VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum-590018

A Mini Project Report On

"THE REAL BOUNCING BALL"

SUBMITTED IN PARTIAL FULFILMENT FOR 6TH SEMESTER

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

Peeyush Rawat 1JB19CS093

Prashant Prasar 1JB19CS098

UNDER THE GUIDANCE OF

Mrs. Manasa B S
Assistant Professor
Dept. of CSE, SJBIT

Mrs. Basamma patil
Assistant Professor
Dept. of CSE, SJBIT

SJBIT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SJB INSTITUTE OF TECHNOLOGY

#67, BGS HEALTH & EDUCATION CITY, DR. VISHNUVARDHAN ROAD, KENGERI, BENGALURU-560060, KARNATAKA, INDIA.

2021 - 2022

|| Jai Sri Gurudev ||

Sri Adichunchanagiri Shikshana Trust ®

SJB INSTITUTE OF TECHNOLOGY

#67, BGS HEALTH & EDUCATION CITY, DR. VISHNUVARDHAN ROAD, KENGERI, BENGALURU-560060, KARNATAKA, INDIA.

Department of Computer Science & Engineering

SJBIT

CERTIFICATE

Certified that the Computer Graphics Mini project work entitled "**THE REAL BOUNCING BALL**" is a bonafide work carried out by Mr. PEEYUSH RAWAT AND Mr. PRASHANT PRASAR and bearing USN **1JB19CS093, 1JB18CS098** of **SJB Institute of Technology** in partial fulfilment for 6th semester in **COMPUTER SCIENCE AND ENGINEERING** of the **Visvesvaraya Technological University**, **Belagavi** during the academic year **2021-22.** It is certified that all corrections/suggestions indicated for Internal Assessment 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 prescribed by the institution.

Mrs. MANASA B S Asst. professor Dept. of CSE, SJBIT	Mrs. BASAMMA PATIL Asst. professor Dept. of CSE, SJBIT	Signature of HOD

1. **Internal Examiner:** ______________________

2. **External Examiner:** ______________________

ACKNOWLEDGEMENT

We would like to express our profound thanks to His Divine Soul **Padmabhushana Sri Sri Sri Dr. Balagangadharanatha MahaSwamiji** and His Holiness **Jagadguru Sri Sri Sri Dr. Nirmalanandanatha MahaSwamiji** for providing us an opportunity to pursue our academics in this esteemed institution.

We would also like to express our profound thanks to **Revered Sri Sri Sri Dr. Prakashnath Swamiji**, Managing Director, SJB Institute of Technology, for his continuous support in providing amenities to carry out this mini project in this admired institution.

We express our gratitude to **Dr. K V Mahendra Prashanth**, Principal, SJB Institute of Technology, for providing an excellent facilities and academic ambience; which have helped us in satisfactory completion of mini project work.

We extend our sincere thanks to **Dr. Krishna A N,** Head of the Department, Computer Science and engineering for providing an invaluable support throughout the period of mini project work.

We wish to express heartfelt gratitude to our guides, **Mrs. Manasa B S,** Assistant Professor**,** Dept of CSE and **Mrs. Basamma Patil**, Assistant Professor, Dept of CSE for their valuable guidance, suggestions and cheerful encouragement during the entire period of this work.

Finally, we take this opportunity to extend our earnest gratitude and respect to our parents, Teaching & Non-teaching staffs of the department, the library staff and all our friends, who have directly or indirectly supported us during the period of this mini project work.

Regards,
PEEYUSH RAWAT 1JB19CS093
PRASHANT PRASAR 1JB19CS098

ABSTRACT

A picture is worth a thousand words' goes the ancient Chinese proverb. This has become a cliché in our society after the advent of inexpensive and simple techniques for producing pictures. Computers have become a powerful medium for the rapid and economical production of pictures. There is virtually no area in which graphical displays cannot be used to some advantage. Graphics provide a so natural means of communicating with a computer that they have become widespread. The fields in which Computer Graphics find their uses are many. Some of them being User Interfaces, Computer Aided Design, Office automation, Desktop Publishing, plotting of mathematical, scientific, or industrial data, Simulation, Art, Presentations, Cartography, to name a few. Here, we have tried to incorporate and present the working environment of The Real Bouncing Ball.

