL/glut.h
- OpenGL 2.0

2.3 Functional Requirements

OpenGL API's

If we want to have a control on the flow of program and if we want to interact with the window system then we use OpenGL API'S. Vertices are represented in the same manner internally, whether they are specified as two-dimensional or three-dimensional entities, everything that we do are here will be equally valid in three dimensions. Although OpenGL is easy to learn, compared with other APIs, it is nevertheless powerful. It supports the simple three dimensional programs and also supports the advanced rendering techniques.

GL/glut.h

We use a readily available library called the OpenGL Utility Toolkit (GLUT), which provides the minimum functionality that should be expected in any modern windowing system. The application program uses only GLUT functions and can be recompiled with the GLUT library for other window system. OpenGL makes a heavy use of macros to increase code readability and avoid the use of magic numbers. In most implementation, one of the include lines

3.ABOUT THE PROJECT

3.1 Overview

This Project is on "3D BIKE SIMULATION" Computer Graphics using OpenGL Functions. It is a User interactive program where in the User can view the required display by making use of the input devices like Keyboard and Mouse. This project mainly consists of an 3D bike on which a robot sits. The bike can be viewed in any direction using Mouse. The bike movement is done with the help of the Keyboard.

3.2 User Interface

A set of keys are used to change the following:

- User can select the view of the bike by using Mouse and Keyboard.
- Acceleration of the bike is controlled by '+'.
- De-acceleration is done using the key '-'.
- Movement of the bike is controlled by the keys '1'-left and '2'-right.
- Zooming in is controlled by the UP arrow key.
- Zoom out is controlled by the DOWN arrow key.
- To move the camera towards left, key to be used is LEFT arrow key.
- To move the camera towards right, key to be used is RIGHT arrow key.
- To reset the scene, key to be used is 'r' or 'R'.

3.3 Objective

- The aim of the project is to demonstrate the 3D Bike Simulation with multiple views.
- As Linux doesn't provide graphics editor, it should be designed in such a way that it provides a very useful graph implementation interface.
- It should be easy to understand, user interactive interface.
- Creation of primitives, i.e. polygons
- Providing human interaction through Mouse and keyboard.

4.IMPLEMENTATION

4.1 Existing System

Existing system for a graphics is the TC++. This system will support only the 2D graphics. 2D graphics package being designed should be easy to use and understand. It should provide various options such as free hand drawing, line drawing, polygon drawing, filled polygons, flood fill, translation, rotation, scaling, clipping etc. Even though these properties were supported, it was difficult to render 2D graphics cannot be very difficult to get a 3 Dimensional object. Even the effects like lighting, shading cannot be provided. So we go for Microsoft Visual Studio software.

4.2 Proposed System

To achieve three dimensional effects, open GL software is proposed. It is software which provides a graphical interface. It is an interface between application program and graphics hardware. The advantages are:

- Open GL is designed as a streamlined.
- It's a hardware independent interface i.e it can be implemented on many different hardware platforms.
- With Open GL we can draw a small set of geometric primitives such as points, lines and polygons etc.
- It provides double buffering which is vital in providing transformations
- It is event driven software.
- It provides call back functions.

4.3 User Defined Functions

- ZCylinder(): This function includes creation of a cylinder within an outer cylinder to give it a 3D effect.
- XCylinder(): This function creates the outer cylinder, it has been called as and when required.
- drawFrame(): This function is used to draw the frame of the bike. XCylinder() has been called several times in this.
- gear( ): This function is mainly used to draw the gear of the bike. It has been declared outside the drawFrame(), since its rotation itself is complex and it should rotate faster once the speed of the bike is increased.
- drawChain( ): This function is used to draw the chain to the gear. It has been created using the api GL_LINE_STIPPLE.
- drawTyre(): This function is used to create the tyres for the bike. In this function the components of the wheel such as spokes, rims and disc brake are included.
- drawSeat(): This function is used to draw the rider's seat. glVertex3f() has been utilized multiple times for top, bottom and side faces of the seat.
- reset(): Once 'r' or 'R' key is pressed, the bike and the camera are brought back to their initial positions.

- Idle(): This function will loop back and display what the user wants. i.e. if no key is pressed the bike will be in its resting place. If any of the movement keys are pressed then motion of the bike and the camera movement(and its orientation) will be observed.
- updateScene(): This function is used to update the viewing screen once the user presses RIGHT or LEFT arrow key.
- landmarks(): This function is used to create the ground for the bike on which it moves. The surface is a giant rectangular mesh whose dimensions are beyond infinity.
- special(): This function is used to move the camera UP, DOWN, LEFT or RIGHT.
- motion(): This function includes forward movement of the bike. As the user presses the '-' key, the de-acceleration of the bike can be observed.
- degrees(): This function is used to convert the given angle from radians to degrees.
- radians(): This function is used to convert the given angle from degrees to radians.
- angleSum(): This function is used to calculate the total angle for handle rotation if long press of key '1' or '2' is made.
- main( ): The main function is used for creating the window for display of the model of the atom. Here, we create menu for ease of use for the user. The callback functions, i.e., mouse callback, keyboard callback, display callback, idle callback, are written in main. The callback functions registered in main ( ) are,

```
glutDisplayFunc(display);
glutKeyboardFunc(Keyboard);
glutSpecialFunc(Special);
```

4.4 OpenGL Functions

- glColor3f (float, float, float):-This function will set the current drawing color
- glClear( ):-Takes a single argument that is the bitwise OR of several values indicating which buffer is to be cleared.
- glClearColor ():-Specifies the red, green, blue, and alpha values used by glClear to clear the color buffers.
- GlLoadIdentity( ):-the current matrix with the identity matrix.
- glMatrixMode(mode):-Sets the current matrix mode, mode can be GL_MODELVIEW, GL_PROJECTION or GL_TEXTURE.
- void glutInit (int *argc, char**argv):-Initializes GLUT, the arguments from main are passed in and can be used by the application.
- void glutInitDisplayMode (unsigned int mode):-Requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model and buffering.
- void glutInitWindowSize (int width, int height):- Specifies the initial position of the topleft corner of the window in pixels
- glutInitCreateWindow (char *title):-A window on the display.The string title can be used to label the window. The return value provides references to the window that can be used when there are multiple windows.
- void glutMouseFunc(void *f(int button, int state, int x, int y):-Register the mouse callback function f. The callback function returns the button,the state of button after the event and the position of the mouse relative to the top-left corner of the window.

