Visvesvaraya Technological University

Belagavi, Karnataka, 590 014.

A Mini Project Report on

"Bounce Ball Game"

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering

in

Computer Science and Engineering

Semester VI

(18CSL67)

Academic Year 2020-21

Submitted By

Ms. Rashmi Ramakantha Pai, (USN: 2KE18CS032)

Ms. Summaiya Dharwad, (USN: 2KE18CS048)

Mr. Rohit Guru M, (USN: 2KE18CS033)

Mr. Pradeep Kumar Natekal, (2KE18CS027)

Under the Guidance of

Prof. Shridhar Allagi

Department of Computer Science & Engineering

K. L. E. SOCIETY'S

K. L. E. INSTITUTE OF TECHNOLOGY,

Opp. Airport, Gokul, Hubballi-580 030

Phone: 0836-2232681 Website: www.kleit.ac.in

K. L. E. SOCIETY'S

K. L. E. INSTITUTE OF TECHNOLOGY,

Opp. Airport, Gokul, Hubballi-580 030

Phone: 0836-2232681 Website: www.kleit.ac.in

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

Certified that the mini project work entitled **"Bounce Ball Game"** is a bonafide work carried out by **Ms. Rashmi Ramakantha Pai, Ms. Summaiya Dharwad, Mr. Rohit Guru M, Mr. Pradeep Kumar Natekal** bearing USN number **2KE18CS032, 2KE18CS048, 2KE18CS033, 2KE18CS027** in partial fulfillment for the award of degree of **Bachelor of Engineering** in **VI Semester, Computer Science and Engineering** of **Visvesvaraya Technological University**, Belagavi, during the year **2020-21**. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the department library. The mini project report has been approved as it satisfies the academic requirements in respect of mini project work prescribed for the said degree.

Signature of the Guide	**Signature of the HOD**	**Signature of the Principal**
(Prof. Shridhar Allagi)	**(Dr. Yerriswamy T.)**	**(Dr. B. S. Anami)**

Name of the Examiners	**Signature with Date**
1.	
2.	

ACKNOWLEDGEMENT

The mini project report on **“Bounce Ball Game”** is the outcome of guidance, moral support and devotion bestowed on me throughout my work. For this I acknowledge and express my profound sense of gratitude and thanks to everybody who have been a source of inspiration during the project work.

First and foremost, I offer my sincere phrases of thanks with innate humility to our **Principal Dr. B. S. Anami** who has been a constant source of support and encouragement. I would like to thank our **Dean Academics Dr. S. G. Joshi** for his constant support and guidance. I feel deeply indebted to our **H.O.D. Dr. Yerriswamy T.** for the right help provided from the time of inception till date. I would take this opportunity to acknowledge our **Guide Prof. Shridhar Allagi,** who not only stood by us as a source of inspiration, but also dedicated his/her time for me to enable me present the project on time. I would be failing in endeavor, if I do not thank our **Coordinator Prof. Vijeeta Patil** who has helped me in every aspect of my mini project work.

Last but not the least, I would like to thank my parents, friends & well wishers who have helped me in this work.

1. Rashmi Ramakantha Pai
2. Summaiya Dharwad
3. Rohit Guru M
4. Pradeep Kumar Natekal

CONTENTS **PAGE**

ABSTRACT 6

1. INTRODUCTION.

1.1 Problem Definition. 7

1.2 Background Study. 7

1.3 Problem Statement. 7

1.4 Aim and Objectives. 7

1.5 Project Overview. 8

1.6 Literature Overview. 8

2. ABOUT COMPUTER GRAPHICS.

2.1 Computer Graphics. 9

2.2 Application of Computer Graphics. 9

2.3 OpenGL. 13

2.4 OpenGL Utility Toolkit (GLUT). 14

3. REQUIREMENT AND SYSTEM ANALYSIS.

3.1 Functional Requirements. 15

3.2 Non-Functional Requirements. 16

3.3 Software and Hardware Specifications. 16

4. SYSTEM DESIGN.

4.1 Data Flow of the Design. 18

5. FUNCTIONS USED.

5.1 In-built Functions. 19

5.2 User-defined Functions. 22

6. IMPLEMENTATION.

6.1 Pseudocode. 25

6.2 Source Code. 26

7. SNAPSHOTS. 42

8. CONCLUSION. 46

9. FUTURE ENHANCEMENTS. 47

10. BIBLOGRAPHY. 48

ABSTRACT

The project entitled "Bounce Ball Game" incorporates as many features of a simple graphics application as possible and adding a few of our own functions using the skills that have learnt in class. Here, spherical ball is drawn using the knowledge of mathematics and OpenGL library. We use the various user-defined functions for the game to work accordingly using the OpenGL library. We also make changes in the speed of bouncing of the ball.

CHAPTER 1

INTRODUCTION

In this chapter, we will see an overview about project definition, background study, aim and objectives, overview of the project and the literature survey.

1.1 Project Definition.

The main idea behind this project is to display the bouncing ball game with computer graphics. The graphics package used here are the OpenGL libraries – free glut and glew32. The programming language used here is C with OpenGL libraries.

1.2 Background Study.

The project entitled "Bounce Ball Game", incorporates as many features of a simple graphics application as possible and adding a few of our own functions using the skills that have learnt in class. Here, a spherical ball and various other objects are drawn using the knowledge of mathematics and OpenGL library. Changes can be made in the number of balls and the speed of bouncing of the ball.

1.3 Problem Statement.

This project demonstrates the game in which the ball will have a random motion and with every mouse click on the ball, the score will be incremented and the speed of the ball will be increased making the game difficult for the player. The ball will collide with the boundaries of the window and move in a random direction.

1.4 Aim & Objectives.

The aim of this project is to develop a graphics package which supports basic operations which include building a game using Open GL. The package must also have a user-friendly interface. The objective of developing this model was to design and apply the skills we learned.

1.5 Project Overview.

- In this project, we are demonstrating the game in which the ball will have a random motion and with every mouse click on the ball, the score will be incremented and the speed of the ball will be increased making the game difficult for the player. The ball will collide with the boundaries of the window and move in a random direction.

- The player needs to click on the ball which will increment the score by 1 point for each consecutive click. The game has two levels. In Level 1, the player needs to score 10 points in order to proceed to the next level and next level, level 2 with be with increased difficulty.

- The time allotted for each level is 10 seconds. The score will be displayed at the top for each level along with time so as to keep a record of the points and time respectively. The final score is displayed to the player at the end of the game.

1.6 Literature Overview.

- “Binding vision to physics-based simulation: The case study of a bouncing ball”. By: Kyriazis et al. (2011) N. Kyriazis, I. Oikonomidis, and A. Argyros.

- “Ball tracking and 3D trajectory approximation with applications to tactics analysis from single-camera volleyball sequences” .
  By: Hua-Tsung Chen & Wen-Jiin Tsai & Suh-Yin Lee & Jen-Yu Yu.

CHAPTER 2

ABOUT COMPUTER GRAPHICS

In this chapter, we will discuss about computer graphics, it's applications and the OpenGL toolkit and libraries.

