PING PONG

Done By:
Muthu Palaniyappan OL 12320

THE PSBB MILLENNIUM SCHOOL
GERUGAMBAKKAM

COMPUTER SCIENCE INVESTIGATORY PROJECT

DONE BY:
Muthu Palaniyappan OL
12320

ACKNOWLEDGMENT

I would like to express my special
thanks of gratitude to my
computer science teacher
“**Mrs.Bhavani**”
for her guidance and support
in completing my project.

I would also like to extend my
gratitude to the **Principal** Madam
“**Mrs.Bhavani Bhaskar**”
for providing with all the facility
that was required. Finally, I would
like to thank my Family for helping me
a lot in finalizing this project
within the limited time frame.

BONAFIDE CERTIFICATE

This is to certify that the project entitled "_____ _____" is a record of bonafide work carried out by _____ of class _____ in **THE PSBB MILLENNIUM SCHOOL, GERUGAMBAKKAM,CHENNAI** during the academic year **2019-20** in partial fulfilment of the requirements in **COMPUTER SCIENCE** prescribed by CBSE.

Submitted for All – India Senior Secondary Practical Examination held in **THE PSBB MILLENNIUM SCHOOL,GERUGAMBAKKAM** at **COMPUTER SCIENCE LAB.**

DATE: _____ PRINCIPAL

INTERNAL EXAMINER EXTERNAL EXAMINER

INDEX

INTRODUCTION TO C++

C++, as we all know is an extension to C language and was developed by **Bjarne stroustrup** at bell labs. C++ is an intermediate level language, as it comprises a confirmation of both high level and low level language features. C++ is a statically typed, free form, multiparadigm, compiled general-purpose language.

C++ is an **Object Oriented Programming language** but is not purely Object Oriented. Its features like Friend and Virtual, violate some of the very important OOPS features, rendering this language unworthy of being called completely Object Oriented. Its a middle level language.

OpenGL is a graphics API and not a platform of its own, it requires a language to operate in and the language of choice is C++ .

GLUT is designed to fill the need for a window system independent programming interface for OpenGL programs. The interface is designed to be simple yet still meet the needs of useful OpenGL programs

LOGIC USED

Header Files Used:

```
#include <windows.h>
#include <iostream>
#include <GL/glut.h>
#include <Math.h>
#include <unistd.h>
#include <cmath>
#include <string>
#include <stdio.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Class Used:

Player
BallClass

Functions Used:

```
DrawPaddle();
MovePaddleUp();
MovePaddleDown();
scoreUpdate();
showScore();
ScoreReached();
ScoreZero();
CPULogics();
alterpaddlelength(float);
resetBall();
YposOfBall();
ApplyBallLogicsForMotion();
```

```
proLogics();
seed();
random(int);
DrawToppedRectangle(float, float, float, float);
DrawBodRectangle(float);
checkwithinbox(float,float,float,float,float,float);
checkwithinlimit(float,float,float,float,float,float);
```

PROGRAM ANALYSIS

A Ping Pong Game Is Created By Muthu Palaniyappan OL Of Class 12 In Acadamic Year 2019 To 2020 For Investigatory Project.The Game Consists Of 3 Modes:

1) Classical Mode
2) Vs CPU Mode
3) Pro Mode

Classical Mode Is Very Basic Mode For Playing.
Vs CPU is Against Computer.
Pro Mode Is A Mode Full Of Twist And Turns.

Code Behind The Game Is Simply Planting The Ball & Paddle Moves Based On User Input And Elastic Collision Is Basic Concept Behind The Game.

MAIN.CPP

```
#include <windows.h>
#include <iostream>
#include <GL/glut.h>
#include <Math.h>
#include <unistd.h>
#include <cmath>
#include <string>
#include <stdio.h>
#define PI 3.14159265f
typedef float GameCoordinates;
using namespace std;

char title[] = "PingPong";
int windowWidth  = 640;
int windowHeight = 480;
int windowPosX   = 50;
int windowPosY   = 50;

bool GameStart = true,
     GameMode1 = false,
     GameMode2 = false,
     GameMode3 = false,
     GameMode1Result = false,
     GameMode2Result = false,
     GameMode3Result = false,
     Credits = false;

int refreshMillis = 30;
GLdouble xMaxNegative, xMaxPositive, yMaxNegative, yMaxPositive;
bool fullScreenMode = false;
bool Player1UpPress = false;
bool Player1DownPress = false;
bool Player2UpPress = false;
bool Player2DownPress = false;

