﻿//Name: Asvene Pathmanathan

//Date: January 10, 2023

//Title: Single Player Pong

//Purpose: Create a single player pong game

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading;

using System.Threading.Taskstatics;

using System.Windows.Forms;

namespace MovingStuffAP

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

//Variable Declaration

static int intSpeed = 4;

static int intDirection = 1;

static int intHMove = intSpeed;

static int intVMove = intSpeed;

static int intBottom = 383;

static int intAngle = 30;

static int intPaddleDirection;

static int intCompCounter = 0;

static int intPlayerCounter = 0;

static Random rnd = new Random();

static int intCompSpeed;

int rndPaddleAI;

int rndDirection;

private void btnStart\_Click(object sender, EventArgs e)

{

this.tmrGame.Enabled = true;

}

//This method resets the game after a winner is claimed

public void Reset()

{

MessageBox.Show("New game is starting");

intCompCounter = 0;

intPlayerCounter = 0;

this.lblCPUScore.Text = "0";

this.lblPlayerScore.Text = "0";

pcbBall.Top = pcbSplit.Top;

pcbBall.Left = pcbSplit.Width / 2;

intAngle = 30;

}

//This method restarts each round of the game

public void Restart()

{

pcbBall.Left = pcbBackground.Width / 2;

pcbBall.Top = pcbSplit.Top;

rndDirection = rnd.Next(1, 5);

intDirection = rndDirection;

this.tmrGame.Enabled = true;

intAngle = 30;

}

//Checking the winner

public void CheckWinner()

{

if (intCompCounter == 10)

{

tmrGame.Enabled = false;

MessageBox.Show("Computer wins!");

Reset();

}

else if (intPlayerCounter == 10)

{

tmrGame.Enabled = false;

MessageBox.Show("You win!");

Reset();

}

}

//Method for the computer paddle to use AI

public void CompSmart()

{

//A random number is used to see if AI for the comp paddle will be executed

Random rnd = new Random();

for (int i = 0; i < 1; i++)

{

//Making sure the computer paddle doesn't go out of bounds when using AI

// the plus six is used incase the paddle wants to supass the boundary to meet a command below

if (pcbCompPaddle.Left > pcbBackground.Width - pcbCompPaddle.Width + 6)

{

intPaddleDirection = -1;

}

else if (pcbCompPaddle.Left == pcbBackground.Left)

{

intPaddleDirection = 1;

}

//1/20 chance per timer interval for computer to use AI when the ball's angle is 90

rndPaddleAI = rnd.Next(0, 20);

if (rndPaddleAI == 0)

{

if (intAngle == 90 && pcbBall.Left == pcbCompPaddle.Left)

{

intCompSpeed = 1;

}

}

//1/25 chance for computer to use AI for other scenerios

rndPaddleAI = rnd.Next(0, 25);

if (rndPaddleAI == 1)

{

if (pcbCompPaddle.Left == pcbBackground.Width - pcbCompPaddle.Width)

{

intPaddleDirection = -1;

}

else if (pcbBall.Left < pcbCompPaddle.Left && pcbBall.Height > pcbSplit.Height)

{

intPaddleDirection = -1;

}

else if (pcbBall.Left > pcbCompPaddle.Left && pcbBall.Height > pcbSplit.Height)

{

intPaddleDirection = 1;

}

}

}

}

private void tmrGame\_Tick(object sender, EventArgs e)

{

intHMove = horizontalVal(intSpeed, intAngle);

intVMove = verticalVal(intSpeed, intAngle);

//Depending on intDirection, the horizontal and vertical move will be determined

if (intDirection == 1)

{

intHMove = Math.Abs(intHMove);

intVMove = Math.Abs(intVMove) \* -1;

}

else if (intDirection == 2)

{

intHMove = Math.Abs(intHMove) \* -1;

intVMove = Math.Abs(intVMove) \* -1;

}

else if (intDirection == 3)

{

intHMove = Math.Abs(intHMove) \* -1;

intVMove = Math.Abs(intVMove);

}

else if (intDirection == 4)

{

intHMove = Math.Abs(intHMove);

intVMove = Math.Abs(intVMove);

}

this.pcbBall.Left += intHMove;

this.pcbBall.Top += intVMove;

//Counter clockwise

if (intDirection == 1 && pcbBall.Left > pcbBackground.Width)

{

intDirection = 2;

}

else if (intDirection == 2 && pcbBall.Top < pcbBackground.Top)

{

//Player gets a point

intPlayerCounter++;

this.lblPlayerScore.Text = intPlayerCounter.ToString();

CheckWinner();

Restart();

}

else if (intDirection == 3 && pcbBall.Left < pcbBackground.Left)

{

intDirection = 4;

}

else if (intDirection == 4 && pcbBall.Top > intBottom)

{

//Computer gets a point

intCompCounter++;

this.lblCPUScore.Text = intCompCounter.ToString();