2.1 Computer Graphics.

Computer Graphics concerns the pictorial synthesis of real and imaginary objects from their computer-based models. It provides one of the most natural means of communication with a computer and permits extensive, high bandwidth user computer interaction. Computer graphics is used today in many different areas like user interfaces. Plotting in business, science, office automation, electronic publishing and computer aided drafting and design, simulation and animation for scientific visualization and entertainment, cartography etc.

Graphics are visual presentation on some surface, such as well, canvas, computer screen etc. Examples are photography, line art, graphs, engineering drawings, maps, typography etc. Graphics always combine text, illusion and color. Graphics design may consist of the deliberation selection creation or arrangement of typography alone etc. Clarity or effective communication may be the objective, association with other cultural elements may be sought or merely the creation of a distinctive style.

The development of computer graphics has made computers easier to interact with, better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized animation, movies and the video game industry.

2.2 Applications of Computer Graphics.

The various applications of computer graphics are:

- Graphs and Charts.
- Computer Aided Design.
- Virtual-Reality Environment.
- Data Visualization.
- Education and Training.
- Computer Art.
- Entertainment.
- Image Processing.
- Graphical User Interfaces.

Graphs and Charts:

An early application for computer graphics is the display of simple data graphs, usually plotted on a character printer. Data plotting is still one of the most common graphics applications, but today one can easily generate graphs showing highly complex data relationships for printed reports or for presentations using 35 mm slides, transparencies, or animated videos. Graphs and charts are commonly used to summarize financial, statistical, mathematical, scientific, engineering, and economic data for research reports, managerial summaries, consumer information bulletins, and other types of publications.

Computer Aided Design:

A major use of computer graphics is in design processes—particularly for engineering and architectural systems, although most products are now computer designed. Generally referred to as CAD, computer-aided design, or CADD, computer-aided drafting and design, these methods are now routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft, computers, textiles, home appliances, and a multitude of other products. The manufacturing process is also tied

in to the computer description of designed objects so that the fabrication of a product can be automated, using methods that are referred to as CAM, computer-aided manufacturing.

Virtual Reality Environment:

It is a recent application of computer graphics which is used to create virtual-reality environments in which a user can interact with the objects in a three-dimensional scene. Specialized hardware devices provide three-dimensional viewing effects and allow the user to "pick up" objects in a scene. Animations in virtual-reality environments are often used to train heavy equipment operators or to analyze the effectiveness of various cabin configurations and control placements. This allows the designer to explore various positions of the bucket or backhoe that might obstruct the operator's view, which can then be taken into account in the overall tractor design.

Data Visualization:

Producing graphical representations for scientific, engineering, and medical data sets and processes is another fairly new application of computer graphics, which is generally referred to as scientific visualization. The term business visualization is used in connection with data sets related to commerce, industry, and other nonscientific areas. Numerical computer simulations, for example, frequently produce data files containing thousands and even millions of values. Similarly, satellite cameras and other recording sources are amassing large data files faster than they can be interpreted. Other visualization techniques include contour plots, renderings for constant-value surfaces or other spatial regions, and specially designed shapes that are used to represent different data types.

Education and Training:

Computer-generated models of physical, financial, political, social, economic, and other systems are often used as educational aids. Models of physical processes, physiological functions, population trends, or equipment, such as the color-coded diagram in for some training applications, special hardware systems are designed. Examples of such specialized systems are the simulators for practice sessions or training of ship captains, aircraft pilots, heavy-equipment operators, and air traffic-control personnel. Some simulators have no video screens; a flight simulator with only a control panel for instrument flying. But most simulators provide screens for visual displays of the external environment with multiple panels is mounted in front of the simulator.

Entertainment:

Television productions, motion pictures, and music videos routinely use computer-graphics methods. Sometimes graphics images are combined with live actors and scenes, and sometimes the films are completely generated using computer-rendering and animation techniques. Many TV series regularly employ computer-graphics methods to produce special effects, such as the scene in Figure from the television series Deep Space Nine. Some television programs also use animation techniques to combine computer-generated figures of people, animals, or cartoon characters with the live actors in a scene or to transform an actor's face into another shape. And many programs employ computer graphics to generate buildings, terrain features, or other backgrounds for a scene.

Computer Art:

Both fine art and commercial art make use of computer-graphics methods. Artists now have available a variety of computer methods and tools, including specialized hardware, commercial software packages (such as Lumena), symbolic mathematics programs (such as Mathematical), CAD packages, desktop publishing software, and animation systems that provide facilities for designing object shapes and specifying object motions. Example: use of a paintbrush program that allows an artist to "paint" pictures on the screen of a video monitor. A paintbrush system, with a Wacom cordless, pressure-sensitive stylus, was used to produce the electronic painting. The stylus translates changing hand pressure into variable line widths, brush sizes, and color gradations.

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, analyze images, or recognize visual patterns for robotics applications. However, image-processing methods are often used in computer graphics, and computer-graphics methods are frequently applied in image processing.

Typically, a photograph or other picture is digitized into an image file before image-processing methods are employed. Then digital methods can be used to rearrange picture parts, to enhance color separations, or to improve the quality of shading 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. The interface consists of over 250 different function calls which can be used to draw complex three dimensional scenes from simple primitives. 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.

2.3 OpenGL.

OpenGL is competing application programming interfaces (APIs), which can be used in applications to render 2D and 3D computer graphics, taking advantage of hardware acceleration when available. Modern graphics processing unit (GPUs) may implement a particular version of the APIs.

OpenGL is a specification implemented in the C language, though it can use other programming languages. It is built on concept of a state machine, though more recent OpenGL version have transformed it into much more object-based system. As an API, OpenGL depends on no particular language feature, and can be made callable from almost any programming language with the proper bindings. Such bindings exist for Ada, BASIC (Blitz Max (often used to Program Games), Pure Basic, Visual Basic), C#, Delphi, Fortran, Haskell, Java, Lisp, Lua, Pascal, Perl, Python, and Ruby.

Graphics provide 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.

OpenGL has become a widely accepted standard for developing graphics application. Most of our applications will be designed to access OpenGL directly through functions in three libraries. Functions in main GL library 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 functions in GLU can be created from the core GL library. The GLU library is available in all OpenGL implementations; functions in the GLU library begin with the letters ‘GLU’.

The third is called the OpenGL Utility Toolkit (GLUT), which provides the minimum functionality that should be expected in any modern windowing system.

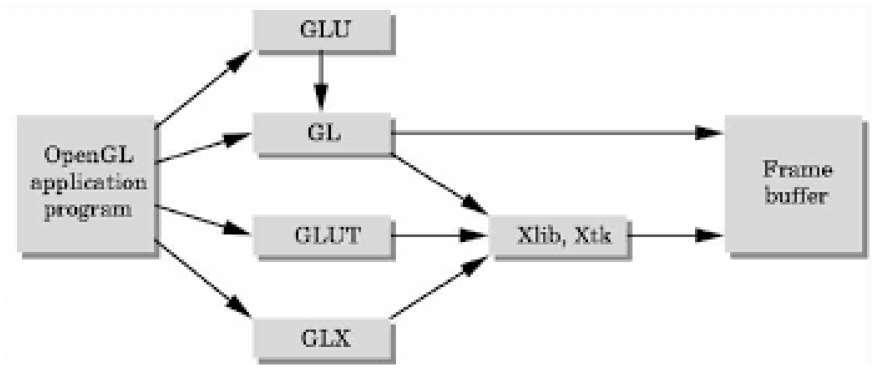


Fig: Library Organization of OpenGL.

