



**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“Jnana Sangama”, Belgaum-590018.**



**Computer Graphics Mini Project On**

**“Ball in the Basket”**

**Submitted in partial fulfillment for the requirements of the VI Semester degree of**

**BACHELOR OF ENGINEERING  
IN  
COMPUTER SCIENCE AND ENGINEERING**

**For The Academic Year**

**2022-23 By**

**Suryateja K (1DB20CS132)**

**Shailesh B S (1DB20CS134)**

**M Rishi Gowda (1DB20CS138)**

**Under The Guidance Of**

**Prof. Vishesh J**

**Asst. Professor**

**Dept. of CSE, DBIT**



**Department of Computer Science and Engineering**

**DON BOSCO INSTITUTE OF TECHNOLOGY**

**Kumbalagodu, Mysore Road, Bengaluru - 560 074.**



# DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Bengaluru – 560 074.



## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### CERTIFICATE

This is to certify that the mini project report entitled “**Ball in the Basket**” is a bonafide work carried out by **Suryateja K(1DB20CS132)**, **Shailesh B S(1DB20CS134)**, **M Rishi Gowda(1DB20CS138)** in partial fulfillment of award of Degree of **Bachelor of Engineering in Computer Science and Engineering** of Visvesvaraya Technological University, Belagavi, during the academic year 2022-2023. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated. The mini project has been approved as it satisfies the academic requirements associated with the degree mentioned.

**Signature of guide**

.....

**Prof. Vishesh J**

Asst. Professor,  
Dept.of CSE,  
DBIT, Bengaluru.

**Signature of HOD**

.....

**Dr K B Shivakumar**

Head of Dept.,  
Dept. of CSE,  
DBIT, Bengaluru.



# DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Bangalore – 560 074.



## DECLARATION

We, student of sixth semester B.E, Department of Computer Science and Engineering, Don Bosco Institute of Technology, Kumbalagodu, Bangalore, declare that the mini project work entitled “**Ball in the Basket**” has been carried out by me and submitted in partial fulfillment of the course requirements for the award of degree in **Bachelor of Engineering in Computer Science and Engineering** of **Visvesvaraya Technological University, Belgaum** during the academic year **2022-2023**. The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

**Place: Bangalore**

**Date: 05/07/2023**

**Suryateja K(1DB20CS132)**

**Shailesh B S(1DB20CS134)**

**M Rishi Gowda(1DB20CS138)**



## ACKNOWLEDGEMENT

Here by we are submitting the CG mini project report on “**BALL IN THE BASKET**”, as per the scheme of Visvesvaraya Technological University, Belgaum.

In this connection, we would like to express my deep sense of gratitude to my beloved institution Don Bosco Institute of Engineering and also, we like to express my sincere gratitude and indebtedness to **Dr. B S Nagabhushana, Principal, DBIT, Bangalore.**

we would like to express are sincere gratitude to **Dr. K.B. Shivakumar, Head of Department of Computer Science and Engineering**, for providing a congenial environment to work in and carryout my mini project.

we would like to express the deepest sense of gratitude to thank my Project Guide **Prof. Vishesh J, Assistant Professor, Department of Computer Science and Engineering, DBIT, Bangalore** for his constant help and support extended towards me during the course of the project.

Finally, we are very much thankful to all the teaching and non-teaching members of the Department of Computer Science and Engineering, my seniors, friends and my parents for their constant encouragement, support and help throughout completion of mini project.

**Suryateja K(1DB20CS132)**  
**Shailesh B S(1DB20CS134)**  
**M Rishi Gowda(1DB20CS138)**





## **ABSTRACT**

Computer Graphics has grown into a very important topic in the branch of Computer Science. This is due to an effective and rapid communication formed between man and the machine. The goal of this mini project is to construct a Bermuda Triangle. I make use of different concepts such as `pushmatrix()`, `translate()`, `popmatrix()`, timer function. The code implemented makes use of various OpenGL functions for translation, rotation and keyboard callback function, built-in functions for solids and many more. The concepts of computer graphics stand a backbone to achieve the aforementioned idea. Primitive drawing, event driven interactions and basic animation have been the important concepts brought out by this application.