Our project is an The Real Bouncing Ball made with OpenGL library in C++. In this animation there are multiple balls, the user can choose the number of balls and the size of the balls too. The balls demonstrate the concept of energy loss with each bounce and hence decrease in maximum height. User can choose the coefficient of restitution for each ball to understand the difference through audial visual of this project.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly. Computers have become a powerful medium for the rapid and economical production of pictures. There is virtually no area in which graphical displays cannot be used to some advantage. Graphics provide a so natural means of communicating with the computer that they have become widespread.

Interactive graphics is the most important means of producing pictures since the invention of photography and television. The project the real bouncing ball is designed using this interactive graphics through open gl. It simulates the animation of a ball bouncing and losing energy with every collision.

1.1 Problem Statement

The projects main aim is to simulate an animation using the OpenGL functions. We have developed an animation of The Real Bouncing Ball. This animation demonstrates the energy loss in a bouncing ball with each collision with the floor according to its coefficient of restitution.

1.2 Objectives

- To develop animation using OpenGL
- To simulate the Bouncing Ball
- Provide motion to required elements in the project
- Show the phases of the Bouncing Ball
- Design the required materials like floor and 3D balls.

1.3 Scope

Competition for jobs is quite stiff, as animation has become a popular choice in career. Many of the upcoming jobs are in the aspects of game designing and other fields. The package can be implemented in various other fields such as

- Advertisements
- Automobile Testing
- Cartoon animations

CHAPTER 2

LITERATURE SURVEY

People use the term “computer graphics” to mean different things in different context. Computer graphics are pictures that are generated by a computer. Everywhere you look today, you can find examples, especially in magazines and on television. Some images look so natural you can’t distinguish them from photographs of a real scene. Others have an artificial look, intended to achieve some visual effects.

There are several ways in which the graphics generated by the program can be delivered.

- Frame- by- frame: A single frame can be drawn while the user waits.
- Frame-by-frame under control of the user: A sequence of frames can be drawn, as in a corporate power point presentation; the user presses to move on to the next slide, but otherwise has no way of interacting with the slides
- Animation: A sequence of frames proceeds at a particular rate while the user watches with delight.
- Interactive program: An interactive graphics presentation is watched, where the user controls the flow from one frame to another using an input device such as a mouse or keyboard, in a manner that was unpredictable at the time the program was written. This can delight the eye.

2.1 History

OpenGL was developed by ‘Silicon Graphics Inc ‘(SGI) in 1992 and is popular in the gaming industry where it competes with the Direct3D in the Microsoft Windows platform. OpenGL is broadly used in CAD (Computer Aided Design), virtual reality, scientific visualization, information visualization, flight simulation and video games development.

OpenGL is a standard specification that defines an API that is multi-language and multi- platform and that enables the codification of applications that output computerized graphics in 2D and 3D.

The interface consists in more than 250 different functions, which can be used to draw complex tri-dimensional scenes with simple primitives. It consists of many functions that help to create a real-world object and a particular existence for an object can be given.

2.2 Characteristics

- OpenGL is a better documented API.
- OpenGL is also a cleaner API and much easier to learn and program.
- OpenGL has the best demonstrated 3D performance for any API.
- Microsoft's Direct3D group is already planning a major API change called Direct Primitive that will leave any existing investment in learning Direct3D immediate mode largely obsolete.

2.3 Computer Graphics Library Organisation

OpenGL stands for Open-Source Graphics Library. Graphics Library is a collection of APIs (Application Programming Interfaces).

Graphics Library functions are divided in three libraries. They are as follows –

i. **GL Library** (OpenGL in Windows)
ii. **GLU** (OpenGL Utility Library)
iii. **GLUT** (OpenGL Utility Toolkit)

Functions in main GL library name function names that begin with the letter ‘gl’.

- **GLU** library uses only GL functions but contains code for creating objects and simplify viewing.
- To interface with the window system and to get input from external devices **GLUT** library is used, which is a combination of three libraries GLX for X windows, ‘wgl’ for Windows and ‘agl’ for Macintosh.
- These libraries are included in the application program using pre-processor directives. E.g.: #include<GL/glut.h>
- The following figure shows the library organization in OpenGL.

