VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum: 590018

A Mini Project Report (17CSL68/18CSL67)

on

Ball in a basket Game

A mini project report submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Engineering

in

Computer Science & Engineering

Submitted by

Rutik Panchal (1AY18CS096)

Sahil khandelwal(1AY18CS102)

Under the guidance of

Mrs. Swati Mohan

Department of Computer Science & Engineering

Acharya Institute of Technology

Department of Computer Science & Engineering

Soladevanahalli, Bangalore-560107

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnanasangama, Belagavi, Karnataka

ACHARYA INSTITUTE OF TECHNOLOGY

(Affiliated to Visvesvaraya Technological University, Belgaum)
Soladevanahalli, Bangalore – 560 107

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Certificate

Certified that the Computer Graphics Mini Project entitled **“Ball in a Basket”** is a bonafide work carried out by **Mr. Rutik Panchal (1AY18CS096) and Mr. Sahil khandelwal(1AY18CS102)** in partial fulfillment for the award of degree of **Bachelor of Engineering in Computer Science & Engineering of the Visvesvaraya Technological University**, Belgaum during the academic year **2020-2021.** It is certified that all corrections/ suggestions indicated for internal assessments have been incorporated in the Report deposited in the departmental library. The Mini project report has been approved as it satisfies the academic requirements in respect of Mini Project work prescribed for the **Bachelor of Engineering Degree**.

Signature of Guides **Signature of H.O.D**

Name of the Examiners **Signature with date**

1.

2.

ACKNOWLEDGEMENT

We express our gratitude to our institution and management for providing us with good infrastructure, laboratory facilities and inspiring staff, and whose gratitude was of immense help in completion of this mini project successfully.

We express our sincere gratitude to our principal, **Dr. M R Prakash** for providing us the required environment and for his valuable suggestions.

Our sincere thanks to **Dr. Prashanth C M,** Head of the Department. Computer Science and Engineering, Acharya Institute of Technology for his valuable support and for rendering us the resources for this mini project.

We heartily thank **Prof. Varalakshmi B D, Prof. Vani K S and Prof. Swathi Mohan** Assistant Professors, Department of Computer Science and Engineering, Acharya Institute of Technology who guided us with valuable suggestions in completing this mini project at every stage.

Our gratitude thanks should be rendered to many people who helped us in all possible ways.

Name: Rutik Panchal(1AY18CS096)
Sahil khandelwal(1AY18CS102)

CONTENTS

Chapter-1
Introduction

<u>Introduction</u>

What Is OpenGL?

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that you use to specify the objects and operations needed to produce interactive three-dimensional applications.

OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, you must work through whatever windowing system controls the particular hardware you're using. Similarly, OpenGL doesn't provide high-level commands for describing models of three-dimensional objects. Such commands might allow you to specify relatively complicated shapes such as automobiles, parts of the body, airplanes, or molecules. With OpenGL, you must build up your desired model from a small set of *geometric primitives* - points, lines, and polygons.

A sophisticated library that provides these features could certainly be built on top of OpenGL. The OpenGL Utility Library (GLU) provides many of the modeling features, such as quadric surfaces and NURBS (Non-Uniform Rational B-Splines) curves and surfaces. GLU is a standard part of every OpenGL implementation. Also, there is a higher-level, object-oriented toolkit, Open Inventor, which is built atop OpenGL, and is available separately for many implementations of OpenGL.

OpenGL-Related Libraries

OpenGL provides a powerful but primitive set of rendering commands, and all higher-level drawing must be done in terms of these commands. Also, OpenGL programs have to use the underlying mechanisms of the windowing system. A number of libraries exist to allow you to simplify your programming tasks, including the following:

- The OpenGL Utility Library (GLU) contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections, performing polygon tessellation, and rendering surfaces. This library is provided as part of every OpenGL implementation. GLU routines use the prefix **glu**.
- For every window system, there is a library that extends the functionality of that window system to support OpenGL rendering. For machines that use the X Window System, the OpenGL Extension to the X Window System (GLX) is provided as an adjunct to OpenGL. GLX routines use the prefix **glX**. For Microsoft Windows, the WGL routines provide the Windows to OpenGL interface. All WGL routines use the prefix **wgl**. For IBM OS/2, the PGL is the Presentation Manager to OpenGL interface, and its routines use the prefix **pgl**.
- The OpenGL Utility Toolkit (GLUT) is a window system-independent toolkit, written by Mark Kilgard, to hide the complexities of differing window system APIs. GLUT routines use the prefix **glut.**
- Open Inventor is an object-oriented toolkit based on OpenGL which provides objects and methods for creating interactive three-dimensional graphics applications. Open Inventor, which is written in C++, provides prebuilt objects and a built-in event model for user interaction, high-level application components for creating and editing three-dimensional scenes, and the ability to print objects and exchange data in other graphics formats. Open Inventor is separate from OpenGL.

