**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“Jnana Sangama”, Belagavi-590018, Karnataka**



**BANGALORE INSTITUTE OF TECHNOLOGY**

**K. R. Road, V. V. Puram, Bengaluru-560 004**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Computer Graphics Laboratory With Mini Project Report-18CSL67**

**on**

**“EGG CATCHING GAME”**

**Submitted By**

**1BI20CS160 SHIVANI**

**for the academic year 2022-23**

Under the guidance of

**Prof. Bhanushree K J** **Prof. Mamatha** Associate Professor Assistant Professor

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“Jnana Sangama”, Belagavi-590018, Karnataka**

**BANGALORE INSTITUTE OF TECHNOLOGY**

**K. R. Road, V. V. Puram, Bengaluru-560 004**



**Department of Computer Science & Engineering**

***Certificate***

This is to certify that the implementation of **Computer Graphics Laboratory With Mini Project (18CSL67)** entitled **“EGG CATCHING GAME”** has been successfully completed by

**1BI20CS160 SHIVANI**

of VI semester B.E. for the partial fulfillment of the requirements for the Bachelor's degree in **Computer Science & Engineering** of the **Visvesvaraya Technological University** during the academic year **2023-2024**.

**Lab In charges:**

|  |  |  |
| --- | --- | --- |
| **Prof. Mamatha** | **Prof. Bhanushree K. J.** | **Dr. Girija J.** |
| Assistant Professor | Assocciate Professor | Professor and Head |
| Dept. of CSE, BIT | Dept. of CSE, BIT | Dept. of CSE, BIT |

Examiners: 1) 2)

**ACKNOWLEDGEMENT**

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

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

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

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

**SHIVANI**

**1BI20CS160**

**Table of contents**

1. Introduction………………………………………………………… ……. 1
   1. Computer Graphics…………………………………………………… 1
   2. Application of Computer Graphics…………………………………... 1
   3. OpenGL……………………………………………………………… 3
   4. Problem Statement…………………………………………………… 4
   5. Objective of The Project …….……………………………………… 4
   6. Organization of The Project..………………………………………… 5
2. System Specification …………………………………………………….. 6
   1. Hardware Requirements……………………………………………….. 6
   2. Software Requirements……………………………………………… 6
3. Design…………………………………………………………………….. 7

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

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

1. Implementation…………………………………………………………… 9
   1. Built In Functions…………………………………………………….. 9
   2. User Defined Functions With Modules………………………………. 13
   3. Psuedocode……………………………………………………………. 18
2. Snapshots………………………………………………………………….. 27
3. Conclusion………………………………………………………………… 33

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

Bibliography……………………………………………………………….. 34

**Chapter -1**

**INTRODUCTION**

**1.1** **Computer Graphics**

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

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

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

**1.2 Applications of Computer Graphics**

Scientific Visualization

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

Graphic Design

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

Computer-aided Design

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

Web Design

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

Digital Art

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

Video Games

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

Virtual Reality

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

Computer Simulation

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

Education and Training

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

Image Processing

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

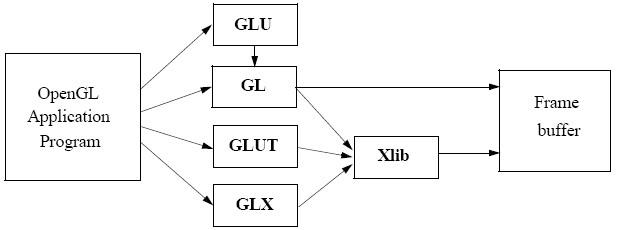
**1.3 OpenGL**

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

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

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

**Figure 1.1 Basic block diagram of OpenGL**



**1.4 Problem Statement**

“To implement a game called Egg Catching Game using OpenGL framework,graphic primitives, textures and keyboard/mouse interactions.”

**1.5 Objectives of the Project**

* To show the working of the orthographic projections in appearance of the objects used in the scene.
* To show the implementation of Textures for a better appearance of the real-world objects.
* To show the implementation of the OpenGL transformation functions.
* To show the user and programme interaction using input devices.

**1.**6 **Organisation of the Project**

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

**Chapter -2**

**SYSTEM SPECIFICATION**

**2.1 Hardware Requirements**

* Main Processor : PENTIUM III
* Processor Speed: 800 MHz
* RAM Size : 128 MB DDR
* Keyboard : Standard qwerty serial or PS/2 keyboard
* Mouse : Standard serial or PS/2 mouse
* Compatibility : AT/T Compatible
* Cache memory : 256 KB
* Diskette drive : 1,44MB,3.5 inches