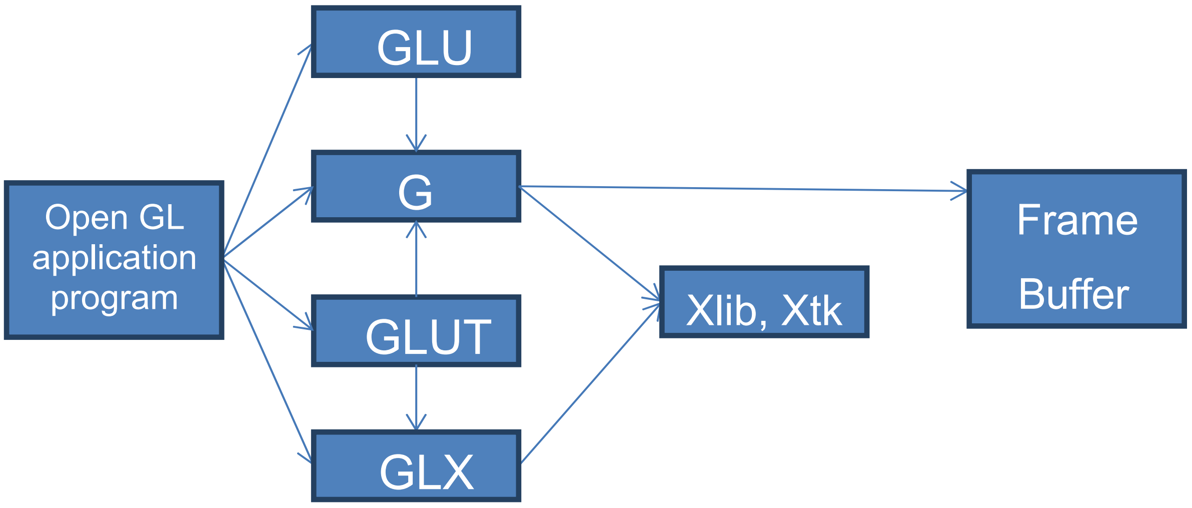


Fig 2.1 Library Organisation

2.4 Graphics System and Functions

- Graphics system and functions can be considered as a black box, a term used to denote a system whose properties are only described by its inputs and output without knowing the internal working.
- Inputs to graphics system are functions calls from application program, measurements from input devices such as mouse and keyboard.
- Outputs are primarily the graphics sent to output devices.

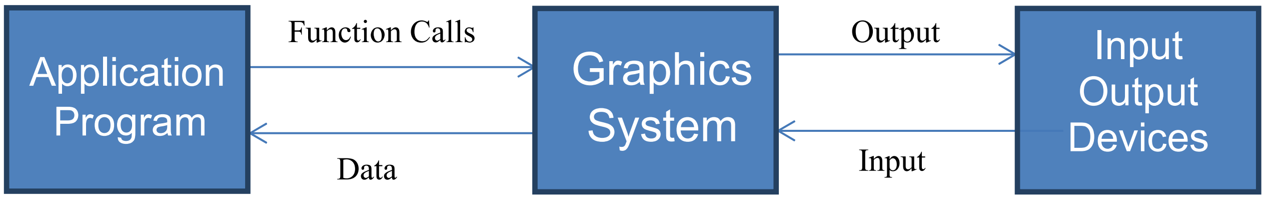


Fig 2.2 Graphics System as a Black Box

APIs are described through functions in its library. These functions are divided into seven major groups.

- Primitive Functions:

  Primitive functions define the low-level objects or atomic entities that a system can display, the primitives include line segments, polygons, pixels, points, text and various types of curves and surfaces.
- Attribute Functions:

  Attribute Functions allow us to perform operations ranging from choosing the colour to display a line segment, to packing a pattern to fill inside any solid figure.
- Viewing Functions:

  Viewing functions allow us to specify various views.
- Transformation Functions:

  Transformation functions allow us to carry out transformation of objects such as rotation, translation, and scaling.
- Input Functions:

  Input functions allow us to deal with the diverse forms of input that characterize modern graphics system. It deals with devices such as keyboard, mouse, and data tablets.
- Control Functions:

  Control Functions enable us to communicate with the window system, to initialize the programs, and to deal with any errors that occur during the execution of the program.
- Query Functions:

  Query Functions provides information about the API.