Include Files

For all OpenGL applications, you want to include the gl.h header file in every file. Almost all OpenGL applications use GLU, the aforementioned OpenGL Utility Library, which requires inclusion of the glu.h header file. So almost every OpenGL source file begins with

```
#include   <GL/gl.h>

#include <GL/glu.h>
```

If you are directly accessing a window interface library to support OpenGL, such as GLX, AGL, PGL, or WGL, you must include additional header files. For example, if you are calling GLX, you may need to add these lines to your code

```
#include <X11/Xlib.h>
#include <GL/glx.h>
```

If you are using GLUT for managing your window manager tasks, you should include

```
#include <GL/glut.h>
```

Note that glut.h includes gl.h, glu.h, and glx.h automatically, so including all three files is redundant. GLUT for Microsoft Windows includes the appropriate header file to access WGL.

GLUT, the OpenGL Utility Toolkit

As you know, OpenGL contains rendering commands but is designed to be independent of any window system or operating system. Consequently, it contains no commands for opening windows or reading events from the keyboard or mouse. Unfortunately, it's impossible to write a complete graphics program without at least opening a window, and most interesting programs require a bit of user input or other services from the operating system or window system.

In addition, since OpenGL drawing commands are limited to those that generate simple geometric primitives (points, lines, and polygons), GLUT includes several routines that create more complicated three-dimensional objects such as a sphere, a torus, and a teapot. This way, snapshots of program output can be interesting to look at. (Note that the OpenGL Utility Library, GLU, also has quadrics routines that create some of the same three-dimensional objects as GLUT, such as a sphere, cylinder, or cone.)

Important features of OpenGL Utility Toolkit (GLUT)

- Provides functionality common to all window systems.
- Open a window.
- Get input from mouse and keyboard.
- Menus.
- Event-driven.
- Code is portable but GLUT lacks the functionality of a good toolkit for a specific platform.

- No slide bars.
- OpenGL is not object oriented so that there are multiple functions for a given logical function :
- glVertex3f
  - glVertex2i
  - glVertex3dv
- Underlying storage mode is the same easy to create overloaded functions in C++ but issue is efficiency.
- **OpenGL Interface**
  - GL (OpenGL in Windows)
  - GLU (graphics utility library) uses only GL functions, creates common objects (such as spheres)
  - GLUT (GL Utility Toolkit) interfaces with the window system
  - GLX: glue between OpenGL and Xwindow, used by GLUT

Window Management

Five routines perform tasks necessary to initialize a window.

- **glutInit**(int **argc*, char ***argv*) initializes GLUT and processes any command line arguments (for X, this would be options like -display and -geometry). **glutInit()** shouldbe called before any other GLUT routine.
- **glutInitDisplayMode**(unsigned int *mode*) specifies whether to use an *RGBA* or color-index color model. You can also specify whether you want a single- or double-buffered window. (If you're working in color-index mode, you'll want to load certain colors into the color map; use **glutSetColor()** to do this.) Finally, you can use this routine to indicatethat you want the window to have an associated depth, stencil, and/or accumulation buffer. For example, if you want a window with double buffering, the RGBA color model, and a depth

buffer, you might call **glutInitDisplayMode**(*GLUT_DOUBLE* | *GLUT_RGB* | *GLUT_DEPTH*).

- **glutInitWindowPosition**(int *x*, int *y*) specifies the screen location for the upper-left corner of your window.
- **glutInitWindowSize**(int *width*, int *size*) specifies the size, in pixels, of your window.
- int **glutCreateWindow**(char **string*) creates a window with an OpenGL context. It returns a unique identifier for the new window. Be warned: Until **glutMainLoop()** is called (see next section), the window is not yet displayed.

The Display Callback

glutDisplayFunc(void (**func*)(void)) is the first and most important event callback function you will see. Whenever GLUT determines the contents of the window need to be redisplayed, the callback function registered by **glutDisplayFunc()** is executed. Therefore, you should put all the routines you need to redraw the scene in the display callback function.

If your program changes the contents of the window, sometimes you will have to call **glutPostRedisplay**(void), which gives **glutMainLoop()** a nudge to call the registered display callback at its next opportunity.

Running the Program

The very last thing you must do is call **glutMainLoop**(void). All windows that have been created are now shown, and rendering to those windows is now effective. Event processing begins, and the registered display callback is triggered. Once this loop is entered, it is never exited!

Graphics Functions

- Primitive functions:
  points, line segments, polygons, pixels, text, curves, surfaces
- Attributes functions:
  color, pattern, typeface

Some Primitive Attributes

glClearColor (red, green, blue, alpha); - Default = (0.0, 0.0, 0.0, 0.0)

glColor3f (red, green, blue); - Default = (1.0, 1.0, 1.0)

glLineWidth (width); - Default = (1.0)

glLineStipple (factor, pattern) - Default = (1, 0xffff)

glEnable (GL_LINE_STIPPLE);

glPolygonMode (face, mode) - Default = (GL_FRONT_AND_BACK, GL_FILL)

glPointSize (size); - Default = (1.0)

- Viewing functions:
  position, orientation, clipping
- Transformation functions:
  rotation, translation, scaling
- Input functions:
  keyboards, mice, data tablets
- Control functions:
  communicate with windows, initialization, error handling
- Inquiry functions: number of colors, camera parameters/values