#include "extra.h"
#include "main.h"

void initGL() {
   glClearColor(0.0, 0.0, 0.0, 1.0);
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    if(GameStart == true){
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0,1.0,1.0);
        renderBitmapString(-0.25,0.2,GLUT_BITMAP_9_BY_15,"Ping Pong Game");
```

```
    glColor3f(1.0,1.0,1.0);
    DrawCenteredRectangle(0,0,1,0.25);
    glColor3f(0.0, 0.0, 0.0);
    renderBitmapString(-0.20,0.0,GLUT_BITMAP_9_BY_15,"Classic Play");

    glColor3f(1.0,1.0,1.0);
    DrawCenteredRectangle(-0.13,-0.155,0.475,0.3);
    glColor3f(0.0, 0.0, 0.0);
    renderBitmapString(-0.38,-0.32,GLUT_BITMAP_9_BY_15,"VS CPU");

    glColor3f(1.0,1.0,1.0);
    DrawCenteredRectangle(0.13,-0.155,0.475,0.3);
    glColor3f(0.0, 0.0, 0.0);
    renderBitmapString(0.17,-0.32,GLUT_BITMAP_9_BY_15,"Pro");
    renderBitmapString(0.17,-0.38,GLUT_BITMAP_9_BY_15,"Dual");

    glColor3f(1.0,1.0,1.0);
    DrawCenteredRectangle(-0.13,-0.27,0.475,0.1);
    glColor3f(0.0, 0.0, 0.0);
    renderBitmapString(-0.38,-0.55,GLUT_BITMAP_9_BY_15,"Credits");

    glColor3f(1.0,1.0,1.0);
    DrawCenteredRectangle(0.13,-0.27,0.475,0.1);
    glColor3f(0.0, 0.0, 0.0);
    renderBitmapString(0.17,-0.55,GLUT_BITMAP_9_BY_15,"Exit");
}
else if(GameMode1 == true){
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.5,0.5,0.5);
    DrawBodRectangle(0.05);
    Ball.PlantBall();
    Ball.ApplyBallLogicsForMotion();
    Player1.DrawPaddle();
    Player2.DrawPaddle();
    int num1 = Player1.showScore();
    char buffer1[10]={'\0'};
    sprintf(buffer1, "%d", num1);
    renderBitmapString(-0.4f, yMaxPositive - 0.2 , GLUT_BITMAP_HELVETICA_18, buffer1);
    int num2 = Player2.showScore();
    char buffer2[10]={'\0'};
    sprintf(buffer2, "%d", num2);
    renderBitmapString(0.4f, yMaxPositive - 0.2 , GLUT_BITMAP_HELVETICA_18, buffer2);
}
else if(GameMode1Result == true){
    if(Player1.showScore()==7){
        glColor3f(1.0, 0.0, 0.0);
        renderBitmapString(-0.25,0,GLUT_BITMAP_9_BY_15,"Player 1 Wins");
    }
    else if(Player2.showScore()==7){
        glColor3f(1.0, 0.0, 0.0);
        renderBitmapString(-0.25,0,GLUT_BITMAP_9_BY_15,"Player 2 Wins");
    }
}
else if(GameMode2 == true){
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.5,0.5,0.5);
    DrawBodRectangle(0.05);
    Ball.PlantBall();
```