2.4 OpenGL Utility Tool Kit (GLUT).

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.

OpenGL drawing commands are limited to those that generate simple geometric primitive, GLUT includes several routines that creates more complicated three-dimensional objects such as a sphere, a torus and a teapot.

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.

The OpenGL Utility Toolkit (GLUT) is a programming interface with ANSIC and FORTAN bindings for writings window system independent OpenGL programs. The toolkit supports the following functionality:

- Multiple windows for OpenGL rendering.
- Call-back driven event processing.
- Sophisticated input devices.
- An “idle” routine and timers.
- A simple, cascading pop-up menu facility.
- Utility routines to generate various solid and wire frame objects.
- Support for bitmap and stroke fonts.
- Miscellaneous window management functions, including managing overlays.

CHAPTER 3

REQUIREMENT AND SYSTEM ANALYSIS

In this chapter, we will discuss and analyze about developing process of Bounce Ball Game including software requirement specification. The functional and non-functional requirements are included in requirement part to provide complete description and overview of system requirement before the developing process is carried out.

3.1 Functional Requirements.

Functional Requirements defines the internal working of the software, i.e., the calculations, technical details, data manipulation and processing and other specific functionality that show how the cases are to be satisfied and how they are supported by non-functional requirements, which impose constraints on the design or the implementation.

The following are the Functional requirements:

- The ability to perform correct operation when the corresponding keys are pressed.
- The ability to display the menu when the right key is clicked.
- When the corresponding option is selected, the corresponding action should be performed.
- Smooth transition between display screens and levels.
- Calculation of the score with precision for each level as well as the final/ total score.
- Correct functioning of the timer during the game.
- The ability to function correctly according the instructions and end the game at the appropriate time and score.
- Clear movement of the ball and change in direction and speed of the ball when colliding with the window edges.
- Correct display of the front screen and transition from the front screen to the next display window.

- The option to quit at any point in the game should be provided to the user, which in this case is achieved by pressing the key ‘q’ or ‘Q’ by the user.

3.2 Non-Functional Requirements.

Nonfunctional requirements are requirements which specify criteria that can be used to judge the operation of the system, rather than specific behaviors. This should be contrasted with functional requirements that specify specific behavior or functions. Typical nonfunctional requirements are reliability and scalability. Nonfunctional requirements are “constraints”, “quality attributes” and “quality of service requirements”.

The following are the Non-Functional requirements:

- The application should provide a simple interface for better understanding of the user.
- The application should be user-friendly and easily understandable even by a person with only basic knowledge of operating a computer.
- The application should provide a pleasant experience for the user.
- The users must get the response within seconds i.e., the response time of a particular function should be minimum.
- The system would exhibit high performance because it would be well optimized.
- There should be an easy back-up feature for the entire data, to prevent losing any data.
- The application should be able to work on any of the modern and any of the common Operating Systems like Linux, Windows and Mac OS.

3.3 Software and Hardware Specifications.

Software Specification:

- Platform: Microsoft Visual Studio 2019.
- Coding Language: C language.
- Libraries Used: OpenGL libraries - free glut, glew32.

Hardware Specification:

- Operating System- Windows.
- RAM: 4 GB RAM or above.
- Hard Disk: 80 GB HDD.
- Keyboard: Any Keyboard.
- Monitor: 15 VGA color or any other.
- System: Intel i5 1.19GHz

CHAPTER 4

SYSTEM DESIGN

In this chapter, we will see the data flow of the application by demonstrating it through a flowchart. Data flow design is as shown below - covering the flow of the data in the system. It describes the relation between user input and the system behavior.

