MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI

INSPECTORATUL ȘCOLAR JUDEȚEAN BIHOR

COLEGIUL NAȚIONAL „ONISIFOR GHIBU” ORADEA

Nr. Întreg. _____/_____

PROIECT PENTRU OBȚINEREA CERTIFICATULUI DE COMPETENȚE PROFESIONALE

-SPECIALIZAREA INFORMATICĂ-

PROFESOR COORDONATOR

Sándor Lukács

ABSOLVENT

Rusu Dana

ORADEA, 2022

MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI

INSPECTORATUL ȘCOLAR JUDEȚEAN BIHOR

COLEGIUL NAȚIONAL „ONISIFOR GHIBU” ORADEA

Nr. Întreg. _____/_____

Tetris

PROFESOR COORDONATOR,

Sándor Lukács

ABSOLVENT,

Rusu Dana

ORADEA, 2022

CONȚINUTUL LUCRĂRII

-PROGRAMARE VIZUALĂ-

I. Motivul alegerii temei

Am ales această temă pentru atestat deoarece, încă de când eram mic, am fost pasionat de lego, iar jocul Tetris îmi amintește de ideea de lego. De asemenea, în alegerea temei am căutat și încercat diferite jocuri, iar Tetris a fost cel care m-a impresionat prin simplitatea sa. Cu toate că regulile sunt nu necesită multă înțelegere, creativitatea jucătorului și strategia sunt cele două puncte cheie ce conduc la atingerea unui scor mare și la finalizarea jocului.

