VISVESVARAYA TECHNOLOGICAL UNIVERSITY
“Jnana Sangama”, Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K. R. Road, V. V. Puram, Bengaluru-560 004

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Computer Graphics Laboratory With Mini Project Report-18CSL68
on

“ROAD ROLLER SIMULATION”

Submitted By

1BI19CS011	**AKASH JAIN**

for the academic year 2021-22

Under the guidance of

Prof. Nagamani D.R
AssistantProfessor

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
“Jnana Sangama”, Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K. R. Road, V. V. Puram, Bengaluru-560 004

Department of Computer Science & Engineering

Certificate

This is to certify that the implementation of **Computer Graphics Laboratory With Mini Project (18CSL68)** entitled **“ROAD ROLLER SIMULATION”** has been successfully completed by

1BI19CS011 **AKASH JAIN**

of VI semester B.E. for the partial fulfillment of the requirements for the Bachelor's degree in **Computer Science & Engineering** of the **Visvesvaraya Technological University** during the academic year **2021-2022**.

Lab In charge:

Prof. Nagamani D.R	**Dr. Girija J.**
Assistant Professor	Professor and Head
Dept. of CSE, BIT	Dept. of CSE, BIT

Examiners: 1) 2)

ACKNOWLEDGEMENT

The knowledge & satisfaction that accompany the successful completion of any task would be incomplete without mention of people who made it possible, whose guidance and encouragement crowned my effort with success. I would like to thank all and acknowledge the help I have received to carry out this Mini Project.

I would like to convey my sincere thanks to **Dr. M. U. Aswath**, Principal, BIT and **Dr. Girija J.**, HOD, Department of CS&E, BIT for being kind enough to provide the necessary support to carry out the mini project.

I am most humbled to mention the enthusiastic influence provided by the lab in-charges **Prof. Nagamani D.R** and on the project for their ideas, time to time suggestions for being a constant guide and co-operation showed during the venture and making this project a great success.

I would also take this opportunity to thank my friends and family for their constant support and help. I'm very much pleasured to express my sincere gratitude to the friendly co-operation showed by all the **staff members** of Computer Science Department, BIT.

AKASH JAIN
1BI19CS011

Table of contents

1. Introduction………………………………………………………………. 1

1.1 Computer Graphics…………………………………………………………1

1.2 Application of Computer Graphics……………………………….….…. 1

1.3 OpenGL…………………………………………………………………. 3

1.4 Problem Statement……………………………………………………… 4

1.5 Objective of The Project …….…………………………………………. 4

1.6 Organization of The Project..…………………………………………… … 5

2. System Specification ……………………………………………….. 6

2.1 Hardware Requirements……………………………………………..… 6

2.2 Software Requirements………………………………………………… 6

3. Design…………………………………………………………………. 7

3.1 Flow Diagram…………………………………………………………… 7

3.2 Description of Flow Diagram…………………………………………… 8

4. Implementation……………………………………………………… 9

4.1 Built In Functions……………………………………………..……… 9

4.2 Source code…………………………………………………………. … 13

5. Snapshots……………………………………………………………….. 29

6. Conclusion……………………………………………………………… 33

6.1 Future Enhancement………………………………………………..…… 33

6.2 Bibliography…………………………………………………………..…… 33

List Of Figures

1.1 OpenGl Block Diagram……………………………………….………......... 4

3.1 Flow Diagram………………………………………………...……. . 7

5.1 Intro Scene……………………………………………………..…… 28

5.2 Road Roller Simualation…………………………………….…….…….. 28

5.3 Road Roller Rotation…………………………………………………. 29

5.4 Road Roller View………………………………………………… …. 29

5.5 Road Roller Translation………………………………………….…… 30

5.6 Road Roller Scaling………………………………………… ………..... 30

5.7 Road Roller Background……………………………………….………... 31

5.8 Road Roller Lighting…………………………………………...…… 31

Chapter -1

INTRODUCTION

1.1 Computer Graphics

Computer graphics is an art of drawing pictures, lines, charts, using computers with the help of programming. Computer graphics is made up of number of pixels. Pixel is the smallest graphical picture or unit represented on the computer screen. Basically, there are 2 types of computer graphics namely,

Interactive Computer Graphics involves a two-way communication between computer and user. The observer is given some control over the image by providing him with an input device. This helps him to signal his request to the computer.

Non-Interactive Computer Graphics otherwise known as passive computer graphics it is the computer graphics in which user does not have any kind of control over the image. Image is merely the product of static stored program and will work according to the instructions given in the program linearly. The image is totally under the control of program instructions not under the user. Example: screen savers.

1.2 Applications of Computer Graphics

Scientific Visualization

Scientific visualization is a branch of science, concerned with the visualization of three-dimensional phenomena, such as architectural, meteorological, medical, biological systems.

Graphic Design

The term graphic design can refer to a number of artistic and professional disciplines which focus on visual communication and presentation

Computer-aided Design

Computer-aided design (CAD) is the use of computer technology for the design of objects, real or virtual. The design of geometric models for object shapes, in particular, is often called computer-aided geometric design (CAGD). The manufacturing process is tied in to the computer description of the designed objects so that the fabrication of a product can be automated using methods that are referred to as CAM, computer-aided manufacturing.

Web Design

Web design is the skill of designing presentations of content usually hypertext or hypermedia that is delivered to an end-user through the World Wide Web, by way of a Web browser.

Digital Art

Digital art most commonly refers to art created on a computer in digital form.

Video Games

A video game is an electronic game that involves interaction with a user interface to generate visual feedback on a raster display device.

Virtual Reality

Virtual reality (VR) is a technology which allows a user to interact with a computer simulated environment. The simulated environment can be similar to the real world. This allows the designer to explore various positions of an object. Animations in virtual reality environments are used to train heavy equipment operators or to analyse the effectiveness of various cabin configurations and control placements.

Computer Simulation

A computer simulation, a computer model or a computational model is a computer program, or network of computers, that attempts to simulate an abstract model of a particular system.

Education and Training

Computer simulations have become a useful part of mathematical modelling of many natural systems in physics, chemistry and biology, human systems in economics, psychology, and social science and in the process of engineering new technology, to gain insight into the operation of those systems, or to observe their behaviour. Most simulators provide screens for visual display of the external environment with multiple panels is mounted in front of the simulator.

Image Processing

The modification or interpretation of existing pictures such as photographs and TV scans, is called image processing. In computer graphics, a computer is used to create a picture. Image processing techniques, on the other hand, are used to improve picture quality, analyse images, or recognize visual patterns for robotics applications

1.3 OpenGL

OpenGL has become a widely accepted standard for developing graphics applications. Most of our applications will be designed to access OpenGL directly through functions in the three libraries. Functions in main GL libraries have names that begin with the letters gl and are stored in a library usually referred to as GL.

The second is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects and simplifying viewing. All function in GLU can be created from the core GL library. The GLU library is available in all OpenGL implementations. Functions in the GLU library starts with the letters glu.

The third is the OpenGL Utility Toolkit (GLUT). It provides the minimum functionality that should be formulated in modern windowing systems.

