Державний вищий навчальний заклад

«Прикарпатський національний університет імені Василя Стефаника» Кафедра комп’ютерних наук та інформаційних систем

**ЛАБОРАТОРНА РОБОТА №6**

з предмету «Програмування С#»

# Тема: «Тетріс»

Виконав:

студент групи КН-2

Гриньків В.І.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2020р.

к.т.н., доц. Ровінський В.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2020р.

Івано-Франківськ – 2020

Мета: Реалізація логічного алгоритму середнього рівня складності.

Код програми:

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.IO;

class Shape

{

public int x;

public int y;

public int[,] myfigure;

public int myfigureSize;

public int[,] figure1 = new int[4, 4]

{

{0,0,1,0},

{0,0,1,0},

{0,0,1,0},

{0,0,1,0},

};

public int[,] figure2 = new int[3, 3]

{

{0,2,0},

{0,2,2},

{0,0,2},

};

public int[,] figure3 = new int[3, 3]

{

{0,0,0},

{3,3,3},

{0,3,0},

};

public int[,] figure4 = new int[3, 3]

{

{4,0,0},

{4,0,0},

{4,4,0},

};

public int[,] figure5 = new int[2, 2]

{

{5,5},

{5,5},

};

public int[,] figure6 = new int[3, 3]

{

{6,6,0},

{6,0,0},

{6,0,0},

};

public int[,] figure7 = new int[3, 3]

{

{0,7,0},

{7,7,0},

{7,0,0},

};

public Shape(int \_x, int \_y)

{

x = \_x;

y = \_y;

myfigure = GenerateFigure();

}

public int [,] GenerateFigure()

{

int[,] \_matrix = figure1;

Random r = new Random();

myfigureSize = 3;

switch(r.Next(1,8))

{

case 1:

{

myfigureSize = 4;

\_matrix = figure1;

break;

}

case 2:

{

\_matrix = figure2;

break;

}

case 3:

{

\_matrix = figure3;

break;

}

case 4:

{

\_matrix = figure4;

break;

}

case 5:

{

myfigureSize = 2;

\_matrix = figure5;

break;

}

case 6:

{

\_matrix = figure6;

break;

}

case 7:

{

\_matrix = figure7;

break;

}

}

return \_matrix;

}

public void povorot()

{

//Код з нету

int[,] tempMatrix = new int[myfigureSize, myfigureSize];

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

{

for (int j = 0; j < myfigureSize; j++)

{

tempMatrix[i, j] = myfigure[j, (myfigureSize - 1) - i];

}

}

myfigure = tempMatrix;

int offset1 = (10 - (x + myfigureSize));

if (offset1 < 0)

{

for (int i = 0; i < Math.Abs(offset1); i++)

MoveLeft();

}

if (x < 0)

{

for (int i = 0; i < Math.Abs(x) + 1; i++)

MoveRight();

}

//Код з нету

}

public void MoveDown()

{

y++;

}

public void MoveRight()

{

x++;

}

public void MoveLeft()

{

x--;

}

}

public partial class Form1 : Form

{

Shape figura;

int size = 25;

int game = 0;

int gamespeed;

string path = @"D:\results.txt";

int[,] map = new int[25, 10];

int linesRemoved;

int score;

public Form1()

{

InitializeComponent();

this.KeyUp += new KeyEventHandler(keyFunc);

Init();

}

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

{

}

public void Init()

{

score = 0;

linesRemoved = 0;

figura = new Shape(3, 0);

label1.Text = "Score: " + score;

timer1.Interval = speed(gamespeed); /\*int.Parse(textBox1.Text);\*/

timer1.Tick += new EventHandler(update);

Invalidate();

}

private void keyFunc(object sender, KeyEventArgs e)

{

switch (e.KeyCode)

{

case Keys.M:

{

ResetArea();

clear();

SaveResults();

score = 0;

game = 0;

StartButton.Enabled = true;

PlayerNameBox.Enabled = true;

SlowRadio.Enabled = true;

MediumRadio.Enabled = true;

FastRadio.Enabled = true;

Synchronization();

break;

}

case Keys.N:

{

game = 0;