**2.2 Software Requirements**

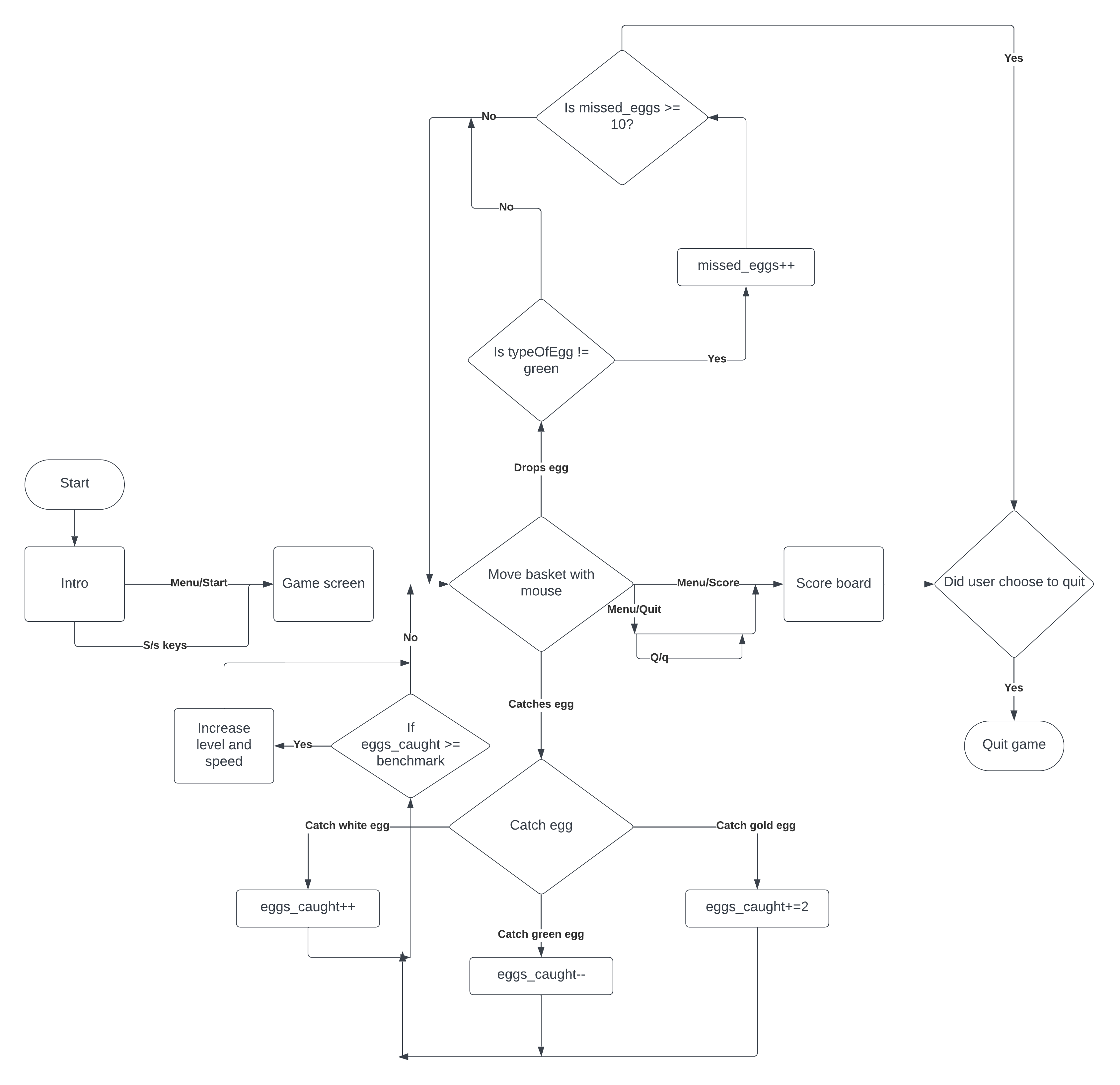
* Operating System: Windows 10 or Linux (Fedora) or macOS
* Hypervisor used : Docker
* Compiler used : gcc
* Language used : C++ language
* Editor : Visual Studio Code / Neovim
* Toolkit : GLUT Toolkit

**Chapter -3**

**DESIGN**

**3.1 Flow Diagram**

**Figure 3.1 : Flow diagram of scene change**



**3.2 Description of Flow Diagram**

The description of the flow diagram is as follows:

**Step 1:** Start

**Step 2:** The user is presented with intro and presses ‘S/s’ or uses the “Menu->Start Game” to proceed to next scene.

**Step 3:** Game starts. User is provided with basket that can be moved with mouse.

**Step 4:** User can either catch or miss eggs. Eggs can be of 3 types, white, gold and green.

**Step 5:** If user catches white, it adds 1 to the score. If user catches gold, it adds 2 to the score. If user catches green, it subtracts 1 from the score. If eggs\_caught crosses the benchmark set for each level, current\_level and speed is increased.

**Step 6:** If user however misses the egg, it is added to another counter as long as it is not a green egg. If user has missed 10 eggs in total, they will be disqualified and score board is automatically displayed.

**Step 7:** If at any point user wishes to see scoreboard, they can use the “Menu->Scoreboard” to check their stats. The user can return to game via “Menu->Start game”

**Step 8:** The user can also quit the game with q/Q or “Menu->Quit”. It displays the scoreboard and terminates the program.

**Step 9:** Stop

**Chapter -4**

**IMPLEMENTATION**

**4.1 Built in Functions**

**1. glutInit()**

glutInit is used to initialize the GLUT library.  
Usage: void glutInit (int \*argc, char \*\*argv);  
Description: glutInit will initialize the GLUT library and negotiate a session with the window system.

**2. glutInitDisplayMode()**

glutInitDisplayMode sets the initial display mode.  
Usage: void glutInitDisplayMode (unsigned int mode);  
Mode-Display mode, normally the bitwise OR-ing GLUT display mode bit masks. Description: The initial display mode is used when creating top-level windows, sub-windows, and overlays to determine the OpenGL display mode for the to-be created window or overlay.