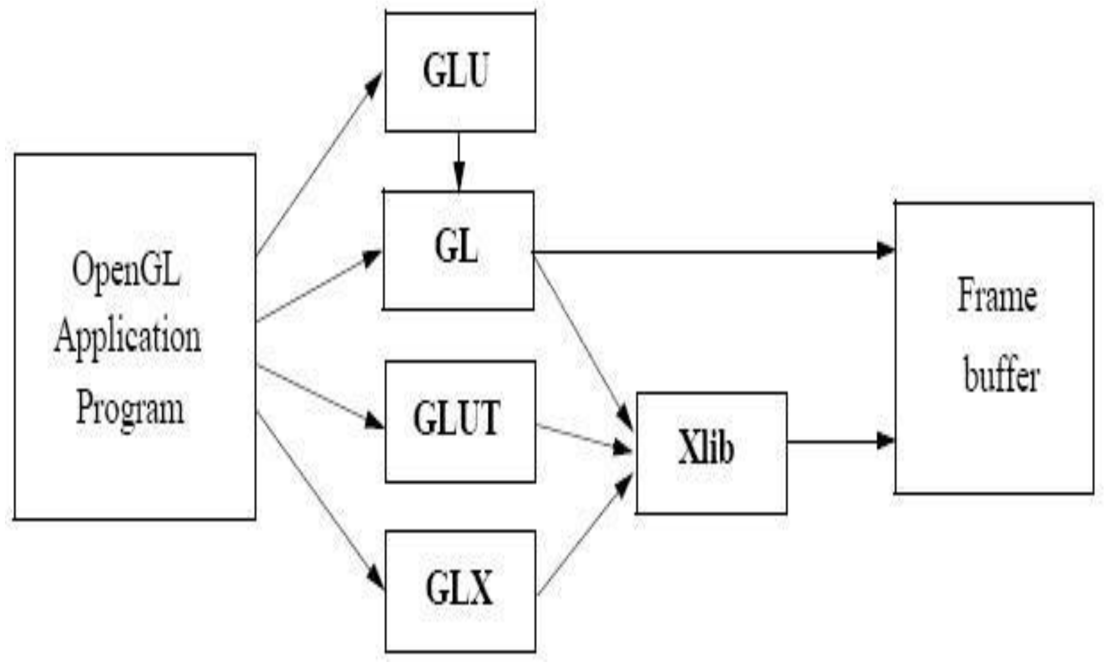


Figure 1.1 Basic block diagram of OpenGL

1.4 Problem Statement

“To develop an animation demonstrating the simulation of Road Roller”.

1.5 Objectives of the Project

This package “Simulation of Road Roller” is to simulate simple movement of a road roller. We have the wheels of the road roller rotating as the roller itself moves forward.

We use mouse for the transformation functions and keyboard keys for changing the camera views.

1.6 Organisation of the Project

The project was organised in a systematic way. First we analysed what are the basic features to be included in the project to make it acceptable. As it is a graphics oriented project, we made the sketches prior, so as to have an idea like how our output must look like. After all these, the source code was formulated as a paper work. All the required software were downloaded. Finally, the successful implementation of the project.

Chapter -2

SYSTEM SPECIFICATION

2.1 Hardware Requirements

- Main Processor : PENTIUM III
- Processor Speed: 800 MHz
- RAM Size : 128 MB DDR
- Keyboard : Standard qwerty serial or PS/2 keyboard
- Mouse : Standard serial or PS/2 mouse
- Cache memory : 256 KB

2.2 Software Requirements

- Operating System: Windows 10 or Linux (Fedora) or macOS
- Hypervisor used : Docker
- Compiler used : g++
- Language used : C++ language
- Editor : Visual Studio Code
- Toolkit : GLUT Toolkit

Chapter -3

DESIGN

3.1 Flow Diagram

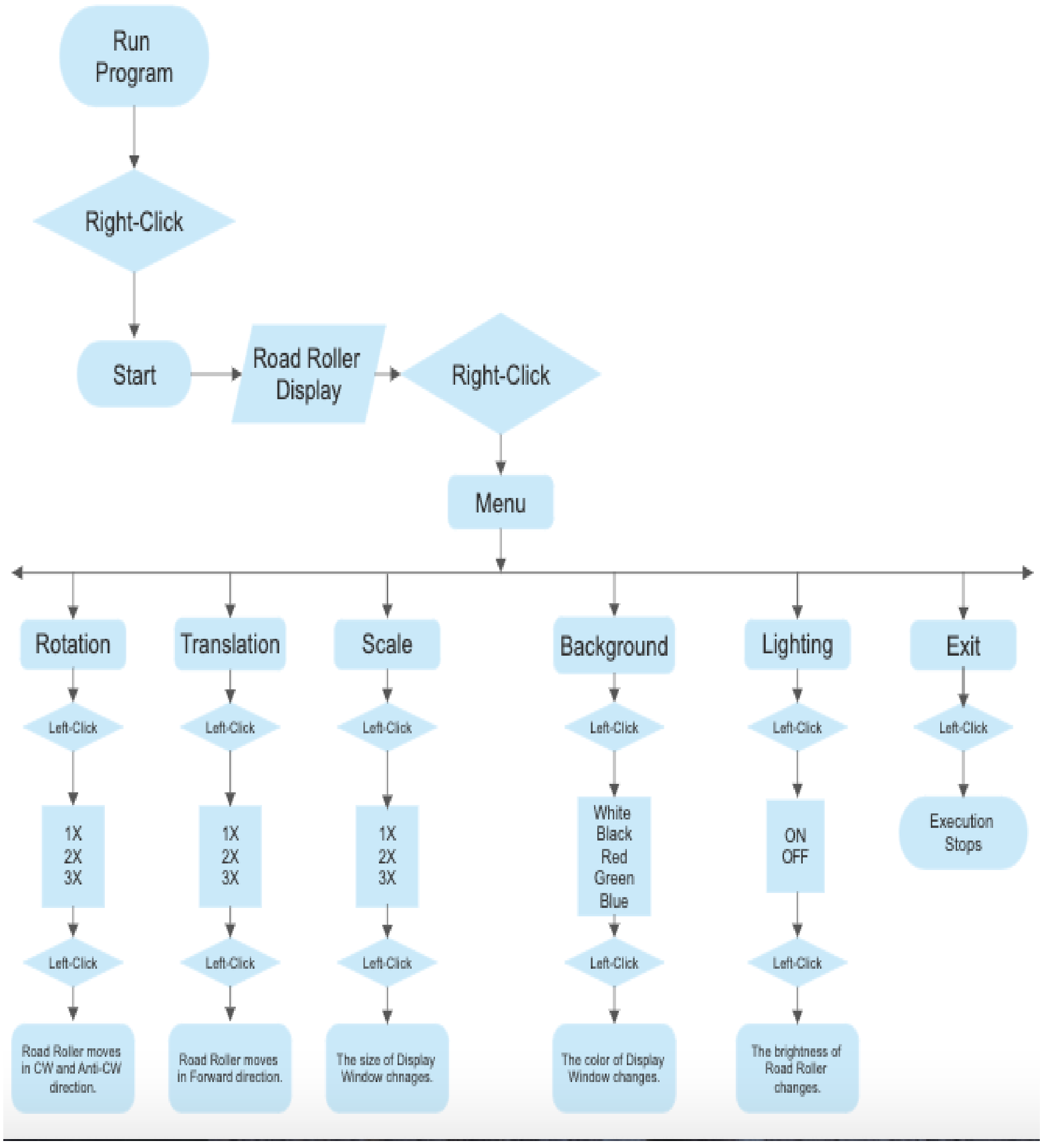

Run Program
Right-Click
Start
Road Roller Display
Right-Click
Menu
Rotation
Translation
Scale
Background
Lighting
Exit
Left-Click
1X 2X 3X
1X 2X 3X
1X 2X 3X
White Black Red Green Blue
ON OFF
Execution Stops
Left-Click
Road Roller moves in CW and Anti-CW direction.
Road Roller moves in Forward direction.
The size of Display Window chnages.
The color of Display Window changes.
The brightness of Road Roller changes.