4.1 Data Flow of the Design.

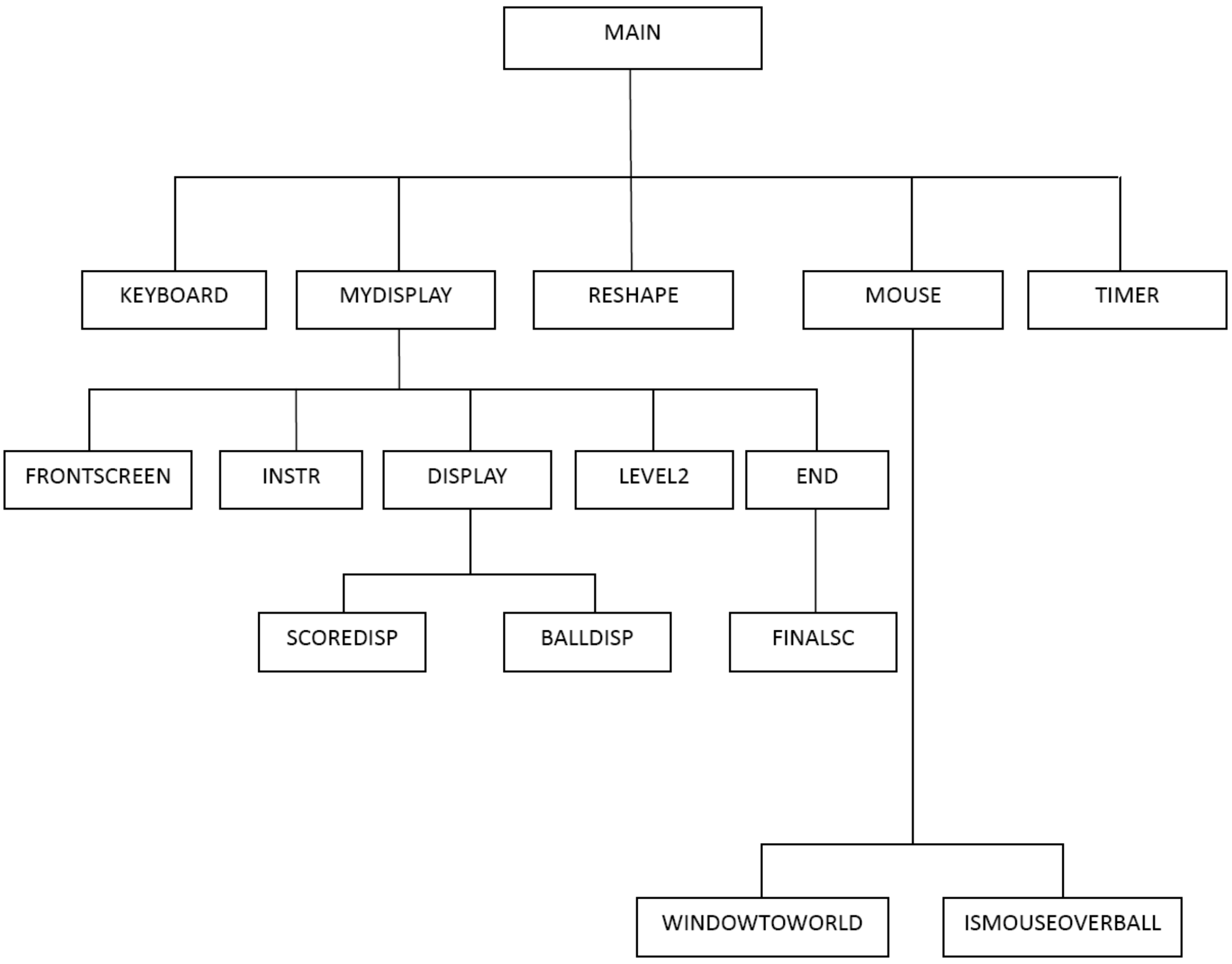


Fig 4.1.1: Flowchart of Design

CHAPTER 5

FUNCTIONS USED

In this chapter, we will discuss the functions used to implement the current system. The in-built functions we have used in this application and the user-defined functions that we have used to perform certain operations are this discussed here.

5.1 In-built Functions.

- **glMatrixMode ():**

  This function sets the current matrix mode. Two commonly used modes are:

  - GL_MODELVIEW: Defines how the objects are transformed.
  - GL_PROJECTON: To set zoom factor and aspect ratio.

- **glutKeyboardFunc ():**

  This function sets the keyboard callback for the current window according to the key pressed. 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.

- **glutMouseFunc ():**

  This function sets the mouse callback for the current window to the mouse clicks. When a user presses and releases mouse buttons in the window, each press and each release generates a mouse callback.

- **glutDisplayFunc ()**:

  This function sets the display callback for the current window and registers the display function that is executed when the window needs to be redrawn.

- **glutPostRedisplay ()**:

  This function marks the current window as needing to be redisplayed and requests that the display callback be executed after the current callback returns.

- **glutMainLoop ()**:

  This function causes the program to enter an event processing loop and calls all the necessary callbacks that have been registered. It should be the last statement in main ().

- **glBegin ():**

  This function initiates and starts the collection of vertices for a new primitive of type mode. The various modes for the new primitive type are:

  - GL_POINTS: Each vertex is treated as a point in GL_POINTS.
  - GL_LINES: Each pair of vertices is considered as a line segment in GL_LINES.
  - GL_TRIANGLE_FAN: It draws a connected group of triangles.

- **glEnd ():**

  This function is used to terminate a list of vertices.

- **glColor3f ():**

  This function specifies colours for the primitives. The colours are specified in RGB codes.

- **glVertex2f ():**

  This function specifies co-ordinates (x, y) values for the vertex positions of the primitives.

- **glClear ():**

  This function clears buffers to pre-set values. The parameter that needs to be passed is the mask value. The two commonly used mask values are:

  - GL_COLOR_BUFFER_BIT: It is used to indicate the buffers that are currently enabled for colour writing.
  - GL_DEPTH_BUFFER_BIT: It is used to indicate the depth buffer.

- **glutBitmapCharacter ():**

  This function renders the bitmap character. The parameters that are needed to be passed are the font type/family and the character which is to displayed.

- **glutInitDisplayMode ():**

  This function is used for initializing the window modes. The bit masks used are:

  - GLUT_DOUBLE: Bit mask to select a double buffered window. Doubling makes the animation smoother.
  - GLUT_RGB: Bit mask to select a colour index mode window. (RGB)

- **glutSwapBuffers ():**

  This function performs a buffer swap on the layer in use for the current window.

- **glPushMatrix ()** and **glPopMatrix ()**:

  These functions push to and pop from the matrix stack corresponding to the current matrix mode.

- **glViewport ():**

  This function sets the extents for the current window. It also specifies the affine transformation of x and y from normalized device coordinates to window coordinates.

- **glClearColor ():**

  This function specifies the background color of the window. It takes four parameters, RGB values and alpha value.

5.2 User-defined Functions.

- **void drawstring ():**

  This function is used to display the strings on the screen. For example: To display name and topic in the front screen.

- **void frontscreen ():**

  This function is used to display the front screen which consists of the topic, names and also the information about which key to be pressed to continue to the game or the instructions menu, as the user desires.

- **void level2 ():**

  This function is used to continue to level 2 when the user has scored greater than or equal to 10 in the level 1.

- **void finalsc ():**

  This function displays the end screen with the final score and scores of the other levels as well as the final screen.

- **void end ():**

  This function is used to display the emoji and strings in the final screen. The finalsc() function is also called in this function to display the scores here.

- **void instr ():**

  This function is used for displaying the instructions screen. It consists of the instructions for the game. The user can proceed to play the game after reading the instructions.

- **void display ():**

  This function is used to display the output screen which includes the ball display, score display as well as the timer.

- **void mydisplay ():**

  This function is used for displaying the appropriate window based on the key pressed. For example: In the front screen, when ‘I’ is pressed the instruction screen must be displayed or when ‘enter’ is pressed then the game screen must be displayed, all these actions are performed by this function.

- **void mykey ():**

  This function is for the keyboard interactions. For the application to interpret which key the user has pressed, there are certain flag bits that are used. These flag bits are set to true through keyboard interaction, so that the application can perform the appropriate action.

- **void balldisp ():**

  This function is used to determine the ball features, such as – direction in which the ball should move, colour of the ball and speed of the ball.

- **void scoredisp ():**

  This function is used to display the timer, scores and the level. It displays score at the top left corner, level at the top centre and timer at the top right corner.

- **void windowToWorld ():**

  This function is used for making appropriate window adjustments.

- **void onMouse ():**

  This function is used to determine what is to be done on mouse clicks. It is used to determine whether the user has clicked on the ball or not and on clicking on the ball, the score is incremented by 1.

- **void reshape ():**

  This function defines what to do when the window is resized.

- **void Timer ():**

  This function marks the current window to be redisplayed and registers the timer callback to be triggered in a specified number of milliseconds.

- **void isMouseOverBall ():**

  This function is used to determine whether the mouse is over the ball or not.

CHAPTER 6

IMPLEMENTATION

In this chapter, we will see the pseudocode and source code to implement the game.

