Liceul Teoretic ”Aurel Vlaicu”

Snake

- proiect de atestat –

Coordonator: Propunător:

prof. Daniel Popa Lomonar Alexandru

2021

Cuprins

[Snake 1](#_Toc70934849)

[Cuprins 2](#_Toc70934850)

[Motivația alegerii temei 3](#_Toc70934851)

[Structura aplicației 4](#_Toc70934852)

[Utilizare 5](#_Toc70934853)

[Detalii tehnice de implementare 6](#_Toc70934854)

[Resurse hardware și software necesare 10](#_Toc70934855)

[Posibilități de dezvoltare ulterioară 11](#_Toc70934856)

[Bibliografie 12](#_Toc70934857)

Motivația alegerii temei

Am ales să fac acest joc deoarece jocurile video sunt o pasiune de-a mea, iar Snake este un joc clasic cunoscut de multă lume.

Structura aplicației

Aplicația are o singură fereastră. În partea stângă a acestei ferestre se află un picture box care reprezintă zona de joc, iar în dreapta sus se află un label care arată scorul curent.

Pentru datele privind șarpele am folosit o listă care memorează Cercuri (clasă care conține coordonate X și Y), fiecare cerc reprezentând un segment al șarpelui.

Mâncarea pe care trebuie să o colecteze șarpele este de asemenea un Cerc. Poziția acesteia este generată folosind clasa Random.

Setările jocului (viteza, mărimea cercurilor) sunt o clasă separată.

Utilizare

Pentru a controla șarpele jucătorul folosește săgețile. Scopul jocului este de a mânca cât mai multe mere (cercuri roșii). Pentru a mânca un măr șarpele trebuie să-l atingă cu capul (cercul negru). Jocul se termină atunci când capul șarpelui se lovește de propriul corp (cercuri verzi). La terminarea jocului acesta poate fi restartat apasând Enter.

Detalii tehnice de implementare

Am ales să fac jocul în programul Visual Studio, limbajul C# deoarece interfața este ușor de utilizat și intuitivă.

Pentru construirea jocului am folosit programarea orientată pe obiect, creând clase pentru: setările jocului, inputul utilizatorului și cercuri. Cercurile reprezintă segmente din corpul șarpelui și mâncarea pe care acesta trebuie să o colecteze, fiind diferențiate prin culori.

Clasa Setări:

public enum Directii

{

Stanga,

Dreapta,

Sus,

Jos

};

class Setari

{

public static int Latime , Inaltime, Viteza, Scor, Puncte;

public static bool GameOver;

public static Directii directie;

public Setari()

{

Latime = 16;

Inaltime = 16;

Viteza = 20;

Scor = 0;

Puncte = 100;

GameOver = false;

directie = Directii.Jos;

}

}

Am folosit un timer pentru ca șarpele să se miște la un anumit interval de timp.

private void updateSreen(object sender, EventArgs e)

{

if (Setari.GameOver == true)

{

if (Input.KeyPress(Keys.Enter))

{

start();

}

}

else

{

if (Input.KeyPress(Keys.Right) && Setari.directie != Directii.Stanga)

{

Setari.directie = Directii.Dreapta;

}

else if(Input.KeyPress(Keys.Left) && Setari.directie!=Directii.Dreapta)

{

Setari.directie = Directii.Stanga;

}

else if (Input.KeyPress(Keys.Up) && Setari.directie != Directii.Jos)

{

Setari.directie = Directii.Sus;

}

else if (Input.KeyPress(Keys.Down) && Setari.directie != Directii.Sus)

{

Setari.directie = Directii.Jos;

}

muta();

}

pictureBox1.Invalidate();

}

Mișcarea șarpelui este simplă: capul merge în direcția dată de jucător, iar celelalte segmente ale corpului iau poziția segmentului anterior. De asemenea, dacă șarpele iese din zona de joc pe o parte acesta va apărea pe partea opusă. Aici se verifică și coliziunea cu mâncarea și segmente ale corpului.

private void muta()

{

for (int i = Snake.Count - 1; i >= 0; i--)

{

if (i == 0)

{

switch (Setari.directie)

{

case Directii.Dreapta:

Snake[i].x++;

break;

case Directii.Stanga:

Snake[i].x--;

break;

case Directii.Sus:

Snake[i].y--;

break;

case Directii.Jos:

Snake[i].y++;

break;

}

int maxXpos = pictureBox1.Size.Width / Setari.Latime;

int maxYpos = pictureBox1.Size.Height / Setari.Inaltime;

if (Snake[i].x < 0) Snake[i].x = maxXpos;

if (Snake[i].y < 0) Snake[i].y = maxYpos;

if (Snake[i].x > maxXpos) Snake[i].x = 0;

if (Snake[i].y > maxYpos) Snake[i].y = 0;

for (int j = 1; j < Snake.Count; j++)

{

if (Snake[i].x == Snake[j].x && Snake[i].y == Snake[j].y)

{

die();

}

}

if (Snake[0].x == mar.x && Snake[0].y == mar.y)

{

eat();

}

}

else

{

Snake[i].x = Snake[i - 1].x;

Snake[i].y = Snake[i - 1].y;

}

}

}

Pentru a genera poziția mâncării am folosit clasa Random.

private void genMancare()

{

int maxXpos = pictureBox1.Size.Width / Setari.Latime;

int maxYpos = pictureBox1.Size.Height / Setari.Inaltime;

Random rnd = new Random();

mar = new Cerc { x = rnd.Next(0, maxXpos), y = rnd.Next(0, maxYpos) };

}

Pentru reprezentarea grafică a jocului am folosit un picture box care este actualizat la fiecare tic al timerului.

private void updateGame(object sender, PaintEventArgs e)

{

Graphics canvas = e.Graphics;

if (Setari.GameOver == false)

{

Brush snakeColour;

for (int i = 0; i < Snake.Count; i++)

{

if (i == 0)

{

snakeColour = Brushes.Black;

}

else

{

snakeColour = Brushes.Green;

}

canvas.FillEllipse(snakeColour, new Rectangle(Snake[i].x \* Setari.Latime, Snake[i].y \* Setari.Inaltime, Setari.Latime, Setari.Inaltime));

canvas.FillEllipse(Brushes.Red, new Rectangle(mar.x \* Setari.Latime,mar.y \* Setari.Inaltime,Setari.Latime, Setari.Inaltime));

}

}

else

{

string gameOver = "Game Over \n" + "Scor final: " + Setari.Scor + "\n Apasati Enter pentru Restart \n";

label3.Text = gameOver;

label3.Visible = true;

}

}

Resurse hardware și software necesare

Program necesar: Visual Studio

Sistem de operare: Windows 7, 8/8.1, 10

Procesor: Dual Core 3.0 Ghz

Memorie: 4GB

Spațiu pe hard disk: 10MB

Posibilități de dezvoltare ulterioară

Adăugare de obstacole.

Adăugare nivele cu diferite forme de obstacole.

Îmbunătățirea graficii.

Bibliografie

<https://docs.microsoft.com/en-us/dotnet/api/system.random?view=net-5.0>

<https://stackoverflow.com/questions/1835062/drawing-circles-with-system-drawing>