Atestat a competenţelor profesionale

Clasa matematică - informatică

**Varga Szabolcs**

XII. A.

**Tetris**

Szakirányítók

Szélyes Emőke

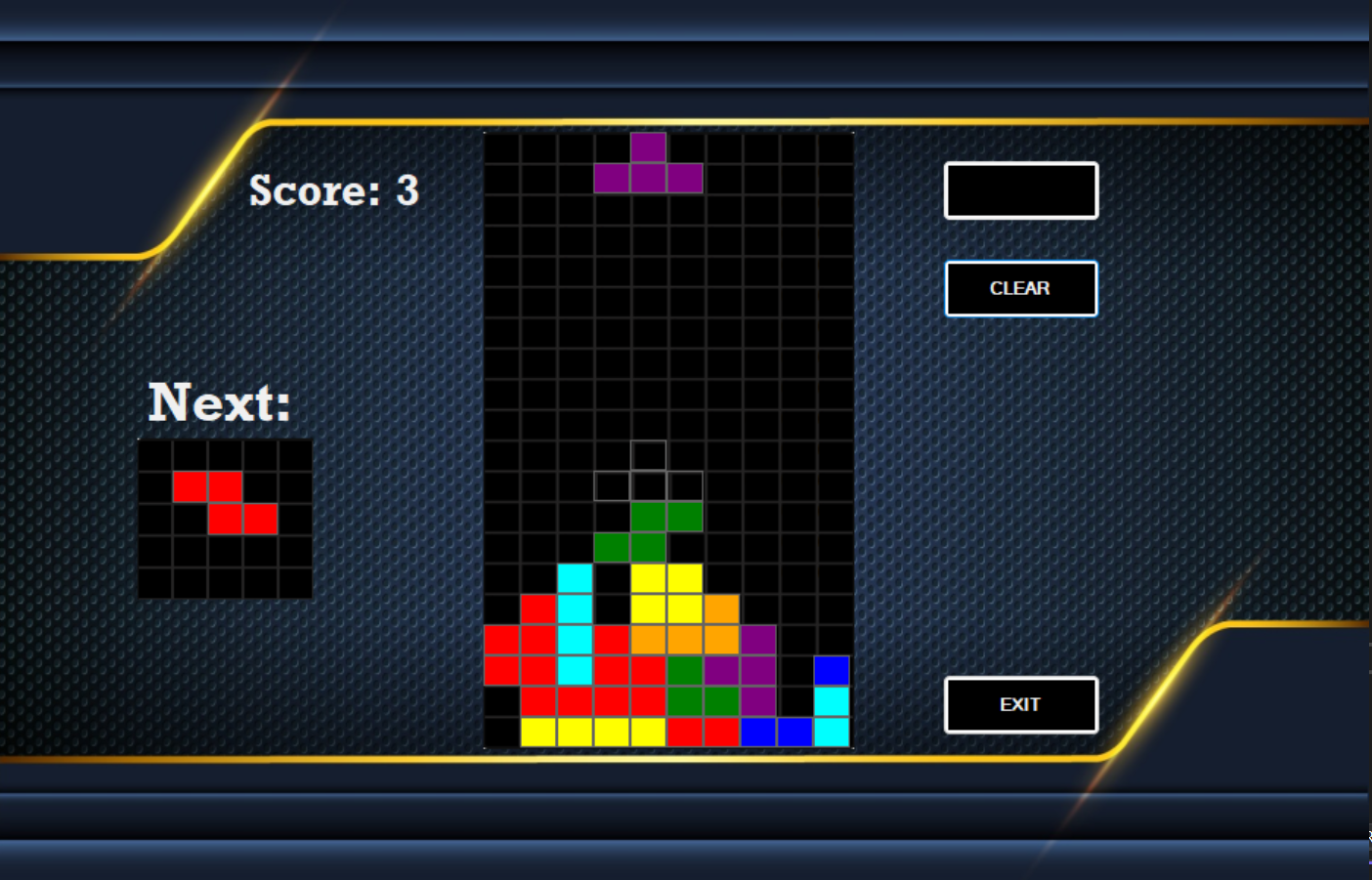
Dénes Ildikó

**- 2023 -**

***Tetris***

Microsoft Visual C# 2022

Ügyességi és logikai videójáték



***Tartalom***

[Ötletforrás és egyéni változtatások 4](#_Toc512968558)