CHAPTER 3

SYSTEM REQUIREMENTS

Requirements analysis is critical for project development. Requirements must be documented, actionable, measurable, testable, and defined to a level of detail sufficient for system design. Requirements can be architectural, structural, behavioural, functional, and non-functional. A software requirements specification (SRS) is a comprehensive description of the intended purpose and the environment for software under development.

3.1 Hardware Requirement

- Minimum of 2GB of main memory
- Minimum of 3GB of storage
- Keyboard
- Mouse
- Display Unit
- Dual-Core or AMD with minimum of 1.5GHz speed

3.2 Software Requirement

- Windows – 8/10/11
- Microsoft Visual Studio C/C++ 10.0 and above versions
- OpenGL Files
- DirectX 8.0 and above versions

Header Files

glut.h

Object File Libraries

glut32.lib

DLL files

glut32.dll

CHAPTER 4

SYSTEM DESIGN AND IMPEMENTATION

4.1 Header Files Used

- **#include<stdlib.h>**

  stdlib.h is the header of the **general-purpose standard library** of C programming language which includes functions involving memory allocation, process control, conversions, and others. It is compatible with C++ and is known as cstdlib in C++. The name "stdlib" stands for "standard library".

- **#include<GL/glut.h>**

  GLUT (pronounced like the glut in gluttony) is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. GLUT provides a portable API so you can write a single OpenGL program that works across all PC and workstation OS platforms.

- **#include<stdio.h>**

  The C programming language provides many standard library functions for file input and output. These functions make up the bulk of the C standard library header <stdio.h>. The first thing you will notice is the first line of the file, the #include "stdio.h" line. This is very much like the #define the pre-processor, except that instead of a simple substitution, an entire file is read in at this point.

  The file is called a header file and you will find several different header files on the source disks that came with your C compiler. Each of the header files has a specific purpose and any or all of them can be included in any program.

- **#include<math.h>**

  math.h is a header file in the standard library of the C programming language designed for basic mathematical operations. Most of the functions involve the use of floating-point numbers. C++ also implements these functions for compatibility reasons and declares them in the header cmath (the C99 functions are not available in the current C++ standard, C++ 98). All functions that take or return an angle work in radians. All functions take doubles for floating-point arguments, unless otherwise specified. In C99, to work with floats or long doubles, append an f or an l to the name, respectively.

- **#include<string.h>**
  string.h is the header in the C standard library for the C programming language which contains macro definitions, constants and declarations of functions and types used not only for string handling but also various memory handling functions. All the string handling functions are prototyped in: #include <string.h>. The common functions are described below: char *stpcpy (char *dest,const char *src) -- Copy one string into another. int strcmp(char *string1,const char *string2) - Compare string1 and string2 to determine alphabetic order etc.

4.2 OpenGL API's Used

- **void glColor3f(float red, float green, float blue);**

  This function is used to mention the colour in which the pixel should appear. The number 3 specifies the number of arguments that the function would take. 'f ' gives the type that is float. The arguments are in the order RGB(Red, Green, Blue). The colour of the pixel can be specified as the combination of these 3 primary colours.

- **void glClearColor(int red, int green, int blue, int alpha);**

  This function is used to clear the colour of the screen. The 4 values that are passed as arguments for this function are (RED, GREEN, BLUE, ALPHA) where the red, green, and blue components are taken to set the background colour and alpha is a value that specifies depth of the window. It is used for 3D images.

- **void glutKeyboardFunc();**

  void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y));

  where func is the new keyboard call-back function. **glutKeyboardFunc** sets the keyboard call-back for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard call-back. The key call-back parameter is the generated ASCII character. The x and y call-back parameters indicate the mouse location in window relative coordinates when the key was pressed. When a new window is created, no keyboard call back is initially registered, and ASCII keystrokes in the window are ignored. Passing NULL to glutKeyboardFunc disables the generation of keyboard call-backs.