II. Structura generală a aplicației

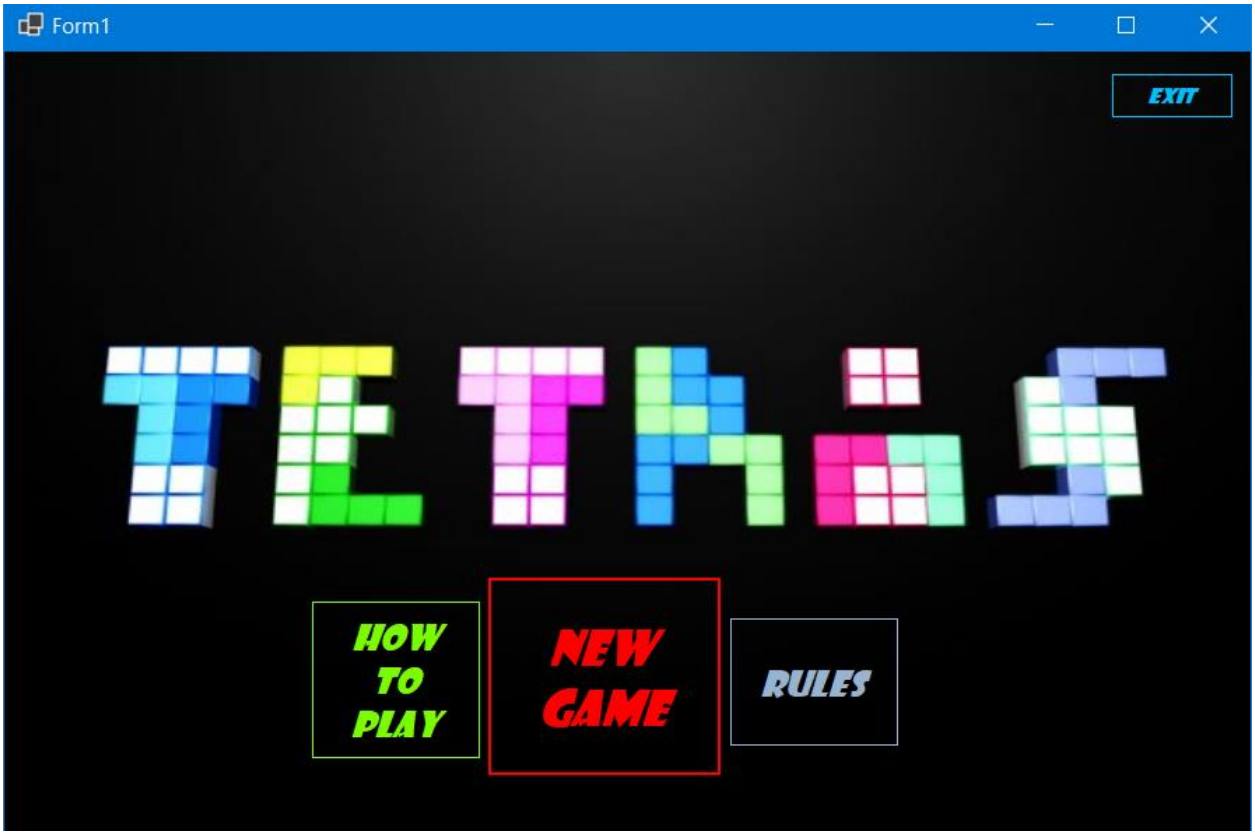


Figura 2.1 Meniu principal

Pentru realizarea acestei aplicații am optat să aleg o structură cât mai simplă și mai user-friendly, meniul principal având doar patru butoane: New Game, Rules și How to Play, precum și un buton de Exit.

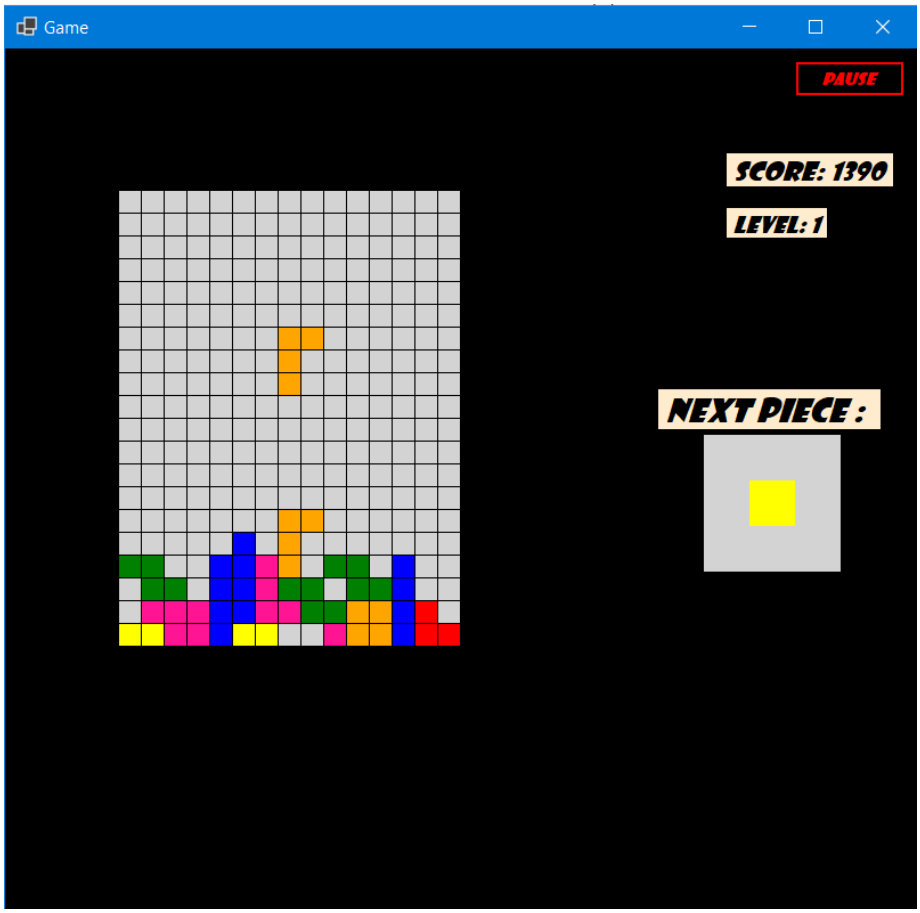


Figura 2.3 Jocul în sine

La apăsarea butonului „New Game”, jocul va porni. Piesele ce trebuie aliniate vor apărea în partea de sus a ecranului și vor coborî treptat, spre a fi aranjate în rânduri complete.