[Játék leírása 5](#_Toc512968560)

[Működési elve 6](#_Toc512968561)

[A játék formjai: 6](#_Toc512968562)

[ Game 6](#_Toc512968563)

[ Menu 13](#_Toc512968564)

[ Settings 14](#_Toc512968565)

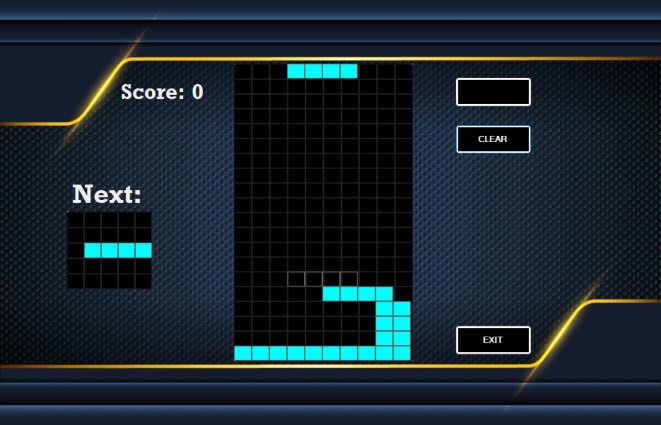
[ End 15](#_Toc512968566)

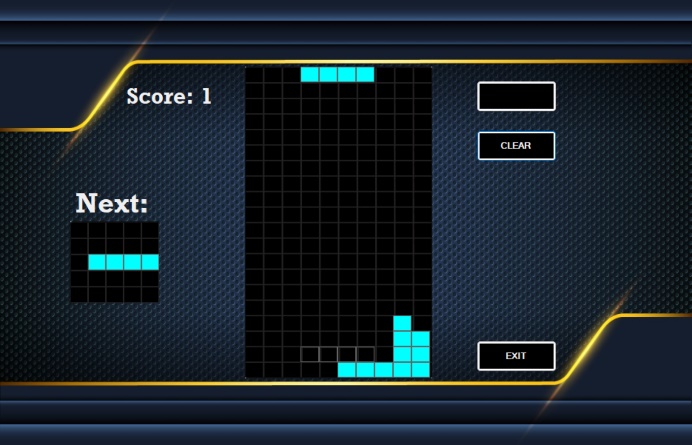
[Fejlesztési lehetőségek 16](#_Toc512968559)

[Végszó 16](#_Toc512968571)

# **Ötletforrás és egyéni változtatások**

* A **Tetris** nevezetű játékot nagyon sokat játszottam gyermekkoromban, a szüleim telefonján, illetve majd később számítógépen is. A tetris az első játékok között volt amit valaha játszottam, azok közül pedig ez maradt meg a legjobban.
* Rengeteg különböző Tetristalálható az interneten, és bár az alap játekmenet ugyanaz, mindegyik picit eltér egymástól. Az általam készített tetris játékmenete is ugyanaz, viszont van egy “gravitáció” beállítási lehetőség, amely bekapcsolásakor, sorkirakás esetén a kockák, a megszokottol eltérően fognak leesni.