Figure 3.1 : Flow diagram of scene change

3.2 Description of Flow Diagram

The description of the flow diagram is as follows:

Step 1: Start

Step 2: The user is presented with intro in **fig 5.1** and presses ‘right-click’ and press START to proceed to next scene.

Step 3: The user is displayed with the road roller display on the screen in **fig 5.2**

Step 4: Using ‘right-click’ user can select any option within the menu showing Rotation, Translation, Scale, Background, Lightening, Exit.

Step 5: The user selects Rotation from the list as in **fig 5.3** and can select any speed from the dropdown menu.

Step 5: In **fig 5.4**, user clicks the keyboard buttons ‘A’ and ‘D’ to rotate the Road Roller.

Step 6: The user selects Translation from the list as in **fig 5.5** and can select any speed from the dropdown menu.

Step 7: Then the Road Roller moves in the forward direction.

Step 8: The user selects Scale from the list as in **fig 5.6** and can select any speed from the dropdown menu.

Step 9: Then the size of Road Roller display screen changes.

Step 10: The user selects Background from the list as in **fig 5.7** and can select any color from the dropdown menu.

Step 11: As the user selects Red color, the background changes to red.

Step 12: The user selects Lighting from the list as in **fig 5.8** and can select ON/OFF from the dropdown menu.

Step 13: As the user selects OFF, the brightness of Road Roller decreases.

Step 14: The user can click Exit to stop execution.

Step 15: Execution stops.

Chapter -4

IMPLEMENTATION

4.1 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 GLUT display mode bit masks. Description: The initial display mode is used when creating 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: intglutCreateWindow (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. 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.

6. glMatrixMode()

The two most important matrices are the model-view and projection matrix. At many times, 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.

7. glTranslate(GLfloat X, GLfloat Y, GLfloat Z)

glTranslate produces a translation by x y z. If the matrix mode is either GL_MODEL_VIEW or GL_PROJECTION, all objects drawn after a call to glTranslate are translated.

8. glRotatef(GLdouble angle, GLdouble X, GLdouble Y, GLdouble Z)

glRotatef produces a rotation of angle degrees around the vector x y z. If the matrix mode is either GL_MODEL_VIEW or GL_PROJECTION, all objects drawn after glRotatef is called are rotated. Use glPushMatrix() and glPopmatrix() to save and restore the unrotated coordinate system.

9. glScalef(GLfloat x, GLfloat y, GLfloat z)

This function multiplies the current matrix by a general scaling matrix. This function does not return a value.

10. glPushMatrix()

There is a stack of matrices for each of the matrix mode. In GL_MODELVIEW mode, the stack depth is atleast 32. In other modes, GL_COLOR, GL_PROJECTION, and GL_TEXTURE, the depth is atleast 2. The current matrix in any mode is the matrix on the top of the stack for that mode.

11. glPopMatrix()

glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack. Initially, each of the stack contains one matrix, an identity matrix. It is an error to push a full matrix stack or pop a matrix stack that contains only a single matrix. In either case, the error flag is set and no other change is made to GL state.

12. glutSwapBuffers()

Usage: void glutSwapBuffers(void);

Description: Performs a buffer swap on the layer in use for the current window. Specifically, glutSwapBuffers promotes the contents of the front buffer. The contents of the back buffer then become undefined.

13. glPointSize(GLfloat size)

glPointSize specifies the rasterized diameter of points. This value will be used rasterize points. Otherwise, the value written to the shading language built-in variable gl-PointSize will be used. The point size specified by glPointSize is always returned when GL_POINT_SIZE is queried.

14. glutKeyboardFunc()

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.

15. glLoadIdentity(void)

glLoadIdentity replaces the current matrix with the identity matrix. It is semantically equivalent to calling glLoadMatrix with the identity matrix.

