**A**

**PROJECT REPORT ON**

**Ping Pong Game**

**Submitted by,**

|  |  |
| --- | --- |
| **Mehul Dharmaraj Patil** | **T184099** |
| **Niranjan Vinod Patil** | **T184104** |
|  |  |

**Guidedby,**

**Ms.Minakshi Vharkate**

**A Report submitted to MIT Academy of Engineering Alandi submitted in partial fulfillment of the requirement for Fifth Semester of BACHELOR OF TECHNOLOGY in Department of Computer Engineering**



**DEPARTMENT OF COMPUTER ENGINEERING**

**MIT Academy of Engineering**

**Dehu Phata, Alandi (D)**

**Pune - 412105, Maharashtra (India)**

**2017-18**

|  |  |
| --- | --- |
| **Alandi (D), Pune – 412105** | Department of Computer Engineering.  **(Accredited by NBA, ISO 9001:2008 Certified)** |

**CERTIFICATE**

This is certify that,

|  |  |
| --- | --- |
| **Mehul Dharmaraj Patil** | **(S174081)** |
| **Niranjan Vinod Patil** | **(S174081)** |

of T. Y. B. Tech. have submitted a Report on,

**Ping Pong Game**

The said work is completed by putting the requirement of hours as per prescribed curriculum during the academic year 2017 – 18. The report is submitted in the partial fulfillment of the requirements for the course **Computer Graphics and Gaming Lab** in the Fifth Semester of Degree of Engineering in Computer Engineering Department of MIT Academy of Engineering.

Date: 18 / 12 /2018.

Place: Alandi-Pune.

Signature: Signature:

Name : Name :

**Internal Examiner External Examiner**

# CONTENTS

|  |  |  |  |
| --- | --- | --- | --- |
|  | Introduction | | 4 |
|  | 1.1 | Aim | 4 |
|  | 1.2 | Objective | 4 |
|  | 1.3 | Introduction Of Project | 5 |
|  | 1.4 | Introduction to OpenGL | 5 |
|  | Identification Of Graphics Objects | | 6 |
|  | 2.1 | Graphics object Required for Project | 6 |
|  | 2.2 | Primitives and Function used for Project | 6 |
|  | Requirement and Specification | | 7 |
|  | Software Design | | 8 |
|  | Implementation | | 9 |
|  | Snapshot | | 16 |

**Chapter 1: Introduction**

**1.1Aim:**

Develop a program to build a ping pong ball game using openGL and c++.

**1.2Objective:**

Develop simple single player game where player have to avoid the ball  
getting hit the down wall. If player unable to avoid collision of ball  
with the down wall using bottom bar in his side he lose one point.

**1.3. Introduction of project:**

When player start ping pong ball game ball is move and collide in areabetween top bar and bottom bar. when ball is collide with top bar and  
move towards down wall in any direction then player change or control  
position of bottom bar which is located at down wall by left key and  
right key to avoid collision of ball with the down wall   so player  
not lose one point. After every avoid collision of ball with the down  
wall ball speed is slightly increased. If Player loses three point or  
ball then game is over. For quit game we use esc key.

**1.4. Introduction to OpenGL**

In recent years OpenGL has become a worldwide standard for 3D computer graphics programming.It’s very widely used: in industry, in research laboratories, in computer games – and for teachingcomputer graphics.

OpenGL itself isn’t a programming language, or a software library. It’s the specification of an Application Programming Interface (API) for computer graphics programming. In other words, OpenGL defines a set of functions for doing computer graphics.

A key feature of the design of OpenGL is the separation of interaction (input and windowing functions) from rendering. OpenGL itself is concerned only with graphics rendering. You can always identify an OpenGL function: all OpenGL function names start with “gl”.Over time, two utility libraries have been developed which greatly extend the low-level (but very efficient) functionality of OpenGL. The first is the “OpenGL Utility Library”, or GLU. The second is the “OpenGL Utility Toolkit”, or GLUT:

1) GLU provides functions for drawing more complex primitives than those of OpenGL, such as curves and surfaces, and also functions to help specify 3D views of scenes. All GLU function names start with “glu”.

2) GLUT provides the facilities for interaction that OpenGL lacks. It provides functions for managing windows on the display screen, and handling input events from the mouse and keyboard.

It provides some rudimentary tools for creating Graphical User Interfaces (GUIs). It also includes functions for conveniently drawing 3D objects like the platonic solids, and a teapot. All GLUT function names start with “glut”

**Chapter 2: Identification of different graphics object**

**2.1 Graphics object required for project**

1. Top Bar (Rectangle)  
2. Bottom Bar (Rectangle)  
3. Ball (Circle)

**2.2 Primitives and functions used for projects.**

1. Top Bar :  Shape of top bar is rectangle. For drawing rectangle we draw 4

vertices using GL\_QUADS. For color effect we use glcolor command.

2. Bottom bar : Shape of bottom bar is also rectangle.For drawing rectangle we

 draw  4 vertices using GL\_QUADS. For color effect we use glcolor

 command.

3. Ball : Shape of ball is Circle. For drawing circle we use glutSolidSphere

command

**Chapter 3:**

**Requirement and specifications**

**3.1. Hardware specification**

Processor: Intel core i5

RAM: 4GB

H/D: 500 GB

Display

**3.2. Software specifications**

Operating system:Linux OS

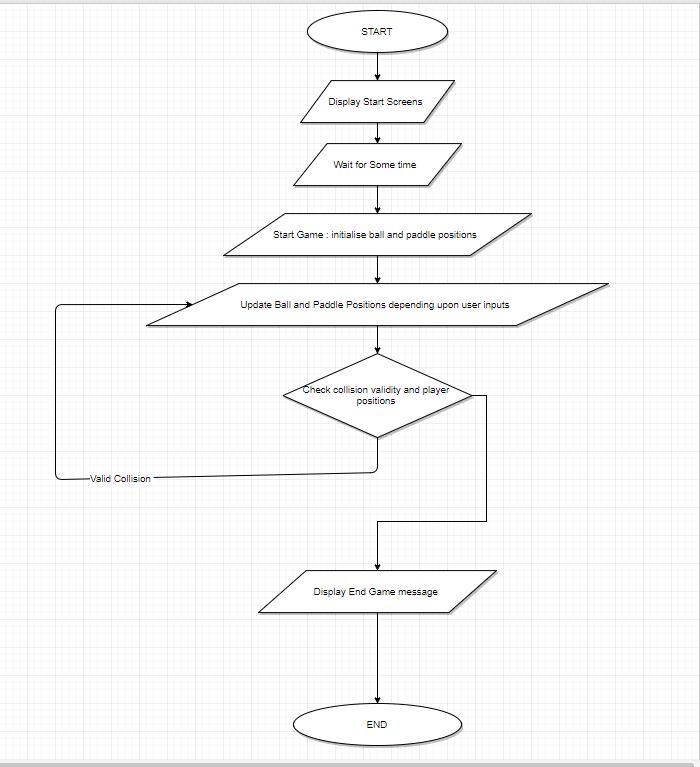
Language: C

Compiler: cpp

**Chapter 4:**

**Software design**

Data flow diagram:



**Chapter 5.Implementation**

#include <GL/gl.h>  
#include <GL/glu.h>  
#include <GL/glut.h>  
#include <cstdio>  
#include <cstring>  
#include <iostream>  
#ifdef \_\_linux\_\_  
#include <unistd.h>  
#endif  
  
#ifdef \_WIN32  
#include <Windows.h>  
#endif  
  
using namespace std;  
  
/\* Generic bar value \*/  
struct Bar  
{  
    float x;  
    float y;  
    float w;  
    float h;  
    float step;  
};  
  
/\* Generic ball value \*/  
struct Ball  
{  
    float r;  
    float x;  
    float y;  
    float velocity;  
    float xstep;  
    float ystep;  
};  
  
/\* Default bar \*/  
Bar bar = {0, -11, 3, 1, 0.5};  
/\* Top bar for bounce and scoring \*/  
Bar top\_bar = {0, 11, 16, 1, 0};  
/\* Main ball \*/  
Ball ball = {0.5, 0, 0, 1.0, 0.10, 0.10};  
  
const float bar\_maxr = 13; // right  
const float bar\_maxl = -13; // left  
  
const float ball\_maxu = 9.5; // up  
const float ball\_maxd = -9.5; // down  
const float ball\_maxr = 15; // right  
const float ball\_maxl = -15; // left  
  
bool isReachedXMax = false;  
bool isReachedYMax = false;  
bool isGameEnd = false;  
bool isGameOver = false;  
  
/\* Start the scene at 30 unit backward from center \*/  
const int zoom = -30;  
  
/\* Levels and points boundary \*/  
const int level\_a = 3;  
const int level\_b = 6;  
const int level\_c = 9;  
  
/\* Player configs \*/  
static int player\_score = 0;  
static int player\_lives = 3;  
static int level = 1;  
  
/\* Text buffer \*/  
char finaltext[100];  
  
static void printscore()  
{  
    char text[50];  
    /\* Printing values with formatted text \*/  
    sprintf(text, "Level %d - Score %d - Lives %d", level,  
player\_score, player\_lives);  
    /\* Yellow color \*/  
    glColor3f(1, 1, 0);  
    /\* Top left corner \*/  
    glRasterPos3f( -15 , 10.75 , zoom);  
    /\* Print using glutBitmapCharacter() \*/  
    for(int i = 0; text[i] != '\0'; i++)  
        glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, text[i]);  
}  
  
static void finalscreen()  
{  
    /\* If game over, show text in red color \*/  
    if(isGameOver)  
    {  
        glColor3f(1, 0, 0);  
    }  
    /\* Or show text in normal yellow color \*/  
    else if(player\_score == level\_c)  
    {  
        glColor3f(1, 1, 0);  
    }  
    /\* Center position \*/  
    glRasterPos3f(-7, 0, zoom);  
    for(int i = 0; finaltext[i] != '\0'; i++)  
        glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, finaltext[i]);  
    glutSwapBuffers();  
}  
  
void render(void)  
{  
    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);  
    glLoadIdentity();  
  
    /\* If game is at end, don't render scene, show final screen \*/  
    if(isGameEnd)  
    {  
        finalscreen();  
        return;  
    }  
  
    /\* Print score in each step \*/  
    printscore();  
  
    // Ball  
    glPushMatrix();  
    glTranslatef(ball.x, ball.y, zoom);  
    glColor3f(0.8, 0, 0.1);  
    glutSolidSphere(ball.r, 20, 20);  
    glPopMatrix();  
  
    // Bottom bar  
    glPushMatrix();  
    glTranslatef(bar.x, bar.y, zoom);  
    glBegin(GL\_QUADS);  
    glColor3f(1, 0, 0);  
    glVertex2f(bar.w, bar.h);  
    glColor3f(0, 1, 0);  
    glVertex2f(bar.w, -bar.h);  
    glColor3f(0, 0, 1);  
    glVertex2f(-bar.w, -bar.h);  
    glColor3f(1, 0, 1);  
    glVertex2f(-bar.w, bar.h);  
    glEnd();  
    glPopMatrix();  
  
    // Top bar  
    glPushMatrix();  
    glTranslatef(top\_bar.x, top\_bar.y, zoom);  
    glBegin(GL\_QUADS);  
    glColor3f(1, 0, 0);  
    glVertex2f(top\_bar.w, top\_bar.h);  
    glColor3f(0, 1, 0);  
    glVertex2f(top\_bar.w, -top\_bar.h);  
    glColor3f(0, 0, 1);  
    glVertex2f(-top\_bar.w, -top\_bar.h);  
    glColor3f(1, 0, 1);  
    glVertex2f(-top\_bar.w, top\_bar.h);  
    glEnd();  
    glPopMatrix();  
  
    glutSwapBuffers();  
}  
void init(void)  
{  
    /\* Init bg with grey color \*/  
    glClearColor( 0.9, 0.9, 0.9, 1);  
    glClearDepth( 1.0 );  
    /\* 32bit \*/  
    glEnable(GL\_DEPTH\_TEST);  
    /\* Smooth rendering \*/  
    glShadeModel(GL\_SMOOTH);  
    /\* Enabling colored objects \*/  
    glEnable(GL\_COLOR\_MATERIAL);  
}  
void reshape(int w, int h)  
{  
    float aspectRatio;  
    h = (h == 0) ? 1 : h;  
    w = (w == 0) ? 1 : w;  
    /\* Setting window dimention as viewport \*/  
    glViewport( 0, 0, w, h );  
    aspectRatio = (float)w/(float)h;  
    /\* Projection mode \*/  
    glMatrixMode(GL\_PROJECTION);  
    glLoadIdentity();  
    /\* Set perspective \*/  
    gluPerspective(45, aspectRatio, 1.0, 100.0);  
    /\* Back to modelview mode \*/  
    glMatrixMode(GL\_MODELVIEW);  
}  
void idle()  
{  
#ifdef \_WIN32  
    Sleep(1); //Sleep = usleep/1000   swap by 100 by mehul  
#else  
    usleep(1); //swap by 10000 by mehul  
#endif  
  