## LIST OF FIGURES

FIGURE:	PAGE:
Fig 5.1: Flow Diagram.....	06
Fig 5.2: Graphical interaction loop.....	07
Fig 7.1: Initial state.....	13
Fig 7.2: Instructions To Play The Game.....	13
Fig 7.3: Options to Start OR Quit the Game.....	14
Fig 7.4: Level 1 of the Game.....	14
Fig 7.5: Level 2 of the Game.....	15
Fig 7.6: GAMEOVER and Displaying of the Score.....	15

## LIST OF ABBREVIATIONS

- OpenGL: Open Graphics Library
- GLU: OpenGL Utility Library
- GLUT : OpenGL Utility Toolkit



# CONTENTS

<b>Acknowledgement</b>	<b>(I)</b>
<b>Abstract</b>	<b>(II)</b>
<b>List of Figures</b>	<b>(III)</b>
<b>List of Abbreviations</b>	<b>(III)</b>

<b>CHAPTERS</b>				<b>Pg. No</b>
<b>1.</b>			<b>INTRODUCTION</b>	<b>1</b>
<b>2.</b>			<b>LITERATURE SURVEY</b>	<b>2</b>
<b>3.</b>			<b>PROPOSED SYSTEM</b>	<b>4</b>
<b>4.</b>			<b>ANALYSIS</b>	
	4.1		Software Requirements	5
	4.2		Hardware Requirements	5
<b>5.</b>			<b>SYSTEM DESIGN</b>	
	5.1		FLOWCHART/ARCHITECTURE	6
	5.2		CONCEPTS AND PRINCIPLES OF OPENGL	7
<b>6.</b>			<b>IMPLEMENTATION</b>	
	6.1		OPENGL RELATED LIBRARIES	8
	6.2		WINDOWS MANAGEMENT IN OPENGL	8
	6.3		CALLBACK FUNCTIONS	9
	6.4		RUNNING THE PROGRAM	12
<b>7.</b>			<b>SNAPSHOTS</b>	<b>13</b>
<b>8.</b>			<b>CONCLUSION</b>	<b>16</b>
			<b>REFERENCES</b>	<b>16</b>



## CHAPTER 1

# INTRODUCTION

Graphics provides one of the natural means of communicating with a computer. Graphics has also become a key technology for communicating ideas, data and trends in most areas of commerce, science, engineering and education. So, graphics refers to pictures, sketch of building, flowcharts, control flow diagrams, bar charts, and pie charts.

Computer graphics is the creation, manipulation and storage of models and images of picture objects by the aid of computers. This was started with the display of plotters and CRT. Computer graphics is also defined as the study of techniques to improve the communication between user and machine. Thus, computer graphics is one of the effective media of communication between machine and user.

### **Classification:**

Computer graphics is broadly classified into three categories:

- Based on type of Object:

- 2-Dimensional Graphics (Ex. Pixel, line, circle etc...)

- 3-Dimensional Graphics (Ex. Cube, polyhedron etc...)

- Based on User Interaction:

- Interactive Computer Graphics

- Non-Interactive Computer Graphics

- Based on Application:

- Business or presentation Graphics

- Scientific Graphics

- Scaled Drawings

- Cartoons, games, and artwork



## CHAPTER 2

### LITERATURE SURVEY

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube (CRT) screens soon after the introduction of computers.

Computer graphics today largely interactive, the user controls the contents, structure, and appearance of objects and of displayed images by using input devices, such as keyboard, mouse, or touch-sensitive panel on the screen. Graphics based user interfaces allow millions of new users to control simple, lowcost application programs, such as spreadsheets, word processors, and drawing programs.

OpenGL (Open Graphics Library) is a standard specification defining a crosslanguage, 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 threedimensional 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.

In addition, SGI had a large number of software customers; by changing to the OpenGL API, they planned to keep their customers locked onto SGI (and IBM) hardware for a few years while market support for OpenGL matured to bring to market 3D hardware, supported by extensions made to the PHIGS standard. In 1992, SGI led the creation of the OpenGL architectural review board (OpenGL ARB), the group of companies that would maintain and expand the OpenGL specification took for years to come.