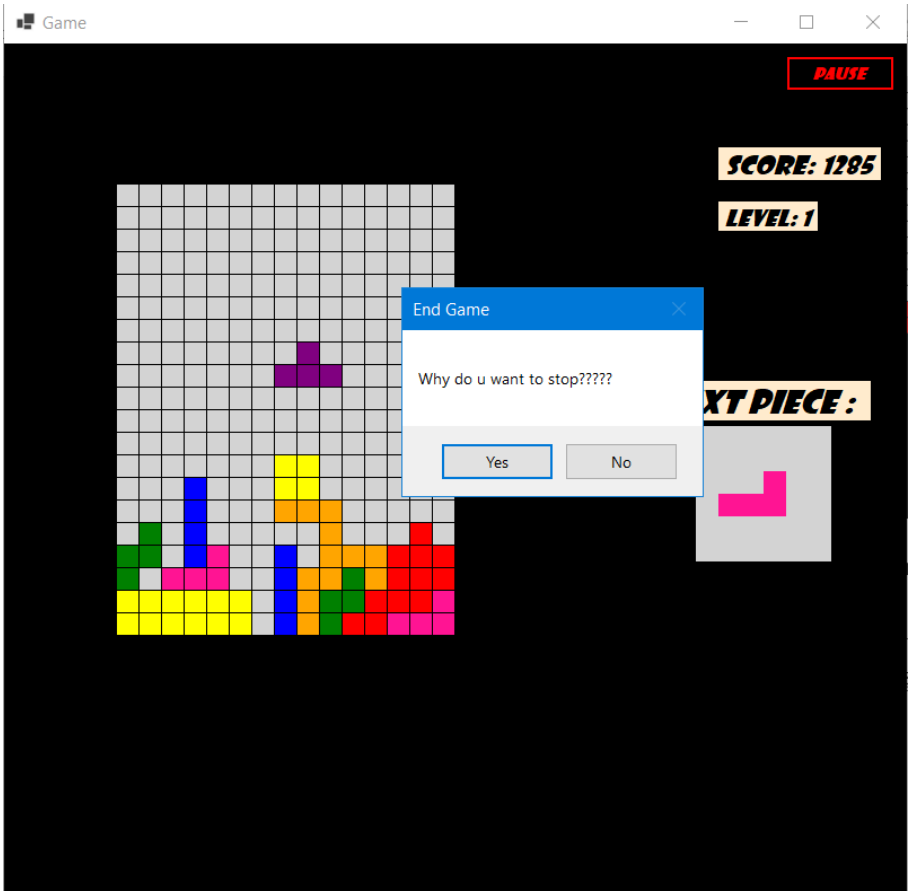


Figura 2.4 Meniu pauză

Când utilizatorul dă click pe butonul ”Pause”, jocul va înghețа, iar pe ecran va apărea un meniu. La apăsarea butonului „Yes”, jocul se va opri definitiv, iar utilizatorul va fi direcționat la meniul principal. La apăsarea butonului „No”, jocul va fi reluat.