Matrix Mode

There are two matrices in OpenGL:

- Model-view: defines COP and orientation
- Projection: defines the projection matrix

glMatrixMode(GL_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, 500.0, 0.0, 500.0);

glMatrixMode(GL_MODELVIEW);

Control Functions

- OpenGL assumes origin is bottom left
- glutInit(int *argcp, char **argv);
- glutCreateWindow(char *title);
- glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE);
- glutInitWindowSize(480,640);

- glutInitWindowPosition(0,0);
- OpenGL default: RGB color, no hidden-surface removal, single buffering

Obtaining Values of OpenGL State Variables

glGetBooleanv (paramname, *paramlist);

glGetDoublev (paramname, *paramlist);

glGetFloatv (paramname, *paramlist);

glGetIntegerv (paramname, *paramlist);

Saving and Restoring Attributes

glPushAttrib (group);

glPopAttrib ( );

where group = GL_CURRENT_BIT, GL_ENABLE_BIT, GL_LINE_BIT, GL_POLYGON_BIT, etc.

Projection Transformations

glMatrixMode (GL_PROJECTION);

glLoadIdentity ( );

glFrustum (left, right, bottom, top, near, far);

gluPerspective (fov, aspect, near, far);

glOrtho (left, right, bottom, top, near, far);

- Default = (-1.0, 1.0, -1.0, 1.0, -1.0, 1.0)

gluOrtho2D (left, right, bottom, top);

Modelview Transformations

glMatrixMode (GL_MODELVIEW);

glLoadIdentity ( );

gluLookAt (eye_x, eye_y, eye_z, at_x, at_y, at_z, up_x, up_y, up_z);

glTranslatef (dx, dy, dz);

glScalef (sx, sy, sz);

glRotatef (angle, axisx, axisy, axisz);

Writing Bitmapped Text

glPixelStorei (GL_UNPACK_ALIGNMENT, 1);

glColor3f (red, green, blue);

glRasterPos2f (x, y);

glutBitmapCharacter (font, character);

where font = GLUT_BITMAP_8_BY_13, GLUT_BITMAP_HELVETICA_10, etc.

Managing the Frame Buffer

glutInit (&argc, argv);

glutInitDisplayMode (GLUT_RGB | mode);

glutInitWindowSize (width, height);

glutInitWindowPosition (x, y);

glutCreateWindow (label);

glClear (GL_COLOR_BUFFER_BIT);

glutSwapBuffers ( );

where mode = GLUT_SINGLE or GLUT_DOUBLE.

Registering Callbacks

glutDisplayFunc (callback);

glutReshapeFunc (callback);

glutDisplayFunc (callback);

glutMotionFunc (callback);

glutPassiveMotionFunc (callback);

glutMouseFunc (callback);

glutKeyboardFunc (callback);

id = glutCreateMenu (callback);

glutMainLoop ( );

Display Lists

glNewList (number, GL_COMPILE);

glEndList ( );

glCallList (number);

glDeleteLists (number, 1);

Managing Menus

id = glutCreateMenu (callback);

glutDestroyMenu (id);

glutAddMenuEntry (label, number);

glutAttachMenu (button);

glutDetachMenu (button);

where button = GLUT_RIGHT_BUTTON or GLUT_LEFT_BUTT

Chapter-2
Design

Design

Ball in the basket:

The **Ball in the basket is a opengl based 3 dimensional game**. It consists of a wired Ball and basket and a pipe from which ball is randomly generated from above.

- If the ball was successfully drop in the basket then it must be generated randomly from pipe to different location, at each successful catch speed of ball dropping will be increased by count value.
- If ball was dropped at bottom the game will be over.

.

Explanation about the package:

This is a mini project on Ball in the Basket 3d game based on OPENGL API for Computer graphics andvisualization laboratory (18CSL67). The project simulates the computer graphics concepts of 3 dimensional api's in the game.
The project implements various geometric transformations like Translation, Rotation, and Scaling. The basic model consists of a wired ball, wired basket and a pipe.The ball initially at top position of window and basket is at bottom position of window.. The Ball are drawn using the GLUT library function glutWireSphere(), and the Basket are drawn using glutWireCone().Initially the user is presented with an introduction scene which has a menu to Start the game.
On pressing the X key on keyboard the actual simulation scene is loaded. The user can

use the mouse wheel to advance through the simulation. An optional animation has been implemented, which shows the movement of ball from top to bottom . The viewing model used is an Orthogonal Projection and model view. The moves to be performed as per the optimal solution are displayed on the top left of the screen is a raster text using the function glutBitmapCharacter(). Lighting has been implemented by the inbuilt OPENGL lightingfunctions. Menus have been provided to modify various features such as Lighting, change background color, Start, exit to automatically simulate the complete solution, restart the simulation and to exit the program.

