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.

TABLE OF CONTENTS

CHAPTER 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 **Ubuntu11.10** 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.

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computers themselves. It has grown to include the creation, storage, and manipulation of models and images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural, and even conceptual structures, natural phenomena, and so on. Computer graphics today is largely interactive. The user controls the contents, structure, and appearance of the objects and of their displayed images by using input devices, such as keyboard, mouse, or touch-screen. Due to close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics. The advantages of the interactive graphics are many in number. Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D patter-recognition abilities allow us to perceive and process data rapidly and efficiently. In many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the 1980s when

the scientists and engineers realized that they could not interpret the prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

1.2OpenGL 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:

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

1.3OpenGL 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.
- *It is a state machine*. At any moment during the execution of a program there is a current model transformation
- *It is a rendering pipeline*. The rendering pipeline consists of the following steps:
    - Defines objects mathematically.

- Arranges objects in space relative to a viewpoint.
- Calculates the color of the objects.
- Rasterizes the objects.

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. 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.

OpenGL (open graphics library) is a standard specification defining a cross language cross platform API for writing applications that produce 2D and 3D computer graphics. OpenGL was developed by silicon graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization and flight simulation. It is also used in video games.

OpenGL serves two main purpose:

- To hide the complexities of interfacing with different 3D accelerators, by presenting 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.

OpenGL has historically been influential on the development of 3D accelerator, promoting a base level of functionality that is now common in consumer level hardware:

- Rasterized points, lines and polygons are basic primitives.
- A transform and lighting pipeline.
- Z buffering.
- Texture Mapping.
- Alpha
- Blending.

Chapter 2

SYSTEM REQUIREMENTS SPECIFICATION

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
- Compiler – C Compiler
- Graphics library – GL/glut.h
- OpenGL 2.0

2.3:FUNCTIONAL REQUIREMENTS:

OpenGL APIs:

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

Chapter 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.

CHAPTER 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:

1. Open GL is designed as a streamlined.
2. It's a hardware independent interface i.e it can be implemented on many different hardware platforms.
3. With Open GL we can draw a small set of geometric primitives such as points, lines and polygons etc.
4. It provides double buffering which is vital in providing transformations.
5. It is event driven software.
6. 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, charargv):-**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 top-left 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.

CHAPTER 5
TESTING

Testing process started with the testing of individual program units such as functions or objects. These were then integrated into sub-systems and systems, and interactions of these units were tested.

Testing involves verification and validation.

Validation: "Are we building right product?"

Verification: "Are we building the product right?"