6.1 Pseudocode.

```
void balldisp ()
{
glTranslatef (ballX, ballY, 0.0f);
glBegin (GL_TRIANGLE_FAN);
if(score>=10)
{
glColor3f (0, 0, 1);
}
else
{
glColor3f (1, 0, 0);
}
glVertex2f (0.0f, 0.0f);
int numSegments = 100;
int i;
for (i=0; i<=numSegments; i++)
{
angle = i * 2.0f * PI/numSegments;
glVertex2f (cos (angle) * ballRadius, sin (angle) * ballRadius);
}
glEnd ();
ballX += xSpeed;
ballY += ySpeed;
if (ballX > ballXMax)
{
```

```
xa=ballX;

ballX = ballXMax;

xSpeed = -xSpeed;

}

else if(ballX < ballXMin)

{

xa=ballX;

ballX = ballXMin;

xSpeed = -xSpeed;

}

if(ballY > ballYMax)

{

ya=ballY;

ballY = ballYMax;

ySpeed = -ySpeed;

}

else if(ballY < ballYMin)

{

ya=ballY;

ballY = ballYMin;

ySpeed = -ySpeed;

}

}
```

6.2 Source Code.

The source code of the “Bounce Ball Game” is as follows:

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

#include<math.h>

#include<stdbool.h>

#define PI 3.14159265f

#include<stdio.h>


GLfloat ballRadius = 0.2, xradius=0.2, xxradius=1.0, xxxradius=0.5;

GLfloat ballX = 0.0f;

GLfloat ballY = 0.0f;

GLfloat ballXMax,ballXMin,ballYMax,ballYMin;

GLfloat xSpeed = 0.02f;

GLfloat ySpeed = 0.007f;

int refreshMills = 30;

int refreshend = 60000;

GLfloat angle=0.0;

int xa,ya;

int flag=0,flag1=0,flag2=0,flagl=0,flagd=0;

int score = 0,score1=0,score2=0,endTimer,timer,i;

void *currentfont;

GLfloat  x, y;

GLdouble clipAreaXLeft,clipAreaXRight,clipAreaYBottom,clipAreaYTop;

void balldisp() ;

void scoredisp();

void end ();


void drawstring (float x,float y,float z,char *string)

{
```

```
char *c;
 glRasterPos3f(x,y,z);
 for (c=string; *c!='\0';c++)
 {
  glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,*c);
 }
}
void frontscreen(void)
{
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_TRIANGLE_FAN);
for (angle=0; angle<360; angle+=1)
{
y =(cos(angle*PI/180)*xradius);
x =(sin(angle*PI/180)*xradius);
glVertex2f(x,y);
}
glEnd();
xradius=xradius+0.005;
glColor3f( 1.0,1.0,0.0 );
glBegin(GL_TRIANGLE_FAN);
for (angle=0; angle<360.0; angle+=1)
{
y =(sin(angle*PI/180)*xxradius);
x =(cos(angle*PI/180)*xxradius);
glVertex2f(x,y);
}
```

```
glEnd();
 xxradius=xxradius-0.005;
 glColor3f(0.0,0.0,0.0);
 drawstring(-0.42,0.9,0.0,"CGV MINI PROJECT");
glColor3f(1.0,0.0,0.0);
 drawstring(-0.3,0.6,0.0,"USING OPENGL");
glColor3f( 0.5,0.0,1.0 );
 drawstring(-0.4,0.4,0.0,"BOUNCE BALL GAME");
 glColor3f( 0.0,0.0,0.0 );
 drawstring(-0.9,-0.0,0.0,"BY:");
 glColor3f( 0.0,0.0,0.0 );
 drawstring(-0.9,-0.1,0.0,"Rashmi Pai (2KE18CS032)");
 drawstring(-0.9,-0.2,0.0,"Rohit Guru (2KE18CS033)");
 drawstring(-0.9,-0.3,0.0,"Pradeep Natekal (2KE18CS027)");
 drawstring(-0.9,-0.2,0.0,"Summaiya Dharwad (2KE18CS048)");
 drawstring(0.45,-0.0,0.0,"GUIDE : ");
drawstring(0.25,-0.15,0.0,"Prof. Shridhar Allagi");
glColor3f( 0.0,0.0,1.5 );
 drawstring(-0.4,-0.7,0.0,"Press i for Instructions");
//glColor3ub( rand()%250, rand()%250, rand()%200 );
 drawstring(-0.4,-0.9,0.0,"Press ENTER to Start");
 drawstring(-0.5,-0.55,0.0,"Academic Year: 2020-2021");
 glColor3f( 0.0,1.0,0.4 );
 glutSwapBuffers();
 }
void level2()
```

```
{
glClear(GL_COLOR_BUFFER_BIT);
drawstring(-0.175,0.0,0.0,"LEVEL 2 ");
drawstring(-0.4,-0.25,0.0,"Press c to Continue...");
glutSwapBuffers();
}

void finalsc()
{
int z,j=0,k=0,y,a=0,b=0, w,d=0,e=0;
z=score;
y=score1;
w=score2;
glColor3f(1.0,1.0,0.0);
glLoadIdentity();
drawstring(-1.0,1.0,0.0,"FINAL SCORE : ");
while(z > 9)
{
k = z % 10;
glRasterPos2f (-0.20,1.0);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,48+k);
z /= 10;
glRasterPos2f(-0.25,1.0);
}
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,48+z);
drawstring(-1.0,0.9,0.0,"LEVEL 1 : ");
```

```
while(w > 9)

{

e = w % 10;

glRasterPos2f (-0.20,0.9);

glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,48+e);

w /= 10;

glRasterPos2f(-0.25,0.9);

}

glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,48+w);

drawstring(-1.0,0.8,0.0,"LEVEL 2 : ");


while(y > 9)

{

b = y % 10;

glRasterPos2f (-0.20,0.8);

glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,48+b);

y /= 10;

glRasterPos2f(-0.25,0.8);

}

glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,48+y);

}

void end()

{

glClear(GL_COLOR_BUFFER_BIT);

glColor3f(1.0,1.0,0.0);

glBegin(GL_TRIANGLE_FAN);
```

```
for(angle=0; angle<360; angle+=1)
{
y =0.3+(cos(angle*PI/180)*xxxradius);
x =(sin(angle*PI/180)*xxxradius);
glVertex2f(x,y);
}
 glEnd();
 glColor3f(0.0,0.0,0.0);
 glPointSize(15.0);
 glBegin(GL_POINTS);
 glVertex2f(-0.2,0.5);
 glVertex2f(0.2,0.5);
 glEnd();
 glLineWidth(10.0);
 glBegin(GL_LINES);
 glVertex2f(0.0,0.2);
 glVertex2f(0.0,0.4);
 glEnd();
 glBegin(GL_LINES);
 glVertex2f(-0.2,0.0);
 glVertex2f(0.0,0.1);
  glVertex2f(0.0,0.1);
   glVertex2f(0.2,0.0);
 glEnd();
 glColor3f(1.0,0.0,0.0);
 drawstring(-0.3,-0.7,0.0,"GAME OVER");
 finalsc();
```

```
glutSwapBuffers();

}


void instr()

{

glClear(GL_COLOR_BUFFER_BIT);

glColor3f( 0.5,0.0,1.0 );

drawstring(-0.25,0.85,0.0,"Instructions");

glColor3f(1.0,1.0,0.0);

drawstring(-0.9,0.55,0.0,"1. The Player needs to click on the ball in order");

drawstring(-0.8,0.45,0.0,"to score. ");

drawstring(-0.9,0.35,0.0,"2. Per click score will be incremented by 1. ");

drawstring(-0.9,0.25,0.0,"3. And the speed of the ball will increase ");

drawstring(-0.9,0.15,0.0,"4. To Quit at any stage press q ");

drawstring(-0.9,0.05,0.0,"5. Timer of 10 secs for two levels");

glColor3ub( rand()%250, rand()%250, rand()%200 );

drawstring(-0.6,-0.9,0.0,"PRESS ENTER TO START");

glutSwapBuffers();

}

void display()

{

glClear(GL_COLOR_BUFFER_BIT);

glMatrixMode(GL_MODELVIEW);

glLoadIdentity();

balldisp();

scoredisp();
```

```
glutSwapBuffers();
endTimer+=refreshMills;
timer=endTimer/1000;
}


void mydisplay(void)
{
if(flag==0&&flag1==0&&flagd==0)
            frontscreen ();
if(flag1==1&&flag==0)
            instr();
if(flag==1&&flagd==0)
{
display();
}
if(score>=10&&flag1==0)
{
level2();
endTimer=0;
flagd=1;
//level1sc=score;
}
if(endTimer>=10000&&flag==1)
{
end();
flagd=1;
}
```

```
else if(endTimer>=20000&&flag1==1) {
        end();
        flagd=1;   }
}
void mykey( unsigned char key, int x, int y )
{
switch(key)    {
case 13:  if(flag==0)
                flag=1;
                //flag1=0;
                //ySpeed=ySpeed+0.01;
        break;
case 'i': if(flag==0)
                flag1=1;
        break;
case 'c':if(flagl==0)
        {
                flag1=1;
                flagd=0;
                //score=0;
                ySpeed=ySpeed+0.05;       }
        break;
case 'q' :
case 'Q' : exit(0);
        break;
}
}
```

```
void balldisp()

{

glTranslatef(ballX,ballY,0.0f);

glBegin(GL_TRIANGLE_FAN);

if(score>=10)

{

glColor3f(0,0,1);

//level1sc=score;

}

else

{

glColor3f(1,0,0);

}

glVertex2f(0.0f,0.0f);

int numSegments = 100;

int i;

for(i=0;i<=numSegments;i++)

{

angle = i*2.0f*PI/numSegments;

glVertex2f(cos(angle)*ballRadius,sin(angle)*ballRadius);

}

glEnd();

ballX += xSpeed;

ballY += ySpeed;


if(ballX > ballXMax)

{
```

```
xa=ballX;

ballX = ballXMax;

xSpeed = -xSpeed;

}

else if(ballX < ballXMin)

{

xa=ballX;

ballX = ballXMin;

xSpeed = -xSpeed;

}

if(ballY > ballYMax)

{

ya=ballY;

ballY = ballYMax;

ySpeed = -ySpeed;

}

else if(ballY < ballYMin)

{

ya=ballY;

ballY = ballYMin;

ySpeed = -ySpeed;

}

}


void scoredisp()

{

int z, j=0, k=0, i, s=0;
```

```
z=score;
glColor3f (1.0,1.0,0.0);
glLoadIdentity ();
glRasterPos2f (0.28,1);
drawstring(0.38,1,0.0,"TIME : ");
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,48+timer);
if(score>=10)
drawstring(-0.25,1,0,"LEVEL 2");
else
drawstring(-0.25,1,0,"LEVEL 1");
drawstring(-1,1,0.0,"SCORE : ");
while(z > 9)
{
k = z % 10;
glRasterPos2f (-0.58,1);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,48+k);
z /= 10;
glRasterPos2f(-0.62,1);
}
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,48+z);
}


bool isMouseOverBall(float worldClickX, float worldClickY, float ballX, float ballY)
{
float distance = sqrt(pow(worldClickX - ballX, 2.0f) + pow(worldClickY - ballY, 2.0f));
return distance < ballRadius;
}
```

```
void windowToWorld(int windowX, int windowY, double *worldX, double *worldY)
{
int x = windowX - 500 / 2;
int y = windowY - 500 / 2;
*worldX = (float)x / 250.0f;
*worldY = -(float)y / 250.0f;
}
void onMouse(int button, int state, int x, int y)
{
GLdouble worldClickX, worldClickY;
windowToWorld(x, y, &worldClickX, &worldClickY);
bool clicked = isMouseOverBall(ballX, ballY, worldClickX, worldClickY);
if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
{
if(clicked)
{
score=score+1;
if(score>10)
{
score1=score1+1;
score2=10;
}
else
score2=score;
}
}
}
```

```
void reshape (GLsizei width,GLsizei height)

{

if(height ==0) height = 1;

GLfloat aspect = (GLfloat)width / (GLfloat)height;

glViewport(0,0,width,height);

glMatrixMode(GL_PROJECTION);

glLoadIdentity();

if(width >=height)

{

clipAreaXLeft = -1.0 * aspect;

clipAreaXRight = 1.0 * aspect;

clipAreaYBottom = -1.0;

clipAreaYTop = 1.0;

}

else

{

clipAreaXLeft = -1.0;

clipAreaXRight = 1.0 ;

clipAreaYBottom = -1.0 / aspect;

clipAreaYTop = 1.0/ aspect;

}

gluOrtho2D (clipAreaXLeft, clipAreaXRight, clipAreaYBottom, clipAreaYTop+0.10);

ballXMin = clipAreaXLeft + ballRadius;

ballXMax =  clipAreaXRight - ballRadius;

ballYMin =  clipAreaYBottom + ballRadius;

ballYMax =  clipAreaYTop - ballRadius;

}
```

```
void Timer(int value)

{

glutPostRedisplay();

glutTimerFunc(refreshMills, Timer, 5);

}


int main(int argc,char* argv[])

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(1080,1080);

glutCreateWindow("Bouncing Ball");

glutMouseFunc(onMouse);

glutDisplayFunc(mydisplay);

glutKeyboardFunc(mykey);

glutReshapeFunc(reshape);

glutTimerFunc(0,Timer,0);

glutMainLoop();

}
```

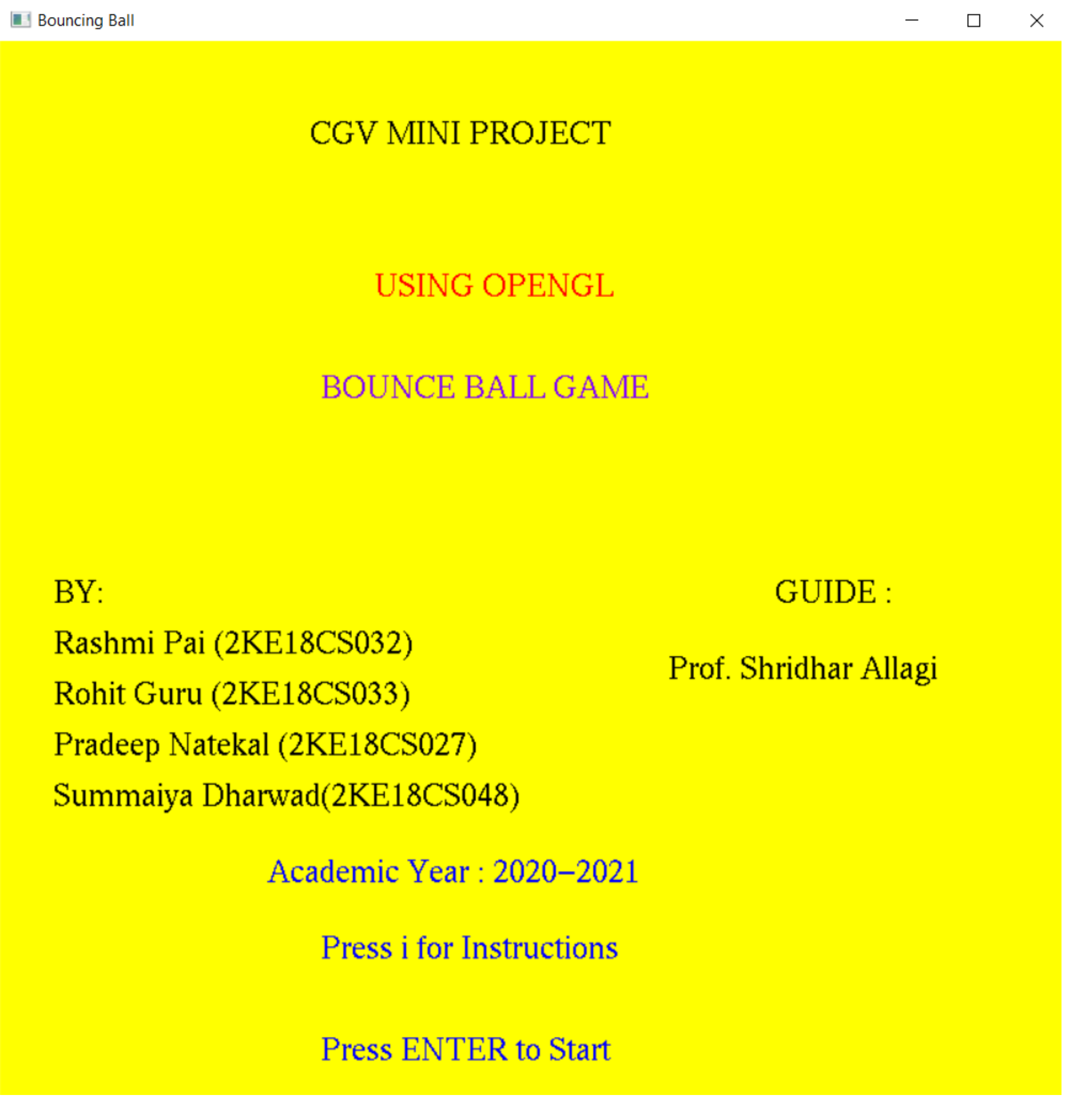
CHAPTER 7

SNAPSHOTS

In this chapter, we will see some screen shots of the various interfaces and displays in the project.

Introduction Page:

This is a snapshot of the page that the user first encounters on running the program. It displays the basic information about the project.



Bouncing Ball

CGV MINI PROJECT

USING OPENGL

BOUNCE BALL GAME

BY:
Rashmi Pai (2KE18CS032)
Rohit Guru (2KE18CS033)
Pradeep Natekal (2KE18CS027)
Summaiya Dharwad(2KE18CS048)

GUIDE :
Prof. Shridhar Allagi

Academic Year : 2020–2021

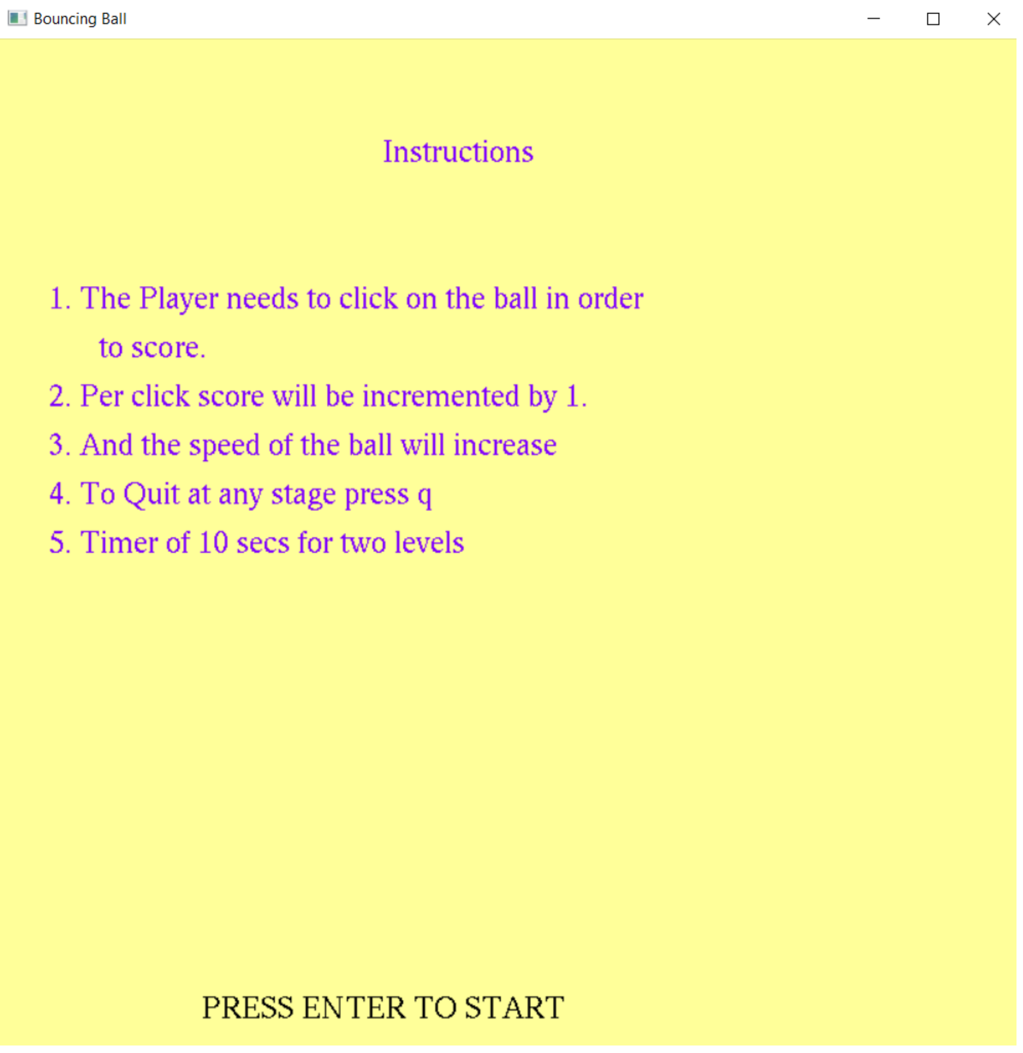
Press i for Instructions

Press ENTER to Start

Fig 7.1: Introduction Page.

Instructions Page:

This is a snapshot of the instructions screen. It displays the instructions of the game and the user can proceed to play the game by pressing ‘ENTER’.



Instructions

1. The Player needs to click on the ball in order
   to score.
2. Per click score will be incremented by 1.
3. And the speed of the ball will increase
4. To Quit at any stage press q
5. Timer of 10 secs for two levels

PRESS ENTER TO START

Fig 7.2: Instructions Page.

Level 1:

This is a snapshot of the first level of the game. It displays the score, level, timer and the ball.

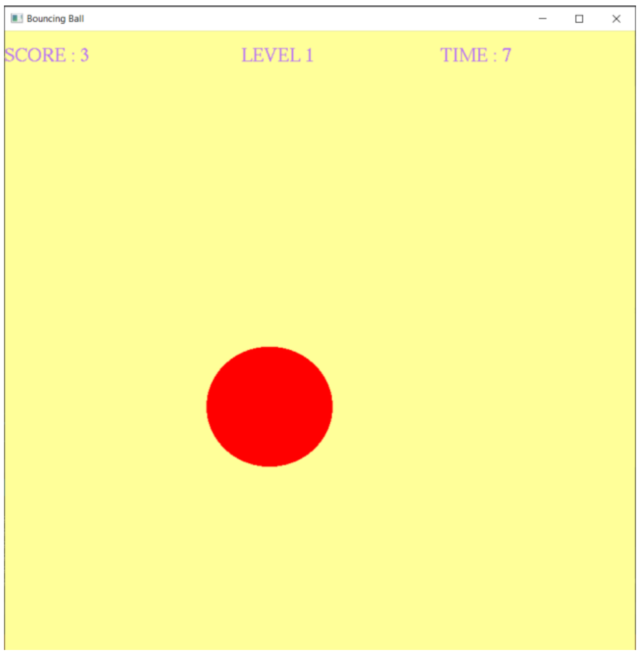


Fig 7.3: Level 1 Screen.

Continue:

This is a snapshot of the continue screen. If the user has passed level 1 then the user can go to the next level by pressing ‘c’.

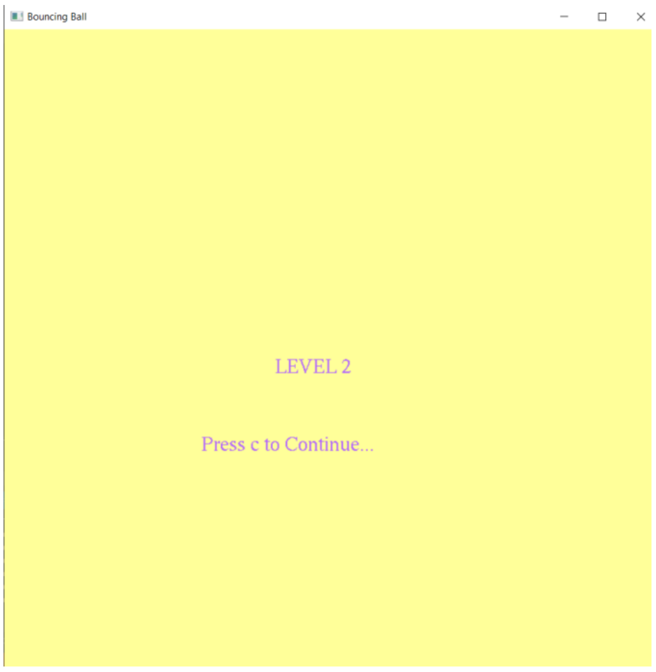


Fig 7.4: Continue Screen.

Level 2:

This is a snapshot of the second level of the game. It displays the score, level, timer and the ball.

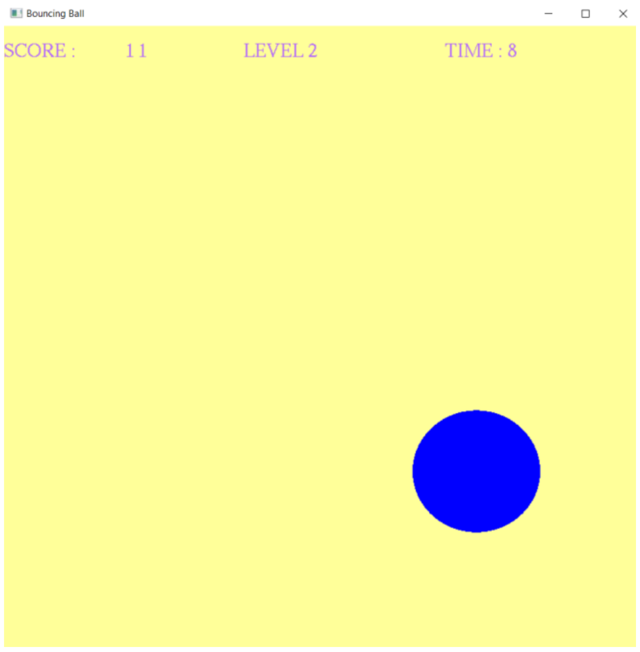


Fig 7.4: Level 2 Screen.

Game Over:

This is a snapshot of the final screen which is displayed when the game is over.

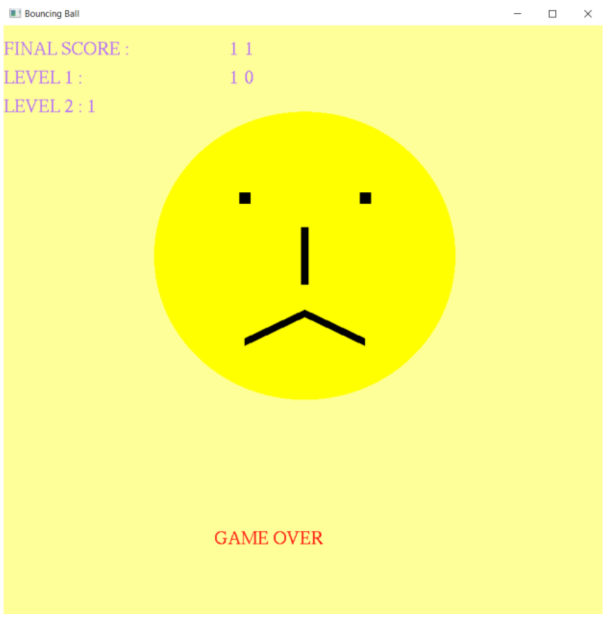


Fig 7.5: Game Over!

CHAPTER 8

CONCLUSION

Presently this application is designed to be very User Friendly with simple interaction. We have fine-tuned our programming skills with regard to OpenGL and many features are enhanced to “Bounce Ball Game”. Also, checking and verification of all possible types of functions are been taken care of.

CHAPTER 8

FUTURE ENHANCEMENTS

The mini project developed has scope further by adding the following enhancements:

a) More levels can be added.

b) Different themes can be added, giving the user an option to choose the theme.

c) Option for the user to set the timer.

d) Option for the user to choose the bouncing object – which can be a ball as usual or a cube.

CHAPTER 9

BIBLOGRAPHY

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

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

[3] https://stackoverflow.com/
[Courtesy: https://stackoverflow.com/questions/7968748/understanding-opengl]

[4] https://docs.microsoft.com/
[Courtesy: https://docs.microsoft.com/en-us/windows/win32/opengl/opengl]

[5] https://community.khronos.org/
[Courtesy: https://community.khronos.org/c/opengl-general/34]