Chapter-3

Implementation

Complete Source Code of the package :

```
#include<GL/glut.h> //for glut libraries
#include<stdio.h>   //for printf functions used
#include<stdlib.h> //for random function
#include<string.h>  //for string use character
#define LIGHT_ON 0
#define LIGHT_OFF 1

int lightflag=0;
void init()
{
	glClearColor(1,1,1,1);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(-2,2,-2,2,-2,2);
	glMatrixMode(GL_MODELVIEW);  //😀

}

void lighting()
{
	GLfloat shininess[] = {50};
	GLfloat white[] = {0.6,0.6,0.6,1};
	glEnable(GL_COLOR_MATERIAL);
	glColorMaterial(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE);
	GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
	GLfloat light_position[] = {100,60, 10, 0.0 };
	glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
	glLightfv(GL_LIGHT0, GL_POSITION, light_position);
	glMaterialfv(GL_FRONT, GL_DIFFUSE, white);
	glMaterialfv(GL_FRONT, GL_SPECULAR, white);
	glMaterialfv(GL_FRONT, GL_SHININESS, shininess);
	glEnable(GL_LIGHT0);
}

void ball()
```

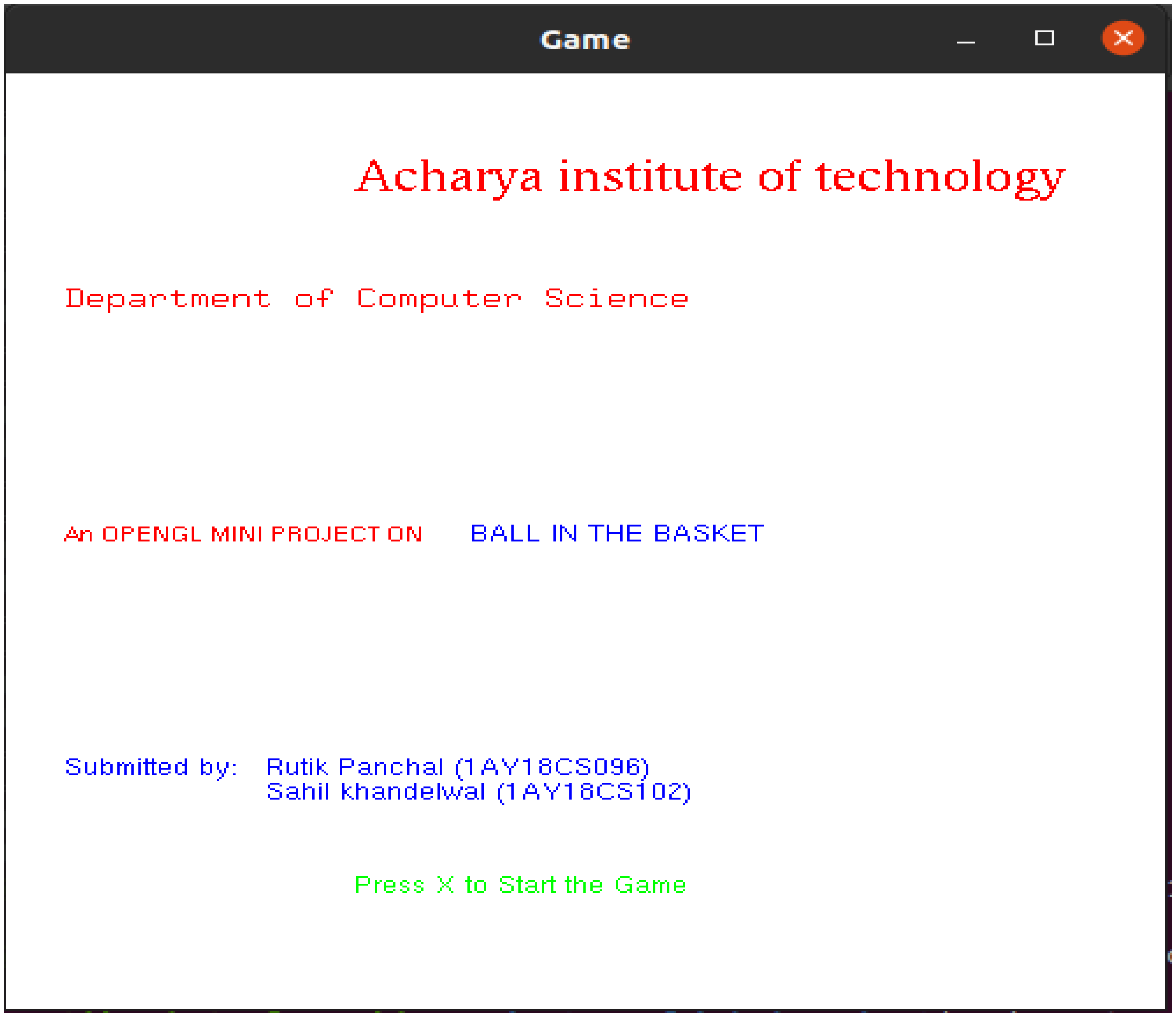
```
{
        glColor3f(1,0,0);

       glutWireSphere(0.2,25,25); //😀
}



void basket()
{
        glColor3f(0,1,0);
        glutWireCone(0.3,0.5,25,25);
}
 float ballx=0,bally=1.8,basketx=0,baskety=-1.5,pollx=0.6,polly=1.8;
 int  rand(),r,flagout=0;   //r to store the random for when ball reaches basket it will
random generate from above.//flag for if ball reaches -1.6 at basket to stop the ball
  float ballspeed=0,count=0;
void display()


{
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

        if(lightflag)glEnable(GL_LIGHTING);
        glLoadIdentity();  // for 😀animation

        printf("ballx=%f\n",ballx);
                printf("bally=%f\n",bally);
                        printf("basketx=%f\n",basketx);
                                printf("baskety=%f\n",baskety);

        if(count==3)
        {
        ballspeed+=0.001;
        count=0;
        }

        if(bally<-1.6 && ballx>basketx-0.3 && ballx<basketx+0.3)
        {
        count++;    //everytime when ball hit basket count incremented
        bally=1.8;
        r=rand()%4; //any random number from 0 to 3 will be generated
        if(r==0)
        {
          ballx=-1.8;
          pollx=-1.8+0.6;

        }
        if(r==1)
        {
          ballx=-1;
                  pollx=-1+0.6;
        }
        if(r==2)
        {
          ballx=1;
                  pollx=1+0.6;

        }
        if(r==3)
        {
          ballx=1.8;
                  pollx=1.8+0.6;
        }
        }

        if(flagout==0)
        bally-=0.01+ballspeed;

        if(bally<-2)
        {
        flagout=1;
        }
        glPushMatrix(); //because ball as well as basket translate so to avoid that
        glTranslatef(ballx,bally,0); //😀tranlsate ball on yaxis
        ball();
        glPopMatrix();

        glPushMatrix(); //other functions not affected
        glTranslatef(basketx,baskety,0);
        glRotatef(90,1,0,0);  //rotate basket 90 about y
        basket();
        glPopMatrix();
```

```
 //create poll
        glPushMatrix();
        glTranslatef(pollx,polly,0);
        glScalef(1.9,0.8,1);
        glColor3f(0,1,1);
        glutSolidCube(0.5);
        glutSolidTorus(0.1,0.2,4,4);
        glPopMatrix();

glutSwapBuffers();
glutPostRedisplay();

}

void welcomeDisplay()
{


glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
glColor3f(1,0,0);
glRasterPos3f(-0.8,1.5,0);
char msg1[]="Acharya institute of technology";
for(int i=0; i<strlen(msg1);i++)

glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, msg1[i]); //nothing will
view so first i have to make cursor to make to that i use raster pos

glRasterPos3f(-1.8,1,0);
char msg2[]="Department of Computer Science";
for(int i=0; i<strlen(msg2);i++)