- **void glFlush();**

Different GL implementations buffer commands in several different locations, including network buffers and the graphics accelerator itself. **glFlush** empties all these buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any time, it does complete in finite time.

- **void glMatrixMode(GLenum mode);**

where mode Specifies which matrix stack is the target for subsequent matrix operations. Three values are accepted: **GL_MODELVIEW, GL_PROJECTION**, and **GL_TEXTURE**. The initial value is GL_MODELVIEW.

glMatrixMode sets the current matrix mode. mode can assume one of three values:

GL_MODELVIEW	Applies subsequent matrix operations to the model view matrix stack.
GL_PROJECTION	Applies subsequent matrix operations to the projection matrix stack.

- **void glViewport(GLint x, GLint y, GLsizei width, GLsizei height)**

where x, y Specify the lower left corner of the viewport rectangle, in pixels. The initial value is (0, 0). The width, height Specify the width and height of the viewport. When a GL context is first attached to a surface (e.g., window), width and height are set to the dimensions of that surface. GlViewport specifies the affine transformation of x and y from normalized device coordinates to window coordinates. Let (xnd, ynd) be normalized device coordinates. Then the window coordinates (xw, yw) are computed as follows:
xw = (xnd + 1) width/2 + x
yw = (ynd + 1) height/2 + y

- **void glutInit(int *argcp, char **argv);**

  glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized. Examples of this situation include the failure to connect to the window system, the lack of window system support for OpenGL, and invalid command line options. glutInit also processes command line options, but the specific options parse are window system dependent.

- **void glutReshapeFunc(void (*func)(int width, int height));**

  glutReshapeFunc sets the reshape call-back for the current window. The reshape call-back is triggered when a window is reshaped. A reshape call-back is also triggered immediately before a window's first display call-back after a window is created or whenever an overlay for the window is established. The width and height parameters of the call-back specify the new window size in pixels. Before the call-back, the current window is set to the window that has been reshaped.

  If a reshape call-back is not registered for a window or NULL is passed to glutReshapeFunc (to deregister a previously registered call-back), the default reshape call-back is used. This default call-back will simply **glOrtho()**

  Syntax: void glOrtho (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far); The function defines an orthographic viewing volume with all parameters measured from the centre of the projection plane.

- **void glutMainLoop(void);**

  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 stop.

- **glutPostRedisplay()**

  glutPostRedisplay, glutPostWindowRedisplay — marks the current or specified window as needing to be redisplayed.

4.3 User Defined Functions

- **Camera (int i ,int j)**

  This function displays the starting screen of the bouncing ball. The starting screen consists of the text "The Real Bouncing Ball" and "Press S for start". This also consists of those 3 balls and 1 checkerboard.

```
class Camera {

    double theta;

    double y;

    double dTheta;

    double dy;

public:

    Camera(){

            theta = 0;

            y = 3;

            dTheta = 0.04;

            dy = 0.2;

    }

    double getX() { return 10 * cos(theta); }

    double getY() { return y; }

    double getZ() { return 10 * sin(theta); }

    void moveRight() { theta += dTheta; }

    void moveLeft() { theta -= dTheta; }

    void moveUp() { y += dy; }

    void moveDown() { if (y > dy) y -= dy; }

};
```

- **Class Ball ()**

  This class displays the balls at the specified locations. We have created three balls using glutSolidSphere.

- **void update ()**

  This function updates the ball location after every call. We have created three balls using glutSolidSphere.

```
void update() {
        y += direction * 0.05;
        if (y >= maximumHeight && direction==1) {
                y = maximumHeight;
                direction = -1;
        }
        else if (y <= radius) {
                y = radius;
                direction = 1;
                maximumHeight = ((maximumHeight - radius) * pow(coeff_of_restitution,
++n)) + radius;
        }
        glPushMatrix();
        glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, color);
        glTranslated(x, y, z);
        glutSolidSphere(radius, 30, 30);
        glPopMatrix();
    }
```