4.3 SOURCE CODE

main.c

```
#include <windows.h>
#include <stdio.h>
#include <GL/glut.h>
#include <math.h>
#define PI  3.1415927
float ang = 0.0;
float angStep = 0.3;
float mov = 2.0;
float movStep = 0.0005;
int frontCleared = 0;
int rev = 0;
int lightOn = 1;
float universalScale = 0.6;
float x = 0.0f, z = 5.0f;

GLfloat ambientCube[] = { 1.0,1.0,0.0,0.0 };
GLfloat diffuseCube[] = { 1.0,1.0,0.0,0.0 };
GLfloat specularCube[] = { 1.0,0.0,0.0,1.0 };
GLfloat shineCube[] = { 10.0 };
GLfloat intensityCube[] = { 1.0,1.0,1.0,1.0 };
GLfloat pos1[] = { 10.0,5.0,-1.0,0.0 };
void stripPieces(int, int, int, int);
void roadPieces(int, int, int, int);
void front();
void menu();
void draw_cylinder(GLfloat radius, GLfloat height, GLfloat R, GLfloat G, GLfloat B) {
GLfloat x = 0.0;
GLfloat y = 0.0;
GLfloat angle = 0.0;
GLfloat angle_stepsize = 0.1;
glDisable(GL_LIGHTING);
```

```
glBegin(GL_QUAD_STRIP);
angle = 0.0;
while (angle < 2 * PI) {
x = radius * cos(angle);
y = radius * sin(angle);
glColor3f(G + x, R + y, B);
glVertex3f(x, y, height);
glVertex3f(x, y, 0.0);
angle = angle + angle_stepsize;
}
glVertex3f(radius, 0.0, height);
glVertex3f(radius, 0.0, 0.0);
glEnd();
/** Draw the circle on top of cylinder */
glBegin(GL_POLYGON);
angle = 0.0;
while (angle < 2 * PI)
{
x = radius * cos(angle);
y = radius * sin(angle);
glColor3f(0.2409, 0.0, 0.0);
glVertex3f(x, y, height);
angle = angle + angle_stepsize;
}
glVertex3f(radius, 0.0, height);
glEnd();
glBegin(GL_POLYGON);
angle = 0.0;
while (angle < 2 * PI) {
x = radius * cos(angle);
y = radius * sin(angle);
glColor3f(0.2409, 0.0, 0.0);
//   lightingWheel();
glVertex3f(x, y, 0);
angle = angle + angle_stepsize;
}
glVertex3f(radius, 0.0, height);
glEnd();
}
GLfloat vertex[][3] = { {-5.5,-1.4,-1.0},{1.0,-1.4,-1.0},{1.0,2.5,-1.0},{-5.5,1.0,-1.0},{-
3.0,-1.4,1.0},{1.0,-1.4,1.0},{1.0,2.5,1.0},{-5.5,1.0,1.0} };
GLfloat normal[][3] = { {-1.0,-1,-1},{1,-1.0,-1.0},{1.0,1.0,-1.0},{-1.0,1.0,-1.0},{-1.0,-
1.0,1.0},{1.0,-1.0,1.0},{1.0,1.0,1.0},{-1.0,1.0,1.0} };
GLfloat color[][3] = {
{0.772549,0.662745,0.48},{0.28627,0,0},{0.28627,0,0},{0.28627,0,0},{0.28627,0,0},{0.
28627,0,0},{0.28627,0,0},{0.28627,0,0} };

GLfloat vertexStrip[][3] = { {-0.5,-1.0,-1.0},{0.5,-1.0,-1.0},{0.5,1.0,-1.0},{-0.5,1.0,-
1.0},{-0.5,-1.0,1.0},{0.5,-1.0,1.0},{0.5,1.0,1.0},{-0.5,1.0,1.0} };

GLfloat roadStrip[][3] = { {-0.5,-1.0,-1.0},{0.5,-1.0,-1.0},{0.5,1.0,-1.0},{-0.5,1.0,-1.0},{-
0.5,-1.0,1.0},{0.5,-1.0,1.0},{0.5,1.0,1.0},{-0.5,1.0,1.0} };

static GLdouble viewer[] = { 8.0,0.0,5.0 };
void polygon(int a, int b, int c, int d)
{
glBegin(GL_POLYGON);
glColor3fv(color[0]);
glNormal3fv(normal[a]);
glVertex3fv(vertex[a]);
glColor3fv(color[0]);
glNormal3fv(normal[b]);
glVertex3fv(vertex[b]);
glColor3fv(color[0]);
glNormal3fv(normal[c]);
glVertex3fv(vertex[c]);
```

```
glColor3fv(color[0]);
glNormal3fv(normal[d]);
glVertex3fv(vertex[d]);
glEnd();
}
void colorCube()
{
polygon(0, 3, 2, 1);
polygon(2, 3, 7, 6);
polygon(0, 4, 7, 3);
polygon(1, 2, 6, 5);
polygon(4, 5, 6, 7);
polygon(0, 1, 5, 4);
}
void strips()
{
glColor3f(1.0, 0.0, 0.0);
stripPieces(0, 3, 2, 1);
glColor3f(0.0, 1.0, 0.0);
stripPieces(2, 3, 7, 6);
glColor3f(0.0, 0.0, 1.0);
stripPieces(0, 4, 7, 3);
glColor3f(1.0, 1.0, 0.0);
stripPieces(1, 2, 6, 5);
glColor3f(0.0, 1.0, 1.0);
stripPieces(4, 5, 6, 7);
glColor3f(1.0, 0.0, 1.0);
stripPieces(0, 1, 5, 4);
}
void roads()
{
roadPieces(0, 3, 2, 1);
roadPieces(2, 3, 7, 6);
roadPieces(0, 4, 7, 3);
roadPieces(1, 2, 6, 5);
roadPieces(4, 5, 6, 7);
roadPieces(0, 1, 5, 4);
}

void stripPieces(int a, int b, int c, int d)
{
glBegin(GL_POLYGON);
glVertex3fv(vertexStrip[a]);
glVertex3fv(vertexStrip[b]);
glVertex3fv(vertexStrip[c]);
glVertex3fv(vertexStrip[d]);
glEnd();
}
void roadPieces(int a, int b, int c, int d)
{
glBegin(GL_POLYGON);
glColor3f(0.0, 0.0, 0.0);
glVertex3fv(roadStrip[a]);
glColor3f(0.0, 0.0, 0.0);
glVertex3fv(roadStrip[b]);
glColor3f(0.0, 0.0, 0.0);
glVertex3fv(roadStrip[c]);
glColor3f(0.0, 0.0, 0.0);
glVertex3fv(roadStrip[d]);
glEnd();
}
void lightingCube()
{
glMaterialfv(GL_FRONT, GL_AMBIENT, ambientCube);
glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuseCube);
glMaterialfv(GL_FRONT, GL_SPECULAR, specularCube);
```

```
glLightfv(GL_LIGHT0, GL_POSITION, pos1);
glLightfv(GL_LIGHT0, GL_DIFFUSE, intensityCube);
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
if (frontCleared == 0)
front();
else
{
gluLookAt(0.0 + x, 0.0, 0.0 + z,
0, 0, 0,
0, 1, 0);
glScalef(universalScale, universalScale, universalScale);
glPushMatrix();
glTranslatef(0.0, -6.5, 0.0);
glScalef(15.0, 5.0, 10.0);
roads();
glPopMatrix();
glTranslatef(mov, 0.0, 0.0);
glPushMatrix();       //front cyl connector1
if (lightOn)
glEnable(GL_LIGHTING);
else
glDisable(GL_LIGHTING);
glTranslatef(1.0, -0.4, 1.19);
glRotatef(15, 0.0, 1.0, 0.0);
glScalef(0.1, 0.7, 0.02);
lightingCube();
strips();
glPopMatrix();
glPushMatrix();       //front connector1 connector to body
```

```
if (lightOn)
glEnable(GL_LIGHTING);
else
glDisable(GL_LIGHTING);
glTranslatef(1.0, 0.35, 1.10);
glRotatef(15, 0.0, 1.0, 0.0);
glRotatef(90, 1.0, 0.0, 0.0);
glScalef(0.1, 0.15, 0.05);
lightingCube();
strips();
glPopMatrix();
glPushMatrix();         //front cyl connector 2
if (lightOn)
glEnable(GL_LIGHTING);
else
glDisable(GL_LIGHTING);
glTranslatef(0.5, -0.4, -0.29);
glRotatef(15, 0.0, 1.0, 0.0);
glScalef(0.1, 0.7, 0.02);
lightingCube();
strips();
glPopMatrix();
glPushMatrix();         //connector2 connector to body
if (lightOn)
glEnable(GL_LIGHTING);
else
glDisable(GL_LIGHTING);
glTranslatef(0.5, 0.30, -0.10);
glScalef(0.1, 0.03, 0.2);
glRotatef(15, 0.0, 1.0, 0.0);
glRotatef(90, 1.0, 0.0, 0.0);
lightingCube();
strips();
```

```
glPopMatrix();
glPushMatrix();       //shelter piller 1
if (lightOn)
glEnable(GL_LIGHTING);
else
glDisable(GL_LIGHTING);
glTranslatef(3.5, 2.0, 0.4);
glScalef(0.1, 0.8, 0.05);
glRotatef(15, 0.0, 1.0, 0.0);
glRotatef(90, 0.0, 0.0, 1.0);
lightingCube();
strips();
glPopMatrix();
glPushMatrix();       //shelter piller 2
if (lightOn)
glEnable(GL_LIGHTING);
else
glDisable(GL_LIGHTING);
glTranslatef(3.0, 2.0, -0.5);
glScalef(0.1, 0.8, 0.05);
glRotatef(15, 0.0, 1.0, 0.0);
glRotatef(90, 0.0, 0.0, 1.0);
lightingCube();
strips();
glPopMatrix();

glPushMatrix();     //shelter
if (lightOn)
glEnable(GL_LIGHTING);
else
glDisable(GL_LIGHTING);
glTranslatef(2.8, 2.3, -0.0);
glScalef(0.6, 0.2, 0.5);
```

```
glRotatef(15, 0.0, 1.0, 0.0);
glRotatef(90, 0.0, 0.0, 1.0);
lightingCube();
strips();
glPopMatrix();
glPushMatrix();
if (lightOn)
glEnable(GL_LIGHTING);
else
glDisable(GL_LIGHTING);
glTranslatef(3.0, 0.5, 0.0);
glRotatef(15, 0.0, 1.0, 0.0);
glScalef(0.5, 0.5, 0.5);
lightingCube();
colorCube();
glPopMatrix();
glPushMatrix();         //front cylinder
glTranslatef(0.5, -1.0, -0.25);
glRotatef(15, 0.0, 1.0, 0.0);
glRotatef(ang, 0.0, 0.0, 1.0);
draw_cylinder(0.48, 1.5, 0.09411, 0.70588, 0.470588);
glPopMatrix();

glPushMatrix();     //back cyl 1
glTranslatef(3.0, -0.48, 0.7);
glRotatef(15, 0.0, 1.0, 0.0);
glRotatef(ang, 0.0, 0.0, 1.0);
draw_cylinder(1.0, 0.3, 0.09411, 0.70588, 0.470588);
glPopMatrix();

glPushMatrix();//back cyl 2
glTranslatef(3.0, -0.48, -1.0);
glRotatef(15, 0.0, 1.0, 0.0);
glRotatef(ang, 0.0, 0.0, 1.0);
draw_cylinder(1.0, 0.3, 0, 0, 0);
glPopMatrix();
}
glutSwapBuffers();
}
void idle()
{
mov -= movStep;
ang += angStep;
if (mov < -5.0)
mov = 2.0;
glutPostRedisplay();
}
void keyboard(unsigned char key, int x1, int y1)
{
switch (key) {
case 'a':
if (x >= -5.0 && z <= 5.0) {
x = x - 0.5;
z = 5.0 - fabs(x);
}
glutPostRedisplay();
break;
case 'd':
if (x <= 5.0 && z <= 5.0)
{
x = x + 0.5;
z = 5.0 - fabs(x);
}
glutPostRedisplay();
break;
}
```

```
}
void reshape(int w, int h)
{
glViewport(0, 0, w, h);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
if (w < h)
glOrtho(-2.0, 2.0, -2.0 * (GLfloat)h / (GLfloat)w, 2.0 * (GLfloat)h / (GLfloat)w, -50,
50.0);
else
glOrtho(-2.0 * (GLfloat)w / (GLfloat)h, 2.0 * (GLfloat)w / (GLfloat)h, -2.0, 2.0, -50.0,
50.0);
glMatrixMode(GL_MODELVIEW);
}
void init()
{
glClearColor(1.0, 1.0, 1.0, 0.0);
glEnable(GL_LIGHT0);
glEnable(GL_LIGHTING);
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LEQUAL);
}
void draw_text(float x1, float y1, float z1, const char* s)
{
int i;
glRasterPos3f(x1, y1, z1);
for (i = 0; s[i] != '\0'; i++)
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, s[i]);
}
void front()
{
glClearColor(1.0, 0.0, 0.0, 0.0);
glColor3f(0.0, 0.0, 1.0);
```

```
draw_text(-1.5, 1.7, 0.0, "    Bangalore Institute Of Technology");
draw_text(-1.5, 1.5, 0.0, "Department of Computer Science & Engineering ");
draw_text(-1.5, 1.2, 0.0, "     COMPUTER GRAPHICS AND VISULIZATION ");
draw_text(-0.5, 0.5, 0.0, "           Package on");
draw_text(-0.5, 0.3, 0.0, "Road Roller Simulation ");
draw_text(-2.0, 0.1, 0.0, "Submitted by:
Submitted to:");
draw_text(-2.0, -0.1, 0.0, "ARYMANN SINHA  (1BI19CS029)
              Mr.Manjunath");
draw_text(-2.0, -0.3, 0.0, "
Asst.Professor");
draw_text(-2.0, -0.9, 0.0, "                         Press A and D to control the road roller
views");
}
void mouse(int btn, int state, int x, int y)
{
if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
{
glutPostRedisplay();
}
}
void main_menu(int index)
{
switch (index)
{
case 2:
movStep = 0.002;
break;

case 3:
movStep = 0.006;
break;
case 4:
```

```
movStep = 0.008;
break;
case 5:
angStep = 0.3;
break;
case 6:
angStep = 0.4;
break;
case 7:
angStep = 0.5;
break;
case 8:
universalScale = 0.6;
break;
case 9:
universalScale = 0.7;
break;
case 10:
universalScale = 0.8;
break;
case 11:
glClearColor(1.0, 1.0, 1.0, 1.0);
break;
case 12:
glClearColor(0.0, 0.0, 0.0, 0.0);
break;
case 13:
glClearColor(1.0, 0.0, 0.0, 0.0);
break;
case 14:
glClearColor(0.0, 1.0, 0.0, 0.0);
break;
case 15:
```