glutBitmapCharacter(GLUT_BITMAP_9_BY_15, msg2[i]);



glRasterPos3f(-1.8,0,0);
char msg3[]="An OPENGL MINI PROJECT ON ";
for(int i=0; i<strlen(msg3);i++)

glutBitmapCharacter(GLUT_BITMAP_HELVETICA_10, msg3[i]);

glColor3f(0,0,1);
glRasterPos3f(-0.4,0,0);
char msg4[]="BALL IN THE BASKET";
```

```
for(int i=0; i<strlen(msg4);i++)

glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, msg4[i]);

glRasterPos3f(-1.8,-1,0);
char msg5[]="Submitted by:";
for(int i=0; i<strlen(msg5);i++)

glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, msg5[i]);

glRasterPos3f(-1.1,-1,0);
char msg6[]="Rutik Panchal (1AY18CS096)";
for(int i=0; i<strlen(msg6);i++)

glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, msg6[i]);

glRasterPos3f(-1.1,-1.1,0);
char msg7[]="Sahil khandelwal (1AY18CS102)";
for(int i=0; i<strlen(msg7);i++)

glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, msg7[i]);



glColor3f(0,1,0);
glRasterPos3f(-0.8,-1.5,0);
char msg8[]="Press X to Start the Game";
for(int i=0; i<strlen(msg8);i++)

glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, msg8[i]);      //character😀

glutSwapBuffers();

}

void skeys(int key,int x,int y)
{
        if(key==GLUT_KEY_LEFT)
        {
        basketx-=0.3;
        }
        if(key==GLUT_KEY_RIGHT)
        {
        basketx+=0.3;
        }
}

void keys(unsigned char key,int x ,int y)
{
if(key=='x')
{
glutDisplayFunc(display);
}
glutPostRedisplay();
}


void mytimer(int val)
{
glutDisplayFunc(display);
glutPostRedisplay();
}

void processstart(int option)
{
 glutDisplayFunc(display);
 glutPostRedisplay();
 }

 void processMenuExit(int option)
{
        if(option==0)exit(0);
}

 void processMenuLighting(int option)
{
        switch(option)
        {
                case LIGHT_OFF:
                        glDisable(GL_LIGHTING);
                        lightflag=0;
                break;
                case LIGHT_ON:
                        glEnable(GL_LIGHTING);
                        lightflag=1;
                        lighting();
                break;
        }
        glutPostRedisplay();
}

void processMenuBgColor(int option)
{
        switch(option)
        {
                case 0:glClearColor(0,0,0,0);break;
                case 1:glClearColor(1,1,1,0);break;
                case 2:glClearColor(1,0,0,0);break;
                case 3:glClearColor(0,1,0,0);break;
                case 4:glClearColor(0,0,1,0);break;
        }
        glutPostRedisplay();
}




 void processMenuMain1(int option)
{
}

void createGLUTMenus1()
{
int menu = glutCreateMenu(processstart);
glutAddMenuEntry("Run",0);
int menuExit = glutCreateMenu(processMenuExit);
        glutAddMenuEntry("Yes",0);
        glutAddMenuEntry("No",1);

int menulight = glutCreateMenu(processMenuLighting);
        glutAddMenuEntry("On",LIGHT_ON);
        glutAddMenuEntry("Off",LIGHT_OFF);

int menuBgColor = glutCreateMenu(processMenuBgColor);
        glutAddMenuEntry("Black",0);
        glutAddMenuEntry("White",1);
        glutAddMenuEntry("Red",2);
        glutAddMenuEntry("Green",3);
        glutAddMenuEntry("Blue",4);
```

```
glutCreateMenu(processMenuMain1);
	glutAddSubMenu("Start",menu);
	glutAddSubMenu("Lightning",menulight);
	glutAddSubMenu("Background Color",menuBgColor);
	glutAddSubMenu("Exit",menuExit);
	glutAttachMenu(GLUT_RIGHT_BUTTON);

}

int main(int argc,char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);
	glutInitWindowSize(500,500);
	glutCreateWindow("Game");
	init();
	glutDisplayFunc(welcomeDisplay);
	createGLUTMenus1();
	glutSpecialFunc(skeys);  //😁rather than keyboard use special because i
have to link to //left and right button of keyboard
	glutKeyboardFunc(keys);  // for press x to strat the game
	glutTimerFunc(10000,mytimer,0);
	glEnable(GL_DEPTH_TEST);
	glutMainLoop();
	return 0;
}
```