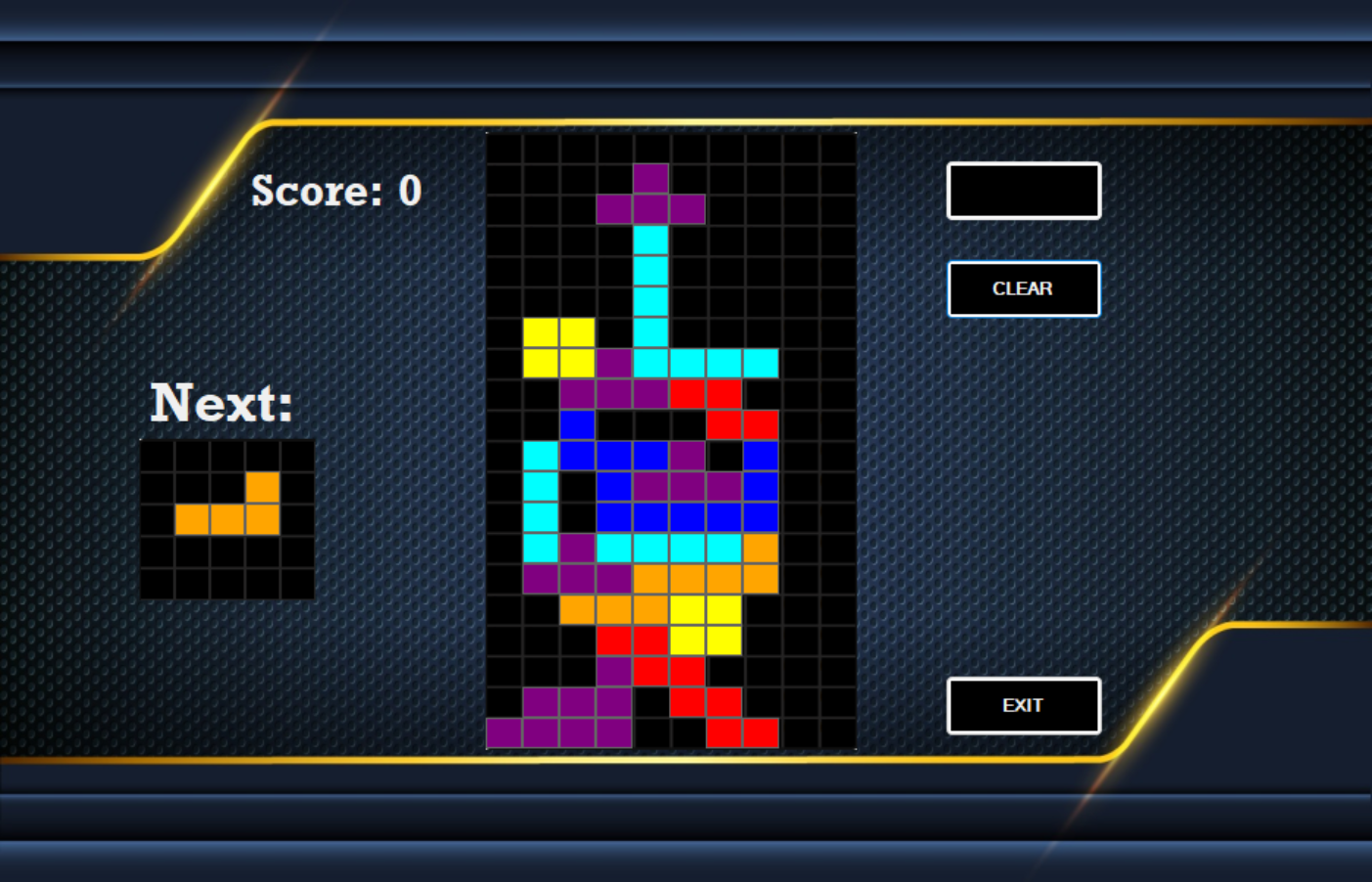


Gravitacióval Gravitació nélkül

# 

# **Játék leírása**

* **A játék kezdete:** A játékos kiválasztja a nehézséget, valamint ki/bekapcsolja a gravitációt és indulhat a játék!
* **A játék célja:** Minél több pont kigyűjtése, és életben maradni ameddig csak lehet
* **Játék vége:** A létrejövő alakzat amennyiben egyből ütközik egy már leesett alakzattal, a játék véget ér



A következő alakzat létrejövetelénél a játék véget ér

# **Működési elve**

A játék 4 formból (Game, Menu, Settings, End) és több fő alprogramból áll a Game formon belül: spawn, ellenoriz, gravitacio stb.

## A game form:

* **button1\_Click(Start gomb)** : létrejön a label mátrix ami maga a pálya, valamint a *kovetkezo* mátrix, melyben a következő alakzat fog kirajzolódni. A játék megkapja a beállításokat(nehézség, gravitáció), és meghívódik a spawn alprogram.

private void button1\_Click(object sender, EventArgs e)

{

button1.Enabled= false;

nehezseg = Settings.nehezseg;

if (nehezseg == 1) { timer1.Interval = 700; }

if (nehezseg == 2) { timer1.Interval = 300; }

if (nehezseg == 3) { timer1.Interval = 100; }

gravitacioertek = Settings.gravitacioertek;

button3.Visible = true;

szam = r.Next(1, 8);

kovszam = r.Next(1, 8);

hossz = panel1.Height / 20;

szel = panel1.Width / 10;

hossz1 = panel2.Height / 5;

szel1 = panel2.Width / 5;

for (int i = 0; i <= 11; i++)