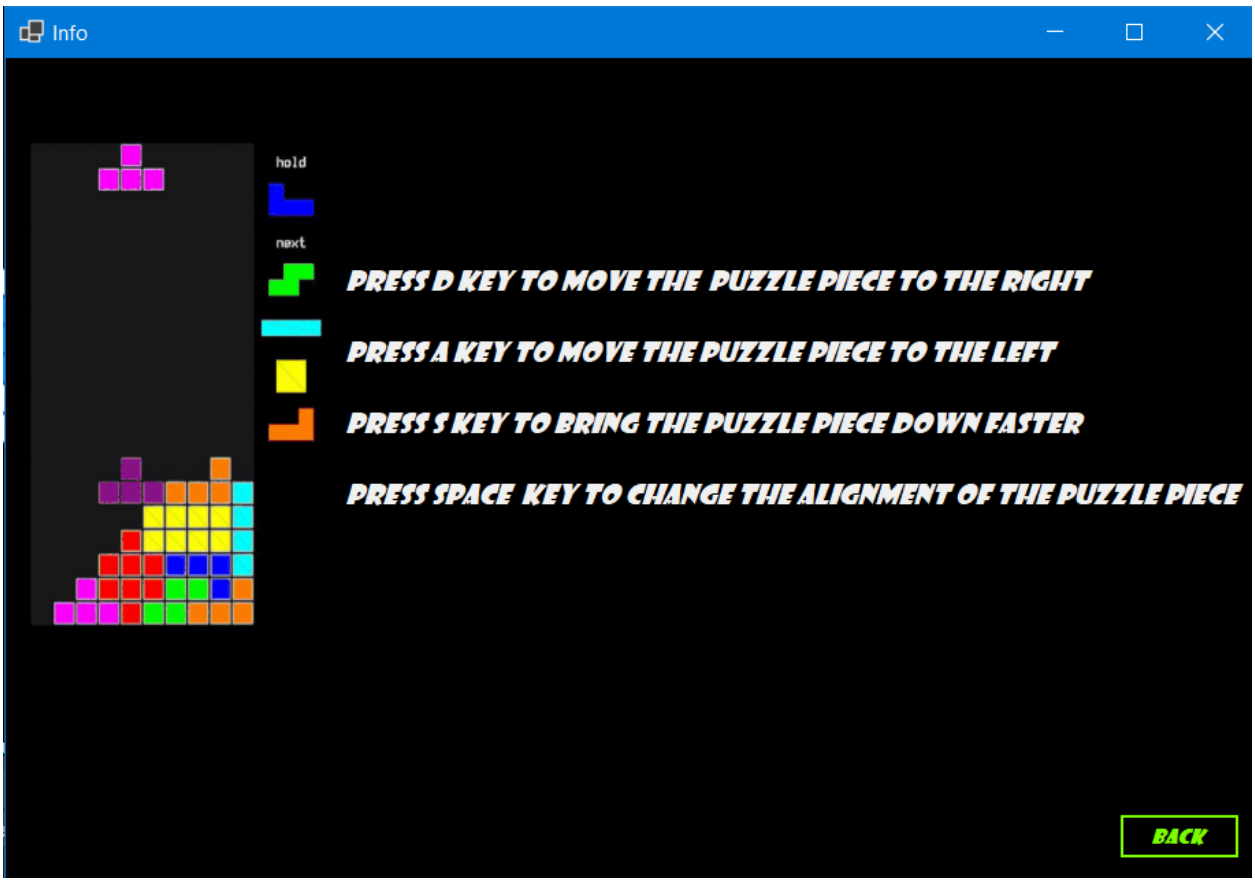


Figura 2.5 How to Play

Când utilizatorul dă click pe butonul ”How to Play” din meniul principal, acesta va fi direcționat către o pagină, unde sunt expuse tastele necesare mutării pieselor și de modificare a poziției piesei.