    /\* Don't calculate any score if game is at end \*/  
  
    if(isGameEnd)  
        return;  
  
    /\* If ball's Y value is equal to maximum\_down value  
then check if it touched the bar \*/  
    if(ball.y <= ball\_maxd)  
    {  
        /\* Check ball's co-ord is between bar's width \*/  
        if(ball.x<=bar.x-bar.w or ball.x>=bar.x+bar.w) // Bar missing  
        {  
            /\* Reduce life \*/  
            player\_lives--;  
            /\* Start the ball from bar, for user's ease \*/  
            bar.x = ball.x;  
            /\* Reset bar \*/  
            bar.y = -11;  
  
            /\* If player has no life left  
end the game and show black banner \*/  
            if(player\_lives==0)  
            {  
                glClearColor(0, 0, 0, 1);  
                isGameEnd = true;  
                isGameOver = true;  
                strcpy(finaltext, "Game Over! You lost all three balls");  
            }  
        }  
        else // Score  
        {  
            /\* Count score \*/  
            player\_score++;  
  
            /\* Increase ball's speed by 0.1 \*/  
            ball.velocity += 1.0;    // swap by 0.1 by  
  
            /\* Check score level \*/  
            if(player\_score==level\_a)  
                level++;  
            else if(player\_score==level\_b)  
                level++;  
            /\* If final level reached, end the game with green banner \*/  
            else if(player\_score==level\_c)  
            {  
                glClearColor(0, 0.8, 0, 1);  
                strcpy(finaltext, "Congratulation! Top score recorded");  
                isGameEnd = true;  
            }  
        }  
    }  
  
    /\* Ball bounce \*/  
  
    /\* If X value of the ball has touched right area \*/  
    if(ball.x>ball\_maxr)  
        isReachedXMax=true;  
    /\* If X value of the ball has touched left area \*/  
    else if(ball.x<ball\_maxl)  
        isReachedXMax=false;  
  
    /\* If Y value of the ball has touched top area \*/  
    if(ball.y>ball\_maxu)  
        isReachedYMax=true;  
    /\* If Y value of the ball has touched down area \*/  
    else if(ball.y<ball\_maxd)  
        isReachedYMax=false;  
  
    /\* Move the ball by step and velocity \*/  
    if(isReachedXMax)  
        ball.x -= ball.xstep \* ball.velocity;  
    else  
        ball.x += ball.xstep \* ball.velocity;  
    if(isReachedYMax)  
        ball.y -= ball.ystep \* ball.velocity;  
    else  
        ball.y += ball.ystep \* ball.velocity;  
  
    glutPostRedisplay();  
}  
  
void keyboard( unsigned char key, int x, int y )  
{  
    /\* If user press ESC key, quit \*/  
    switch(key)  
    {  
    case 27:  
        exit(1);  
    }  
}  
  
void specialkey(int key, int x, int y)  
{  
    /\* Move the bar with left right key \*/  
    if(key==GLUT\_KEY\_LEFT and bar.x > bar\_maxl)  
    {  
        bar.x -= bar.step;  
    }  
    else if(key==GLUT\_KEY\_RIGHT and bar.x < bar\_maxr)  
    {  
        bar.x += bar.step;  
    }  
}  
  
int main(int argc, char\*\* argv)  
{  
    glutInit(&argc,argv);  
    glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH );  
    glutInitWindowSize(640, 480);  
    glutCreateWindow("Ping Pong Game");  
    init();  
    glutReshapeFunc(reshape);  
    glutDisplayFunc(render);  
    glutKeyboardFunc(keyboard);  
    glutSpecialFunc(specialkey);  
    glutIdleFunc(idle);  
    glutMainLoop();  
    return 0;  
}

**Chapter6 :Snapshot**