{

for (int j = 0; j <= 21; j++)

{

kockak[i, j] = new Label();

kockak[i, j].Left = (i - 1) \* szel;

kockak[i, j].Top = (j - 1) \* hossz;

kockak[i, j].Width = szel;

kockak[i, j].Height = hossz;

kockak[i, j].BackColor = Color.Transparent;

kockak[i, j].TabIndex = 0;

if (j == 21 || j == 0 || i == 0 || i == 11)

kockak[i, j].TabIndex = 1;

panel1.Controls.Add(kockak[i, j]);

kockak[i, j].BringToFront();

kockak[i, j].Text = kockak[i, j].TabIndex.ToString();

}

}

for (int i = 0; i <= 4; i++)

{

for (int j = 0; j <= 4; j++)

{

kovetkezo[i, j] = new Label();

kovetkezo[i, j].Left = i \* szel1;

kovetkezo[i, j].Top = j \* hossz1;

kovetkezo[i, j].Width = szel1;

kovetkezo[i, j].Height = hossz1;

kovetkezo[i, j].BackColor = Color.Transparent;

kovetkezo[i, j].Visible = false;

kovetkezo[i, j].BorderStyle = BorderStyle.None;

panel2.Controls.Add(kovetkezo[i, j]);

kovetkezo[i, j].BringToFront();

}

}

spawn();

}

* **Spawn:** létrejön az alakzat a *kockak* mátrix tetején, illetve kirajzolódik a következő alakzat is. Elindul a timer1 ami az alakzat soronkénti lehozását hajtja végre, valamint a timer3, ami megmutatja hogy hova fog esni a jelenlegi alakzat. Emellett itt hívódik meg az End form is ha a létrejövő alakzat egyből ütközik egy másikkal.

private void spawn()

{

szam = kovszam;

kovszam = r.Next(1, 8);

for (int i = 1; i <= 4; i++)

{

for (int j = 1; j <= 4; j++)

{

block1[i, j] = new Label();

block1[i, j].Left = (i + 2) \* szel;

block1[i, j].Top = (j - 1) \* hossz;

block1[i, j].Width = szel;

block1[i, j].Height = hossz;

block1[i, j].BackColor = Color.Transparent;

block1[i, j].BorderStyle = BorderStyle.None;

panel1.Controls.Add(block1[i, j]);

}

}

for (int i = 0; i <= 4; i++)

{

for (int j = 0; j <= 4; j++)

{

kovetkezo[i, j].BackColor = Color.Transparent;

kovetkezo[i, j].Visible = false;

kovetkezo[i, j].BorderStyle = BorderStyle.None;

panel2.Controls.Add(kovetkezo[i, j]);

kovetkezo[i, j].BringToFront();

}

}

if (kovszam == 1)

if (kovszam == 2)

if (kovszam == 3)

if (kovszam == 4)

if (kovszam == 5)

if (kovszam == 6)

if (kovszam == 7)

if (szam == 1)

if (szam == 2)

if (szam == 3)

if (szam == 4)

if (szam == 5)

if (szam == 6)

if (szam == 7)

for (int i = 1; i <= 4 && !jatekvege; i++)

{

for (int j = 1; j <= 4 && !jatekvege; j++)

{

if (block[i, j].Visible == true)

{

int x = block[i, j].Left / szel + 1;

int y = block[i, j].Top / hossz + 1;

if (kockak[x, y].TabIndex == 1)

{

timer1.Stop();

timer3.Stop();

jatekvege = true;

Form vege = new End();

vege.Show();

this.Close();

}

}

}

}

if (!jatekvege)

{

timer1.Enabled = true;

timer3.Enabled = true;

}

this.Focus();

}

* A Game form KeyDown eventjében történik meg az alakzatok forgatása, úgy hogy csak akkor forduljon el az alakzat ha ez lehetséges, és nem fog beleütközni semmibe. Pl.:

if (szam == 2)

{

if (e.KeyCode == Keys.Up)

{

if (kettofel == true)