Many OpenGL functions are used for rendering and transformation purposes.

Transformations functions like `glRotate()`, `glTranslate()`, `glScaled()` can be used. 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 and rendering surfaces.

OpenGL Utility Toolkit (GLUT) is a window-system-independent toolkit, written by Mark Kill guard, to hide the complexities of differing window APIs.

To achieve the objective of the project, information related to the light sources is required with OpenGL we can manipulate the lighting and objects in a scene to create many different kinds of effects. It explains how to control the lighting in a scene, discusses the OpenGL conceptual model of lighting, and describes in detail how to set the numerous illumination parameters to achieve certain effects.

The properties of a light source like its material, diffuse, emissive, has to mention in the project. So, to design the light source and the objects, programming guide of an OpenGL is used.

## CHAPTER 3

### PROBLEM DEFINITION

#### PROPOSED SYSTEM

In the proposed system, OpenGL is a graphic software system 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, we must work through whatever windowing system controls the hardware you're using.

The problem is to develop a computer graphics project called "Ball in the Basket." The goal is to create a realistic simulation where a ball is thrown towards a basket, and the player's objective is to successfully shoot the ball into the basket. The project aims to provide an engaging and visually appealing experience, incorporating principles of physics and graphics rendering techniques.

Physics Engine, User Interaction, Graphics Rendering, Collision Detection, Scoring System, User Interface, Sound Effects, Game Modes these components are used.

Overall, the "Ball in the Basket" project aims to create an immersive and entertaining computer graphics experience by combining realistic physics simulation, visually appealing graphics, accurate collision detection, and interactive gameplay elements.

## CHAPTER 4

# ANALYSIS

### 4.1 SOFTWARE REQUIREMENTS

- Operating System : Windows 10
- Language : C++
- Compiler : Microsoft Visual Studio Code 2010
- OpenGL : library Files And dll files
- Editor : Visual C++

### 4.2 HARDWARE REQUIREMENTS

- Processor : Intel Core i3
- RAM : 8GB
- Hard Disk : 20 GB(approx.)
- Keyboard : Standard qwerty serial or PS/2 keyboard
- Mouse : Standard serial or PS/2 mouse
- Monitor : VGA color Monitor

## CHAPTER 5

# SYSTEM DESIGN

### 5.1 FLOWCHART/ARCHECTURE

The system is designed with the Bermuda Triangle with the following system design description. The program starts with start with the main function. There are init, display, keyboard function. Then the control flows to the init function then the control flows to display function then followed by keyboard function these are used to operate step by step process of transmission.

