

# Proiect Atestat

Nume Proiect: Apararea Castelului

Profesor coordonator: Aldea Cristina

Realizat de: Sorohan Serban

Clasa: XII A

Liceu: Alexandru Ioan Cuza

Realizat in: Visual Studio 2015

Limbaj de programare utilizat: C# (C Sharp)

Aplicatia consta intr-un joc de tip "Tower Defense", in care inamicii, in acest caz reprezentati de buline albastre, vor sa ajunga in baza jucatorului. Jucatorul, la randul sau, poate incerca sa impiedice inamicii prin intermediul unor turnuri plasate strategic.



Cele doua butoane de pe prima forma:



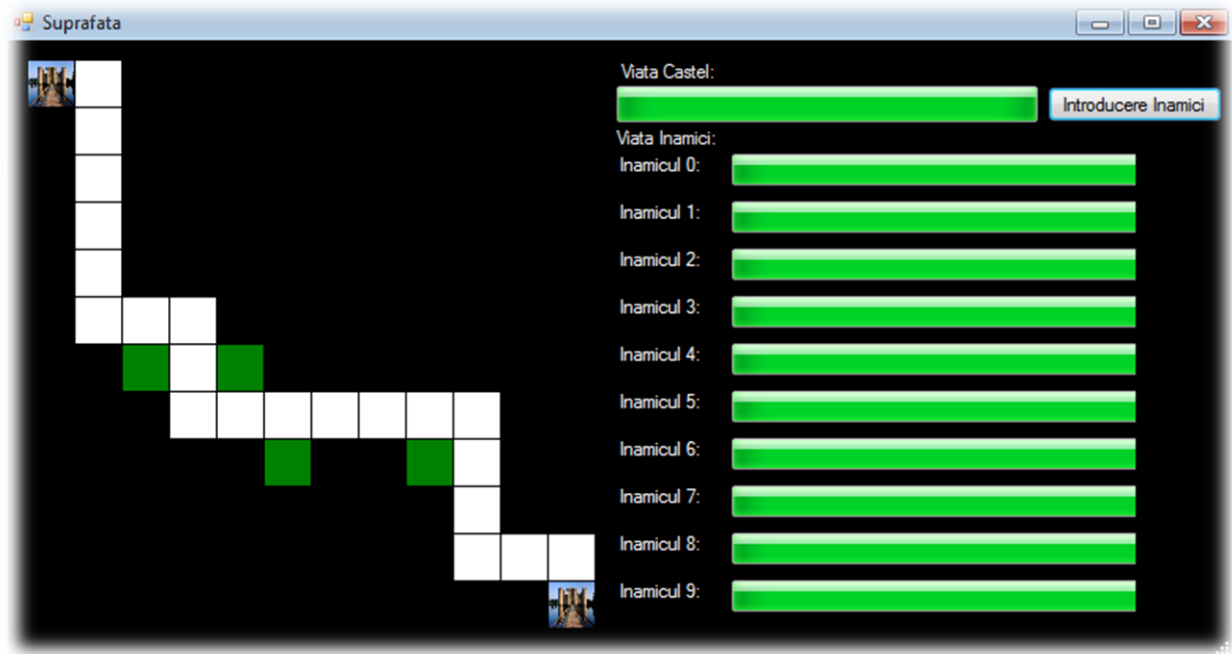
button1 reprezinta "Start", iar button2 "About"

```
private void button1_Click(object sender, EventArgs e)
{
    //Aprinde fereastra de joc
    Suprafata forma = new Suprafata();
    forma.ShowDialog();
    this.Close();
}

private void Form1_Load(object sender, EventArgs e)
{
}

private void button2_Click(object sender, EventArgs e)
{
    //Pentru butonul "About"
    if (ok == 1)
    {
        label1.Show();
        ok *= -1;
    }
    else
    {
        label1.Hide();
        ok *= -1;
    }
}
```

Cea de a doua forma:



Toate variabilele folosite in aceasta forma:

```
//Matricea de joc
int[,] matriceSuprafata;

//Pozitia inamicului in structura DrumInamici
List<int> nrPozitieInamic = new List<int>();

//Dimensiunea unui patrat
int dimensiunePatrat;

//Numarul de turnuri si de patrate din care e alcatuit drumul
int nrTurnuri, nrPatrateDrum;

//Numar de linii/coloane
int nrLinii;

//Inamici
List<Inamici> inamic = new List<Inamici>();
int nrInamici = 10;

//Turnuri + nr curent de turnuri
TurnTier1[] turn;
int nrTurn;

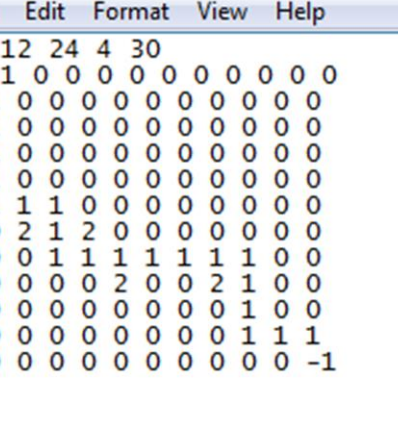
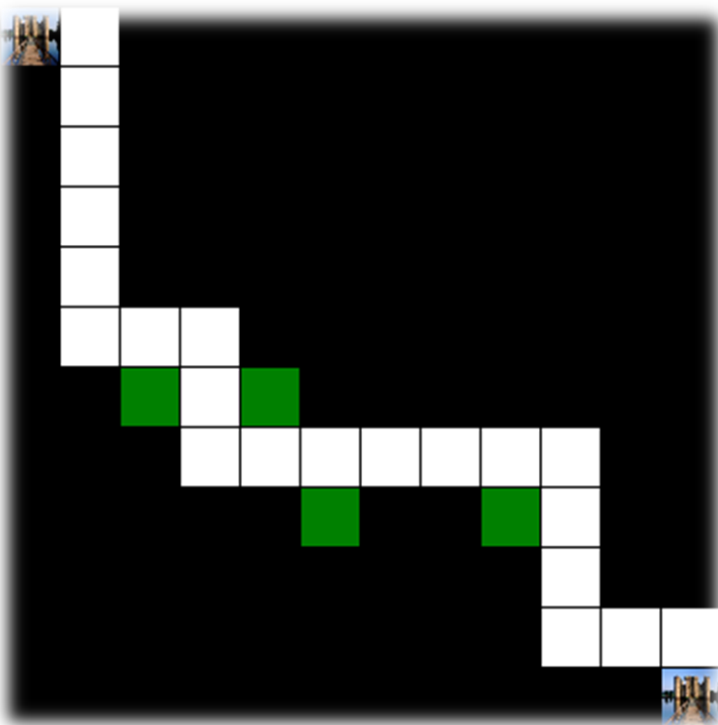
//pozitia pe axa Ox si Oy a drumului inamicilor + pozitia acestora la un moment de timp
struct DrumInamici{...}
DrumInamici[] drumInamici;

//pozitie pe axa Ox si Oy a turnurilor
struct PozitieTurnuri{...}
PozitieTurnuri[] pozitieTurnuri;

//Bara hp inamici
List<ProgressBar> baraHP = new List<ProgressBar>();
List<Label> numeHPInamic = new List<Label>();

//score
int score = 0;
```

Suprafata propriu-zisa de joc este alcatuita dintr-o matrice patratica situata in fisierul Map.txt



Map.txt - Notepad

File Edit Format View Help

```

12 12 24 4 30
-1 1 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 2 1 2 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 2 0 0 2 1 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 -1

```

Pe prima linie sunt, in aceasta ordine: numarul de linii al suprafetei de joc, numarul de coloane al suprafetei de joc, numarul maxim de patrate care pot fii folosite pentru

deplasarea inamicilor, numarul turnurilor si dimensiunea unui patrat (in pixels). In matricea propriu-zisa, “-1” reprezinta pozitia de incepere a inamicilor, respective pozitia unde trebuie sa ajunga, “1” reprezinta drumul pe care trebuie sa il parcurga, iar “2” reprezinta locatiile unde pot fii plasate turnuri.