```
        Ball.ApplyBallLogicsForMotion();
        Player1.DrawPaddle();
        Player2.DrawPaddle();
        Player2.CPULogics();
        int num1 = Player1.showScore();
        char buffer1[10]={'\0'};
        sprintf(buffer1, "%d", num1);
        renderBitmapString(-0.4f, yMaxPositive - 0.2 , GLUT_BITMAP_HELVETICA_18, buffer1);
        int num2 = Player2.showScore();
        char buffer2[10]={'\0'};
        sprintf(buffer2, "%d", num2);
        renderBitmapString(0.4f, yMaxPositive - 0.2 , GLUT_BITMAP_HELVETICA_18, buffer2);
    }
    else if(GameMode2Result == true){
        if(Player1.showScore()==3){
            glColor3f(1.0, 0.0, 0.0);
            renderBitmapString(-0.25,0,GLUT_BITMAP_9_BY_15,"Player Wins");
        }
        else if(Player2.showScore()==3){
            glColor3f(1.0, 0.0, 0.0);
            renderBitmapString(-0.25,0,GLUT_BITMAP_9_BY_15,"CPU Wins");
        }
    }
    else if(GameMode3 == true){
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(0.5,0.5,0.5);
        DrawBodRectangle(0.05);
        Ball.PlantBall();
        Ball.ApplyBallLogicsForMotion();
        Ball.proLogics();
        Player1.DrawPaddle();
        Player2.DrawPaddle();
        int num1 = Player1.showScore();
        char buffer1[10]={'\0'};
        sprintf(buffer1, "%d", num1);
        renderBitmapString(-0.4f, yMaxPositive - 0.2 , GLUT_BITMAP_HELVETICA_18, buffer1);
        int num2 = Player2.showScore();
        char buffer2[10]={'\0'};
        sprintf(buffer2, "%d", num2);
        renderBitmapString(0.4f, yMaxPositive - 0.2 , GLUT_BITMAP_HELVETICA_18, buffer2);
    }
    else if(GameMode3Result == true){
        if(Player1.showScore()==10){
            glColor3f(1.0, 0.0, 0.0);
            renderBitmapString(-0.25,0,GLUT_BITMAP_9_BY_15,"Player 1 Wins");
        }
        else if(Player2.showScore()==10){
            glColor3f(1.0, 0.0, 0.0);
            renderBitmapString(-0.25,0,GLUT_BITMAP_9_BY_15,"Player 2 Wins");
        }
    }
    else if(Credits == true){
        glColor3f(0.0, 1.0, 0.0);
        renderBitmapString(-0.5,0.1,GLUT_BITMAP_9_BY_15,"Done By : Muthu Palaniyapan Ol");
        renderBitmapString(-0.25,0.0,GLUT_BITMAP_9_BY_15,"Country : India");
        renderBitmapString(-0.35,-0.1,GLUT_BITMAP_9_BY_15,"Game Name : PingPong");
        renderBitmapString(-0.5,-0.2,GLUT_BITMAP_9_BY_15,"Completed On : 20-OCTOBER-
2019");
```

```
        renderBitmapString(-0.35,-0.3,GLUT_BITMAP_9_BY_15,"Version : 3.00 00 00");
    }
    else{
        exit(0);
    }
    glutSwapBuffers();
}

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) {
        xMaxNegative   = -1.0 * aspect;
        xMaxPositive  = 1.0 * aspect;
        yMaxNegative = -1.0;
        yMaxPositive    = 1.0;
    } else {
        xMaxNegative   = -1.0;
        xMaxPositive  = 1.0;
        yMaxNegative = -1.0 / aspect;
        yMaxPositive    = 1.0 / aspect;
    }
    gluOrtho2D(xMaxNegative, xMaxPositive, yMaxNegative, yMaxPositive);
    glutReshapeWindow( 640, 480);
}

void Timer(int value) {
    glutPostRedisplay();
    glutTimerFunc(refreshMillis, Timer, 0);
}

void keyboard(unsigned char key, int x, int y) {
   switch (key) {
        case 27:
            break;
        case 'w':
            Player1UpPress = true;
            break;
        case 's':
            Player1DownPress = true;
            break;
        default:
            break;
   }
}

void keyboardUp(unsigned char key, int x, int y) {
   switch (key) {
        case 'w':
            Player1UpPress = false;
            break;
        case 's':
            Player1DownPress = false;
            break;
        default:
```

```
            break;
   }
}

void specialKeys(int key, int x, int y) {
   switch (key) {
      case GLUT_KEY_F1:
         break;
         case GLUT_KEY_UP:
            Player2UpPress = true;
            break;
        case GLUT_KEY_DOWN:
            Player2DownPress = true;
            break;
        default:
            break;
   }
}

void specialKeysUp(int key, int x, int y) {
   switch (key) {
      case GLUT_KEY_F1:
         break;
         case GLUT_KEY_UP:
            Player2UpPress = false;
            break;
        case GLUT_KEY_DOWN:
            Player2DownPress = false;
            break;
        default:
            break;
   }
}