**3. glutCreateWindow()**

glutCreateWindow creates a top-level window.

Usage: intglutCreateWindow (char \*name); Name-ASCII character string for use as window name

Description: glutCreateWindow creates a top-level window. The name will be provided to the window system as the window’s name. The intent is that the window system will label the window with the name. Implicitly, the current window is set to the newly created window.

Each created window has a unique associated OpenGL context.

**4. glutDisplayFunc()**

glutDisplayFunc sets the display callback for the current window.  
Usage: void glutDisplayFunc (void(\*func)(void));  
Func: The new display callback function.  
Description: glutDisplayFunc sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the current window is set to the window needing to be redisplayed and the layer in use is set to the normal plane. The display callback is called with no parameters. The entire normal plane region should be redisplayed in response to the callback.

**5. glutMainLoop()**

glutMainLoop enters the GLUT event processing loop.

Usage: void glutMainLoop(void);

Description: glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

**6. glMatrixMode()**

The two most important matrices are the model-view and projection matrix. At many times, the state includes values for both of these matrices, which are initially set to identity matrices. There is only a single set of functions that can be applied to any type of matrix. Select the matrix to which the operations apply by first set in the matrix mode, a variable that is set to one type of matrix and is also part of the state.

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

glTranslate produces a translation by x y z. If the matrix mode is either GL\_MODEL\_VIEW or GL\_PROJECTION, all objects drawn after a call to glTranslate are translated.

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

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

**9. glPushMatrix()**

There is a stack of matrices for each of the matrix mode. In GL\_MODELVIEW mode, the stack depth is atleast 32. In other modes, GL\_COLOR, GL\_PROJECTION, and GL\_TEXTURE, the depth is atleast 2. The current matrix in any mode is the matrix on the top of the stack for that mode.

**10. glPopMatrix()**

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

**11. glutSwapBuffers()**

**Usage:** void glutSwapBuffers(void);

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

**12. glPointSize(GLfloat size)**

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

**13. glutKeyboardFunc()**

**Usage:** void glutKeyboardFunc(void(\*func)(unsigned char key, int x, int y)

**Func:** The new keyboard callback function

**Description:** glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character.

**14. glLineWidth(GLfloat width)**

**Parameters:** width- Specifies the width of rasterized lines. The initial value is 1. **Description:** glLineWidth specifies the rasterized width of lines. The actual width is determined by rounding the supplied width to the nearest integer. i pixels are filled in each column that is rasterized, where I is the rounded value of width.

**15. glLoadIdentity(void)**

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

**4.2 User Defined Functions**

**1. void myinit()**

Used to set Matrix Mode and clipping coordinates and set background color.

**2. void display(int)**

Used to make the apple disappear from the crocodile’s mouth

**3. void start\_screen()**

Display function to display start scene.

**4. void displayScoreboard()**

Display function to display scoreboard.

**5. void drawLine(int\*, int\*)**

Used to display line on the scene.

**6. void keys(unsigned char,int,int)**

Function to handle keyboard events

**7. void LoadIntro()**

Function used to load texture assets for intro scene.

**8. void displayIntro(int, int, int)**

Function used to bind textures for intro..

**9. void menu()**

Function to display menu and process menu events.

**10. void mouse(int,int,int,int)**

Function to handle mouse events.

**11. void cloud1()**

Function to display cloud 1 in the scene.

**12. void cloud2()**

Function to display cloud 2 in the scene.

**13. void egg()**

Function to display egg in the scene with dynamic co-ordinates (as it is falling).

**14. void egg\_start()**

Function to reset egg in the scene with fixed co-ordinates, i.e, fixed height and one of the chickens chosen at random.

**15. void basket\_set()**

Function to display basket in the scene with dynamic co-ordinates, i.e, changes with the mouse. Collision code for the egg and basket is also defined here.

**15. void myReshape()**

Reshape function to handle changing display.

**4.3 PSUEDOCODE**

**main.cpp**

#include <GL/freeglut\_std.h>

#include <GL/gl.h>

#include <GL/glut.h>

#include <cstdlib>

#include <iostream>

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <string>

#define STB\_IMAGE\_IMPLEMENTATION

#include "headers/stb\_image.h"

#define CIRCLE\_RADIUS 0.15f

#define PI 3.14159265358979323846

unsigned int intro;

int eggs\_caught = 0, missed\_eggs = 0, level\_count = 1, points = 0;

// typeOfEgg denotes the type of egg, if 1 it is white egg (normal), else it

// is gold (special)

int typeOfEgg;

int egg\_xc, egg\_yc;

// for coordinates of egg

int basket\_x, basket\_y;

// for coordinates of basket

int a = 600, b = 650; // for default size of the screen

int s = 0;

// for menu option

int dropped\_eggs = 0;

double speed\_1 = 1, speed\_2 = 1.5, speed\_3 = 2, speed\_4 = 2.5;

int w = 48, h = 48, t = 10, e = 9, g = 12;

void myinit();

void start\_screen(int, int);

void displayIntro();

void cloud1();

void egg();

void drawLine(int \*, int \*);

void basket(int, int);

void duck(int, int);

void print\_score();

void egg\_start();

void color();

void score();

void display(void);

void basket\_set(int, int);

void myReshape(int, int);

void keys(unsigned char, int, int);

void menu(int);

void myinit()