- void glutKeyboardFunc(void(*func) (void)):-This function is called every time when you press enter key to resume the game or when you press ‘b’ or ‘B’ key to go back to the initial screen or when you press esc key to exit from the application.
- void glutDisplayFunc (void (*func) (void)):-Register the display function func that is executed when the window needs to be redrawn.
- void glutSpecialFunc(void(*func)(void)):-This function is called when you press the special keys in the keyboard like arrow keys, function keys etc. In our program, the func is invoked when the up arrow or down arrow key is pressed for selecting the options in the main menu and when the left or right arrow key is pressed for moving the object(car) accordingly.
- glutPostReDisplay ( ) :-which requests that the display callback be executed after the current callback returns.
- void MouseFunc (void (*func) void)):-This function is invoked when mouse keys are pressed. This function is used as an alternative to the previous function i.e., it is used to move the object(car) to right or left in our program by clicking left and right button respectively.
- void glutMainLoop () Cause the program to enter an event-processing loop.It should be the last statement in main function.

4.5 Source Code

```
#include<GL/glut.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<math.h>
#define PI 3.14159
#define WIN_WIDTH 1920
#define WIN_HEIGHT 1080
#define BIKE_LENGTH 3.3f
#define ROD_RADIUS 0.05f
#define GAS_TANK 0.3f
#define NUM_SPOKES 5
#define SPOKE_ANGLE 72
#define RADIUS_WHEEL 1.0f
#define TUBE_WIDTH 0.20f
#define RIGHT_ROD 2.35f
#define RIGHT_ANGLE 33.0f
#define MIDDLE_ROD 1.8f
#define MIDDLE_ANGLE 125.0f
#define BACK_CONNECTOR 0.5f
#define LEFT_ANGLE 50.0f
#define WHEEL_OFFSET 0.11f
#define WHEEL_LEN 0.8f
#define TOP_LEN 1.0f
#define FRONT_ROD 2.5f
#define CRANK_ROD 1.9f
#define CRANK_RODS 2.5f
#define CRANK_ANGLE 0.0f
```

```
#define HANDLE_ROD 1.2f
#define FRONT_INCLINE 38.0f
#define HANDLE_LIMIT 30.0f
#define INC_STEERING 2.0f
#define INC_SPEED 0.05f
GLfloat pedalAngle, speed, steering;
GLfloat camx,camy,camz;
GLfloat anglex,angley,anglez;
int prevx,prevy;
GLenum Mouse;
GLfloat xpos,zpos,direction;
void welcome_window();
void operations_window();
void ZCylinder(GLfloat radius,GLfloat length);
void XCylinder(GLfloat radius,GLfloat length);
void drawFrame(void);
void gear( GLfloat inner_radius, GLfloat outer_radius,
 GLfloat width,GLint teeth, GLfloat tooth_depth );
void drawChain(void);
void drawPedals(void);
void drawTyre(void);
void drawSeat(void);
void init(void);
void reset(void);
void display_bike(void);
void idle(void);
void updateScene(void);
void landmarks(void);
void special(int key,int x,int y);
void keyboard(unsigned char key,int x,int y);
```

```
void mouse(int button,int state,int x,int y);
void motion(int x,int y);
GLfloat Abs(GLfloat);
GLfloat degrees(GLfloat);
GLfloat radians(GLfloat);
GLfloat angleSum(GLfloat, GLfloat);
void ZCylinder(GLfloat radius,GLfloat length)
{
 GLUquadricObj *cylinder;
 cylinder=gluNewQuadric();
 glPushMatrix();
 glTranslatef(0.0f,0.0f,0.0f);
 gluCylinder(cylinder,radius,radius,length,15,5);
 glPopMatrix();
}
void XCylinder(GLfloat radius,GLfloat length)
{
 glPushMatrix();
 glRotatef(90.0f,0.0f,1.0f,0.0f);
 ZCylinder(radius,length);
 glPopMatrix();
}
void updateScene()
{
 GLfloat xDelta, zDelta; //Distance