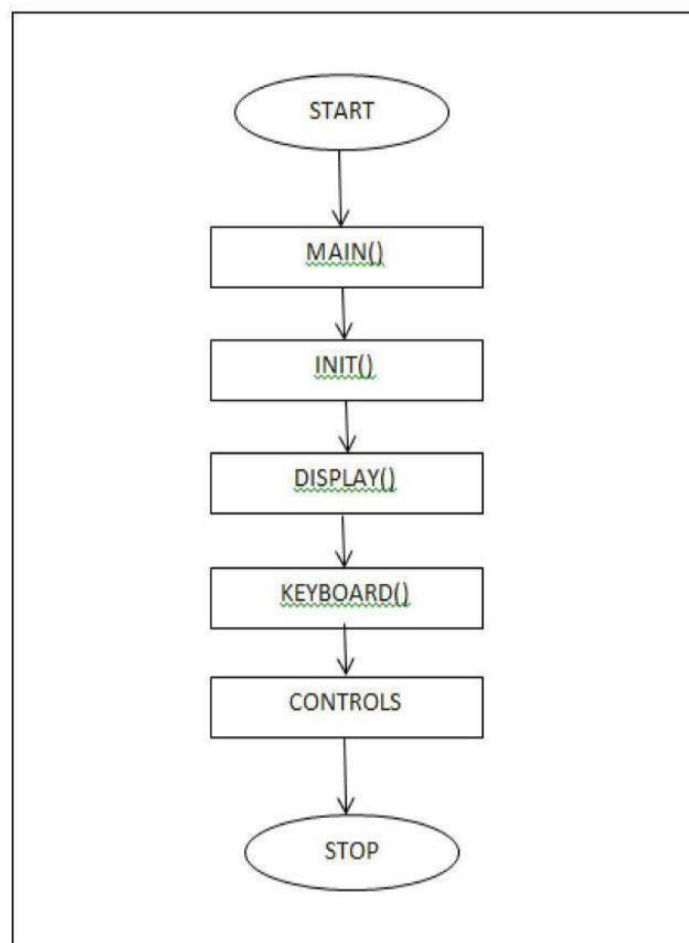


Figure 5.1: Flow Diagram

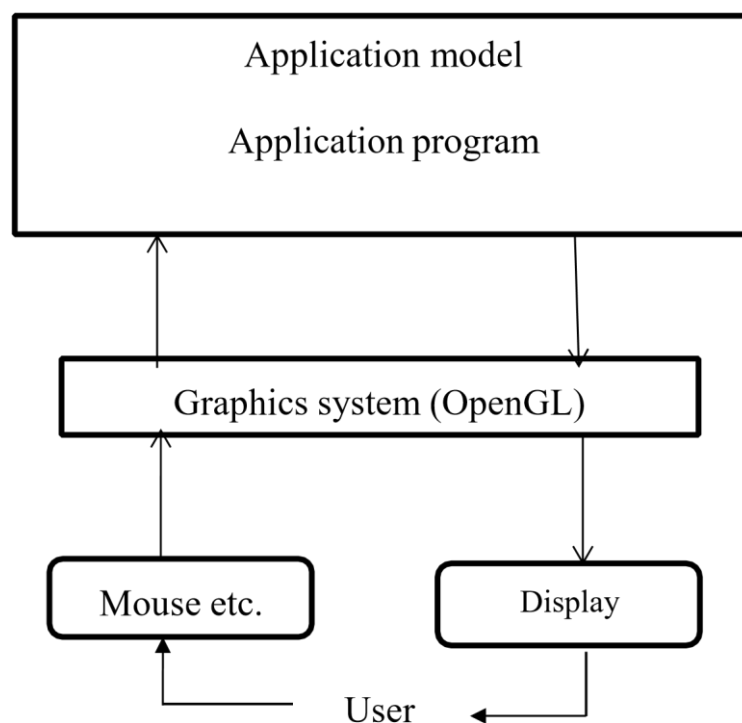
## 5.2 CONCEPTS AND PRINCIPLES OF OPENGL

### 5.2.1 THE OPENGL MODEL:

Figure relationships between an application program, the graphics system, input and output devices, and the user.

### 5.2.2 THE GRAPHICAL INTERFACE:

The application program has its own internal model. It draws the graphics using the facilities of the graphics system. The user views the graphics, and uses input devices, such as a mouse, to interact. Information about the users is sent back to the applications are sent back to the application, which decides what action to take. Typically, it will make changes to its internal model, which will cause the graphics to be updated, and so another loop in the interaction cycle begins.



**Fig 5.2: Graphical interaction loop**

## CHAPTER 6

### IMPLEMENTATION

#### 6.1 OPENGL related libraries

##### 6.1.1 LIBRARIES:

OpenGL provides a powerful but primitive set of rendering command, and all the higher-level drawing must be done in terms of these commands. There are several libraries that allow you to simplify your programming tasks, including the following:

- 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 and rendering surfaces.
- OpenGL Utility Toolkit (GLUT) is a window-system-independent toolkit, written by

Mark Kilgard, to hide the complexities of differing window APIs.

##### 6.1.2 INCLUDE FILES:

For all OpenGL applications, you want to include the `gl.h` header file in every file. Almost all OpenGL applications use GLU, which also requires inclusion of the `glu.h` header file. So almost every OpenGL source file begins with: `#include<GL/gl.h>;`  
`#include<GL/glut.h>`

#### 6.2 WINDOWS MANAGEMENT IN OPENGL

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`). It should be called before any other GLUT routine.
- **glutInitDisplayMode**(unsigned int mode) specifies whether to use an RGBA or color- index color model. We can also specify whether we want a single or double

buffered window. GLUT SINGLE: selects a single-buffered window which is the default if isn't called; GLUT DOUBLE: selects a double-buffered window.