{

glViewport(0, 0, a, b);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0, (GLdouble)a, 0, (GLdouble)b);

glMatrixMode(GL\_MODELVIEW);

glutPostRedisplay();

}

void sun()

{

float theta;

GLfloat angle;

glLineWidth(1.5);

glColor3f(1, 0.2, 0);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(500 + 80 \* cos(theta) / 2, 600 + 80 \* sin(theta) / 2);

}

glEnd();

glFlush();

}

void cloud1()

{

float theta;

GLfloat angle;

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(100 + 50 \* cos(theta) / 2, 590 + 50 \* sin(theta) / 2);

}

glEnd();

// GLfloat angle;

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(130 + 50 \* cos(theta) / 2, 580 + 50 \* sin(theta) / 2);

}

glEnd();

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(140 + 50 \* cos(theta) / 2, 600 + 50 \* sin(theta) / 2);

}

glEnd();

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(170 + 50 \* cos(theta) / 2, 590 + 50 \* sin(theta) / 2);

}

glEnd();

glFlush();

}

void cloud2()

{

float theta;

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(420 + 45 \* cos(theta) / 2, 540 + 45 \* sin(theta) / 2);

}

glEnd();

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(455 + 45 \* cos(theta) / 2, 550 + 45 \* sin(theta) / 2);

}

glEnd();

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(455 + 45 \* cos(theta) / 2, 530 + 45 \* sin(theta) / 2);

}

glEnd();

glLineWidth(1.5);

glColor3f(1, 1, 1);

glBegin(GL\_POLYGON);

for (int i = 0; i < 360; i++)

{

theta = i \* PI \* i / 180;

glVertex2f(490 + 45 \* cos(theta) / 2, 540 + 45 \* sin(theta) / 2);

}

glEnd();

glFlush();

}

void line(int i, int j)

{

glBegin(GL\_QUADS);

glColor3f(1.0, .5, 0.5);

glVertex2f(0.0 + i, 10.0 + j);

glVertex2f(0.0 + i, 15.0 + j);

glVertex2f(600.0 + i, 15.0 + j);

glVertex2f(600.0 + i, 10.0 + j);

glEnd();

glFlush();

}

void backk(int i, int j)

{

glColor3f(0, .5, 1);

glBegin(GL\_QUADS);

glVertex2f(0.0 + i, 0.0 + j);

glVertex2f(600.0 + i, 0.0 + j);

glVertex2f(600.0 + i, -500 + j);

glVertex2f(0.0 + i, -500 + j);

glEnd();

glFlush();

}

void ground(int i, int j)

{

glBegin(GL\_QUADS);

glColor3f(0, 1.0, 0);

glVertex2f(0.0 + i, 0.0 + j);

glVertex2f(600.0 + i, 0.0 + j);

glVertex2f(600.0 + i, -j);

glVertex2f(0.0 + i, -j);

glEnd();

}

void start\_screen()

{

// Set background color

glClearColor(0.0, 0.0, 0.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

// Draw background

glBegin(GL\_QUADS);

glColor3f(0.0, 0.5, 1.0);

glVertex2f(-800.0, 900.0);

glVertex2f(-800.0, -800.0);

glColor3f(0.0, 0.2, 0.5);

glVertex2f(800.0, -800.0);

glVertex2f(800.0, 800.0);

glEnd();

// Draw stars

glPointSize(4.0);

glBegin(GL\_POINTS);

glColor3f(1.0, 1.0, 1.0);

for (int i = 0; i <= 10; i++)

{

int x = rand() % 600;

int y = rand() % 600;

glVertex2f(x, y);

}

glEnd();

int k;

char cat[4] = "EGG";

char orr[9] = "Catching";

char hatch[5] = "Game";

char start[20] = "Press S for start";

char authors[8] = "Made by";

char author1[16] = "Vaitheeswaran J";

char author2[8] = "Shivani";

glColor3f(0, 1, 0);

glRasterPos2i(150, 320);

for (k = 0; k < 4; k++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, cat[k]);

glColor3f(1, 1, 1);

glRasterPos2i(200, 320);

for (k = 0; k < 9; k++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, orr[k]);

glColor3f(1, 0, 0);

glRasterPos2i(300, 320);

for (k = 0; k < 5; k++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, hatch[k]);

glColor3f(1, 1, 0);

glRasterPos2i(210, 200);

for (k = 0; k < 20; k++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, start[k]);

glRasterPos2i(400, 50);

for (k = 0; k < 8; k++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, authors[k]);

glRasterPos2i(400, 30);

for (k = 0; k < 16; k++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, author1[k]);

glRasterPos2i(400, 10);

for (k = 0; k < 8; k++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, author2[k]);

glColor3f(1, .5, .5);

glFlush();

glutSwapBuffers();

}

void loadIntro(void)

{

glGenTextures(1, &intro);

glBindTexture(GL\_TEXTURE\_2D, intro);

// set the bg1 wrapping/filtering options (on the currently bound bg1 object)

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

// load and generate the bg1

int width, height, nrChannels;

unsigned char \*data = stbi\_load("introBanner.psd", &width, &height,

&nrChannels, STBI\_rgb\_alpha);

if (data)

{

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGBA, width, height, 0, GL\_RGBA,

GL\_UNSIGNED\_BYTE, data);

// glGenerateMipmap(GL\_TEXTURE\_2D);

}

else

{

std::cout << "Failed to load bg1" << std::endl;