Mai jos este codul prin care se citesc datele din fisier:

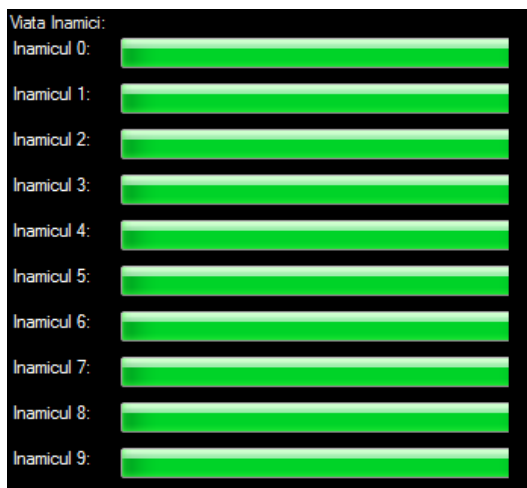
```
private void Initializare_Variabile_Start(StreamReader sr)
{
    string _s = sr.ReadLine();
    string[] _sir = _s.Split(' ');
    matriceSuprafata = new int[int.Parse(_sir[0]), int.Parse(_sir[1])];
    nrPatrateDrum = int.Parse(_sir[2]);
    nrTurnuri = int.Parse(_sir[3]);
    dimensiunePatrat = int.Parse(_sir[4]);
    nrLinii = int.Parse(_sir[0]);
}

private void Initializare_Suprafata_Joc(StreamReader sr)
{
    // Initializare suprafata de joc
    string _s;
    string[] _sir = new string[dimensiunePatrat];
    for (int i = 0; i < nrLinii; i++)
    {
        _s = sr.ReadLine();
        _sir = _s.Split(' ');
        for (int j = 0; j < nrLinii; j++)
        {
            matriceSuprafata[i, j] = int.Parse(_sir[j]);
        }
    }
}
```

Viata castelului care trebuie aparat este contruita prin intermediul unui progressbar:



Viata inamicilor este contruita prin intermediul unei liste de progressbar-uri:



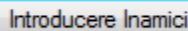
```

private void Initializare_HP_Inamici()
{
    for(int i=0;i<nrInamici;i++)
    {
        baraHP.Add(new ProgressBar());
        baraHP[i].Parent = pictureBox3;
        baraHP[i].Size = new Size(275, 20);
        baraHP[i].Location = new Point(75, i * 30);
        baraHP[i].Maximum = 100;
        baraHP[i].Minimum = 0;
        baraHP[i].Step = -10;
        baraHP[i].Value = 100;

        numeHPInamic.Add(new Label());
        numeHPInamic[i].Parent = pictureBox3;
        numeHPInamic[i].Size = new Size(70, 13);
        numeHPInamic[i].Location = new Point(1,i*30);
        numeHPInamic[i].Text = "Inamicul " + i.ToString() + ":";
        numeHPInamic[i].ForeColor = Color.White;
    }
}

```

Butonul “Introducere Inamici” introduce inamici pe suprafata de joc:



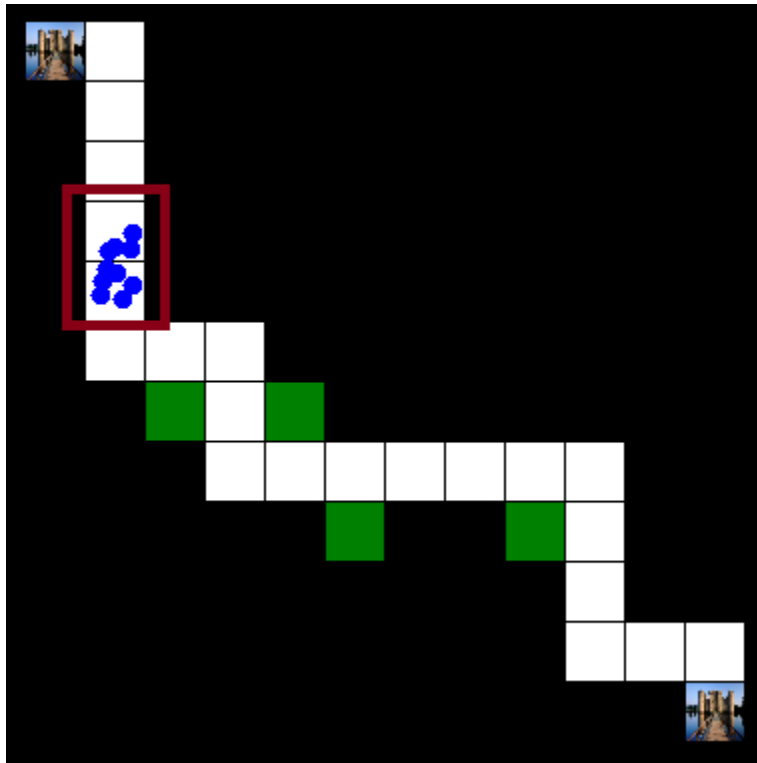
```

private void button1_Click(object sender, EventArgs e)
{
    Random r = new Random();
    int x = r.Next(0, dimensiunePatrat - 10);
    int y = r.Next(0, dimensiunePatrat - 10);
    inamic.Add(new Inamici(drumInamici[0].x + x, drumInamici[0].y + y, 10, 100, x, y, r.Next(1, 4)));
    nrPozitieInamic.Add(1);
    baraHP.Add(new ProgressBar());
    int i = baraHP.Count - 1;
    baraHP[i].Parent = pictureBox3;
    baraHP[i].Size = new Size(275, 20);
    baraHP[i].Location = new Point(75, i * 30);
    baraHP[i].Maximum = 100;
    baraHP[i].Minimum = 0;
    baraHP[i].Step = -2;
    baraHP[i].Value = 100;
    nrInamici++;

    numeHPInamic.Add(new Label());
    numeHPInamic[i].Parent = pictureBox3;
    numeHPInamic[i].Size = new Size(70, 13);
    numeHPInamic[i].Location = new Point(1, i * 30);
    numeHPInamic[i].Text = "Inamicul " + i.ToString() + ":";
    numeHPInamic[i].ForeColor = Color.White;
}

```

Inamicii, initializare si miscarea acestora:



```
private void Initializare_Inamici()
{
    //rx și ry pun o pozitie random in casuta principala
    Random r = new Random();
    for (int i = 0; i < nrInamici; i++)
    {
        int x = r.Next(0, dimensiunePatrat - 10);
        int y = r.Next(0, dimensiunePatrat - 10);
        inamic.Add(new Inamici(drumInamici[0].x + x, drumInamici[0].y + y, 10, 100, x, y, r.Next(1, 4)));
        nrPozitieInamic.Add(1);
    }
}
```

```
private void Schimbare_De_Directie()
{
    //Pentru schimbare de directie
    Random r = new Random();
    for (int i = 0; i < nrInamici; i++)
    {
        if (inamic[i].pozitie.X == drumInamici[nrPozitieInamic[i]].x * dimensiunePatrat + inamic[i].pRx && inamic[i].pozitie.Y == drumInamici[nrPozitieInamic[i]].y * dimensiunePatrat + inamic[i].pRy)
        {
            nrPozitieInamic[i]++;
        }
        //Pentru miscare pe axa Ox
        if (inamic[i].pozitie.X < drumInamici[nrPozitieInamic[i]].x * dimensiunePatrat + inamic[i].pRx)
        {
            inamic[i].pozitie.X += r.Next(1, 4);
        }
        if (inamic[i].pozitie.X > drumInamici[nrPozitieInamic[i]].x * dimensiunePatrat + inamic[i].pRx)
        {
            inamic[i].pozitie.X -= r.Next(1, 4);
        }
    }
}
```

```

//Pentru miscare pe axa Oy
if (inamic[i].pozitie.Y < drumInamici[nrPozitieInamic[i]].y * dimensiunePatrat + inamic[i].pRy)
{
    inamic[i].pozitie.Y += r.Next(1, 4);
}
if (inamic[i].pozitie.Y > drumInamici[nrPozitieInamic[i]].y * dimensiunePatrat + inamic[i].pRy)
{
    inamic[i].pozitie.Y -= r.Next(1, 4);
}
if (inamic[i].pozitie.X / dimensiunePatrat == drumInamici[DrumInamici.lungime].x && inamic[i].pozitie.Y / dimensiunePatrat == drumInamici[DrumInamici.lungime].y)
{
    inamic.Remove(inamic[i]);
    nrInamici--;
    nrPozitieInamic.Remove(nrPozitieInamic[i]);
    baraHP[i].Visible = false;
    baraHP.Remove(baraHP[i]);
    numeHPInamic[i].Visible = false;
    numeHPInamic.Remove(numeHPInamic[i]);
    progressBar1.PerformStep();
    if (progressBar1.Value == 0)
    {
        timer1.Enabled = false;
        timer2.Enabled = false;
        MessageBox.Show("Ai pierdut cu un scor de: " + score);
        this.Close();
    }
    i--;
    refreshHP();
}
}
}
}