- **class Checkerboard{ };**

This class is used to create the checkerboard floor on which the ball hits.

```
class Checkerboard {
        int displayListId;
        int width;


        int depth;
public:
        Checkerboard(int width, int depth){
                this->width = width;
                this->depth = depth;
        }
        double centerx() { return width / 2; }
        double centerz() { return depth / 2; }
        void create() {
                displayListId = glGenLists(1);
                glNewList(displayListId, GL_COMPILE);
                glBegin(GL_QUADS);
                glNormal3d(0, 1, 0);
                for (int x = 0; x < width - 1; x++) {
                        for (int z = 0; z < depth - 1; z++) {
                                glMaterialfv(GL_FRONT,
GL_AMBIENT_AND_DIFFUSE,         (x + z) % 2 == 0 ? RED : WHITE);
                                glVertex3d(x, 0, z);
                                glVertex3d(x + 1, 0, z);
                                glVertex3d(x + 1, 0, z + 1);
                                glVertex3d(x, 0, z + 1);


                        }
                }
                glEnd();
                glEndList();
        }
        void draw() {
                glCallList(displayListId);
        }
};
```

- void display()

  This function draws one frame, the checkerboard then the balls, from the current camera position.

```
void display() {
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        gluLookAt(camera.getX(), camera.getY(), camera.getZ(),
                checkerboard.centerx(), 0.0, checkerboard.centerz(),
                0.0, 1.0, 0.0);
        checkerboard.draw();
        for (int i = 0; i < sizeof(balls) / sizeof(Ball); i++) {
                balls[i].update();
        }
        glFlush();
        glutSwapBuffers();
}
```

- void timer(int v)

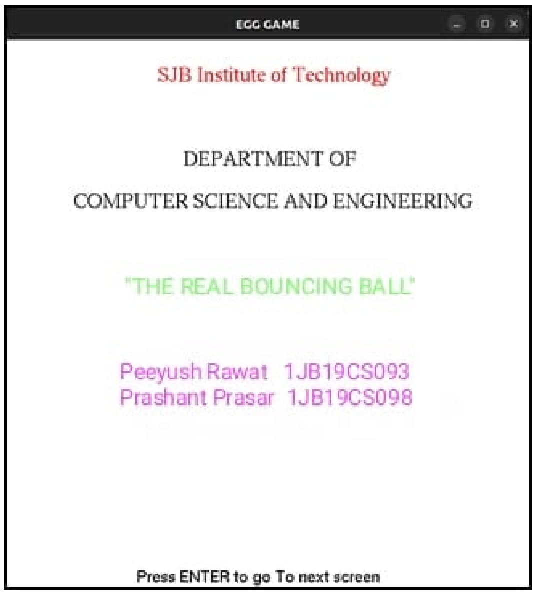
  This function requests to draw the next frame.

```
void timer(int v) {
        glutPostRedisplay();
        glutTimerFunc(1000 / 60, timer, v);
}
```

CHAPTER 5

SNAPSHOTS

In Fig 5.1 the window consists of the contents like name of the project, department. We need to press enter to go to the next screen.



EGG GAME

SJB Institute of Technology

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

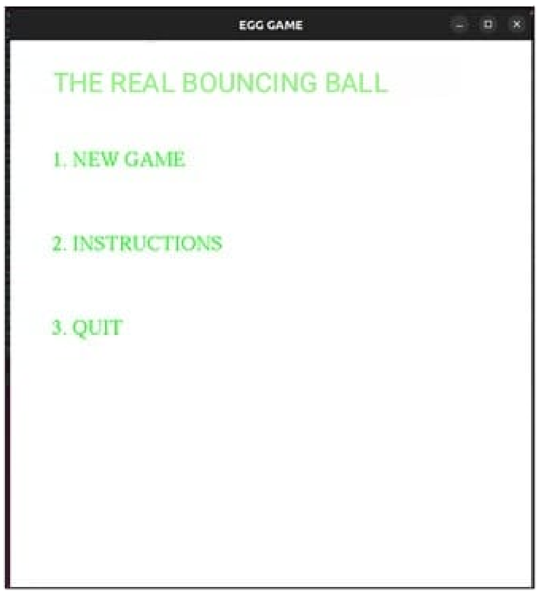
"THE REAL BOUNCING BALL"

Peeyush Rawat 1JB19CS093
Prashant Prasar 1JB19CS098

Press ENTER to go To next screen

Fig 5.1 Starting Page of bouncing ball animation.