}

stbi\_image\_free(data);

}

void displayIntro()

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glEnable(GL\_TEXTURE\_2D);

glColor3f(1, 1, 1);

glBindTexture(GL\_TEXTURE\_2D, intro);

glBegin(GL\_QUADS);

glVertex3f(0, 0, 10);

glTexCoord2f(0, 0);

glVertex3f(0, 5000, 10);

glTexCoord2f(0, 1);

glVertex3f(5000, 5000, 10);

glTexCoord2f(1, 1);

glVertex3f(5000, 0, 10);

glTexCoord2f(1, 0);

glEnd();

glFlush();

glDisable(GL\_TEXTURE\_2D);

glutSwapBuffers();

}

void egg()

{

float x, y, z;

int t;

if (typeOfEgg == 1)

glColor3f(1.0, 1.0, 1.0);

else if (typeOfEgg == 2)

glColor3f(0.05, 0.33, 0.04);

else

glColor3f(1.0, 0.843, 0);

glBegin(GL\_POLYGON);

for (t = 0; t <= 360; t += 1)

{

x = egg\_xc + 8 \* (cos(t));

y = egg\_yc + 16 \* (sin(t));

z = 0;

glVertex3f(x, y, z);

}

glEnd();

}

void basket(int i, int j)

{

j = 10;

if (i >= a - 60)

i = a - 60;

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_QUADS);

glVertex2f(0.0 + i, 30.0 + j);

glVertex2f(10.0 + i, 10.0 + j);

glVertex2f(50.0 + i, 10.0 + j);

glVertex2f(60.0 + i, 30.0 + j);

glEnd();

}

void duck(int i, int j)

{

int h;

glColor3f(1.0, 1.0, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(45 + i, 45 + j);

glVertex2f(70 + i, 20 + j);

glVertex2f(95 + i, 20 + j);

glVertex2f(120 + i, 45 + j);

glVertex2f(95 + i, 70 + j);

glVertex2f(70 + i, 70 + j);

glVertex2f(95 + i, 95 + j);

glVertex2f(82.5 + i, 107.5 + j);

glVertex2f(32.5 + i, 57.5 + j);

glEnd();

glFlush();

for (h = 0; h < 13; h += 4)

{

glBegin(GL\_LINES);

glColor3f(0.7, 0.4, 0);

glVertex2f(57.5 + h + i, 52.5 + h + j);

glVertex2f(100 + h + i, 30 + h + j);

glEnd();

glFlush();

}

glColor3f(0.0, 1.0, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(82.5 + i, 107.5 + j);

glVertex2f(65 + i, 107.5 + j);

glVertex2f(50 + i, 95 + j);

glVertex2f(70 + i, 95 + j);

glEnd();

glFlush();

glColor3f(0.0, 0.0, 0.0);

glPointSize(5);

glBegin(GL\_POINTS);

glVertex2f(76 + i, 101 + j);

glEnd();

glColor3f(0.0, 1.0, 0.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(72.5 + i, 107.5 + j);

glVertex2f(67.5 + i, 112.5 + j);

glVertex2f(72.5 + i, 110 + j);

glVertex2f(77.5 + i, 112.5 + j);

glEnd();

glFlush();

}

void displayScore()

{

std::string::iterator k;

// Set background color

glClearColor(0.0, 0.0, 0.0, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

// Draw background

glBegin(GL\_QUADS);

glColor3f(0.0, 0.5, 1.0);

glVertex2f(-800.0, 900.0);

glVertex2f(-800.0, -800.0);

glColor3f(0.0, 0.2, 0.5);

glVertex2f(800.0, -800.0);

glVertex2f(800.0, 800.0);

glEnd();

// Draw stars

glPointSize(2.0);

glBegin(GL\_POINTS);

glVertex2f(300, 300);

glColor3f(1.0, 1.0, 1.0);

for (int i = 0; i <= 10; i++)

{

int x = rand() % 600;

int y = rand() % 600;

glVertex2f(x, y);

}

glEnd();

char cat[11] = "Scoreboard";

char eggTotal[11] = "Total eggs";

char eggCaught[12] = "Eggs caught";

char eggMissed[12] = "Eggs missed";

std::string eggTotalValue = std::to\_string(dropped\_eggs);

std::string eggCaughtValue = std::to\_string(eggs\_caught);

std::string eggMissedValue = std::to\_string(missed\_eggs);

int kc;

glColor3f(0, 1, 0);

glRasterPos2i(260, 540);

for (kc = 0; kc < 11; kc++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, cat[kc]);

glColor3f(1, 1, 1);

glRasterPos2i(180, 450);

for (kc = 0; kc < 11; kc++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, eggTotal[kc]);

glRasterPos2i(180, 390);

for (kc = 0; kc < 11; kc++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, eggCaught[kc]);

glColor3f(1, 0, 0);

glRasterPos2i(180, 330);

for (kc = 0; kc < 12; kc++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, eggMissed[kc]);

glColor3f(1, 1, 1);

glRasterPos2i(320, 450);

for (k = eggTotalValue.begin(); k != eggTotalValue.end(); k++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, \*k);

glRasterPos2i(320, 390);

for (k = eggCaughtValue.begin(); k != eggCaughtValue.end(); k++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, \*k);

glColor3f(1, 0, 0);

glRasterPos2i(320, 330);

for (k = eggMissedValue.begin(); k != eggMissedValue.end(); k++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, \*k);

glColor3f(1, 1, 1);

int p1[2] = {300, 300};

int p2[2] = {300, 500};

drawLine(p1, p2);