Chapter-4
Results



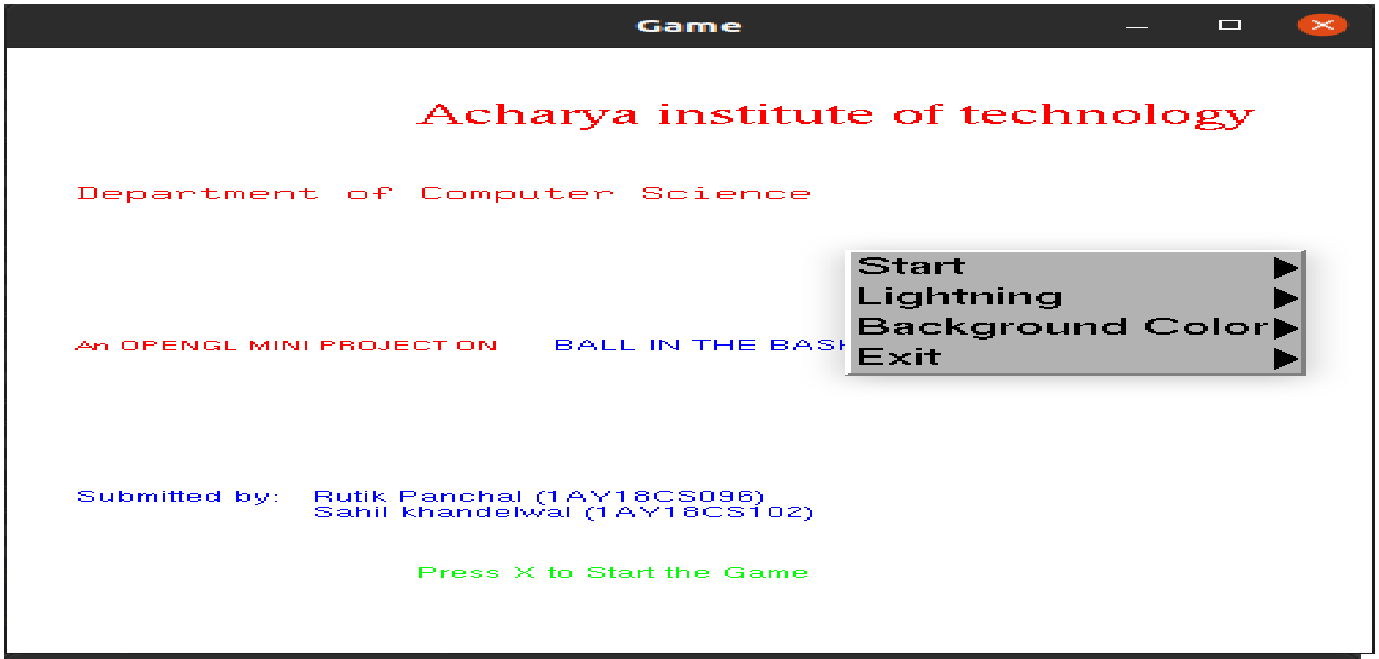
Acharya institute of technology

Department of Computer Science

An OPENGL MINI PROJECT ON BALL IN THE BASKET

Submitted by: Rutik Panchal (1AY18CS096)
Sahil khandelwal (1AY18CS102)