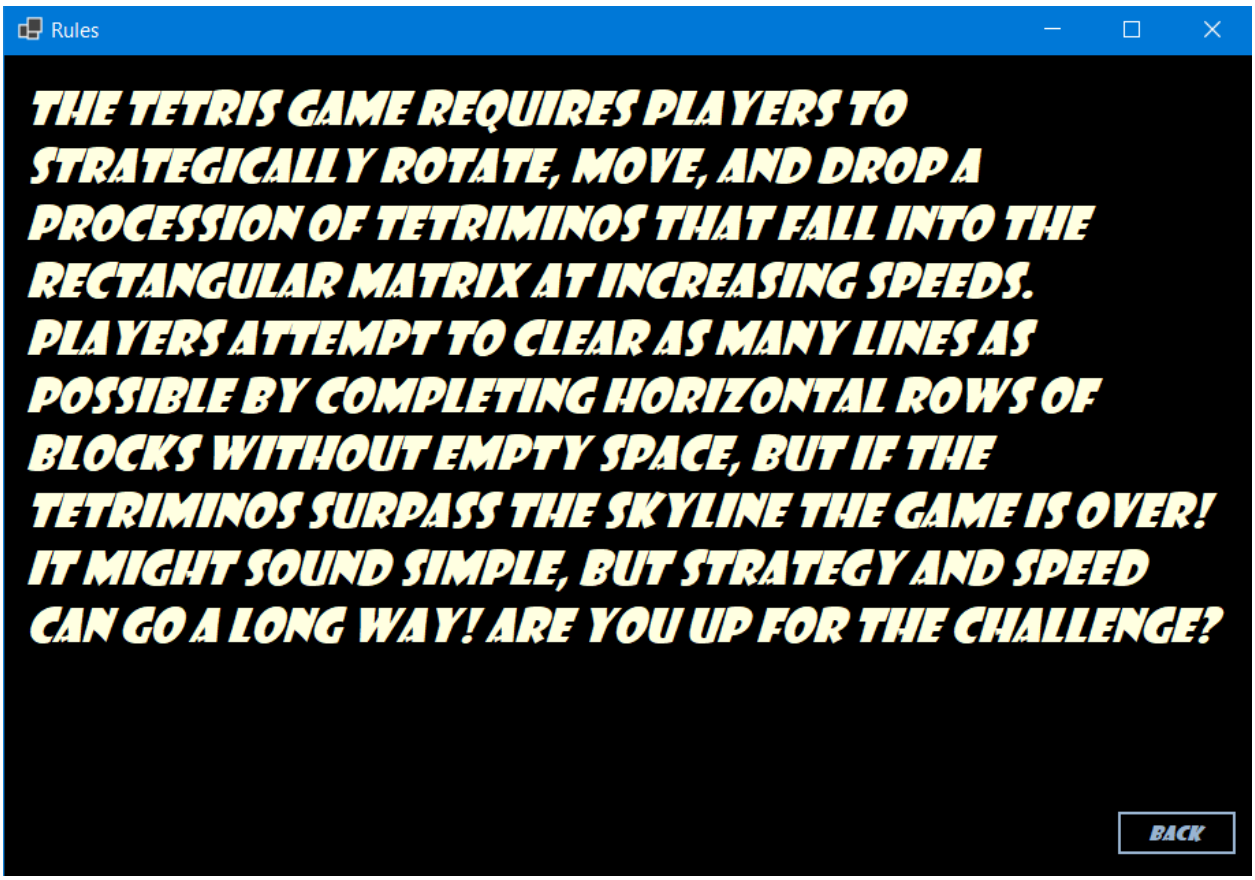


Figura 2.6 Rules

Când este apăsat butonul ”Rules” din meniul principal, utilizatorul va fi direcționat către o pagină, unde sunt prezentate regulile jocului.