void MouseFunc(int button, int state, int x, int y)
{
    float X = (float) x / (windowWidth/2) - 1.0;
    float Y = (float) (windowHeight-y) / (windowHeight/2) - 1.0;

       if(button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        if((GameStart == true) && checkwithinbox(X,Y,0,0,0.75,0.25)) {
            GameMode1 = true;
            GameStart = false;
        }
        if((GameStart == true) && checkwithinbox(X,Y,-0.20,-0.3,0.36,0.3)) {
            GameMode2 = true;
            GameStart = false;
        }
        if((GameStart == true) && checkwithinbox(X,Y,0.19,-0.3,0.36,0.3)) {
            GameMode3 = true;
            GameStart = false;
        }
        else if((GameStart == true) && checkwithinbox(X,Y,-0.2,-0.55,0.35,0.1)) {
            Credits = true;
            GameStart = false;
        }
        else if((GameStart == true) && checkwithinbox(X,Y,0.2,-0.55,0.35,0.1)) {
            exit(0);
```

```
        }
        else if((GameMode1Result == true)) {
            GameMode1Result = false;
            GameStart = true;
            Player1.scoreZero();
            Player2.scoreZero();
        }
        else if((GameMode2Result == true)) {
            GameMode2Result = false;
            GameStart = true;
            Player1.scoreZero();
            Player2.scoreZero();
        }
        else if((GameMode3Result == true)) {
            GameMode3Result = false;
            GameStart = true;
            Player1.scoreZero();
            Player2.scoreZero();
        }
        else if((Credits == true)) {
            Credits = false;
            GameStart = true;
        }
       }
       else if(button == GLUT_LEFT_BUTTON && state == GLUT_UP) {

       }
}

void Updater(){
    if(Player1UpPress == true){
        Player1.MovePaddleUp();
    }
    else{
        ;
    }
    if(Player1DownPress == true){
        Player1.MovePaddleDown();
    }
    else{
        ;
    }
    if(Player2UpPress == true && (GameMode1 == true||GameMode3==true)){
        Player2.MovePaddleUp();
    }
    else{
        ;
    }
    if(Player2DownPress == true && (GameMode1 == true||GameMode3==true)){
        Player2.MovePaddleDown();
    }
    else{
        ;
    }
}

int main(int argc, char** argv) {
   seed();
```

```
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE);
    glutInitWindowSize(windowWidth, windowHeight);
    glutInitWindowPosition(windowPosX, windowPosY);
    glutCreateWindow(title);
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutTimerFunc(0, Timer, 0);
    glutSpecialFunc(specialKeys);
    glutSpecialUpFunc(specialKeysUp);
    glutKeyboardFunc(keyboard);
    glutKeyboardUpFunc(keyboardUp);
    glutMouseFunc(MouseFunc);
    glutIdleFunc(Updater);
    initGL();
    glutMainLoop();
    return 0;
}
```

MAIN.H

```
#ifndef MAIN_H_INCLUDED
#define MAIN_H_INCLUDED

GLfloat ballX = 0.0f;
GLfloat ballY = 0.0f;

class Player{
    private:
        float paddlelength = 0.4;
        float paddlewidth = 0.01;
        char playerName;
        float controller = 0.0f;
        int score = 0;
    public:
        Player(char a){
            playerName = a;
        }
        void DrawPaddle(){
            glPushMatrix();
            if(playerName=='a'){
                glBegin(GL_QUADS);
                glColor3f(1,0.0,0.0);
                glVertex2f(xMaxNegative +0.05 +0.01 + paddlewidth,controller +
paddlelength/2);
                glVertex2f(xMaxNegative +0.05 +0.01,controller + paddlelength/2);
                glVertex2f(xMaxNegative +0.05 +0.01,controller - paddlelength/2);
                glVertex2f(xMaxNegative +0.05 +0.01 + paddlewidth,controller -
paddlelength/2);
                glEnd();
            }
            else if(playerName=='b'){
                glBegin(GL_QUADS);
                glColor3f(0,0.0,1.0);
                glVertex2f(xMaxPositive -0.05 -0.01 - paddlewidth,controller +
paddlelength/2);
                glVertex2f(xMaxPositive -0.05 -0.01,controller + paddlelength/2);
                glVertex2f(xMaxPositive -0.05 -0.01,controller - paddlelength/2);
                glVertex2f(xMaxPositive -0.05 -0.01 - paddlewidth,controller -
paddlelength/2);
                glEnd();
            }
            glPopMatrix();
        }
        void MovePaddleUp(){
            if(controller + paddlelength/2 + 0.01 + 0.05 > yMaxPositive){
                controller = yMaxPositive - 0.05 - 0.01 - paddlelength/2;
            }
            else{
                controller +=0.0000005;
            }
```

```
        }
        void MovePaddleDown(){
            if( controller - paddlelength/2 - 0.01 - 0.05 < yMaxNegative){
                controller = yMaxNegative + 0.05 + 0.01 + paddlelength/2;
            }
            else{
                controller -=0.0000005;
            }
        }
        bool hitingpaddle(float bally){
            if((bally+0.05>controller-paddlelength/2) && (bally-
0.05<controller+paddlelength/2)){
                return true;
            }
            else {
                return false;
            }
        }
        void scoreUpdate(){
            score++;
        }
        int showScore(){
            return score;
        }
        bool Scorereached(){
            if((score==7&&GameMode1==true)||(score==3&&GameMode2==true)||
(score==10&&GameMode3==true)){
                return true;
            }
            else{
                return false;
            }
        }
        void scoreZero(){
            score =0;
        }
        void CPULogics(){
            if(controller < ballY){
                controller+=0.02;
            }
            else if(controller > ballY){
                controller-=0.02;
            }
        }
        void alterpaddlelength(float a){
                paddlelength = a;
        }
}Player1('a'),Player2('b');

class BallClass{
    private:
        GLfloat ballRadius = 0.03f;
        GLfloat ballXMax, ballXMin, ballYMax, ballYMin;
        GLfloat xSpeed = 0.01f;
        GLfloat ySpeed = 0.003f;
    public:
        BallClass(){
            xSpeed = pow((-1),random(10))*(random(10)/1000 + 0.03);
```

```
    ySpeed = pow((-1),random(10))*(random(10)/1000 + 0.03);
}
void resetBall(){
    Sleep(1000);
    ballRadius = 0.03f;
    xSpeed = pow((-1),random(10))*(random(10)/1000 + 0.03);
    ySpeed = pow((-1),random(10))*(random(10)/1000 + 0.03);
    ballX =0.0f;
    ballY =0.0f;
}
float YposOfBall(){return ballY;}
void PlantBall(){
    ballXMin = xMaxNegative + ballRadius + 0.05 + 0.01 +0.02;
    ballXMax = xMaxPositive - ballRadius - 0.05 - 0.01 -0.02;
    ballYMin = yMaxNegative + ballRadius + 0.05 +0.03;
    ballYMax = yMaxPositive - ballRadius - 0.05 -0.03;
    glPushMatrix();
    glTranslatef(ballX, ballY, 0.0f);
    glBegin(GL_TRIANGLE_FAN);
    glColor3f(1.0f, 1.0f, 1.0f);
    glVertex2f(0.0f, 0.0f);
    int numSegments = 100;
    GLfloat angle;
    for (int i = 0; i <= numSegments; i++) {
        angle = i * 2.0f * PI / numSegments;
        glVertex2f(cos(angle) * ballRadius, sin(angle) * ballRadius);
    }
    glEnd();
    glPopMatrix();
}
void ApplyBallLogicsForMotion(){
    if(Player1.Scorereached()==false || Player2.Scorereached()==false)
    if((ballX > ballXMax)){
        if((Player2.hitingpaddle(ballY)==true))
        {
            ballX = ballXMax;
            xSpeed = -xSpeed+(-xSpeed/20);
        }
        else{
            Player1.scoreUpdate();
            resetBall();
        }
    } else if((ballX < ballXMin)){
        if((Player1.hitingpaddle(ballY)==true))
        {
            ballX = ballXMin;
            xSpeed = -xSpeed+(-xSpeed/20);
        }
        else{
            Player2.scoreUpdate();
            resetBall();
        }
    }
    else {
        ballX += xSpeed;
    }

    if(Player1.Scorereached()==false || Player2.Scorereached()==false)
```

```
            if(ballY > ballYMax){
                ballY = ballYMax;
                ySpeed = -ySpeed;
            } else if(ballY < ballYMin){
                ballY = ballYMin;
                ySpeed = -ySpeed;
            }
            else {
                ballY += ySpeed;
            }