{

int x = block[1, 2].Left / szel + 1;

int x1 = block[2, 2].Left / szel + 1;

int y = block[1, 2].Top / hossz + 1;

int y1 = block[1, 2].Top / hossz + 1;

if (kockak[x, y + 1].TabIndex == 0 && kockak[x1, y1 + 1].TabIndex == 0 && kockak[x, y].TabIndex == 0 && kockak[x1 - 1, y1].TabIndex == 0)

{

for (int i = 1; i <= 4; i++)

{

for (int j = 1; j <= 4; j++)

{

block[i, j].Visible = false;

if ((i == 2 && j < 4) || (i == 1 && j == 3))

{

block[i, j].Visible = true;

}

}

}

kettofel = false;

kettojobb = true;

}

}

else if (kettojobb == true)

{

int x = block[2, 1].Left / szel + 1;

int x1 = block[2, 2].Left / szel + 1;

int y = block[2, 1].Top / hossz + 1;

int y1 = block[2, 2].Top / hossz + 1;

if (x < 11 && kockak[x - 1, y].TabIndex == 0 && ((kockak[x + 1, y].TabIndex == 0 && kockak[x + 1, y + 1].TabIndex == 0) || x == 10))

{ for (int i = 1; i <= 4; i++)

{

for (int j = 1; j <= 4; j++)

{

if (!(kockak[x + 1, y].TabIndex == 0 && kockak[x1 + 1, y1].TabIndex == 0))

block[i, j].Left -= szel;

block[i, j].Visible = false;

if ((j == 1 && i < 4) || (i == 3 && j == 2))

{

block[i, j].Visible = true;

}

}

}

kettojobb = false;

kettole = true;

}

}

else if (kettole == true)

{

int x = block[1, 1].Left / szel + 1;

int y = block[1, 1].Top / hossz + 1;

if (kockak[x, y + 1].TabIndex == 0 && kockak[x, y + 2].TabIndex == 0)

for (int i = 1; i <= 4; i++)

{

for (int j = 1; j <= 4; j++)

{

block[i, j].Visible = false;

if ((i == 1 && j < 4) || (i == 2 && j == 1))

{

block[i, j].Visible = true;

}

}

}

kettole = false;

}

else

{

int x = block[2, 1].Left / szel + 1;

int y = block[2, 1].Top / hossz + 1;

if (x < 11 && kockak[x, y + 1].TabIndex == 0 && (kockak[x + 1, y + 1].TabIndex == 0 || x == 10))

{ for (int i = 1; i <= 4; i++)

{

for (int j = 1; j <= 4; j++)

{

if (kockak[x + 1, y + 1].TabIndex == 1)

block[i, j].Left -= szel;

block[i, j].Visible = false;

if (!(j > 2 || i > 3 || (j == 1 && (i == 2 || i == 3))))

{

block[i, j].Visible = true;

}

}

}

kettofel = true;

}

}

}

* **lemegy**: az alprogram a lefele nyíl megnyomására, a jelenlegi alakzatot soronként hozza le, amíg nem ütközik egy másik alakzattal, vagy amíg az utolsó sorba ér.

private void lemegy()

{

if (timer2.Enabled == false)

{

timer2.Enabled = true;

bool okes = true;

while (okes)

{

for (int i = 1; i <= 4; i++)

{

for (int j = 1; j <= 4; j++)

{

if (block[i, j].Visible == true)

{

int x = block[i, j].Left / szel + 1;

int y = block[i, j].Top / hossz + 1;

/\*label1.Text= x.ToString();

label2.Text= y.ToString();\*/

if (kockak[x, y + 1].TabIndex == 1)

{

okes = false;

for (int k = 1; k <= 4; k++)

{

for (int l = 1; l <= 4; l++)

{

if (block[k, l].Visible == true)

{

x = block[k, l].Left / szel + 1;

y = block[k, l].Top / hossz + 1;

/\*label1.Text = x.ToString();

label2.Text = y.ToString();\*/

kockak[x, y].TabIndex = 1;

kockak[x, y].Text = "";

kockak[x, y].BackColor = block[i, j].BackColor;

kockak[x, y].BorderStyle = BorderStyle.FixedSingle;

kockak[x, y].BringToFront();