CheckWinner();

Restart();

}

//Clockwise

if (intDirection == 2 && pcbBall.Left < pcbBackground.Left)

{

intDirection = 1;

}

else if (intDirection == 1 && pcbBall.Top < pcbBackground.Top)

{

//Player gets a point

intPlayerCounter++;

this.lblPlayerScore.Text = intPlayerCounter.ToString();

CheckWinner();

Restart();

}

else if (intDirection == 4 && pcbBall.Left > pcbBackground.Width)

{

intDirection = 3;

}

else if (intDirection == 3 && pcbBall.Top > intBottom)

{

//Computer gets a point

intCompCounter++;

this.lblCPUScore.Text = intCompCounter.ToString();

CheckWinner();

Restart();

}

//Making the pcbBall bounce off from the pcbPlayerPaddle

if (this.pcbPaddle.Bounds.IntersectsWith(this.pcbBall.Bounds))

{

if (intDirection == 4)

{

intDirection = 1;

}

else if (intDirection == 3)

{

intDirection = 2;

}

//Angles For player paddle

if (pcbBall.Left > pcbPaddle.Left && pcbBall.Left < pcbPaddle.Left + pcbPaddle.Width / 3)

{

intAngle = 20;

}

else if (pcbBall.Left > pcbPaddle.Left && pcbBall.Left < pcbPaddle.Left + (pcbPaddle.Width / 3) \* 2)

{

intAngle = 90;

}

else if (pcbBall.Left > pcbPaddle.Left && pcbBall.Left < (pcbPaddle.Width - (pcbPaddle.Width / 3)) \* 2)

{

intAngle = 20;

}

}

//Making the computer paddle move automatically

if (pcbCompPaddle.Left == pcbBackground.Width - pcbCompPaddle.Width)

{

intPaddleDirection = -1;

}

else if (pcbCompPaddle.Left == pcbBackground.Left)

{

intPaddleDirection = 1;

}

//Method is called to see if comp paddle uses AI

CompSmart();

this.pcbCompPaddle.Left += 5 \* intPaddleDirection;

//Programs how the ball bounces off of the comp paddle

if (this.pcbCompPaddle.Bounds.IntersectsWith(this.pcbBall.Bounds))

{

if (intDirection == 1)

{

intDirection = 4;

}

else if (intDirection == 2)

{

intDirection = 3;

}

//Angles For Computer paddle

if (pcbBall.Left > pcbCompPaddle.Left && pcbBall.Left < pcbCompPaddle.Left + pcbCompPaddle.Width / 3)

{

intAngle = 20;

}

else if (pcbBall.Left > pcbCompPaddle.Left && pcbBall.Left < pcbCompPaddle.Left + (pcbCompPaddle.Width / 3) \* 2)

{

intAngle = 90;

}

else if (pcbBall.Left > pcbCompPaddle.Left && pcbBall.Left < (pcbCompPaddle.Width - (pcbCompPaddle.Width / 3)) \* 2)

{

intAngle = 20;

}

this.pcbCompPaddle.Left += 5 \* intPaddleDirection;

}

}

//Horizontal Movement

public int horizontalVal(int intHyp, int intDegree)

{

return (int)(intHyp \* Math.Cos((double)intDegree \* Math.PI / 180));

}

//Vertical movement

public int verticalVal(int intHyp, int intDegree)

{

return (int)(intHyp \* Math.Sin((double)intDegree \* Math.PI / 180));

}

private void btnStop\_Click(object sender, EventArgs e)

{

this.tmrGame.Enabled = false; //Timer is false, therefore game is paused

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

//Makes the ball move faster

private void btnFast\_Click(object sender, EventArgs e)

{

intSpeed += 1;

}

//Makes the ball move slower

private void btnSlow\_Click(object sender, EventArgs e)

{

if (intSpeed < 5)

{

intSpeed += 0;

}

else

{

intSpeed -= 1;

}

}

//Exits application

private void btnExit\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

//Allows the A and D keys to be used for left and right movement of the user paddle

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyData == Keys.D)

{

if (pcbPaddle.Left + pcbPaddle.Width == pcbBackground.Width + pcbExtra.Width)

{

this.pcbPaddle.Left += 0; //makes sure paddle does not exceed boundary

}

else if (pcbPaddle.Left + pcbPaddle.Width < pcbBackground.Width + pcbExtra.Width)

{

this.pcbPaddle.Left += 5;

}

}

else if (e.KeyData == Keys.A)

{

if (pcbPaddle.Left == pcbBackground.Left)

{

this.pcbPaddle.Left -= 0; //makes sure paddle does not exceed boundary

}

else

{

this.pcbPaddle.Left -= 5;

}

}

}

private void pcbCompPaddle\_Click(object sender, EventArgs e)

{

if (this.pcbPaddle.Bounds.IntersectsWith(this.pcbBall.Bounds))

{

}

}

}

}