```

## Subprogramul de atac al turnurilor si “proiectilele acestora”

```

private void Atac_Turnuri()
{
    for (int i = 0; i < nrTurnuri; i++)
    {
        if (turn[i] != null) //Daca exista turnul verifica coliziune
        {
            //MessageBox.Show((pozitieCurenta[0].y / dimensiunePatrat).ToString());
            for (int z = 0; z < nrInamici; z++)
            {
                int pozX = inamic[z].pozitie.X / dimensiunePatrat;
                int pozY = inamic[z].pozitie.Y / dimensiunePatrat;
                //MessageBox.Show(pozY.ToString() + " " + turn[i].patrate[0].y.ToString());
                if (pozX >= turn[i].patrate[0].x && pozX <= turn[i].patrate[0].x + 2 && pozY >= turn[i].patrate[0].y && pozY <= turn[i].patrate[0].y + 2)
                {
                    for (int j = 0; j < turn[i].nrAcoperire; j++) // Verifica pentru fiecare patrat pe care il are in aria de acoperire Daca contine vreun inamic
                    {
                        if (pozX == turn[i].patrate[j].x && pozY == turn[i].patrate[j].y)
                        {
                            proiectile(pozitieTurnuri[i].x * dimensiunePatrat + dimensiunePatrat / 2, pozitieTurnuri[i].y * dimensiunePatrat + dimensiunePatrat / 2, inamic[z].pozitie.X, inamic[z].pozitie.Y, z);
                            break;
                        }
                    }
                }
            }
        }
    }
}

```



```

private void proiectiles(int x,int y,int destx,int desty,int z)
{
    timer2.Enabled = false;
    Graphics g = pictureBox1.CreateGraphics();
    while (x != destx && y != desty)
    {
        g.FillEllipse(new SolidBrush(Color.Yellow), x, y, 5, 5);
        if(destx>x)
        {
            x++;
        }
        else if(destx<x)
        {
            x--;
        }
        if (desty > y)
        {
            y++;
        }
        else if (destx < y)
        {
            y--;
        }
    }
    baraHP[z].PerformStep();
    if (baraHP[z].Value == 0)
    {
        inamic.Remove(inamic[z]);
        nrPozitieInamic.Remove(nrPozitieInamic[z]);
        baraHP[z].Visible = false;
        baraHP.Remove(baraHP[z]);
        numeHPInamic[z].Visible = false;
        numeHPInamic.Remove(numeHPInamic[z]);
        nrInamici--;
        z--;
        refreshHP();
        score += 10;
    }
    timer2.Enabled = true;
}

```

Aceasta este cea mai importanta parte de cod din proiect. Mai exista cateva subprograme, in sa nu sunt esentiale pentru intelegerea programului in mare.

Bibliografie:

www.stackoverflow.com, www.reddit.com/r/learnprogramming, [www.github.com](http://www.github.com),  
 “Head First C#”