            if(Player1.Scorereached()==true || Player2.Scorereached()==true){
                if(GameMode1==true)
                {
                    GameMode1Result = true;
                    GameMode1 = false;
                }
                if(GameMode2==true){
                    GameMode2Result = true;
                    GameMode2 = false;
                }
                if(GameMode3==true)
                {
                    GameMode3Result = true;
                    GameMode3 = false;
                }
            }
        }
        void proLogics(){
            if(Player1.showScore()==1||Player1.showScore()==1){
                ballRadius=0.02;
            }
            if(Player1.showScore()==2){
                Player1.alterpaddlelength(0.2);
            }
            if(Player2.showScore()==2){
                Player2.alterpaddlelength(0.2);
            }
        }
}Ball;

#endif
```

EXTRA.H

```
#ifndef EXTRA_H_INCLUDED
#define EXTRA_H_INCLUDED

#include <stdio.h>
#include <stdlib.h>
#include<time.h>

void seed(){
    srand (time(NULL));
}

int random(int a){
    return (rand()%a);
}

void renderBitmapString(float x, float y, void *font,const char *string){
    const char *c;
    glRasterPos2f(x, y);
    for (c=string; *c != '\0'; c++) {
        glutBitmapCharacter(font, *c);
    }
}

void DrawCenteredRectangle(float x, float y, float width, float height){
    glPushMatrix();
    glTranslatef(x, y, 0.0f);
    glBegin(GL_QUADS);
    glVertex2f(x+width/2, y+height/2);
    glVertex2f(x-width/2, y+height/2);
    glVertex2f(x-width/2, y-height/2);
    glVertex2f(x+width/2, y-height/2);
    glEnd();
    glPopMatrix();
}

void DrawToppedRectangle(float x, float y, float width, float height){
    glPushMatrix();
    glTranslatef(x, y, 0.0f);
    glBegin(GL_QUADS);
    glVertex2f(x+width, y);
    glVertex2f(x, y);
    glVertex2f(x, y+height);
    glVertex2f(x+width, y+height);
    glEnd();
    glPopMatrix();
};