// int p3[2] = {180, 320};

// int p4[2] = {320, 320};

// drawLine(p3, p4);

// int p5[2] = {180, 350};

// int p6[2] = {320, 350};

// drawLine(p5, p6);

glFlush();

glutSwapBuffers();

}

void drawLine(int \*p1, int \*p2)

{

glLineWidth(2);

glBegin(GL\_LINES);

glVertex2iv(p1);

glVertex2iv(p2);

glEnd();

}

void print\_score()

{

printf("\nLevel reached: %d\n\n", level\_count);

printf("\nNo. of eggs dropped= %d \n", dropped\_eggs);

printf("\nNo. of eggs caught = %d\n", eggs\_caught);

printf("\nNo. of eggs missed = %d\n", missed\_eggs);

getchar();

exit(0);

}

void egg\_start()

{

egg\_yc = 375;

if (missed\_eggs >= 10)

{

printf("\n\n\t\t\t\tGAME OVER\n\n");

print\_score();

}

int temp = rand() % 8;

if (temp == 0)

typeOfEgg = 0;

else if (temp == 7 || temp == 3)

typeOfEgg = 2;

else

typeOfEgg = 1;

if (typeOfEgg != 2)

dropped\_eggs++;

switch (rand() % 9)

{

case 0:

egg\_xc = 115;

break;

case 1:

egg\_xc = 255;

break;

case 2:

egg\_xc = 390;

break;

case 5:

egg\_xc = 115;

break;

case 3:

egg\_xc = 255;

break;

case 4:

egg\_xc = 390;

break;

case 7:

egg\_xc = 115;

break;

case 6:

egg\_xc = 255;

break;

case 8:

egg\_xc = 390;

break;

}

}

void score()

{

if (egg\_yc <= 50 && (egg\_xc >= basket\_x && egg\_xc <= basket\_x + 60))

{

printf("\a");

if (typeOfEgg == 0)

eggs\_caught += 3;

else if (typeOfEgg == 2)

eggs\_caught--;

else

eggs\_caught++;

egg\_yc = -10;

}

missed\_eggs = dropped\_eggs - eggs\_caught;

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

ground(0, 650);

backk(0, 650);

duck(40, 375);

duck(180, 375);

duck(320, 375);

sun();

cloud1();

cloud2();

line(0, 375);

int i;

char level1[12] = "LEVEL 1";

char level2[12] = "LEVEL 2";

char level3[12] = "LEVEL 3";

char level4[12] = "LEVEL 4";

if (s >= 1)

{

if (level\_count == 1)

{

glColor3f(1, 1, 1);

glRasterPos2i(500, 300);

for (i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, level1[i]);

}

else if (level\_count == 2)

{

glColor3f(1, 0, 0);

glRasterPos2i(500, 300);

for (i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, level2[i]);

}

else if (level\_count == 3)

{

glColor3f(0, 1, 1);

glRasterPos2i(500, 300);

for (i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, level3[i]);

}

if (level\_count == 4)

{

glColor3f(1, 0, 1);

glRasterPos2i(500, 300);

for (i = 0; i < 12; i++)

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, level4[i]);

}

if (egg\_yc <= 10)

egg\_start();

egg();

basket(basket\_x, basket\_y);

if (eggs\_caught >= 10)

{

egg\_yc -= speed\_2;

level\_count = 2;

}

if (eggs\_caught >= 20)

{

egg\_yc -= speed\_3;

level\_count = 3;

}

if (eggs\_caught >= 30)

{

egg\_yc -= speed\_4;

level\_count = 4;

}

else

egg\_yc -= speed\_1;

score();

}

glFlush();

glutSwapBuffers();

}

void basket\_set(int a, int b)

{

basket\_x = a;

basket\_y = b;

glutPostRedisplay();

}

void myReshape(int w, int h)

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glViewport(0, 0, w, h);

a = w;

b = h;

}

void keys(unsigned char key, int x, int y)

{

if (key == 'q' || key == 'Q')

{

glutDisplayFunc(displayScore);

glutIdleFunc(displayScore);

glutPostRedisplay();

// printf("\n\n\n\t\tQUIT BY PLAYER\n\n");

// print\_score();

}

if (key == 's' || key == 'S')

{

s += 1;

glutDisplayFunc(display);

glutIdleFunc(display);

glutPostRedisplay();

}

if (key == 'a' || key == 'A')

{

egg\_xc -= 10;

if (egg\_xc <= 0)

egg\_xc = 10;

}

if (key == 'd' || key == 'D')

{

egg\_xc += 10;

if (egg\_xc >= 500)

egg\_xc = 490;

}

}

void menu(int id)

{

switch (id)

{

case 1:

glutDisplayFunc(display);

glutIdleFunc(display);

glutPostRedisplay();

s += 1;

break;

case 2:

glutDisplayFunc(displayScore);

glutIdleFunc(displayScore);

glutPostRedisplay();

break;

case 3:

printf("\n\n\n\t\tQUIT BY PLAYER\n");

exit(0);

break;

default:

exit(0);

}

glutPostRedisplay();

}

int main(int argc, char \*\*argv)