- **glutInitWindowPosition**(intx,inty) specifies the screen location for the upperleft corner of your window.
- **glutInitWindowSize** (intwidth,intsize) specifies the size, pixels, of window.
- **glutCreateWindow** (char \*string) creates a window with an OpenGL context. It returns a unique identifier for the new window.

### 6.3 CALLBACK FUNCTIONS

All callback function more often just called a callback, in C function, written by the application programmer. But there's one important difference between a callback function and an ordinary C function: application never calls the callback function directly. Instead, the callback function is called by OpenGL.

#### 6.3.1 THE DISPLAY CALLBACK:

**glutDisplayFunc** (void (\*func)(void)) is the first and most important event callback function. Whenever GLUT determines the contents of the window to be redisplayed.

- The callback function registered by glutDisplayFunc () is executed. Therefore, we should put all the routines you need to redraw the scene in the display callback function.
- Void glutDisplayFunc(void(\*func)(void));

glutDisplayFunc () registers the name of the callback function to be invoked when OpenGL needs to redisplay the contents of the window. The application must register a display function – it is not optional.

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.



### 6.3.2 THE KEYBOARD CALLBACK:

- void **glutKeyboardFunc**(void(\**func*) (unsigned char *key*, intx, inty));
- **glutKeyboardFunc** () registers the application function to call when OpenGL detects a key press generating an ASCII character. This can only occur when the mouse focus is inside the OpenGL window. It expects a function *func* () which returns void, and has three arguments *key*, *x* and *y*. So, it's function is:

```
void keyboard (unsigned char key, intx, inty)
{
/* called when a key is pressed */
}
```

### 6.3.3 THE MOUSE CALLBACK

- void **glutMouseFunc** (void (\**func*) (int*button*, int*state*, intx, inty)); **glutMouseFunc** () register an application callback function which GLUT will call when the user presses a mouse button within the window. The following values are passed to the callback function:
- button records which button was pressed, and can be
  - GLUT LEFT BUTTON
  - GLUT MIDDLE BUTTON
  - GLUT RIGHT BUTTON
- state records whether the event was generated by pressing the button (GLUT DOWM), or releasing it (GLUT UP).
- x, y give the current mouse position in pixels. Note: when using OpenGL with X, the mouse y position is measured from the top of the window.

### 6.3.4 HANDLING INPUT EVENTS:

You can use these routines to register callback commands that are invoked when specified events occur.

- **glutReshapeFunc**(void(\*func) (intw, inth)) indicates what action should be taken when the window is resized.
- **glutKeyboardFunc**(void(\*func)(unsignedcharkey,intx,inty))and **glutMouseFunc**(void(\*func)(intbutton,intstate,intx,ints)) allow you to link a keyboard key or a mouse button with routine that is invoked when the key or mouse button is pressed or released.
- **glutMotionFunc** (void (\*func) (intx,inty)) registers a routine to call back when the mouse is moved while a mouse button is also pressed

### 6.3.5 TRANSLATION:

void **glTranslatef**(GLfloatx,GLfloaty,GLfloatz); **glTranslatef** () creates a matrix M which performs a translation by (x,y,z) and then post-multiplies the current matrix by M.

### 6.3.6 SCALING:

- void **glScalef** (GLfloatx, GLfloaty, GLfloatz); **glScalef**() creates a matrix M which performs a scale by (x,y,z) and then post-multiplies the current matrix by M.

### 6.3.7 ROTATION:

- void **glRotatef**(GLfloatangle,GLfloatx,GLfloaty,GLfloatz); **glRotatef**() creates a matrix M which performs a counter-clockwise rotation of angle in degrees. The axis about which the rotation occurs in the vector from the origin (0, 0, 0) to the point (x,y,z),and then post-multiplies the current matrix by M.