III. Descrierea algoritmilor, structurilor de date folosite

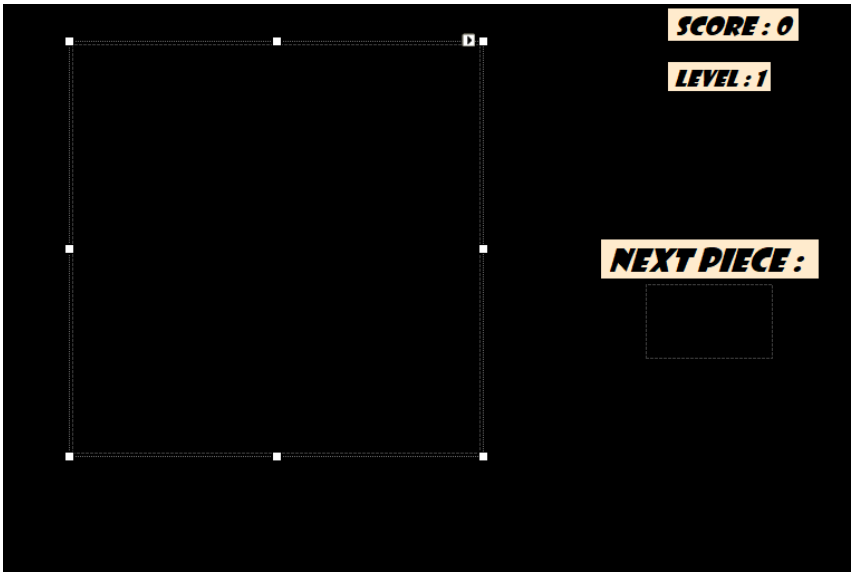


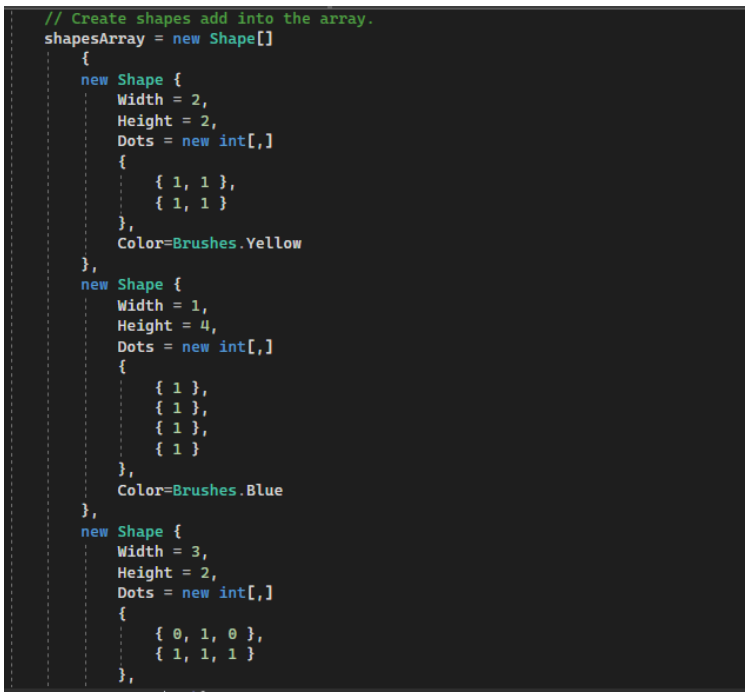
Figura 3.1 pitureBox1

Tabla de joc unde vor apărea piesele este de fapt un pictureBox, care va fi transformat ulterior într-un Canvas.

Întâi am creat tabla de joc.

```
private void loadCanvas()
{
    // Resize the picture box based on the dotsize and canvas size
    pictureBox1.Width = canvasWidth * dotSize;
    pictureBox1.Height = canvasHeight * dotSize;

    // Create Bitmap with picture box's size
    canvasBitmap = new Bitmap(pictureBox1.Width, pictureBox1.Height);

    canvasGraphics = Graphics.FromImage(canvasBitmap);

    canvasGraphics.FillRectangle(Brushes.LightGray, 0, 0, canvasBitmap.Width, canvasBitmap.Height);

    #region Draw Grid
    for (int i = 0; i < canvasHeight; i++)
    {
        canvasGraphics.DrawLine(gridColor, i * dotSize, 0, i * dotSize, canvasHeight * dotSize);
    }

    for (int i = 0; i < canvasHeight; i++)
    {
        canvasGraphics.DrawLine(gridColor, 0, i * dotSize, canvasHeight * dotSize, i * dotSize);
    }
    #endregion

    // Load bitmap into picture box
    pictureBox1.Image = canvasBitmap;

    // Initialize canvas dot array. by default all elements are zero
    canvasDotArray = new int[canvasWidth, canvasHeight];
    canvasBrushArray = new Brush[canvasWidth, canvasHeight];
}
```

Figura 3.2 Formarea tablei de joc



```
// Create shapes add into the array.
shapesArray = new Shape[]
    {
    new Shape {
        Width = 2,
        Height = 2,
        Dots = new int[,]
        {
            { 1, 1 },
            { 1, 1 }
        },
        Color=Brushes.Yellow
    },
    new Shape {
        Width = 1,
        Height = 4,
        Dots = new int[,]
        {
            { 1 },
            { 1 },
            { 1 },
            { 1 }
        },
        Color=Brushes.Blue
    },
    new Shape {
        Width = 3,
        Height = 2,
        Dots = new int[,]
        {
            { 0, 1, 0 },
            { 1, 1, 1 }
        },
```

Figura 3.3 Crearea pieselor

În această funcție am creat piesele care urmează să fie utilizate în joc, prin intermediul unei matrici.

```
    // Get a shape form the array in a random basis
    1 reference
    public static Shape GetRandomShape()
    {
        var shape = shapesArray[new Random().Next(shapesArray.Length)];

        return shape;
    }
}
int currentX;
int currentY;
2 references
private Shape getRandomShapeWithCenterAligned()
{
    var shape = ShapesHandler.GetRandomShape();

    // Calculate the x and y values as if the shape lies in the center
    currentX = 7;
    currentY = -shape.Height;

    return shape;
}
```