In Fig 5.2 the window consists of the text "WELCOME TO THE REAL BOUNCING BALL". It also consists of the options like new game, instructions and quit.



EGG GAME

THE REAL BOUNCING BALL

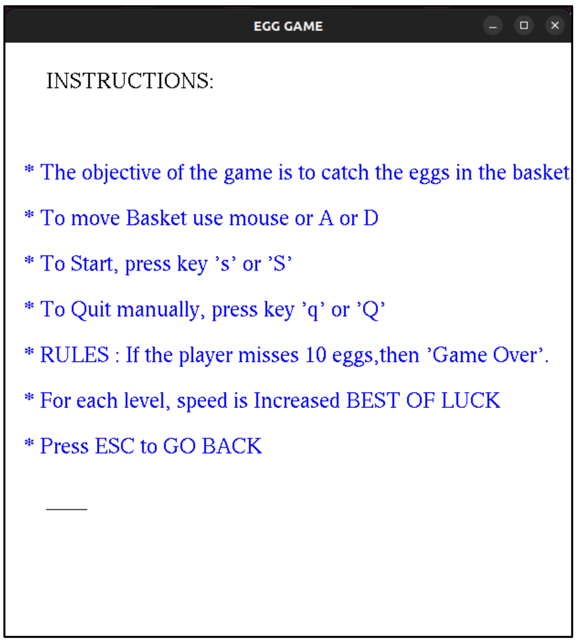
1. NEW GAME

2. INSTRUCTIONS

3. QUIT

Fig 5.2 Menu Page

Fig 5.3 demonstrates that on clicking on 2 in the previous window, we get this window which consists of instructions. This window consists of those instructions that are to be followed in the game.



EGG GAME

INSTRUCTIONS:

* The objective of the game is to catch the eggs in the basket
* To move Basket use mouse or A or D
* To Start, press key ’s’ or ’S’
* To Quit manually, press key ’q’ or ’Q’
* RULES : If the player misses 10 eggs,then ’Game Over’.
* For each level, speed is Increased BEST OF LUCK
* Press ESC to GO BACK

Fig 5.3 Instructions about Project

Fig 5.4 shows the main window of the game. This window consists of the name of the game. On pressing S the will start.

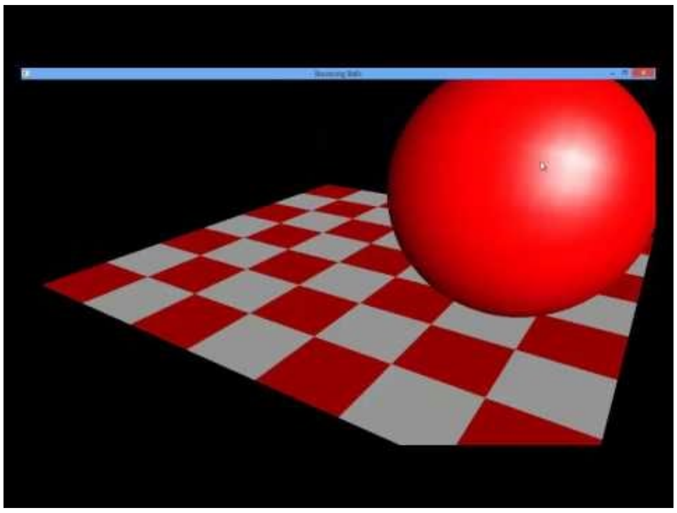


Fig 5.4 Start Screen of bouncing ball

In Fig 5.5 the animation has been started. The ball losses kinetic energy according to the coefficient of restitution between the floor and ball.

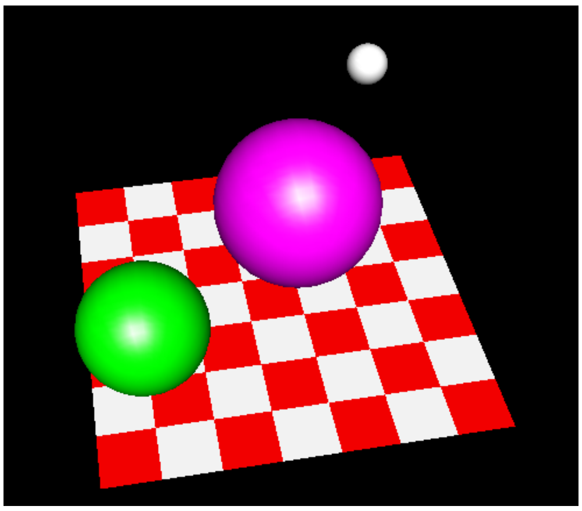
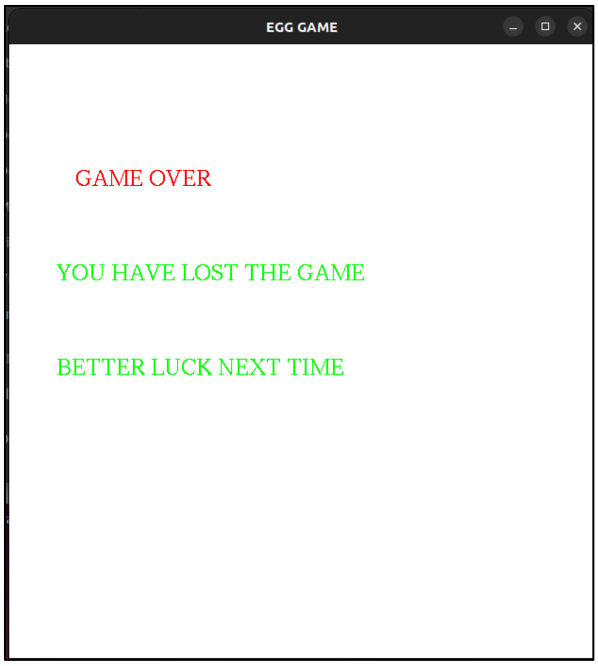


Fig 5.5 the real bouncing ball

Fig 5.6 shows us that the animation has ended and displays the end screen.



EGG GAME

GAME OVER

YOU HAVE LOST THE GAME

BETTER LUCK NEXT TIME

Fig 5.6 Game Over Page

CONCLUSION AND FUTURE ENHANCEMENTS

The aim of the project is to represent the basic animation of The Real Bouncing Ball using the openGL functions available. In this animation there are 3 balls and a checkerboard. The balls will start dropping from the Height H_i and those balls will bounce back to some height H_f after loosing kinetic energy according to the coefficient of restitution of the ball.

The real bouncing ball demonstrates the loss in kinetic energy in a ball after every collision with the floor. After every collision of ball with the checkerboard, i.e., the floor, the ball will bounce back to the height H_f which will be less than or equal to the initial height of the ball before the collision.

$$H_n = \frac{H(1-e^n)}{(1-e)}$$

Where,

H_n : Maximum height reached after the nth collision with the floor

H : Initial maximum height from the ground

e : The coefficient of restitution between the floor and ball.

n : nth collision (n = 1, 2, 3,…)

Once your animation starts, the ball will gradually move towards the floor, which will create an animation indicating that the balls are falling. Then with the help of loops, we can constantly check that the balls have been have touched by the ground. When the ball touches the ground it will move back in the positive Y direction to a height H_n. This process continues till the kinetic energy of the ball reaches to zero and ball comes to halt.

Some future enhancements can be:

Drawing Ball : We can change the size of the ball by modifying the glutSolidSphere, the colour of the ball can also be modified similarly. Background of the colour can also be added and changed accordingly. We can draw any new object in the background as the ball bounces.

Modify : We can modify this Simple Opengl Program by adding some user interaction that will let the ball start and stop. We can add sound to the program as well. The 3D version of the program can also be made as future enchantments.

BIBLIOGRAPHY

[1]. Programming OpenGL for the X Window System by Addison-Wesley

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

[3]. Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL, 5th edition. Pearson Education, 2008

[4]. James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer graphics with OpenGL: pearson education

[5]. M M Raikar & Shreedhara K S Computer Graphics using OpenGL

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

[7]. http://www.opengl-tutorial.org/

[8]. https://ogldev.org/