void DrawBodRectangle(float rectangleborder){
    glPushMatrix();
    glBegin(GL_QUADS);
```

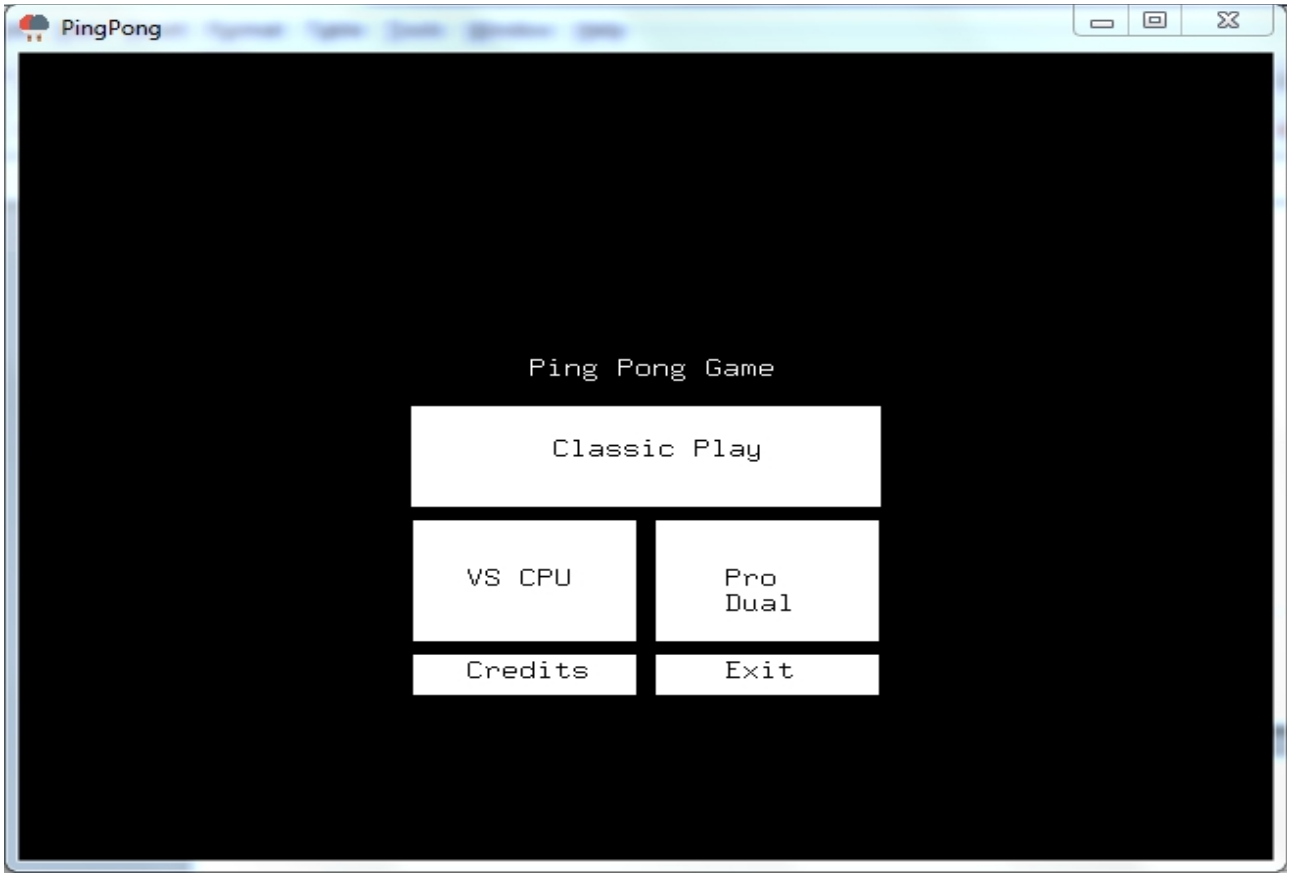
```
    glVertex2f(xMaxPositive,yMaxPositive);
    glVertex2f(xMaxNegative,yMaxPositive);
    glVertex2f(xMaxNegative,yMaxPositive-rectangleborder);
    glVertex2f(xMaxPositive,yMaxPositive-rectangleborder);
    glEnd();
    glBegin(GL_QUADS);
    glVertex2f(xMaxNegative+rectangleborder,yMaxPositive);
    glVertex2f(xMaxNegative,yMaxPositive);
    glVertex2f(xMaxNegative,yMaxNegative);
    glVertex2f(xMaxNegative+rectangleborder,yMaxNegative);
    glEnd();
    glBegin(GL_QUADS);
    glVertex2f(xMaxPositive,yMaxNegative+rectangleborder);
    glVertex2f(xMaxNegative,yMaxNegative+rectangleborder);
    glVertex2f(xMaxNegative,yMaxNegative);
    glVertex2f(xMaxPositive,yMaxNegative);
    glEnd();
    glBegin(GL_QUADS);
    glVertex2f(xMaxPositive,yMaxPositive);
    glVertex2f(xMaxPositive-rectangleborder,yMaxPositive);
    glVertex2f(xMaxPositive-rectangleborder,yMaxNegative);
    glVertex2f(xMaxPositive,yMaxNegative);
    glEnd();
    glPopMatrix();
}

bool checkwithinbox(float X,float Y,float Xcoord,float Ycoord,float Width,float Height){
if(X>(Xcoord-(Width/2)) && X<(Xcoord+(Width/2)) && Y>(Ycoord-(Height/2)) && Y<(Ycoord+
(Height/2)))
{
    return true;
}
else{
    return false;
}
}

bool checkwithinlimit(float X,float Y,float x1,float y1,float x2,float y2){
if(X>x2 && X<x1 && Y>y1 && Y<y2)
{
    return true;
}
else{
    return false;
}
}

#endif
```

OUTPUT


PingPong
Ping Pong Game
Classic Play
VS CPU
Pro
Dual
Credits
Exit


PingPong
3
2

BIBLIOGRAPHY

- Computer Science With C++, Class 11th, Sumita Arora, 10th Edition
- Computer Science With C++, Class 12th, Sumita Arora, 11th Edition

WEBLIOGRAPHY

- http://www.lighthouse3d.com/tutorials/glut-tutorial/
- https://www2.cs.arizona.edu/classes/cs433/spring02/opengl/index.html
- https://www.geeksforgeeks.org/getting-started-with-opengl/