Press X to Start the Game



Game

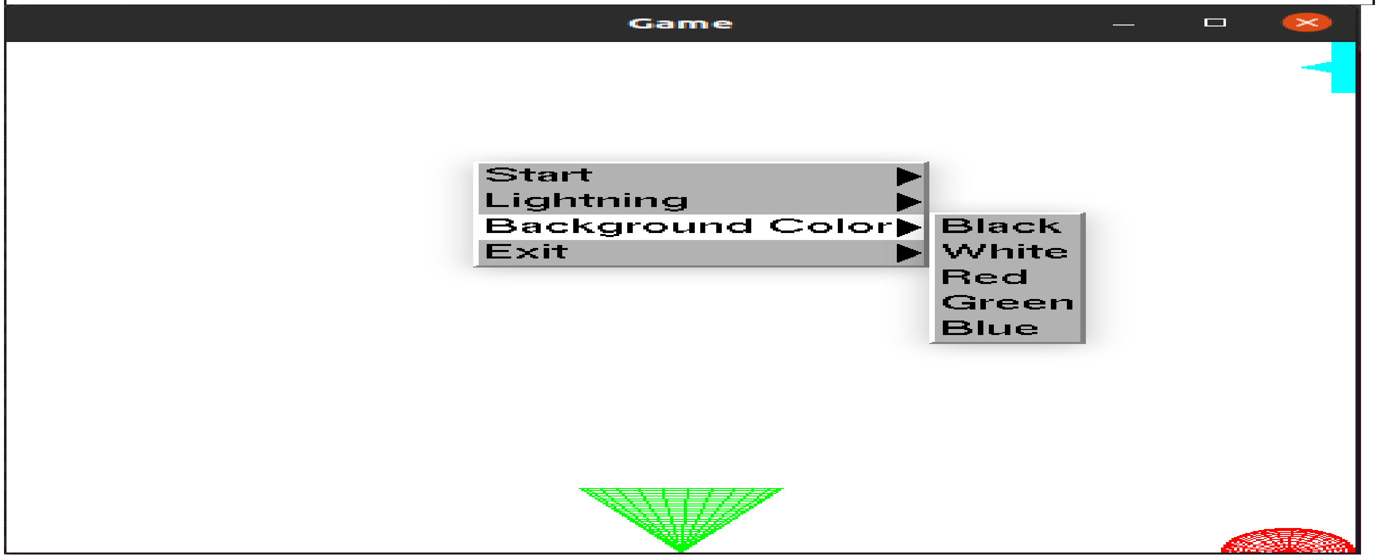
Acharya institute of technology

Department of Computer Science

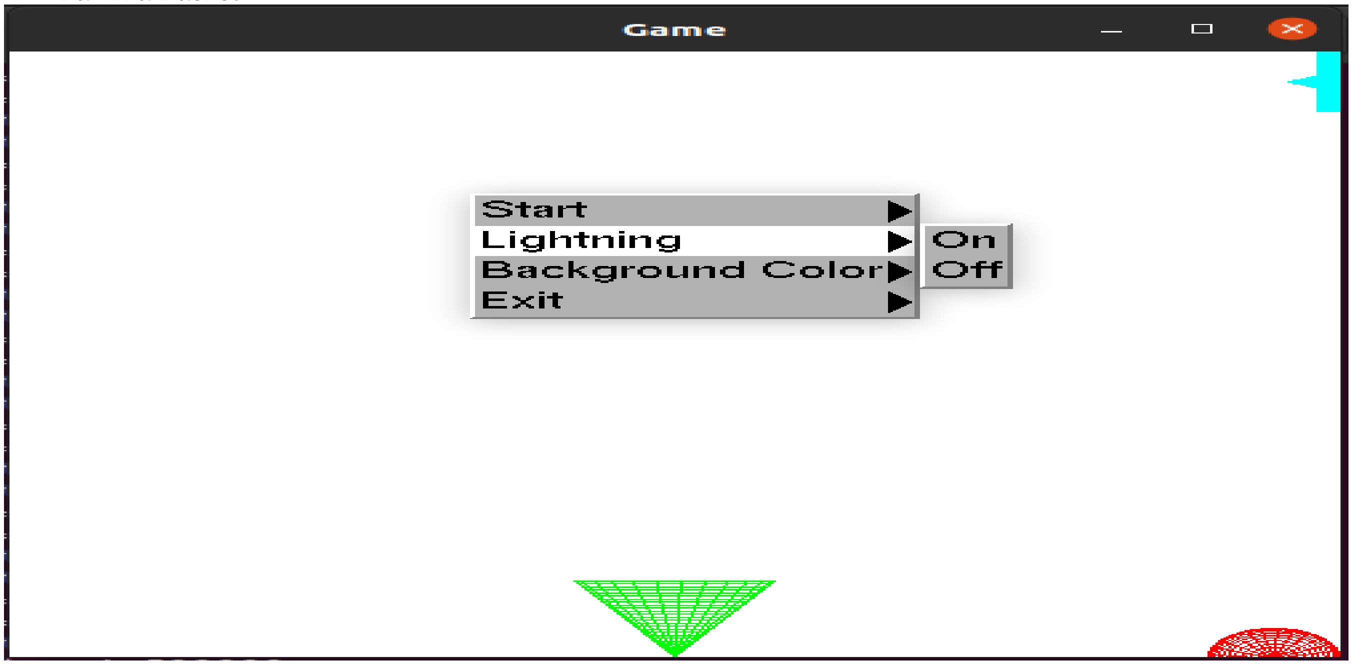
An OPENGL MINI PROJECT ON BALL IN THE BASK