 GLfloat rotation;
 GLfloat sin_steering, cos_steering;
if(speed < 0.0f)
 pedalAngle = speed = 0.0f;
```

```
xDelta = speed*cos(radians(direction + steering));
 zDelta = speed*sin(radians(direction + steering));
 xpos += xDelta;
 zpos -= zDelta;
 pedalAngle = degrees(angleSum(radians(pedalAngle), speed/RADIUS_WHEEL));
sin_steering = sin(radians(steering));
 cos_steering = cos(radians(steering));
rotation = atan2(speed * sin_steering, BIKE_LENGTH + speed * cos_steering);
 direction = degrees(angleSum(radians(direction),rotation));
}
GLfloat angleSum(GLfloat a, GLfloat b)
{
 a += b;
 if (a < 0) return a+2*PI; //If returned angle is towards left and lesser than its threshold value,increment its value to zero & do nothing
 else if (a > 2*PI) return a-2*PI; //If returned angle is towards right and greater than its threshold value, decrement its value to zero & do nothing
 else return a; //If nothing, return the radians equivalent of angle 'a'
}
void drawFrame()
{
 glColor3f(1.0f,0.0f,0.0f);
glPushMatrix();
glPushMatrix();
 glColor3f(0.0f,1.0f,0.0f);
glPushMatrix();
 glTranslatef(0.0f,0.0f,0.06f);
 glRotatef(-2*pedalAngle,0.0f,0.0f,1.0f);
 gear(0.08f,0.3f,0.03f,30,0.03f);
 glPopMatrix();
```

```
 glColor3f(1.0f,0.0f,0.0f);
 glTranslatef(0.0f,0.0f,-0.2f);
 ZCylinder(0.08f,0.32f);
 glPopMatrix();
glRotatef(RIGHT_ANGLE,0.0f,0.0f,1.0f);
 XCylinder(ROD_RADIUS,RIGHT_ROD);
glRotatef(MIDDLE_ANGLE-RIGHT_ANGLE,0.0f,0.0f,1.0f);
 XCylinder(ROD_RADIUS,MIDDLE_ROD);
glColor3f(1.0f,1.0f,1.0f);
 glTranslatef(MIDDLE_ROD,0.0f,0.0f);
 glRotatef(-MIDDLE_ANGLE,0.0f,0.0f,1.0f);
 glScalef(0.3f,ROD_RADIUS,0.25f);
 drawSeat();

 glColor3f(1.0f,0.0f,0.0f);
 glPopMatrix();
glPushMatrix();
 glRotatef(-180.0f,0.0f,1.0f,0.0f);
 XCylinder(ROD_RADIUS,BACK_CONNECTOR);
glPushMatrix();
 glTranslatef(0.5f,0.0f,WHEEL_OFFSET);
 XCylinder(ROD_RADIUS,RADIUS_WHEEL+TUBE_WIDTH);
 glPopMatrix();
 glPushMatrix();
 glTranslatef(0.5f,0.0f,-WHEEL_OFFSET);
 XCylinder(ROD_RADIUS,RADIUS_WHEEL+TUBE_WIDTH);
 glPopMatrix();
 glPopMatrix();
glPushMatrix();
 glTranslatef(-(BACK_CONNECTOR+RADIUS_WHEEL+TUBE_WIDTH),0.0f,0.0f
```

```
glTranslatef(TOP_LEN,0.0f,0.0f); //Supporting rod position
 glRotatef(-FRONT_INCLINE,0.0f,0.0f,1.0f);
glTranslatef(-0.3f,0.0f,0.0f);
glPushMatrix();
 glRotatef(FRONT_INCLINE,0.0f,0.0f,1.0f);
 glPushMatrix();
 glTranslatef(-0.6f,0.5f,-HANDLE_ROD/2);
 ZCylinder(ROD_RADIUS,HANDLE_ROD);
 glPopMatrix();
 glPushMatrix();
 glColor3f(1.0f,1.0f,0.0f);
 glTranslatef(-0.6f,0.5f,-HANDLE_ROD/2); //Handle
 ZCylinder(0.07f,HANDLE_ROD/4); //Rods
 glTranslatef(0.0f,0.0f,HANDLE_ROD*3/4); //Are
 ZCylinder(0.07f,HANDLE_ROD/4); //Drawn
 glColor3f(1.0f,0.0f,0.0f); //Here
 glPopMatrix();
 glPopMatrix();
glPushMatrix();
glTranslatef(-0.75,0.0,0.0);
 XCylinder(ROD_RADIUS,FRONT_ROD);
glTranslatef(CRANK_ROD,0.0f,0.0f); //Position set to the end of the main big connector rod
in the front to draw the front connector rods at that position
 glRotatef(CRANK_ANGLE,0.0f,0.0f,1.0f);
glPushMatrix();
 glTranslatef(0.0f,0.0f,WHEEL_OFFSET-0.35);
 XCylinder(ROD_RADIUS,CRANK_RODS);
 glPopMatrix();
 glPushMatrix();
 glTranslatef(0.0f,0.0f,-WHEEL_OFFSET+0.35);
 XCylinder(ROD_RADIUS,CRANK_RODS);
```

```
 glPopMatrix();
glTranslatef(CRANK_RODS,0.0f,0.0f);
 glRotatef(-2*pedalAngle,0.0f,0.0f,1.0f);
 drawTyre();
 glPopMatrix();
 glPopMatrix(); /* End of the rotation of the handle effect */
 glPopMatrix();
glPushMatrix();
 glColor3f(1.0,1.0,0.0);
 glRotatef(360.0,1.0,0.0,0.0);
 glTranslatef(1.0,2.6,0.0);
 glutSolidSphere(0.2,160.0,180.0);
 glPopMatrix();
 glPushMatrix();
 glRotatef(-steering/2.0,1.0f,0.0f,0.0f);
 glColor3f(0.5f,0.3f,0.0f);
 glRotatef(180.0,1.0,0.0,0.0);
 glTranslatef(-1.0,-3.5f,0.0f);
 glutSolidSphere(0.4,160.0,180.0);
 glPopMatrix();
glPushMatrix();
 glRotatef(-steering/2.0,1.0f,0.0f,0.0f);
 glColor3f(1.0f,1.0f,1.0f);
 glRotatef(180.0,1.0,0.0,0.0);
 glTranslatef(-0.6,-3.6f,-0.15f);
 glutSolidSphere(0.05,160.0,180.0); //Right white eye
 glPopMatrix();
 glPushMatrix();
 glRotatef(-steering/2.0,1.0f,0.0f,0.0f);
 glColor3f(1.0f,1.0f,1.0f);
```

```
glRotatef(180.0,1.0,0.0,0.0);
glTranslatef(-0.6,-3.6f,0.15f);
glutSolidSphere(0.05,160.0,180.0); //Left white eye
glPopMatrix();
glPushMatrix();
glRotatef(-steering/2.0,1.0f,0.0f,0.0f);
glColor3f(0.0f,0.0f,0.0f);
glRotatef(180.0,1.0,0.0,0.0);
glTranslatef(-0.6,-3.6f,-0.15f);
glutSolidSphere(0.025,160.0,180.0); //Right black eye lid
glPopMatrix();
glPushMatrix();
glRotatef(-steering/2.0,1.0f,0.0f,0.0f);
glColor3f(0.0f,0.0f,0.0f);
glRotatef(180.0,1.0,0.0,0.0);
glTranslatef(-0.6,-3.6f,0.15f);
glutSolidSphere(0.025,160.0,180.0); //Left black eye lid
glPopMatrix();
glPushMatrix();
glRotatef(-steering/2.0,1.0f,0.0f,0.0f);
glColor3f(0.5f,0.3f,0.0f);
glRotatef(-270.0,-1.0,0.0,0.0);
glTranslatef(-0.6,0.0f,3.4f);
glutSolidCone(0.1,0.2,50.0,50.0);
glPopMatrix();
glPushMatrix();
glRotatef(-steering/2.0,1.0f,0.0f,0.0f);
glColor3f(1.0f,0.0f,1.0f); //It is made pink to indicate the rider's lips
glRotatef(90.0,0.0,1.0,0.0);
glTranslatef(-0.1,3.3f,-0.6f);
```

```
 glVertex3f(-0.1f,1.0f,-0.5f);
 glVertex3f(-0.1f,-1.0f,-0.5f);
 glVertex3f(1.0f,-1.0f,-0.3f);
 glVertex3f(-0.1f,1.0f,0.5f);
 glVertex3f(-0.5f,1.0f,1.0f);
 glVertex3f(-0.5f,-1.0f,1.0f);
 glVertex3f(-0.1f,-1.0f,0.5f);
 glVertex3f(-0.1f,1.0f,-0.5f);
 glVertex3f(-0.5f,1.0f,-1.0f);
 glVertex3f(-0.5f,-1.0f,-1.0f);
 glVertex3f(-0.1f,-1.0f,-0.5f);
 glVertex3f(-0.5f,1.0f,1.0f);
 glVertex3f(-1.0f,1.0f,1.0f);
 glVertex3f(-1.0f,-1.0f,1.0f);
 glVertex3f(-0.5f,-1.0f,1.0f);
 glVertex3f(-0.5f,1.0f,-1.0f);
 glVertex3f(-1.0f,1.0f,-1.0f);
 glVertex3f(-1.0f,-1.0f,-1.0f);
 glVertex3f(-0.5f,-1.0f,-1.0f);
 glVertex3f(-1.0f,1.0f,1.0f);
 glVertex3f(-1.0f,1.0f,-1.0f);
 glVertex3f(-1.0f,-1.0f,-1.0f);
 glVertex3f(-1.0f,-1.0f,1.0f);
 glEnd();
}
void drawPedals() //Drawn for foot rest
{
 glColor3f(0.0f,0.0f,1.0f);
 glPushMatrix();
 glTranslatef(0.0f,0.0f,0.105f);
```

```
 glTranslatef(0.25f,0.0f,0.0f);
 glPushMatrix();
 glScalef(0.5f,0.1f,0.1f);
 glutSolidCube(1.0f);
 glPopMatrix();
 glPushMatrix();
 glTranslatef(0.25f,0.0f,0.15f);
 glScalef(0.2f,0.02f,0.3f);
 glutSolidCube(1.0f);
 glPopMatrix();
 glPopMatrix();
 glPushMatrix();
 glTranslatef(0.0f,0.0f,-0.105f);
 glTranslatef(0.25f,0.0f,0.0f);
 glPushMatrix();
 glScalef(0.5f,0.1f,0.1f);
 glutSolidCube(1.0f);
 glPopMatrix();
 glPushMatrix();
 glTranslatef(0.25f,0.0f,-0.15f);
 glScalef(0.2f,0.02f,0.3f);
 glutSolidCube(1.0f);
 glPopMatrix();
 glPopMatrix();
 glColor3f(1.0f,0.0f,0.0f);
}
void drawTyre(void)
{
 int i;
glColor3f(1.0f,1.0f,1.0f);
```

```
GLfloat light_diffuse[]={1.0,1.0,1.0};
 reset();
 GLfloat radians(GLfloat a)
{
 return a*PI/180.0f;
}
void idle(void)
{
 updateScene();
 glutPostRedisplay();
}
void special(int key,int x,int y)
{