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\t\t\t\t EGG GAME\n\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\tHow to Play..?\n\n <1>The objective of the game is to catch the "

"eggs in the basket.\n\t To move Basket use mouse.\n");

printf("\n <2> To Start, press key 's' or 'S' or \n\tClick Right mouse "

"button then click 'Start Game'.\n");

printf("\n <3> To Quit manually, press key 'q' or 'Q' or\n\t Click Right "

"mouse button then click 'Quit'.\n");

printf("\n\n RULES : If the player misses 10 eggs,then 'Game Over'.\n");

printf("\nFor each level, speed is Increased\n\n");

printf("\n\nBEST OF LUCK\n");

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(a, b);

glutCreateWindow("EGG GAME");

myinit();

glutInitWindowPosition(100, 100);

glutCreateMenu(menu);

glutAddMenuEntry("Start game", 1);

glutAddMenuEntry("View score", 2);

glutAddMenuEntry("Quit without score", 3);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutDisplayFunc(start\_screen);

// loadIntro();

// glutDisplayFunc(display);

glutKeyboardFunc(keys);

glutPassiveMotionFunc(basket\_set);

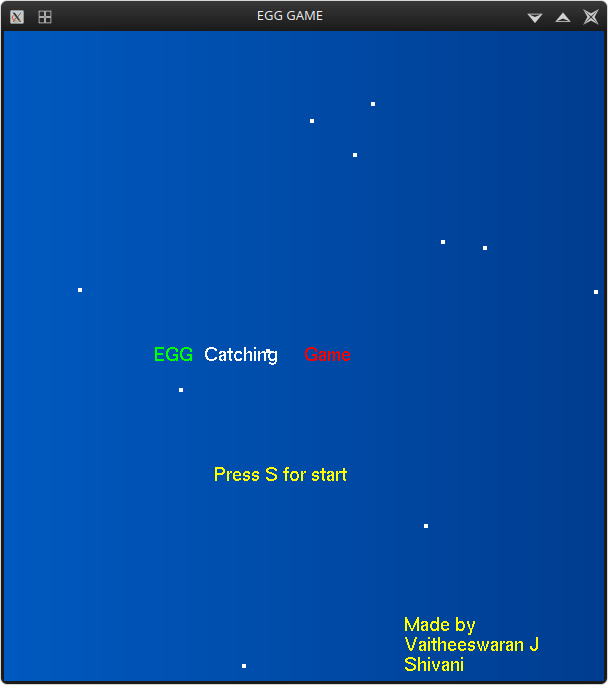
glutReshapeFunc(myReshape);

glutMainLoop();