StartButton.Enabled = true;

break;

}

case Keys.Space:

{

timer1.Interval = 10;

break;

}

case Keys.Up:

{

if (!IsIntersects())

{

ResetArea();

figura.povorot();

Synchronization();

Invalidate();

}

else

{

}

break;

}

case Keys.Right:

{

if (!Colizija\_Horizontal(1))

{

ResetArea();

figura.MoveRight();

Synchronization();

Invalidate();

}

break;

}

case Keys.Left:

{

if (!Colizija\_Horizontal(-1))

{

ResetArea();

figura.MoveLeft();

Synchronization();

Invalidate();

}

break;

}

}

}

private void update(object sender, EventArgs e)

{

if (game == 1)

{

ResetArea();

if (!Colizija())

{

figura.MoveDown();

}

else

{

Synchronization();

DeleteMap();

timer1.Interval = speed(gamespeed);/\*int.Parse(textBox1.Text);\*/

figura = new Shape(3, 0);

if (Colizija())

{

//timer1.Tick -= new EventHandler(update);

//timer1.Stop();

SaveResults();

score = 0;

game = 0;

StartButton.Enabled = true;

PlayerNameBox.Enabled = true;

SlowRadio.Enabled = true;

MediumRadio.Enabled = true;

FastRadio.Enabled = true;

//Init();

}

}

//currentShape.MoveDown();

Synchronization();

Invalidate();

}

}

public void DeleteMap() //Якщо лінія по іксах повна, цей рядок чиститься

{

int temp = 0;

int deletedlines = 0;

//перевіряє поле

for(int i =0; i<25;i++)

{

temp = 0;

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

{

if (map[i, j] != 0)

{

temp++;

}

}

//якщо рядок повний

if(temp == 10)

{

deletedlines++;

for(int h = i; h>=1; h--)

{

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

{

//верхній рядок стає нижнім

map[h, t] = map[h-1, t];

}

}

}

}

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

{

score += 10;

}

label1.Text = "Score: " + score;

}

public void Synchronization() //Cинхронізує карту на формі і фігуру з класу

{

for (int i = figura.y; i < figura.y + figura.myfigureSize; i++)

{

for (int j = figura.x; j < figura.x + figura.myfigureSize; j++)

{

if (figura.myfigure[i - figura.y, j - figura.x] != 0)

map[i, j] = figura.myfigure[i - figura.y, j - figura.x];

}

}

}

public bool Colizija() // Колізія по вертикалі(переміщення фігури)

{

for (int i = figura.y + figura.myfigureSize - 1; i >= figura.y; i--)

{

for (int j = figura.x; j < figura.x + figura.myfigureSize; j++)

{

if (figura.myfigure[i - figura.y, j - figura.x] != 0)

{

if (i + 1 == 25)//дно

{

return true;

}

if (map[i + 1, j] != 0)//низ фігура

{

return true;

}

}

}

}

return false;

}

public bool Colizija\_Horizontal(int check) // Колізія по горизонталі

{

for (int i = figura.y; i < figura.y + figura.myfigureSize; i++)

{

for (int j = figura.x; j < figura.x + figura.myfigureSize; j++)

{

if (figura.myfigure[i - figura.y, j - figura.x] != 0)

{

if (j + 1 \* check > 9 || j + 1 \* check < 0)

return true;

if (map[i, j + 1 \* check] != 0)

{

if (j - figura.x + 1 \* check >= figura.myfigureSize || j - figura.x + 1 \* check < 0)

{

return true;

}

if (figura.myfigure[i - figura.y, j - figura.x + 1 \* check] == 0)

return true;

}

}

}

}

return false;

}

public void ResetArea() //Очищує клітинку, де стояла наша фігура в попередньому ході (з 1 на 0)

{

for (int i = figura.y; i < figura.y + figura.myfigureSize; i++)

{

for (int j = figura.x; j < figura.x + figura.myfigureSize; j++)

{

if (i >= 0 && j >= 0 && i < 25 && j < 10)

{

if (figura.myfigure[i - figura.y, j - figura.x] != 0)

{

map[i, j] = 0;