}

}

}

}

}

}

}

if (okes)

for (int i = 1; i <= 4; i++)

{

for (int j = 1; j <= 4; j++)

{

block[i, j].Top += hossz;

}

}

}

ellenoriz();

spawn();

}

}

* **ellenoriz**: minden alkalommal ha egy alakzat leért, és jönne a következő, az alprogram ellenőrzi hogy van-e teljes sor, amennyiben van, kitörli ezt, és meghívja azt az alprogramot mely a gravitáció beállításától függően, lehozza a fölötte levő kockákat.

private void ellenoriz()

{

bool cserelt = false;

bool oksor = true;

for (int j = 20; j > 0; j--)

{

oksor = true;

for (int i = 1; i <= 10 && oksor; i++)

{

if (kockak[i, j].TabIndex == 0)

oksor = false;

}

for (int i = 1; i <= 10 && oksor; i++)

{

kockak[i, j].TabIndex = 0;

kockak[i, j].BackColor = Color.Transparent;

kockak[i, j].BorderStyle = BorderStyle.None;

for (int k = j; k > 0; k--)

{

if (kockak[i, k].TabIndex == 1)

{

cserelt = true;

if (gravitacioertek == 1)

gravitacio(i, k);

else

lees(i, k);

}

}

}

if (oksor)

{

pontszam++;

label2.Text = pontszam.ToString();

}

}

if (cserelt)

{

ellenoriz();

}

}

## A Menu form:

A játék elindításakor megjelenő form. Innen lehet elindítani a játékot, vagy a beállításokba lépni.



public partial class Menu : Form

{

public Menu()

{

InitializeComponent();

}

private void button2\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button4\_Click(object sender, EventArgs e)

{

Form jatek = new Game();

jatek.Show();

this.Hide();

}

private void Form2\_Load(object sender, EventArgs e)

{

this.CenterToScreen();

}

private void button1\_Click(object sender, EventArgs e)

{

Form beallitas = new Settings();

beallitas.Show();

this.Hide();

}

}

## A Settings form:

A játék alapbeállításaiért felel. A játékos itt tudja beállítani a nehézséget, valamint a gravitációt is.



public partial class Settings : Form

{

public Settings()

{

InitializeComponent();

}

public static int nehezseg=1;

public static int gravitacioertek=1;

private void Form3\_Load(object sender, EventArgs e)

{

this.CenterToScreen();

}

private void button2\_Click(object sender, EventArgs e)

{

Form menu = new Menu();

menu.Show();

this.Close();

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

if(radioButton1.Checked) { nehezseg = 1; }

else if(radioButton2.Checked) { nehezseg = 2;}

else { nehezseg = 3;}

}

private void radioButton6\_CheckedChanged(object sender, EventArgs e)

{

if(radioButton6.Checked) { gravitacioertek = 1; }

else { gravitacioertek = 2;}

}

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

{

label2.Visible = true;

label3.Visible = true;

}

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

{

label2.Visible=false;

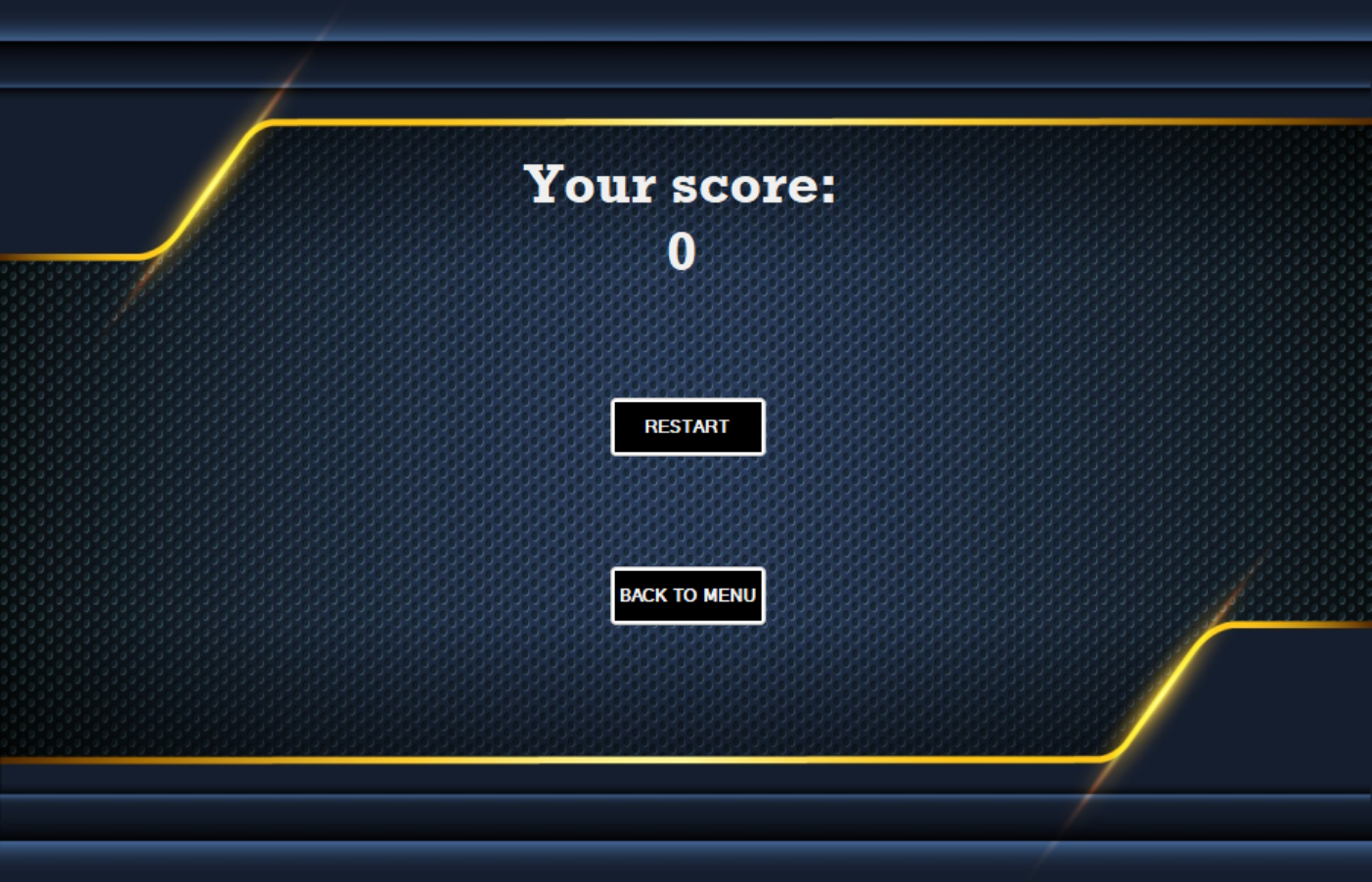
label3.Visible=false;

}

}

## Az End form:

A játék végén megjelenő form, amennyiben a játékos elbukott. A játékosnak 2 lehetősége van: visszalépni a menübe, vagy újra megpróbálni a játékot.



public partial class End : Form

{

public End()

{

InitializeComponent();

}

public static int pontszam;

private void Form4\_Load(object sender, EventArgs e)

{

this.CenterToScreen();

pontszam = Game.pontszam;

label1.Text = pontszam.ToString();

}

private void button1\_Click(object sender, EventArgs e)

{

Form jatek= new Game();

Game.pontszam = 0;

jatek.Show();

this.Close();

}

private void button2\_Click(object sender, EventArgs e)

{

Form menu = new Menu();

menu.Show();

this.Close();

}

}

# **Fejlesztési lehetőségek**

* Egyetlenegy label matrix használata a kockák kirajzolásához
* Az alakzatok középpontjának rögzítése, hogy a forgatás akörül történjen, ne a bal felső sarok szerint
* A következő alakzatot ne kirajzolja, hanem kép legyen
* Az eddigi pontszámok elmentése, illetve ezekből ranglista
* Háttérhangok hozzáadása, mint például háttérzene, valamint hangeffektek sorkirakáskor
* Több alakzat hozzáadása

# **Végszó**

A játék elkészítésében nem voltak nehézségek, viszont idő hiánya miatt több tervezett fejlesztést/javítást is ki kellett hagyjak, de ennek ellenére nagyon élveztem elkészíteni. Remélem a jövőben sor kerül majd arra hogy újra neki fogjak ennek a játéknak, és a lehető legtöbbet hozzam ki belőle.