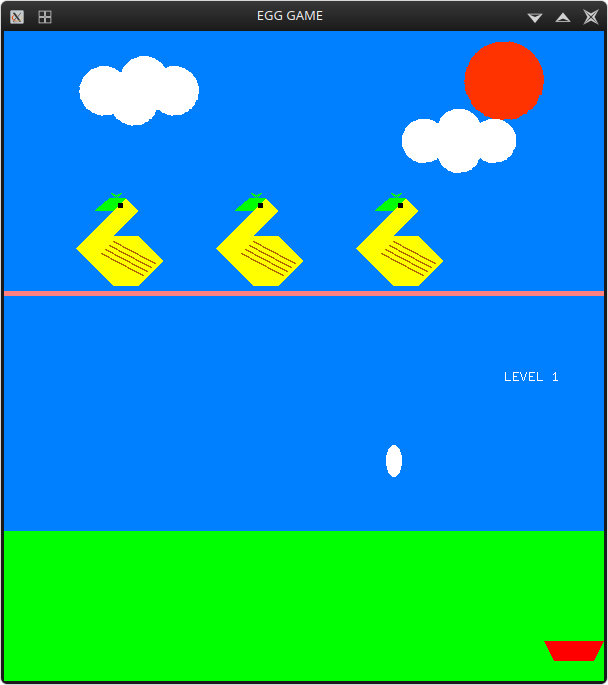
}

**Chapter -5**

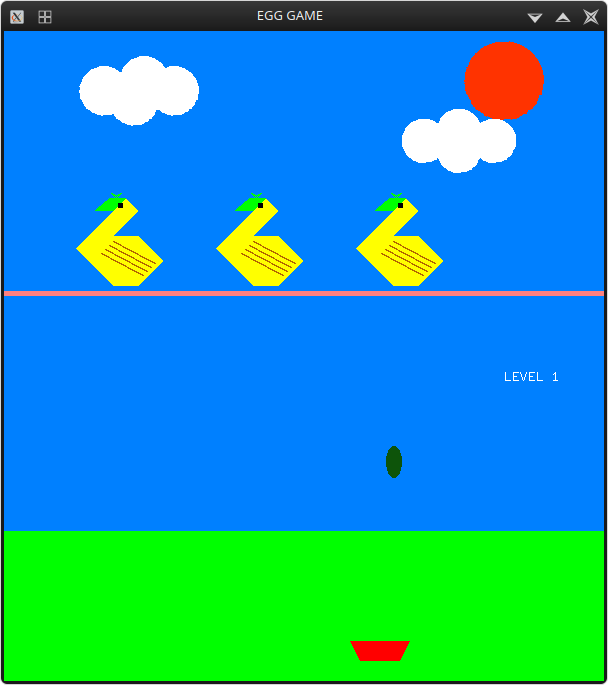
**SNAPSHOTS**



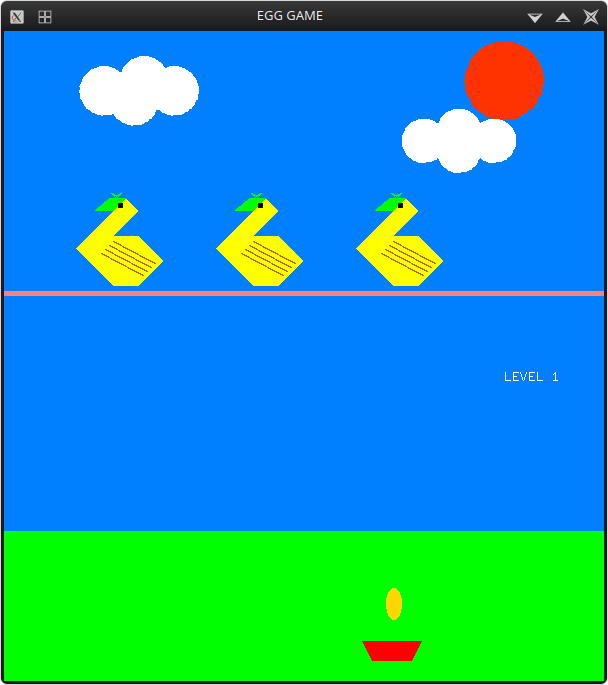
**Fig5.1 Start screen**



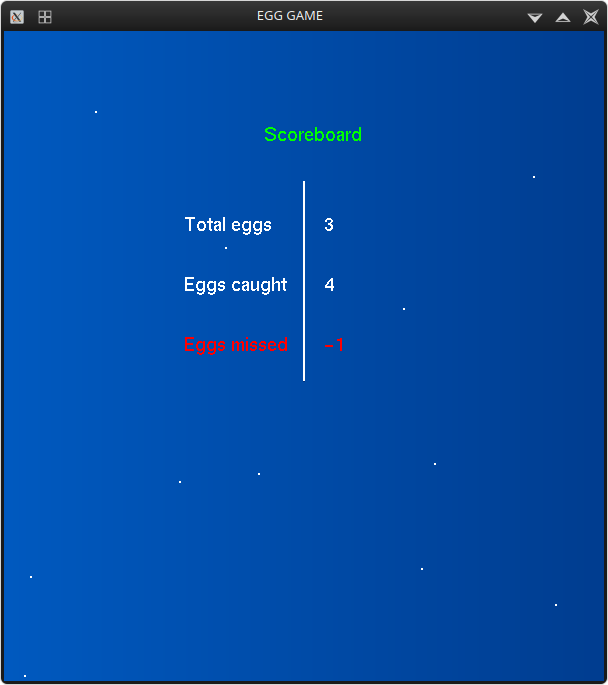
**Fig5.2 Game screen (white egg)**



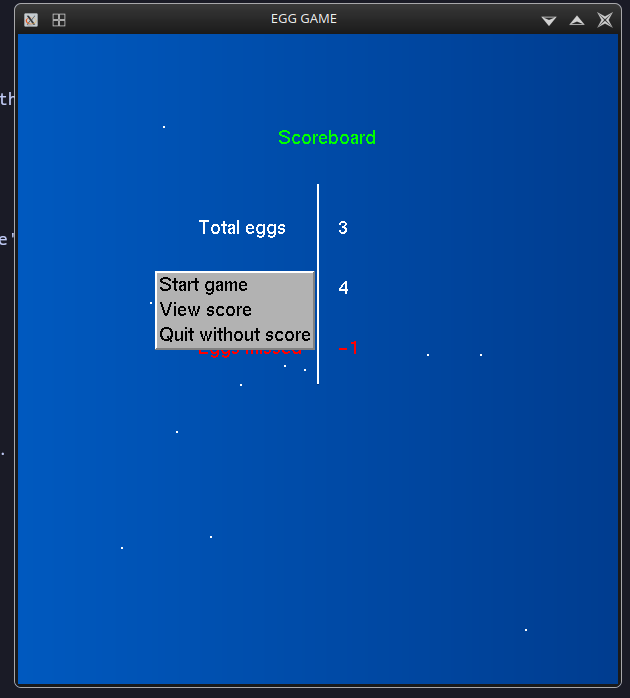
**Fig5.3 Game screen (green egg)**



**Fig5.4 Game screen (gold egg)**



**Fig5.5 Scoreboard screen**



**Fig5.6 Menu options**

**Chapter -6**

**CONCLUSION**

“Egg catching game” project was done in a perspective of understanding the openGL software toolkit. Through this project , we have acquired a much deeper knowledge of the openGL constraints and computer graphics as a whole.

We thus would like to emphasize the importance of this project to many other perspectives of Technical, mathematical, graphical and software concepts which we were unaware of.

**6.1 Future Enhancements**

* In future the same project can be enhanced in such a way that we can interact more with the project. Also the project can be implemented in 3D Space.
* A vast amount of future work can be possible by following investigations and strategies.
* More features can be included and can be modified in a more versatile way.
* Writing shader code for GPUs to render pictures

**BIBLIOGRAPHY**

**Reference Books**

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

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

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

[4] Macro Cantu: Mastering Delphi

**Websites**

[1] https://www.opengl.org/  
[2] https://learnopengl.com/