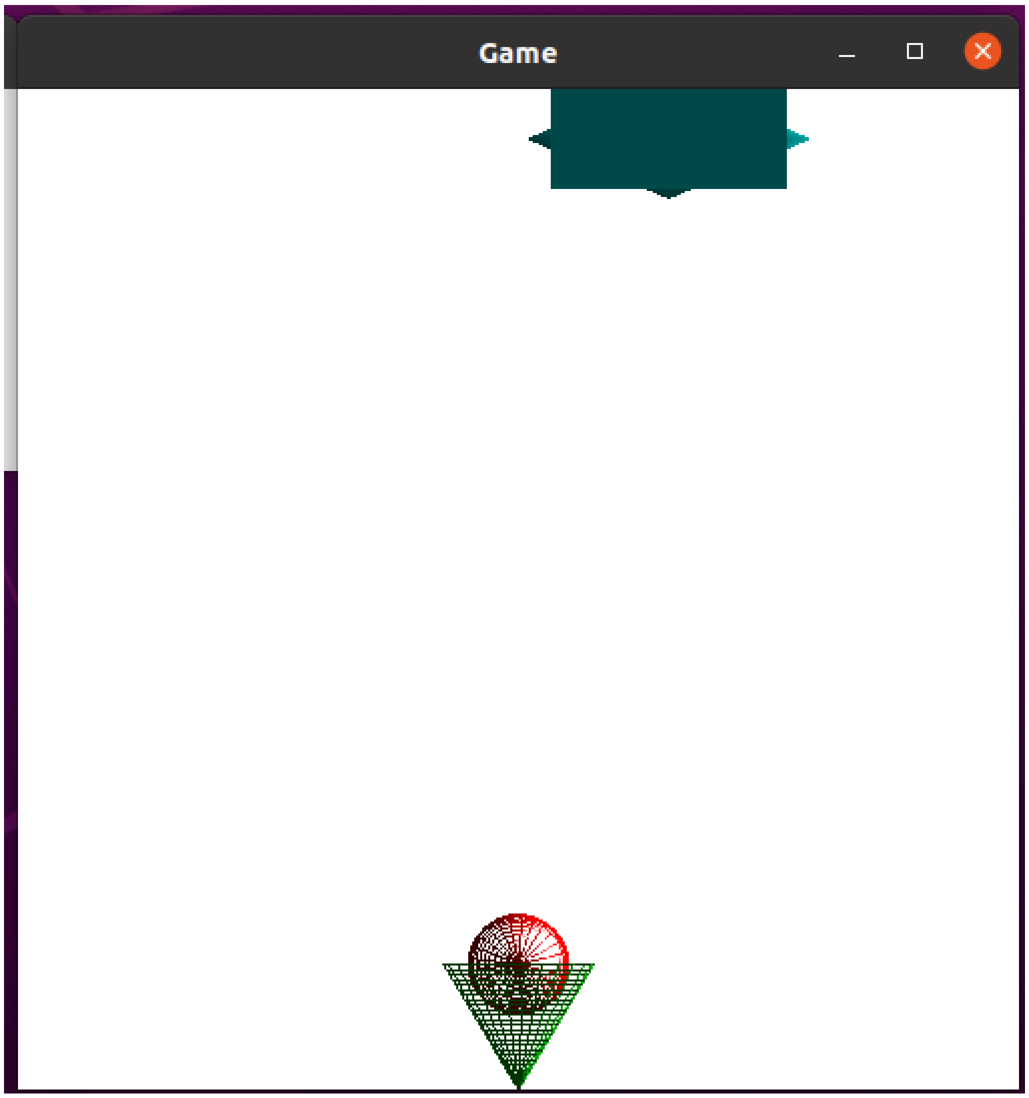
Submitted by: Rutik Panchal (1AY18CS096)
Sahil khandelwal (1AY18CS102)

Press X to Start the Game



Game


Game
Start
Lightning
Background Color
Exit
On
Off


Game

Chapter-5
Conclusion

Conclusion

It was a wonderful learning experience for me working on this project. This project 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 project.

Chapter-6
Bibliography

Bibliography

- Interactive Computer Graphics – Edward Angel
- Official OPENGL Documentation at https://www.**opengl**.org/**documentation**/
- OpenGL Programming Guide: The Official Guide to Learning OpenGL- Dave Shreiner.