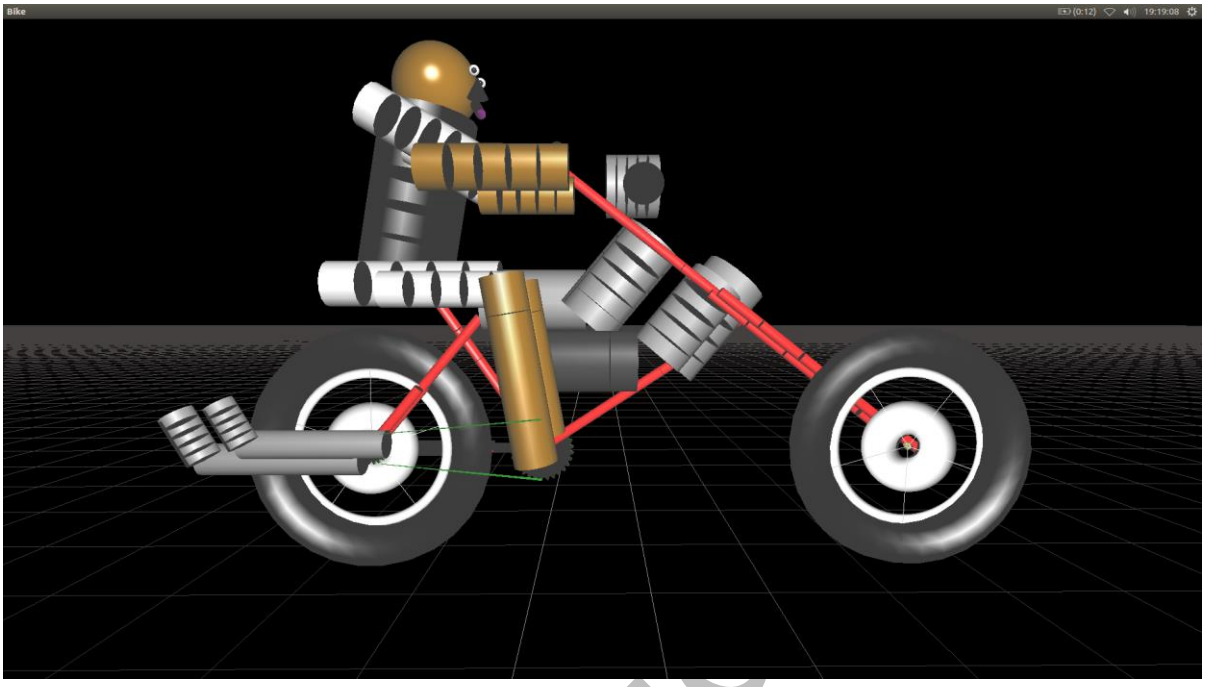
The ultimate goal of the verification and validation process is to establish confidence that the software system is 'fit for purpose'. The level of required confidence depends on the system's purpose, the expectations of the system users and the current marketing environment for the system.

With the verification and validation process, there are two complementary approaches to the system checking and analysis:

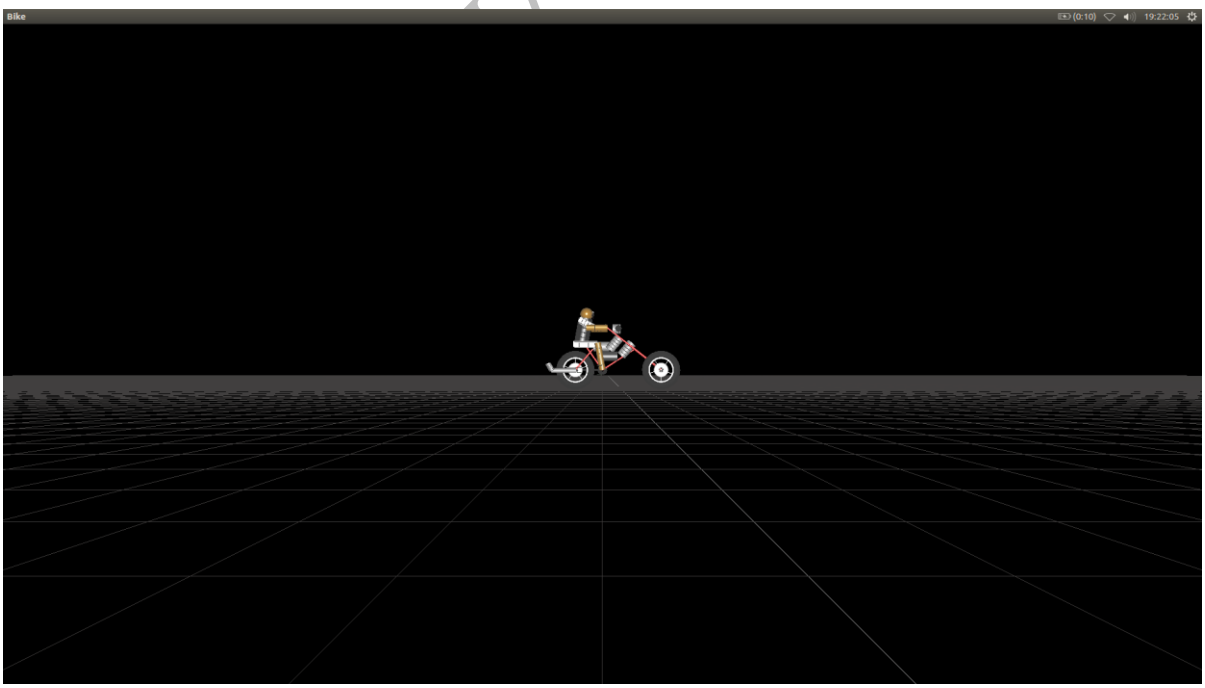
Software inspections or peer reviews analyses and check system representations such as the requirements document, design diagrams, and the program source code. Software testing involves running an implementation of the software with test data.

CHAPTER 6

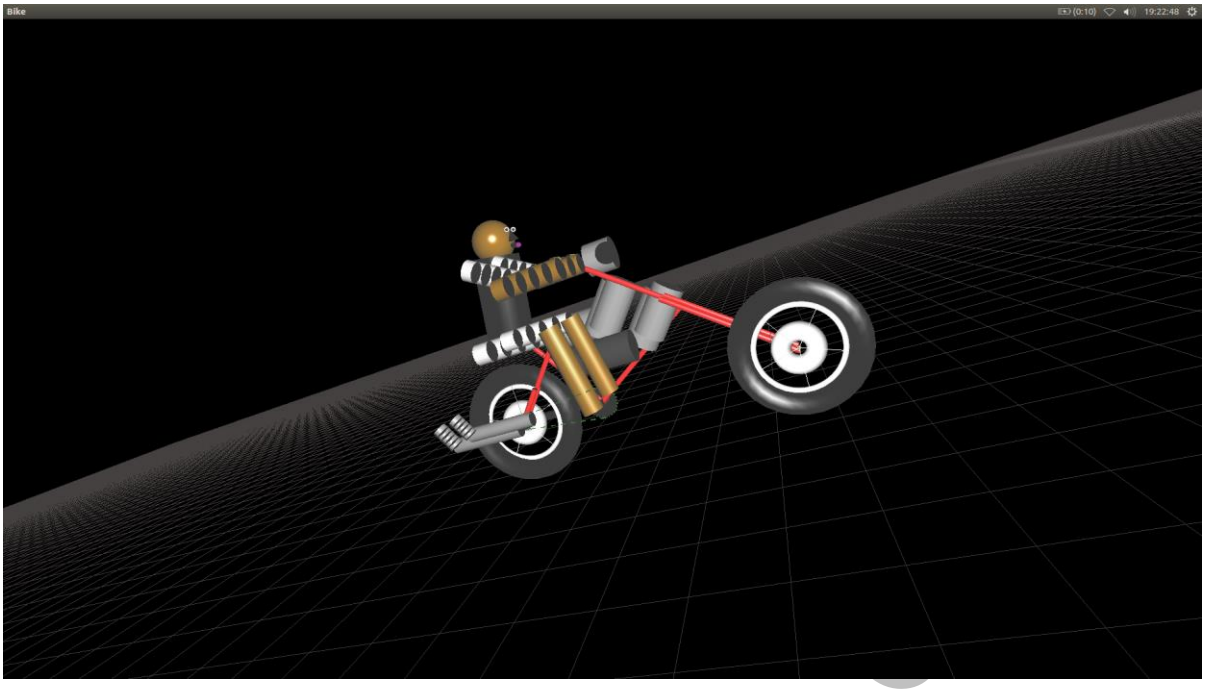
SNAPSHOTS



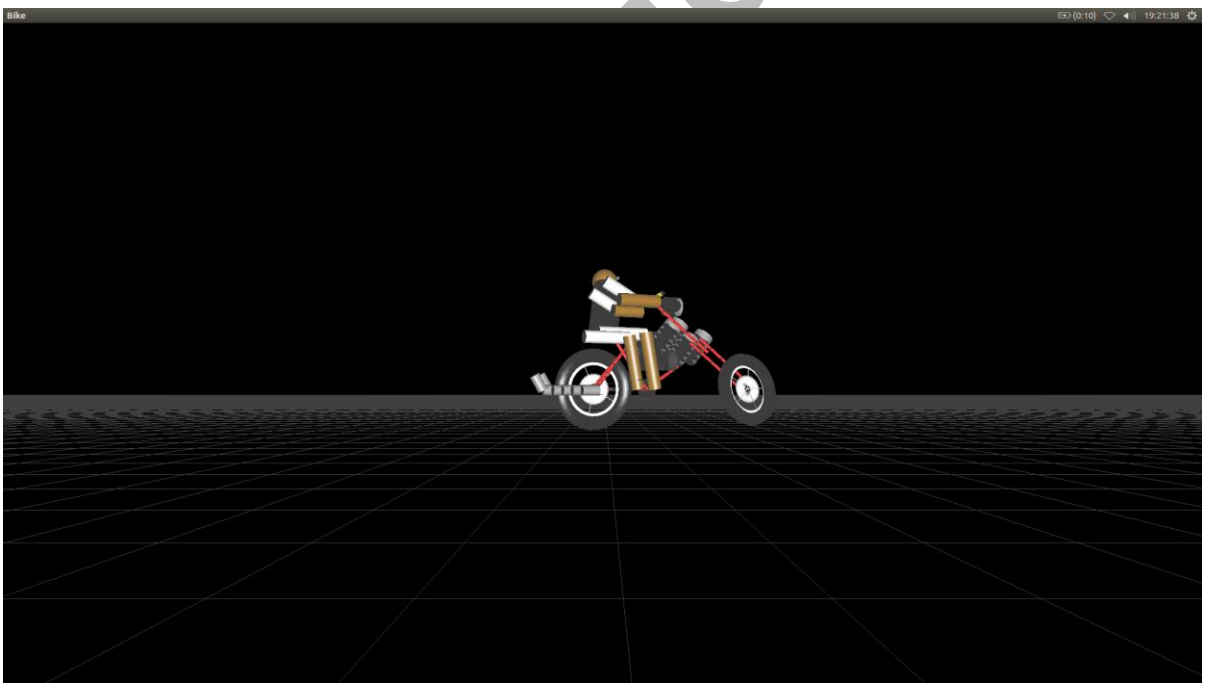
Bike after running the code



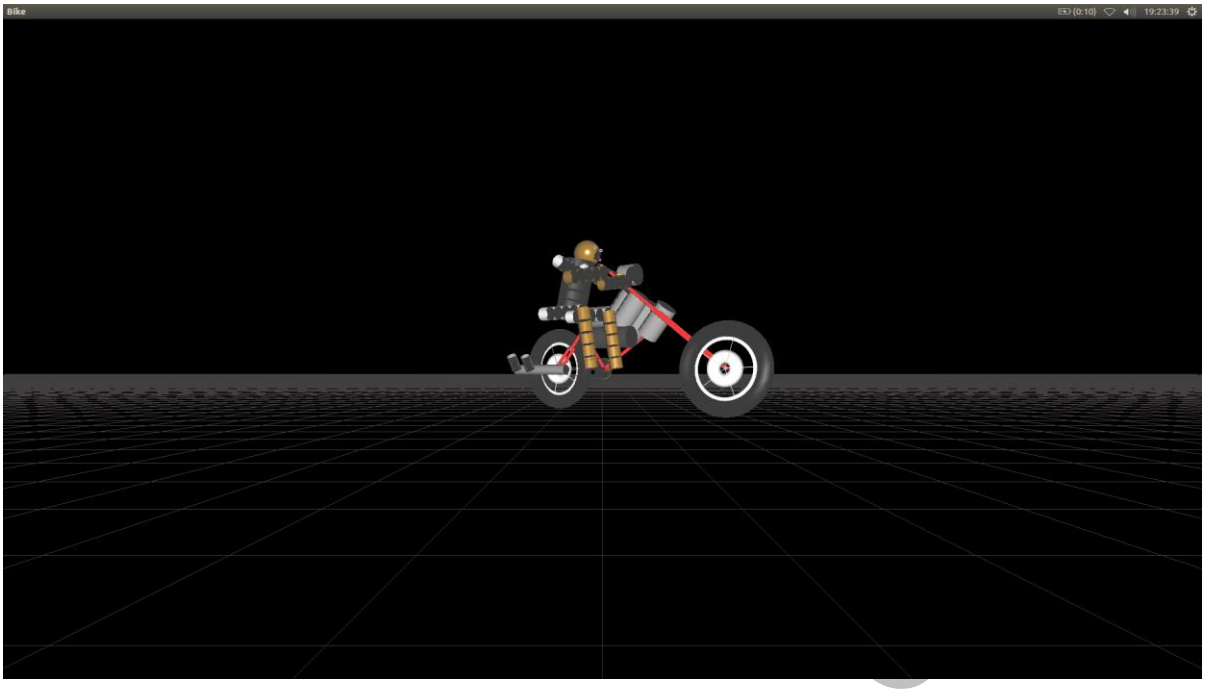
Bike moved away from the viewer



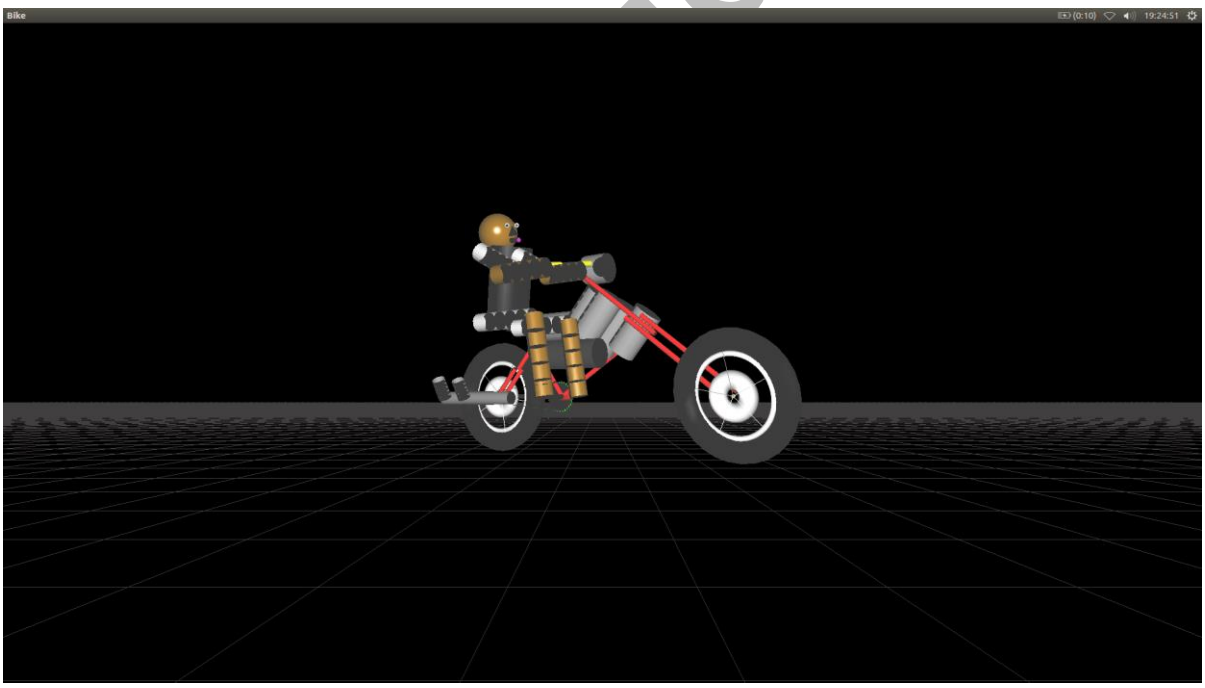
Mouse orientation towards x-y-axes



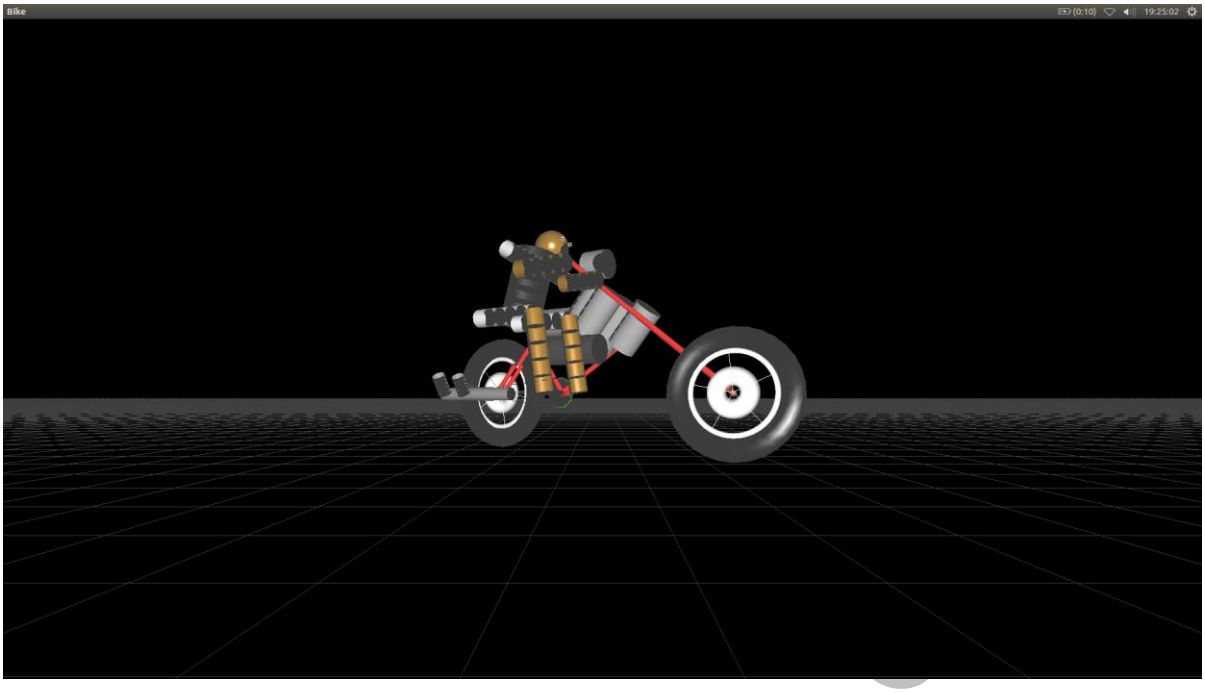
Bike when it turns left



Bike when it approaches the viewer



The rider inclines towards right when handle rotates right



The rider inclines towards left when handle rotates left

Chapter 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

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.

7.2 FUTURE ENHANCEMENTS

These are the features that are planned to be supported in the future

* Simulating smoke from exhaust pipe
* Perform wheelie by the rider
* To improve the looks of the rider
* To implement shadow using more built-in functions

7.3 LIMITATIONS

As with all types of parallel projection, objects drawn with isometric projection do not appear larger or smaller as they extend closer to or away from the viewer. Since lighting has been given from only one side, the light will not be illuminated on the other side. Hence, the left part of the bike is seen as dark. As we have generated cylinders within a cylinder, it so happens that when we move from right to left or vice-versa, the surface of inner cylinder is seen through the outer one.

BIBLIOGRAPHY

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

[2] Interactive computer Graphics --A top down approach using open GL--by Edward Angle

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

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

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

Project on '3D BIKE SIMULATION'

Commands to execute:

In terminal, type the following command to compile the program

"gcc project.c -lglut -lGL –lGLU -lm"

After compiling to run the program type

"./a.out"