 switch(key)
 {
 case GLUT_KEY_UP:
 camz -= 0.1f;
 break;
 case GLUT_KEY_DOWN:
 camz += 0.1f;
 break;
 case GLUT_KEY_LEFT:
 camx -= 0.1f;
 break;
 case GLUT_KEY_RIGHT:
 camx += 0.1f;
 break;
 }
 glutPostRedisplay();
}
```

```
void reset()
{
glColor3f(1.0f,1.0f,1.0f);
 anglex=angley=anglez=0.0f;
 pedalAngle=steering=0.0f;
 Mouse=GLUT_UP;
 pedalAngle=speed=steering=0.0f;
 camx=camy=0.0f;
 camz=5.0f;
 xpos=zpos=0.0f;
 direction=0.0f;
 glPushMatrix();
 glColor3f(0.5f,0.3f,0.0f);
 glRotatef(100.0,0.0,0.0,1.0);
 glTranslatef(-1.0,0.0,-0.6);
 XCylinder(GAS_TANK-0.1,2.7);
 glPopMatrix();
}
void keyboard(unsigned char key,int x,int y)
{
 GLfloat r=0.0f,g=0.0f;
 switch(key)
 {
 case 'r':
 case 'R':
 reset();
 break;
 case 's':
 case 'S':
 glutDisplayFunc(operations_window);
```

```
break;
case 'c':
case 'C':
glutDisplayFunc(display_bike);
break;
case '1':
if(steering < HANDLE_LIMIT)
steering += INC_STEERING;
break;
case '2':
if(steering > -HANDLE_LIMIT)
steering -= INC_STEERING;
break;
case '+':
speed += INC_SPEED;
break;
case '-':
speed -= INC_SPEED;
break;
case 27:
exit(1);
}
pedalAngle += speed;
if(speed < 0.0f)
speed = 0.0f;
if(pedalAngle < 0.0f)
pedalAngle = 0.0f;
if(pedalAngle >= 360.0f)
pedalAngle -= 360.0f;
glutPostRedisplay();
```

```
}
void mouse(int button,int state,int x,int y)
{
 switch(button)
 {
 case GLUT_LEFT_BUTTON:
 if(state==GLUT_DOWN)
 {
 Mouse=GLUT_DOWN;
 prevx=x;
 prevy=y;
 }
 if(state==GLUT_UP)
 {
 Mouse=GLUT_UP;
 }
}
void welcome_window()
{
 glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glClearColor(0,0,0,0);
glColor3f(1.0,1.0,1.0);
bitmap_output(-1.25,1.8,0.50,"VISVESVARAYA TECHNOLOGICAL UNIVERSITY");
bitmap_output(-0.6,1.6,0.50,"BELGAUM,KARNATAKA");
bitmap_output(-0.3,0.70,0.50,"Project On");
bitmap_output(-0.85,0.50,0.50,"'SIMULATION OF A 3D BIKE'");
bitmap_output(-0.6,-1.5,0.50,"PLEASE PRESS S TO START");
 glutSwapBuffers();
glFlush();
}
```

```
void operations_window()
{
 glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glClearColor(0,0,0,0);
glColor3f(1.0,1.0,1.0);
bitmap_output(-1.25,1.7,0.50,"OPERATIONS THAT CAN BE PERFORMED BY THE BIKE");
bitmap_output(-1.25,1.2,0.50,"1. RESET THE CAMERA - USE 'R' OR 'r'");
bitmap_output(-1.25,1.0,0.50,"2. ACCELERATE THE BIKE - USE '+'");
bitmap_output(-1.25,0.8,0.50,"3. DEACCELERATE THE BIKE - USE '-'");
 bitmap_output(-1.25,0.6,0.50,"4. TURN RIGHT - USE '2'");
 bitmap_output(-1.25,0.4,0.50,"5. TURN LEFT - USE '1'");
bitmap_output(-1.25,0.2,0.50,"6. ZOOM IN - USE 'UPWARD ARROW'");
bitmap_output(-1.25,0.0,0.50,"7. ZOOM OUT - USE 'DOWNWARD ARROW'");
bitmap_output(-1.25,-0.2,0.50,"8. MOVE LEFT - USE 'LEFT ARROW'");
bitmap_output(-1.25,-0.4,0.50,"9. MOVE RIGHT - USE 'RIGHT ARROW'");
bitmap_output(-1.25,-0.6,0.50,"10. USE MOUSE TO CHANGE THE ANGLE OF VIEWING");
bitmap_output(-1.25,-1.0,0.50,"PLEASE PRESS C TO CONTINUE");
 glutSwapBuffers();
glFlush();
}
int main(int argc,char *argv[])
{
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
 glutInitWindowPosition(100,100);
 glutInitWindowSize(WIN_WIDTH,WIN_HEIGHT);
 glutCreateWindow("Bike");
 init();
 glutDisplayFunc(welcome_window);
```

```
 glutReshapeFunc(reshape);
 glutIdleFunc(idle);
 glutSpecialFunc(special);
 glutKeyboardFunc(keyboard);
 glutMouseFunc(mouse);
 glutMotionFunc(motion);
 glutPassiveMotionFunc(passive);
 glutSetCursor(GLUT_CURSOR_CROSSHAIR);
 glutMainLoop();
}
```