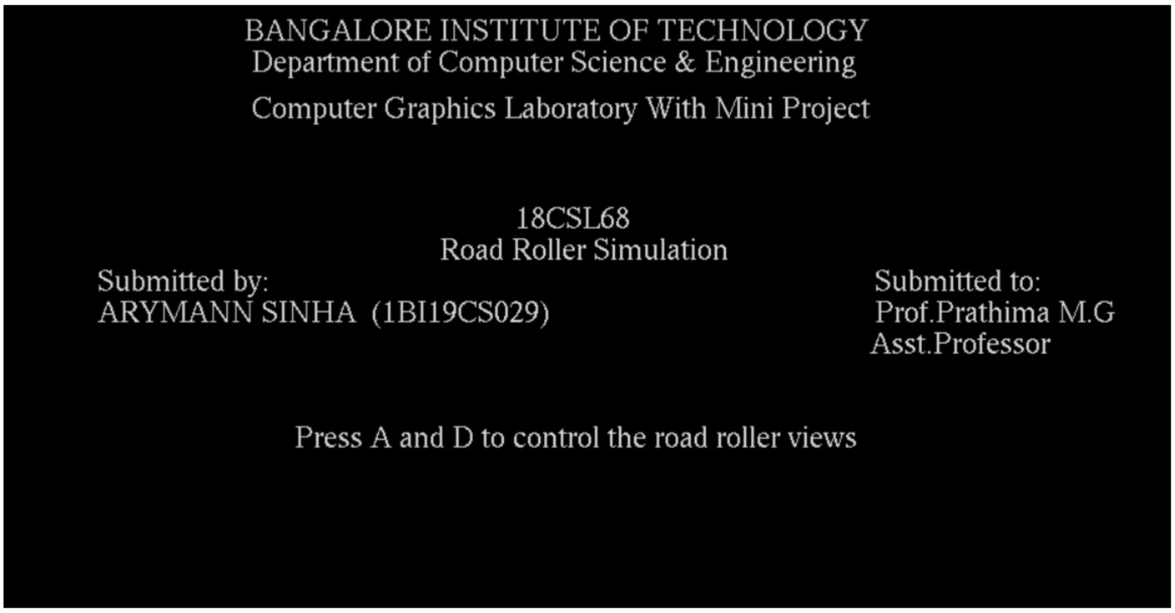
```
glClearColor(0.0, 0.0, 1.0, 0.0);
break;
case 16:
exit(0);
break;
case 17:
lightOn = 1;
glutPostRedisplay();
break;
case 18:
lightOn = 0;
glutPostRedisplay();
break;
case 100:
menu();
frontCleared = 1;
glClearColor(1.0, 1.0, 1.0, 1.0);
glutPostRedisplay();
}
}
void menu()
{
int primaryMenu, scaleMenu, translationMenu, rotationMenu, backgroundMenu,
lightMenu;
translationMenu = glutCreateMenu(main_menu);
glutAddMenuEntry("1x", 2);
glutAddMenuEntry("2x", 3);
glutAddMenuEntry("3x", 4);

rotationMenu = glutCreateMenu(main_menu);
glutAddMenuEntry("1x", 5);
glutAddMenuEntry("2x", 6);
glutAddMenuEntry("3x", 7);
```

```
scaleMenu = glutCreateMenu(main_menu);
glutAddMenuEntry("1x", 8);
glutAddMenuEntry("2x", 9);
glutAddMenuEntry("3x", 10);

backgroundMenu = glutCreateMenu(main_menu);
glutAddMenuEntry("White", 11);
glutAddMenuEntry("Black", 12);
glutAddMenuEntry("Red", 13);
glutAddMenuEntry("Green", 14);
glutAddMenuEntry("Blue", 15);

lightMenu = glutCreateMenu(main_menu);
glutAddMenuEntry("On", 17);
glutAddMenuEntry("Off", 18);

primaryMenu = glutCreateMenu(main_menu);
glutAddSubMenu("Rotation", rotationMenu);
glutAddSubMenu("Translation", translationMenu);
glutAddSubMenu("Scale", scaleMenu);
glutAddSubMenu("Background", backgroundMenu);
glutAddSubMenu("Lighting", lightMenu);
glutAddMenuEntry("Exit", 16);
glutAttachMenu(GLUT_RIGHT_BUTTON);
}
void menu1()
{
glutCreateMenu(main_menu);
glutAddMenuEntry("Start", 100);
glutAttachMenu(GLUT_RIGHT_BUTTON);
}
```

```
int main(int arc, char** argv)
{
glutInit(&arc, argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
glutInitWindowSize(750, 500);
glutInitWindowPosition(0, 0);
glutCreateWindow("RoadRoller");
glutKeyboardFunc(keyboard);
init();
menu1();
glutDisplayFunc(display);
glutIdleFunc(idle);
glutReshapeFunc(reshape);
glutMainLoop();
return 0;
}
```

Chapter -5

SNAPSHOTS



BANGALORE INSTITUTE OF TECHNOLOGY
Department of Computer Science & Engineering
Computer Graphics Laboratory With Mini Project

18CSL68
Road Roller Simulation

Submitted by:
ARYMANN SINHA (1BI19CS029)

Submitted to:
Prof.Prathima M.G
Asst.Professor

Press A and D to control the road roller views

Figure 5.1 : Intro scene

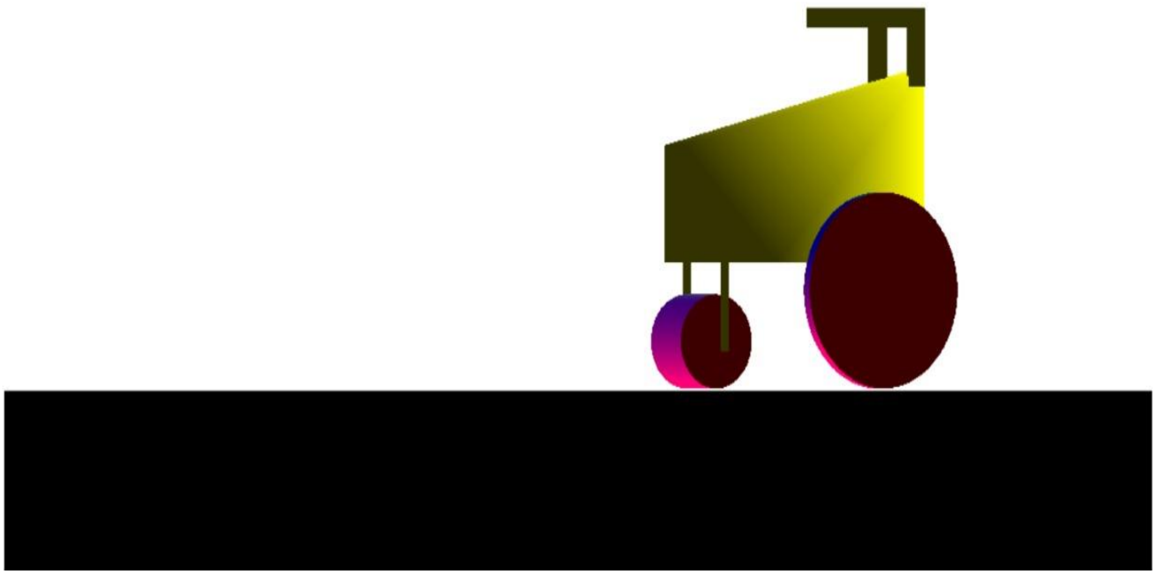


Figure 5.2 : Road Roller Simulating

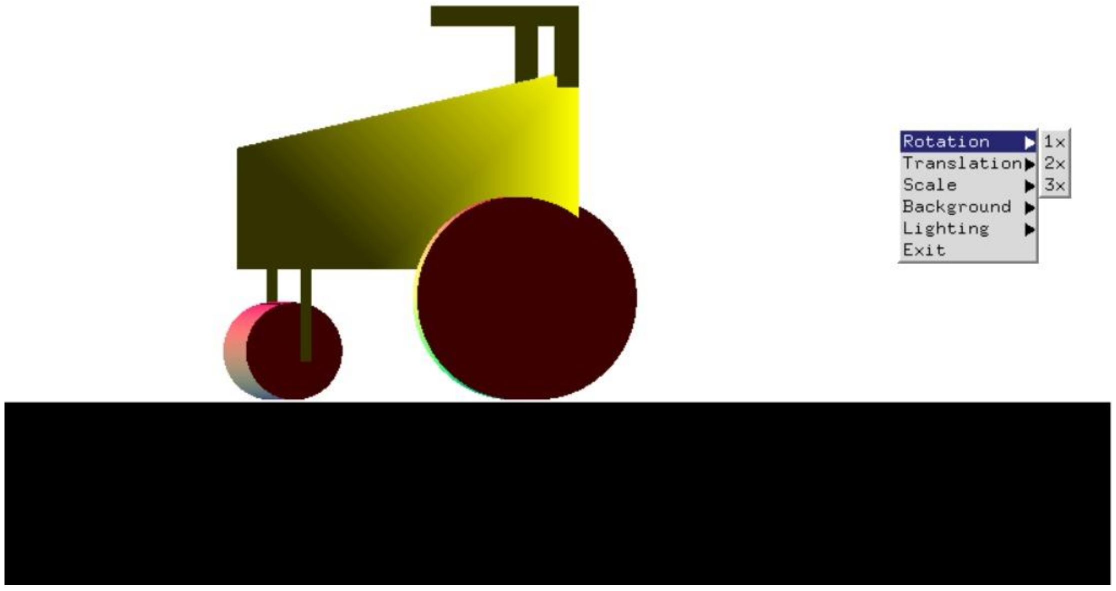


Figure 5.3: Road Roller Rotation(1x,2x,3x)

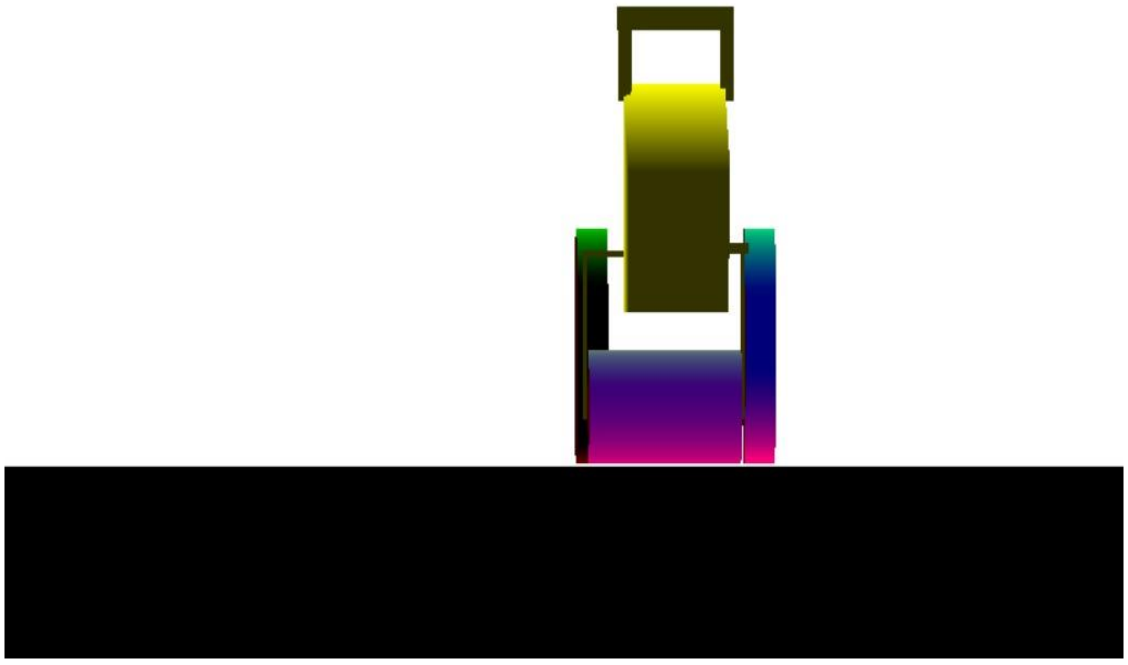


Figure 5.4 Road Roller View(Using A and D)

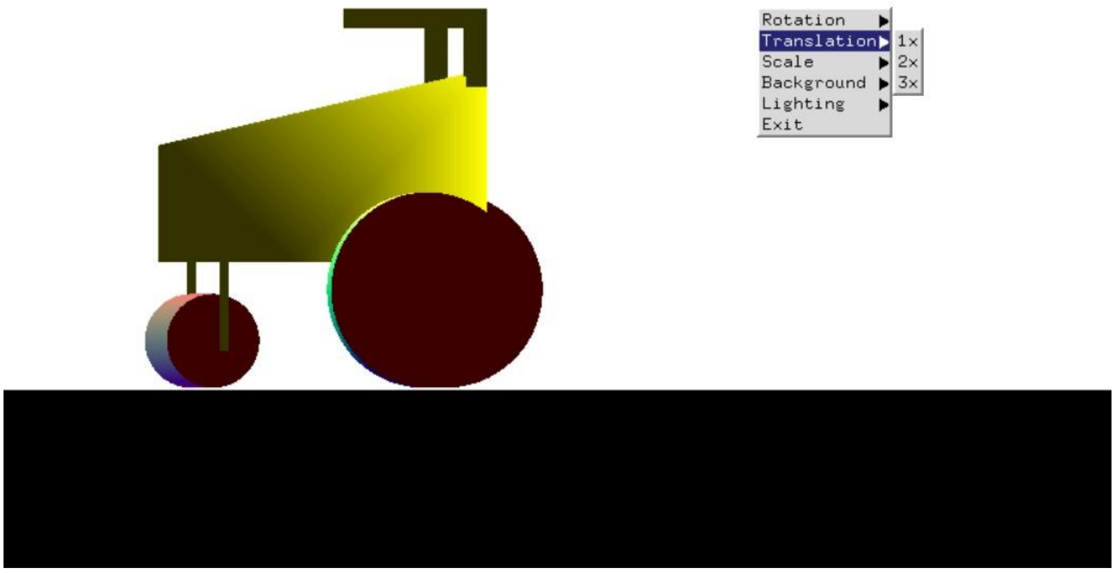


Figure 5.5: Road Roller Translation(1x,2x,3x)

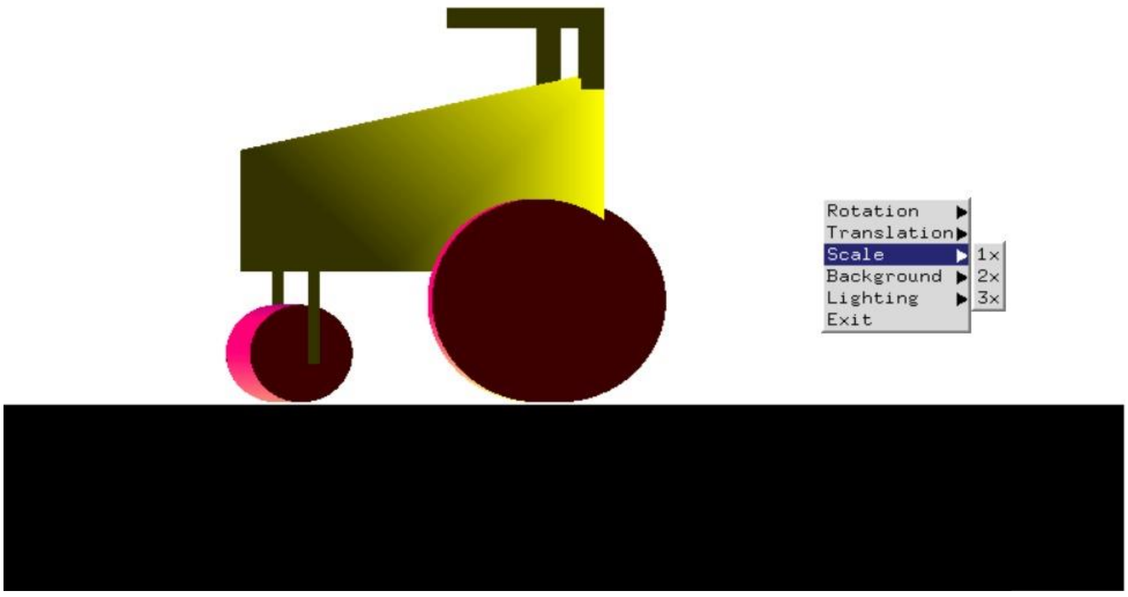


Figure 5.6: Road Roller Scaling(1x,2x,3x)