}

}

}

}

}

public bool IsIntersects()//обмежити поворот між фігурами

{

for (int i = figura.y; i < figura.y + figura.myfigureSize; i++)

{

for (int j = figura.x; j < figura.x + figura.myfigureSize; j++)

{

if (j >= 0 && j <= 9)

{

if (map[i, j] != 0 && figura.myfigure[i - figura.y, j - figura.x] == 0)

return true;

}

}

}

return false;

}

public void DrawMap(Graphics e) // малюю фігуру

{

for (int i = 0; i< 25; i++)

{

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

{

if (map[i, j] == 0)

{

e.FillRectangle(Brushes.LightBlue, new Rectangle(50 + j \* (size) + 1, 50 + i \* (size) + 1, size - 1, size - 1));

}

if (map[i,j] == 1)

{

e.FillRectangle(Brushes.Red, new Rectangle(50 + j \* (size)+1, 50 + i \* (size)+1, size-1, size-1));

}

if (map[i, j] == 2)

{

e.FillRectangle(Brushes.Green, new Rectangle(50 + j \* (size) + 1, 50 + i \* (size) + 1, size - 1, size - 1));

}

if (map[i, j] == 3)

{

e.FillRectangle(Brushes.Yellow, new Rectangle(50 + j \* (size) + 1, 50 + i \* (size) + 1, size - 1, size - 1));

}

if (map[i, j] == 4)

{

e.FillRectangle(Brushes.HotPink, new Rectangle(50 + j \* (size) + 1, 50 + i \* (size) + 1, size - 1, size - 1));

}

if (map[i, j] == 5)

{

e.FillRectangle(Brushes.Purple, new Rectangle(50 + j \* (size) + 1, 50 + i \* (size) + 1, size - 1, size - 1));

}

if (map[i, j] == 6)

{

e.FillRectangle(Brushes.GreenYellow, new Rectangle(50 + j \* (size) + 1, 50 + i \* (size) + 1, size - 1, size - 1));

}

if (map[i, j] == 7)

{

e.FillRectangle(Brushes.Orange, new Rectangle(50 + j \* (size) + 1, 50 + i \* (size) + 1, size - 1, size - 1));

}

}

}

}

public void DrawArea(Graphics g) //Малює лінії

{

for(int i = 0; i<= 25;i++)

{

g.DrawLine(Pens.Black, new Point(50, 50+i\*size), new Point(50+10\*25, 50+i\*size));

}

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

{

g.DrawLine(Pens.Black, new Point(50 + i \* size, 50), new Point(50 + i \* size, 50 + 25 \* size));

}

}

public void SaveResults()

{

try

{

using(StreamWriter sw = new StreamWriter( path, true ))

{

if(score != 0)

sw.Write(PlayerNameBox.Text + ": " + score + " ");

}

}

catch

{

}

}

private void OnPaint(object sender, PaintEventArgs e)

{

DrawArea(e.Graphics);

DrawMap(e.Graphics);

}

void clear()

{

for (int i = 0; i < 25; i++)

{

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

{

map[i, j] = 0;

}

}

}

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

{

clear();

game = 1;

StartButton.Enabled = false;

SlowRadio.Enabled = false;

MediumRadio.Enabled = false;

FastRadio.Enabled = false;

PlayerNameBox.Enabled = false;

}

int speed(int n) //Мало би давати швидкість, а дає проблеми

{

gamespeed = n;

if(SlowRadio.Checked == true)

{

n = 500;

return n;

}

if (MediumRadio.Checked == true)

{

n = 400;

return n;

}

if (FastRadio.Checked == true)

{

n = 300;

return n;

}

else

{

n = 500;

return n;

}

}

private void SlowRadio\_CheckedChanged(object sender, EventArgs e)

{

textBox1.Text = Convert.ToString(500);

}

private void MediumRadio\_CheckedChanged(object sender, EventArgs e)

{

textBox1.Text = Convert.ToString(400);

}

private void FastRadio\_CheckedChanged(object sender, EventArgs e)

{

textBox1.Text = Convert.ToString(300);

}

private void PlayerNameBox\_TextChanged(object sender, EventArgs e)

{

}

}

Результат виконання:



