Dílenská praxe

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A4** | 6. Tenis | | | |
| Dobeš Daniel | |  | 1/3 | Známka: |
| 23. 1. 2019 | | Datum odevzdání: | 20. 2. 2019 | Odevzdáno: |

Zadání:

Sestavte program v jazyce **C#**, který bude podporovat činnost nejméně dvou hráčů, kteří hrají jednoduchou videohru – tenis. Hra má pravidla:

1. Hraje se v pravoúhlém poli. Rozměr pole je nejméně 200x300 obrazových bodů. Větší počet bodů je vždy ve vodorovném směru.
2. Každý hráč si vybere svoji stranu hřiště.
3. Hrací pálka je znázorněna krátkou úsečkou pohybující se po zadní straně hřiště. Její délka je maximálně 0,2 \* délka hřiště.
4. „míč“ je odražen zpět k soupeři, pokud je mu v přechodu zadní strany zabráněno „přítomností“ hrací pálky. Jinak má soupeř bod. Při odrazech míče platí zákony o úhlu dopadu a odrazu s respektováním rotace míčku.
5. Hráči ovládají svoji pálku nezávisle.
6. Program průběžně sleduje a zobrazuje průběžný stav hry a toho hráče, který první splní podmínky výhry, vyhlásí za vítěze.
7. Volitelně může program nahradit jednoho z hráčů.
8. Volitelně může program měnit obtížnost hry, například po dosažení jistého stavu bodů může zrychlit pohyb míčku.

Stavový diagram:

Viz příloha.

Výpis programu:

Viz příloha.

Komentář:

Hru lze pozastavit klávesou „P“ nebo „mezerníkem“, přepínání mezi modem pro 2 hráče a pro 1 hráče a počítač lze klávesou „C“. Je možné měnit počet kol a rychlost pohybu pálek (doporučuji rozmezí 10-20).

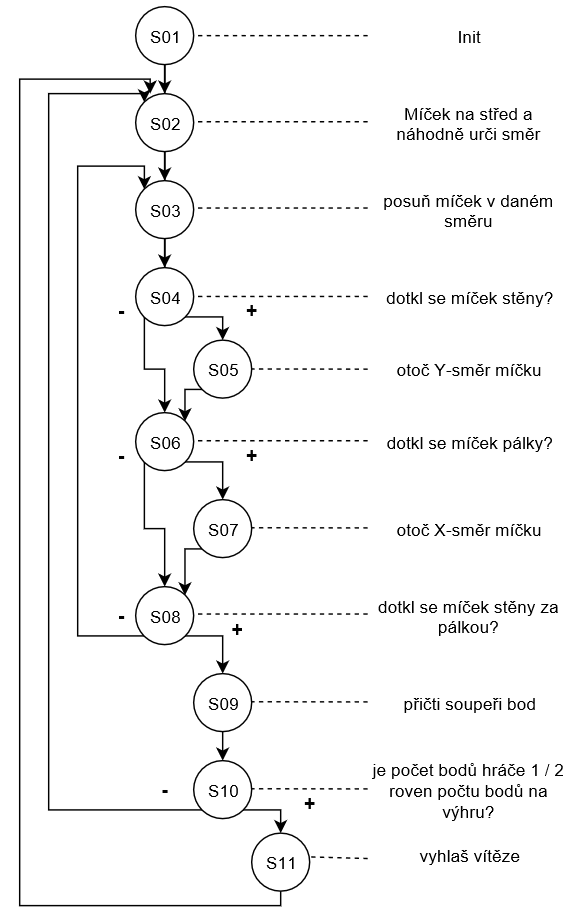
Závěr:

Program je plně funkční, navíc jsem do programu přidal možnost pozastavení hry, během pozastavení se míček nehýbe a není možné pohybovat pálkami.

Přílohy:

* Stavový diagram (1 strana)
* Výpis programu (4 strany)

Příloha 1 – Stavový diagram:



Příloha 2 – Výpis programu:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing.Drawing2D; //Must be defined

using System.Reflection; //Must be defined

namespace U6\_Tenis

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

timer\_preparation.Enabled = true;

timer\_ball.Enabled = false;

timer\_pad.Enabled = true;

timer\_pause.Enabled = true;

label\_pause.Visible = false;

}

int pad1length = 125; //left pad length

int pad2length = 125; //left pad length

int xball = 430, yball = 280; //xy ball coordinates

int pad1 = 238, pad2 = 238; //y pad coordinates

int pad1up, pad1down, pad2up, pad2down; //key control

int ballspeed = 0; //default ball speed

int xballdirection = 1, yballdirection = 1; //default ball direction

int RightPlayerScore = 0, LeftPlayerScore = 0; //score each player

int difficulty = 3; //default

int Round = 0; //actual round

int RoundCount = 5; //number of round to win

int padspeed = 15; //10 - 20

int pause = 0; //if 1 -> game is paused

int computer = 0; //if 1 -> player 2 = computer

Random ball = new Random(); //random ball direct

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

{

//Double buffer used when panel is refreshed -> no flashing

typeof(Panel).InvokeMember("DoubleBuffered", BindingFlags.SetProperty

| BindingFlags.Instance | BindingFlags.NonPublic,

null, panel\_playground, new object[] { true });

}

private void panel\_playground\_Paint(object sender, PaintEventArgs e)

{

Graphics tenis = e.Graphics;

Pen BlackPen = new Pen(Color.Black, 3); //definitions of brushes and pen

SolidBrush RedBrush = new SolidBrush(Color.DarkRed);

SolidBrush GreenBrush = new SolidBrush(Color.LightGreen);

SolidBrush YellowBrush = new SolidBrush(Color.Yellow);

SolidBrush BlueBrush = new SolidBrush(Color.DarkSlateBlue);

tenis.FillRectangle(GreenBrush, 0, 0, 900, 600);//background of playground

tenis.DrawLine(BlackPen, 450, 0, 450, 600); //middle line

tenis.DrawRectangle(BlackPen, 0, 0, 900, 600); //border line

tenis.FillEllipse(YellowBrush, xball, yball, 40, 40); //ball

tenis.DrawEllipse(BlackPen, xball, yball, 40, 40); //ball

//pad - player 1 & 2 – left & right

tenis.FillRectangle(RedBrush, 10, pad1, 25, pad1length);

tenis.FillRectangle(BlueBrush, 865, pad2, 25, pad2length);

}

//pad control

protected override void OnKeyDown(KeyEventArgs e) //Keystroke test

{

//if "key" is pressed -> set pad divergence

if ((e.KeyCode == Keys.Space) || (e.KeyCode == Keys.P)) pause++;

if (e.KeyCode == Keys.C)

{

computer++;

label\_player2.Text = Convert.ToString("Computer");

pad2length = 600; //set full wall pad

pad2 = 0;

}

if (e.KeyCode == Keys.W) pad1up = 1;

if (e.KeyCode == Keys.S) pad1down = 1;

if (e.KeyCode == Keys.Up) pad2up = 1;

if (e.KeyCode == Keys.Down) pad2down = 1;

}

protected override void OnKeyUp(KeyEventArgs e) //Key Release Test

{

if (((e.KeyCode == Keys.Space) || (e.KeyCode == Keys.P)) && (pause == 2))

{

pause = 0;

timer\_ball.Enabled = true;

timer\_pad.Enabled = true;

label\_pause.Visible = false;

}

if ((e.KeyCode == Keys.C) && (computer == 2))

{

computer = 0;

label\_player2.Text = Convert.ToString("Player 2");

pad2length = 125; //reset pad 2

pad2 = 238;

}

//if "key" is unpressed -> reset pad divergence

if (e.KeyCode == Keys.W) pad1up = 0;

if (e.KeyCode == Keys.S) pad1down = 0;

if (e.KeyCode == Keys.Up) pad2up = 0;

if (e.KeyCode == Keys.Down) pad2down = 0;

}

private void timer\_pad\_Tick(object sender, EventArgs e)

{

//if some key was pressed -> set new y pad coordinates & refresh panel

if (pad1up == 1 && pad1 > 0) pad1 = pad1 - padspeed;

if (pad1down == 1 && pad1 < 475) pad1 = pad1 + padspeed;

if (computer != 1)

{

if (pad2up == 1 && pad2 > 0) pad2 = pad2 - padspeed;

if (pad2down == 1 && pad2 < 475) pad2 = pad2 + padspeed;

}

Refresh();

} //end of timer\_pad\_Tick

private void timer\_ball\_Tick(object sender, EventArgs e)

{

ballspeed = difficulty;

xball = xball + (xballdirection \* ballspeed); //ball x movement

yball = yball + (yballdirection \* ballspeed); //ball y movement

//bounce from the wall

if ((yball <= 0) || (yball >= 560)) yballdirection = -yballdirection;

//bounce from the pad

if (((xball <= 35) && (yball > pad1) && (yball < pad1 + pad1length))

|| (xball >= 830) && (yball > pad2) && (yball < pad2 + pad2length))

xballdirection = -xballdirection;

if ((xball < 1) || (xball > 850))

{

if (xball < 1) RightPlayerScore++; //left wall hit

if (xball > 850) LeftPlayerScore++; //right wall hit

ballspeed = 0; //stop ball

xball = 430; //xy ball coordinates

yball = 280;

timer\_ball.Enabled = false; //turn off timer - ball control

timer\_preparation.Enabled = true; //turn on timer - preparation

}

if ((LeftPlayerScore == RoundCount) || (RightPlayerScore == RoundCount))

{

timer\_preparation.Enabled = false;

if (computer == 1)

{

MessageBox.Show("You lose, try again", "",

MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

else

{

if (LeftPlayerScore == RoundCount) //Left player win

MessageBox.Show("Left player win", "Congratulations!", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

if (RightPlayerScore == RoundCount) //Right player win

MessageBox.Show("Right player win", "Congratulations!", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

LeftPlayerScore = 0; //Score reset

RightPlayerScore = 0;

difficulty = 3; //difficulty reset

Round = 0; //round reset

timer\_preparation.Enabled = true;

}

Refresh();

} //end of timer\_ball\_Tick

private void timer\_preparation\_Tick(object sender, EventArgs e)

{ //delay 2 sec - preparation for next round

difficulty++; //increase ball speed -> higher difficulty

Round++; //increment round

//show statistic

label\_LeftPlayerScore.Text = Convert.ToString(LeftPlayerScore);

label\_RightPlayerScore.Text = Convert.ToString(RightPlayerScore);

label\_round.Text = Convert.ToString(Round);

//generation random number -> random ball direction

if (ball.Next(0, 1) == 1) xballdirection = -xballdirection;

if (ball.Next(0, 1) == 1) yballdirection = -yballdirection;

timer\_ball.Enabled = true; //turn on timer - ball control

timer\_preparation.Enabled = false; //turn off timer - preparation

} //end of timer\_preparation\_Tick

private void timer\_pause\_Tick(object sender, EventArgs e)

{

if (pause == 1)

{

timer\_ball.Enabled = false;

timer\_pad.Enabled = false;

timer\_preparation.Enabled = false;

label\_pause.Visible = true;

}

} //end of timer\_pause\_Tick

}

}