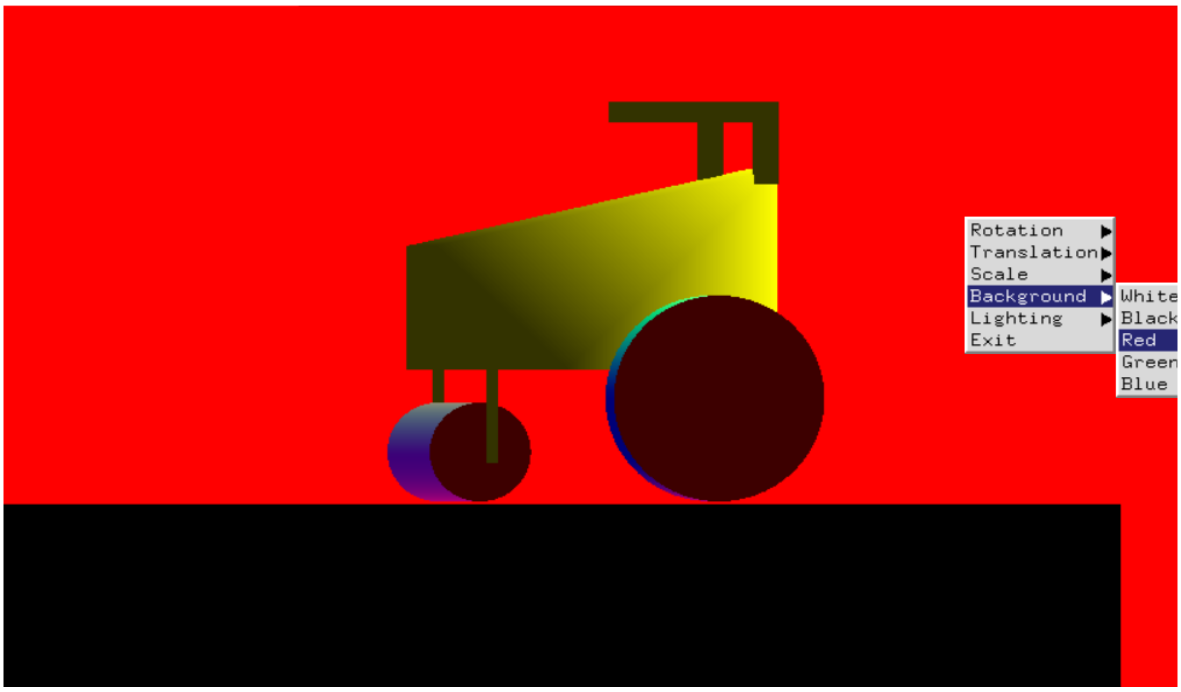
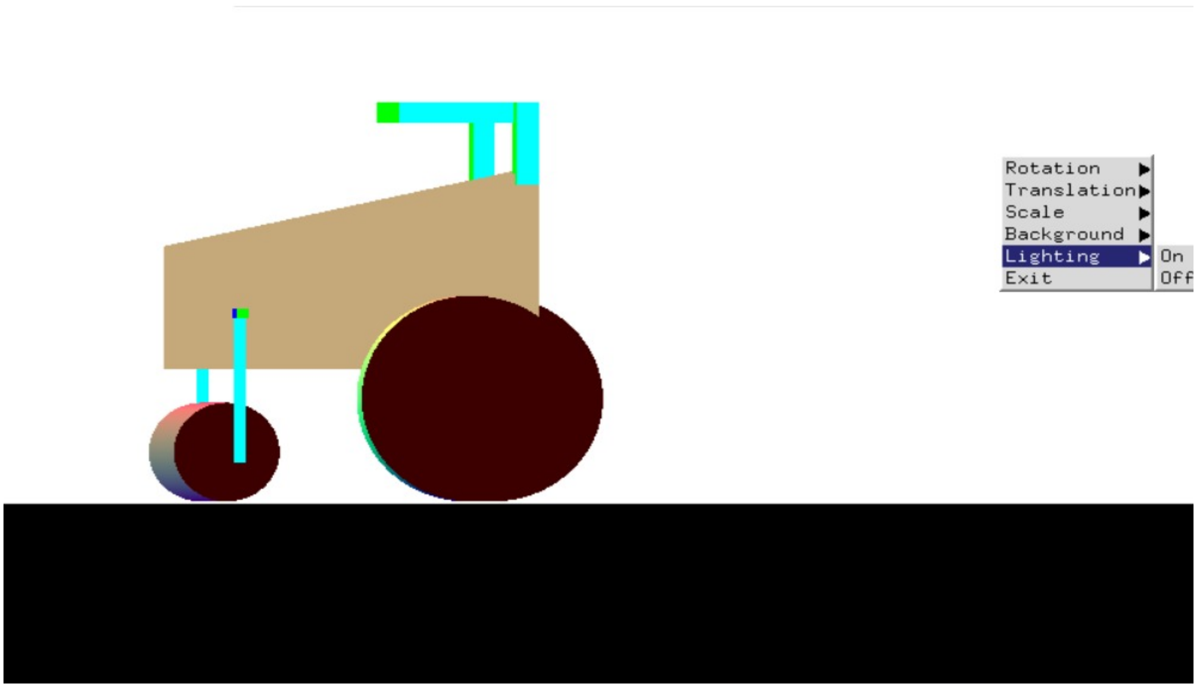


Figure 5.7: Road Roller Background(White,Black,Red,Green,Blue)



Chapter -6

CONCLUSION

It was a wonderful learning experience for me working on this package. This package took me through various phases of project development and gave me real insight into the world of software engineering. The joy of working and the thrill involved while tackling the various problems and challenges gave me a feel of developers industry.

It was due to project that I came to know how professional software's are designed.

I enjoyed each and every bit of work I had to put into this package.

6.1 Future Enhancements

- A vast amount of future work can be possible by following investigations and strategies.
- More features can be included and can be modified in a more versatile way.

BIBLIOGRAPHY

Reference Books

[1] Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd/4th Edition, Pearson Education, 2011

[2] James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer Graphics with OpenGL, Pearson Education

[3] Macro Cantu: Mastering Delphi

Websites

[1] https://www.opengl.org/

[2] https://learnopengl.com/