5.RESULTS

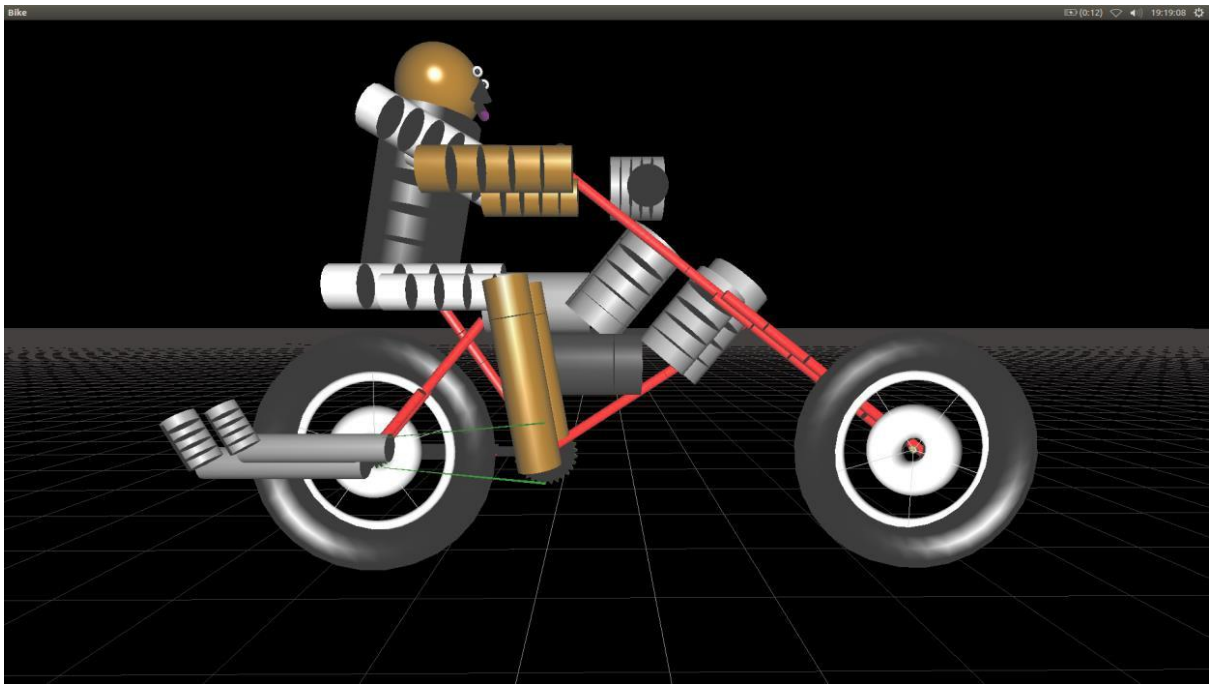

Fig 5.1 Bike after running the code

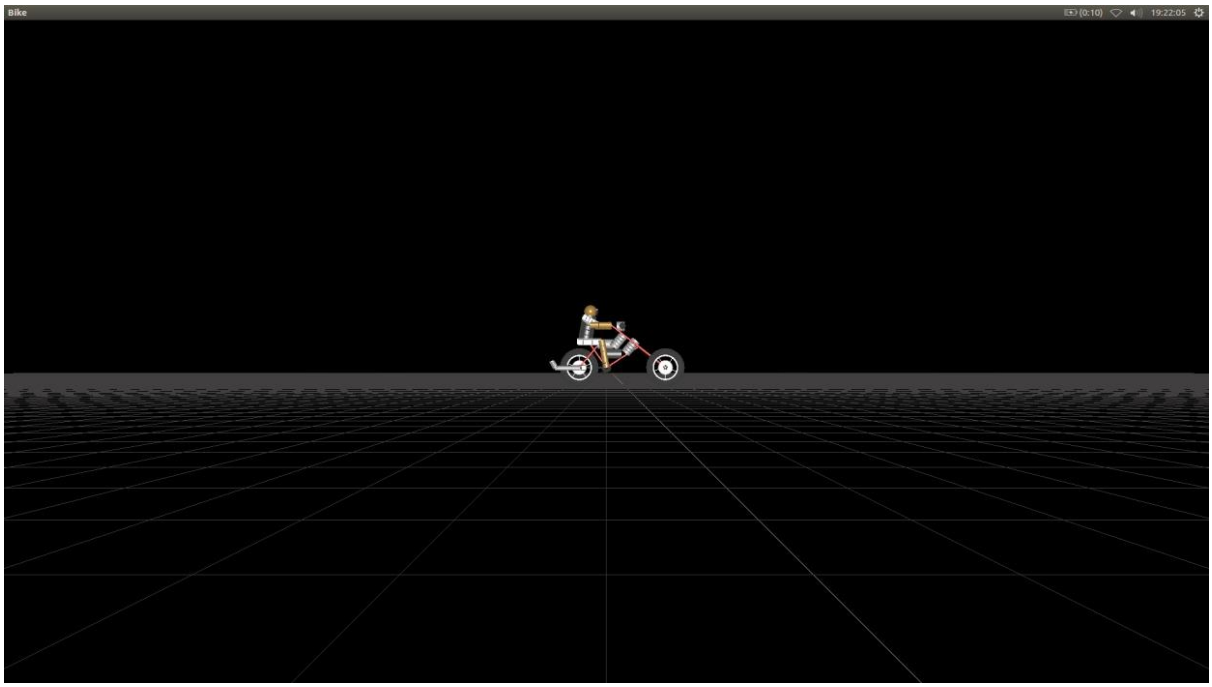

Fig 5.2 Bike moved away from the viewer

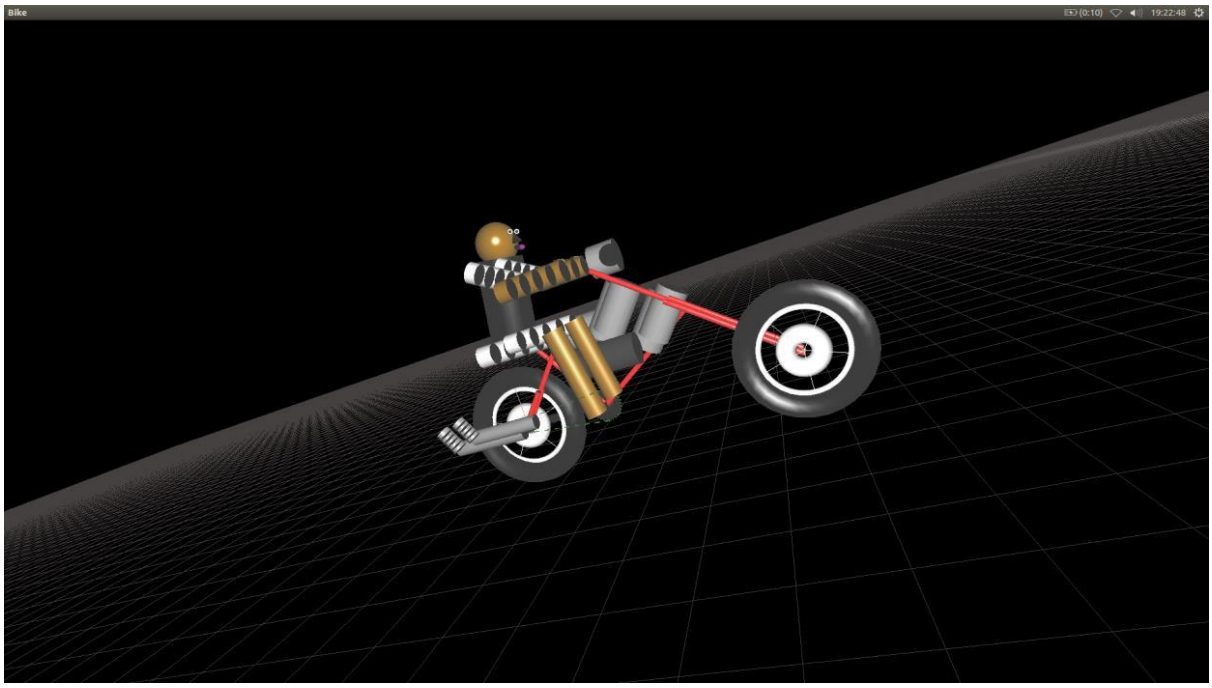

Fig 5.3 Mouse orientation towards x-y-axes

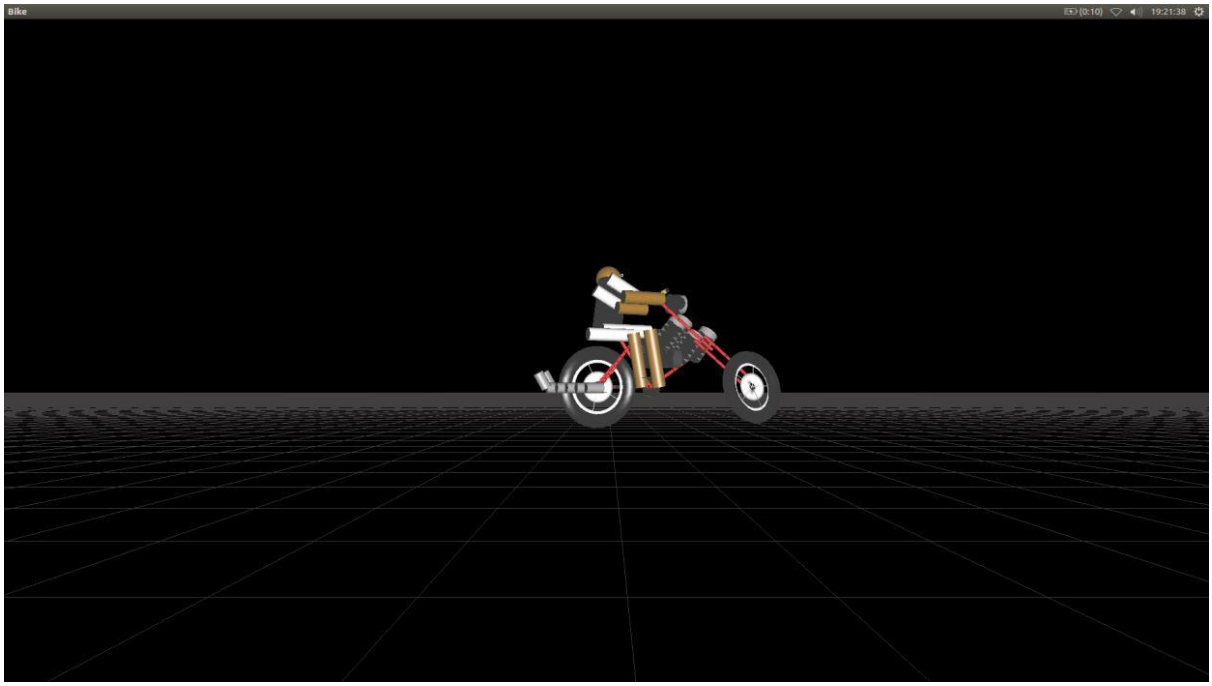

Fig 5.4 Bike when it turns left

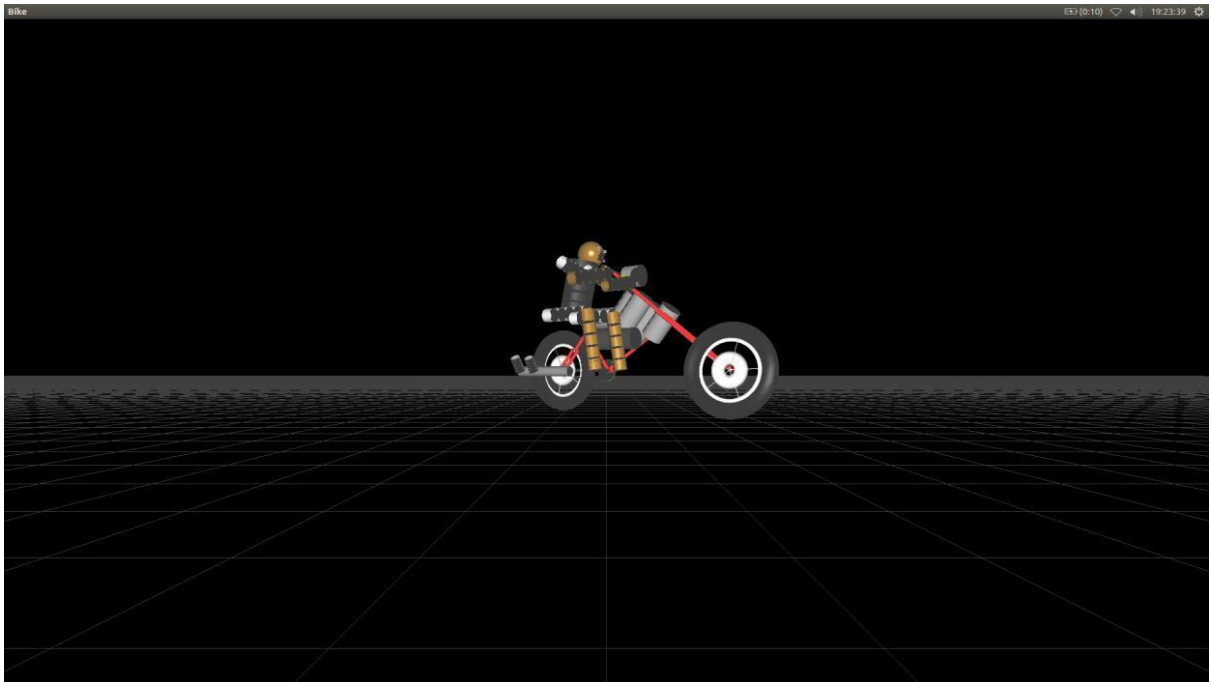

Fig 5.5 Bike when it approaches the viewer

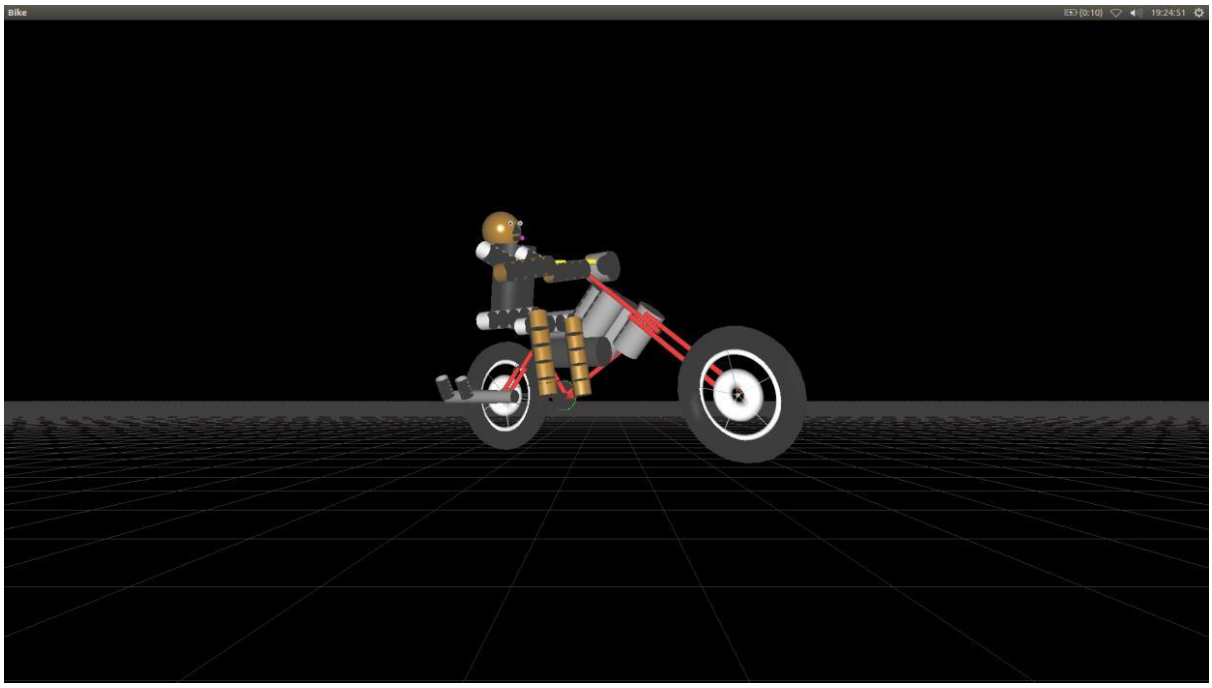

Fig 5.6 The rider inclines towards right when handle rotates right

6.CONCLUSION AND FUTURE SCOPE

The 3D Bike Simulation has been tested under Ubuntu 11.10, and has been found to provide ease of use and manipulation to the user. The 3D Bike Simulation created for the Ubuntu 11.10 operating system can be used to draw lines, boxes, circles, ellipses, and polygons. It has a very simple and effective user interface.

We found designing and developing this 3D Bike as a very interesting and learning experience. It helped us to learn about computer graphics, design of Graphical User Interfaces, interface to the user, user interaction handling and screen management. The graphics editor provides all and more than the features that have been detailed in the university syllabus.

REFERENCES

[1] Donald D Hearn and M.Pauline Baker,"Computer Graphics with OpenGL", 3rd Edition.

[2] Portion of the code for implementing GEAR has been borrowed from Brian Paul's MESA.

[3] Edward Angel's Interactive Computer Graphics Pearson Education 5th Edition

[4] Jackie.L.Neider,Mark Warhol,Tom.R.Davis,"OpenGL Red Book",Second Revised Edition,2005.