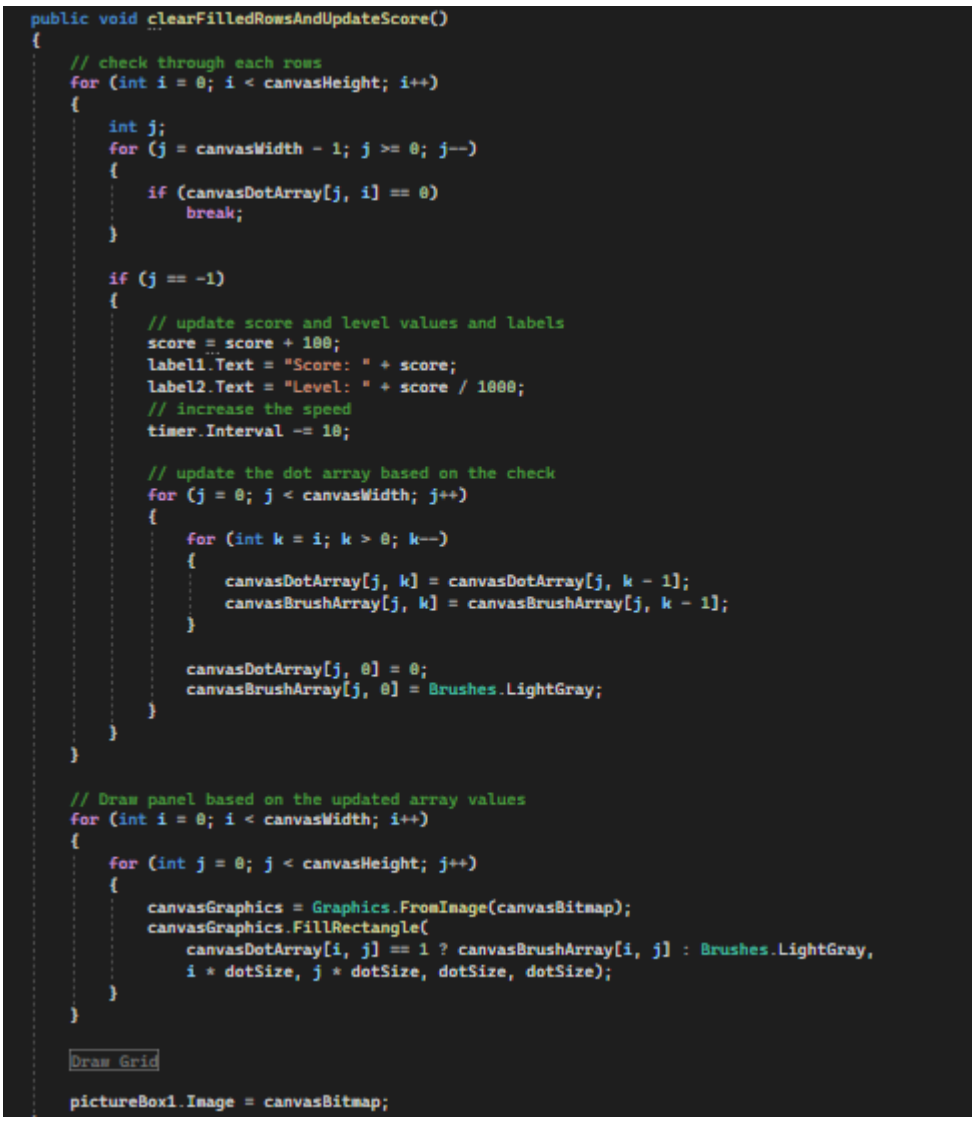
Figura 3.4 Generare piesă aleatorie

Prin această funcție, generez o piesă aleatorie care urmează să apară în mijlocul tablei de joc.

```
protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
{
    var verticalMove = 0;
    var horizontalMove = 0;
    // calculate the vertical and horizontal move values
    // based on the key pressed
    switch (keyData)
    {
        case Keys.Escape:
            timer.Stop();
            string message = "I hope you'll start the game soon";
            string title = "Need a break?";
            MessageBoxButtons buttons = MessageBoxButtons.YesNo;
            DialogResult result = MessageBox.Show(message, title, buttons);
            if (result == DialogResult.Yes)
            {
                this.BackToMainForm.Visible = true;
                this.Close();
            }
            else
            {
                this.Visible = true;
                timer.Start();
            }
            break;

        // move shape left
        case Keys.Left:
            verticalMove--;
            break;

        // move shape right
        case Keys.Right:
            verticalMove++;
            break;

        // move shape down faster
        case Keys.Down:
            pressing_down=true;
            horizontalMove++;
            break;

        // rotate the shape clockwise
        case Keys.Up:
            currentShape.turn();
            break;
        default:
            break;
    }
```