### 6.3.8 PUSH AND POP OPERATIONS:

There are two functions which operate on the current matrix stack:

**glPushMatrix() glPopMatrix() void glPushMatrix(void);** Pushes the current matrix stack down one level. The matrix on the top of the stack is copied into the next-to-top position.

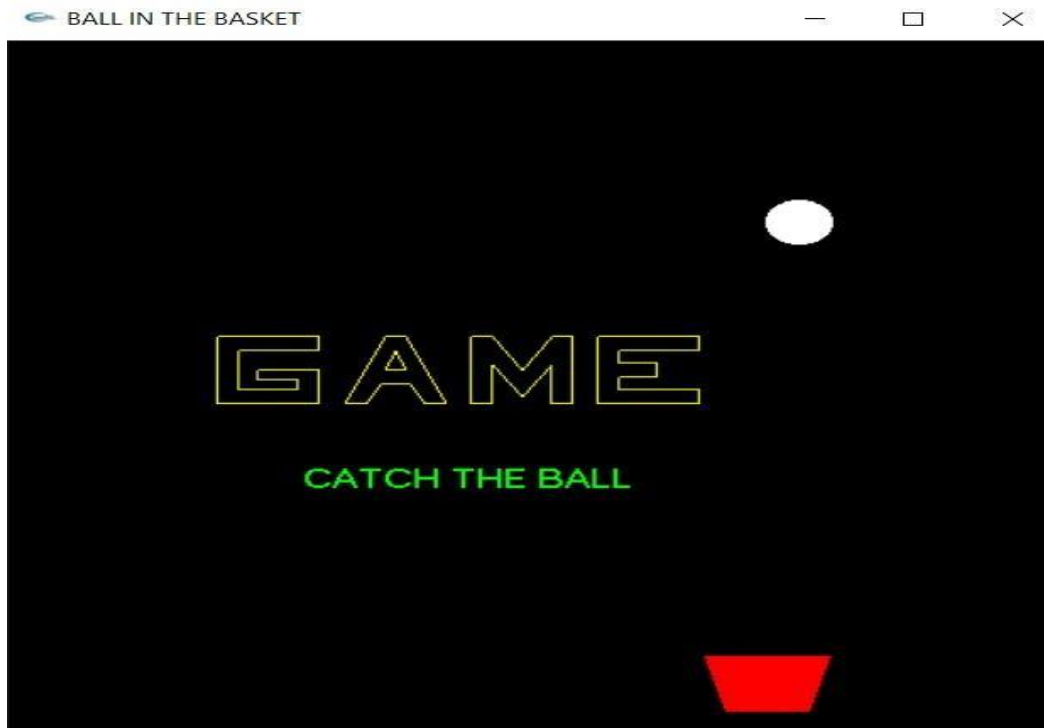
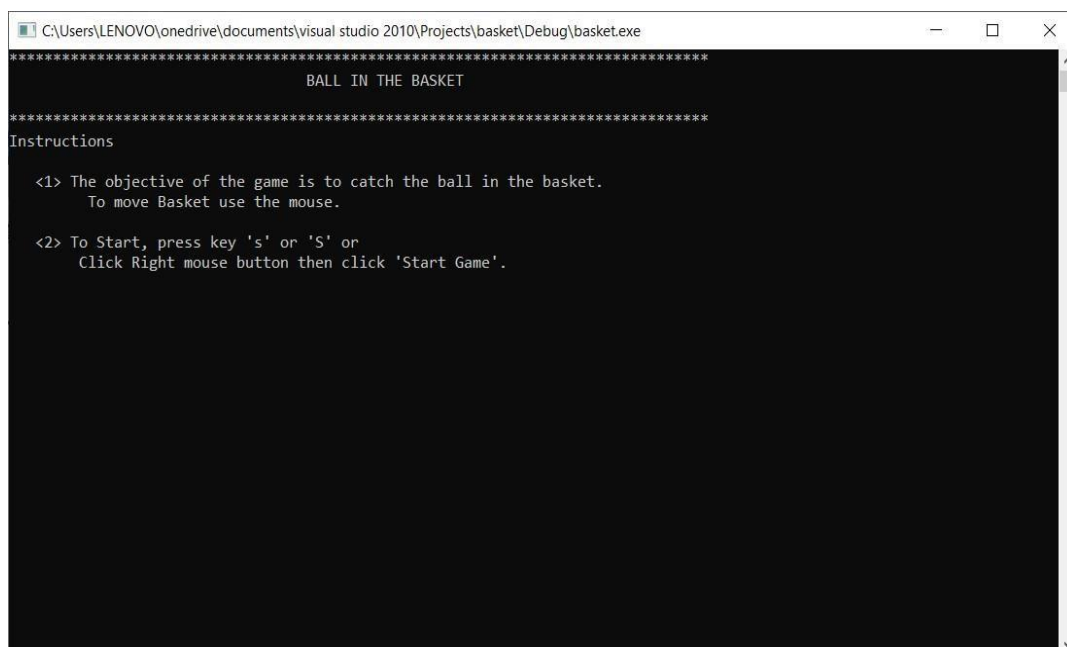
**void glPopMatrix(void);** Pops the current matrix stack, moving each matrix in the stack one position towards the top of the stack.

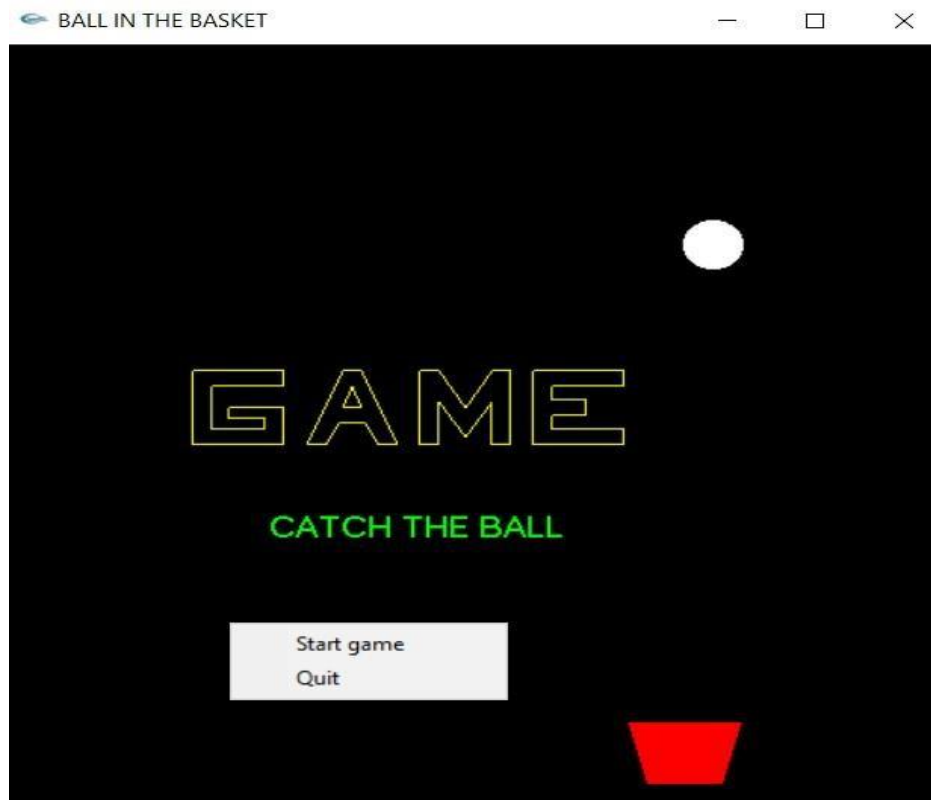
The current matrix stack is determined by the most recent call to **glMatrixMode()**.

### 6.4 RUNNING THE PROGRAM :

After all the setup is completed, GLUT programs enter an event processing loop, **glutMainLoop()**.

**void glutMainLoop(void)** enters the GLUT processing loop, never to return. Registered callback functions will be called when the corresponding events instigate them.