Figura 3.5 Mișcarea pieselor

Prin aceste instrucțiuni, realizez procesarea tastelor, în urma apăsării cărora, piesa va fi mișcată, fie la dreapta, fie la stânga, fie în jos cu viteză sporită.



```
public void clearFilledRowsAndUpdateScore()
{
    // check through each rows
    for (int i = 0; i < canvasHeight; i++)
    {
        int j;
        for (j = canvasWidth - 1; j >= 0; j--)
        {
            if (canvasDotArray[j, i] == 0)
                break;
        }

        if (j == -1)
        {
            // update score and level values and labels
            score = score + 100;
            label1.Text = "Score: " + score;
            label2.Text = "Level: " + score / 1000;
            // increase the speed
            timer.Interval -= 10;

            // update the dot array based on the check
            for (j = 0; j < canvasWidth; j++)
            {
                for (int k = i; k > 0; k--)
                {
                    canvasDotArray[j, k] = canvasDotArray[j, k - 1];
                    canvasBrushArray[j, k] = canvasBrushArray[j, k - 1];
                }

                canvasDotArray[j, 0] = 0;
                canvasBrushArray[j, 0] = Brushes.LightGray;
            }
        }
    }

    // Draw panel based on the updated array values
    for (int i = 0; i < canvasWidth; i++)
    {
        for (int j = 0; j < canvasHeight; j++)
        {
            canvasGraphics = Graphics.FromImage(canvasBitmap);
            canvasGraphics.FillRectangle(
                canvasDotArray[i, j] == 1 ? canvasBrushArray[i, j] : Brushes.LightGray,
                i * dotSize, j * dotSize, dotSize, dotSize);
        }
    }

    Draw Grid

    pictureBox1.Image = canvasBitmap;
```

Figura 3.6

În cele de mai sus, voi verifica dacă în urma mutării efectuate, s-au umplut unul sau mai multe rânduri. În caz afirmativ, piesele de pe rândurile pline vor dispărea, si vor fi înlocuite de piesele de deasupra lor.

```
private Shape getNextShape()
{
    var shape = getRandomShapeWithCenterAligned();

    // Codes to show the next shape in the side panel
    nextShapeBitmap = new Bitmap(6 * dotSize, 6 * dotSize);
    nextShapeGraphics = Graphics.FromImage(nextShapeBitmap);

    nextShapeGraphics.FillRectangle(Brushes.LightGray, 0, 0, nextShapeBitmap.Width, nextShapeBitmap.Height);

    // Find the ideal position for the shape in the side panel
    var startX = (6 - shape.Width) / 2;
    var startY = (6 - shape.Height) / 2;

    for (int i = 0; i < shape.Height; i++)
    {
        for (int j = 0; j < shape.Width; j++)
        {
            nextShapeGraphics.FillRectangle(
                shape.Dots[i, j] == 1 ? shape.Color : Brushes.LightGray,
                (startX + j) * dotSize, (startY + i) * dotSize, dotSize, dotSize);
        }
    }

    pictureBox2.Size = nextShapeBitmap.Size;
    pictureBox2.Image = nextShapeBitmap;

    return shape;
}
```

Figura 3.7 Următoarea piesă

În această funcție, prin intermediul unui pictureBox, afișez în partea dreaptă a tablei de joc, piesa care urmează să apară pe tabla de joc, după poziționarea piesei curente

IV. Detalii tehnice de implementare

Pentru implementare am folosit limbajul C# și am lucrat în Visual Studio.

V. Cerințe soft și hard

Sistem de operare: Windows XP/7/8/10

Procesor: 1000 MHz (minim)

RAM: 1 GB (minim)

VI. Posibilități de dezvoltare

Sunt mai multe posibile dezvoltări ale acestei aplicații: spre exemplu, design-ul pieselor și al tablei de joc ar putea fi îmbunătățite, iar în mișcarea pieselor cât și în dispariția pieselor de pe rândurile pline ar putea fi utilizate animații. De asemenea, ar putea fi implementat un algoritm mai compex cu privire la generarea aleatorie a pieselor, întrucât în algoritmul meu am utilizat doar o funcție predefinită.

VII. Concluzii

În urma realizării acestei lucrări de atestat pot afirma că am dobândit numeroase cunoștințe în ceea ce privește utilizarea limbajului de programeare C#, cât și o mai bună înțelegere a algoritmicii în general. Am descoperit că, în spatele jocurilor pe calculator, pe care le consideram banale când vine vorba doar a le juca, se află ore de trudă și numeroase eșecuri. De asemenea, am avut ocazia de a lăsa frâu liber imaginației și creativității în momentul creării interfaței.