**CHAPTER 7****SNAPSHOTS****Fig:7.1** Initial state**Fig:7.2** Instructions To Play The Game



**Fig:7.3** Options to Start OR Quit the Game



**Fig:7.4** Level 1 of the Game

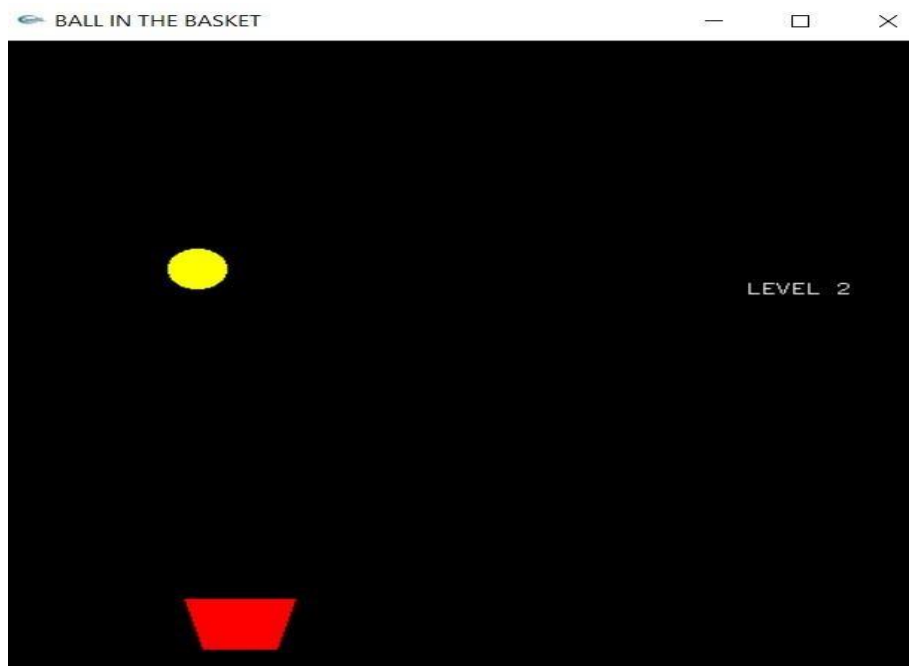


Fig:7.5 Level 2 of the Game

```
C:\Users\LENOVO\onedrive\documents\visual studio 2010\Projects\basket\Debug\basket.exe
*****
                        BALL IN THE BASKET
*****
Instructions
<1> The objective of the game is to catch the ball in the basket.
      To move Basket use the mouse.
<2> To Start, press key 's' or 'S' or
      Click Right mouse button then click 'Start Game'.

                        GAME OVER

Level reached: 3

No. of balls dropped      = 23
No. of balls caught      = 13
No. of balls missed      = 10
White balls   = 0      points gained   = 0
Red balls    = 4      points gained   = 20
Green balls   = 2      points gained   = 20
Blue balls    = 2      points gained   = 30
Yellow balls  = 3      points gained   = 300

Your score = 390
```

Fig:7.6 GAMEOVER and Displaying of the Score

## CHAPTER 8

### CONCLUSION

In conclusion, the project "Ball in the Basket" in the computer graphics aimed to create a realistic and engaging simulation of a ball being shot into a basket. By utilizing various computer graphics techniques, including rendering, physics simulation, and animation, the project successfully achieved its goal. Through the implementation of accurate physics-based motion, realistic lighting and shading, and visually appealing textures, the final result provided an immersive experience for the users. The project not only demonstrated the practical application of computer graphics concepts but also showcased the potential of technology in creating interactive and visually stunning simulations. Overall, "Ball in the Basket" served as a testament to the power of computer graphics in creating virtual environments that can engage and entertain users.

# REFERENCES

## Books :

**Edward Angel** “Interactive Computer Graphics”, 5th edition, Pearson Education, 2005

**Donald D Hearn** and **M. Pauline Baker**, “Computer Graphics with OpenGL”, 3rd Edition

## Websites:

- <http://www.opengl.org>
- <http://glprogramming.com/red>
- <http://jerome.jouvie.free.fr/OpenGL>
- <https://www.3dgep.com/introduction-to-opengl-and-glsl/>