Joc cu oglinzi

Descrierea programului

Aplicatia este un joc in care jucatorul trebuie sa orienteze oglinzile astfel incat laserul sa ajunga din partea de jos dreapta a jocului in partea de stanga sus. Orientarea oglinzilor se face cu un click pe oglinda care trebuie orientata. Aplicatia genereaza aleatoriu o aranjare a oglinzilor.

Codul sursa

```
namespace MirrorGame
{
   public partial class Form1 : Form
   {
      Bitmap draw = new Bitmap(1000, 1000);
      Graphics g;
      int tileWidth = 60;
      public int[,] mirrors = new int[8, 8];
      public Coords[] dir = new Coords[] { Coords.Left, Coords.Right, Coords.Up, Coords.Down
};
      public int[,] lee = new int[8,8];

      public Form1()
      {
         InitializeComponent();
      }

      public bool CheckBorder(int width, Coords pos)
      {
         if (pos.x >= 0 && pos.y >= 0 && pos.x < width && pos.y < width)
            return true;
         else
            return false;
      }
```

```
public Coords MapIsFull (int width)
{
   for (int i = 0; i < width; i ++)
   {
      for (int j = 0; j < width; j ++)
      {
         if (lee[i, j] == 0)
         {
            for (int k = 0; k < 4; k ++)
            {
               Coords pos = new Coords(i, j) + dir[k];
               if (CheckBorder(width, pos))
               {
                  if (lee[pos.x, pos.y] != 0)
                     return new Coords(i, j);
               }
            }
         }
      }
   }
   return null;
}

public void MakeRandomMap()
{
   int width = 8;

   int[] xd = new int[4] { 1, 0, 0, -1 };
   int[] yd = new int[4] { 0, 1, -1, 0 };
   Coords start = new Coords(width - 1, width - 1);
   lee[start.x, start.y] = 1;
   Random r = new Random();
   #region Lee generation
   do
   {
      Coords pos = start;
      bool ok = true;
      while (ok == true)
      {
         ok = false;
         for (int k = 0; k < 4; k ++)
         {
            Coords nextPos = pos + dir[k];
```

```
            if (CheckBorder(width, nextPos))
            {
                if (lee[nextPos.x, nextPos.y] == 0)
                {
                    ok = true;
                }
            }
        }
        if (ok == true)
        {
            while (true)
            {
                int k = r.Next(0, 4);
                Coords nextPos = pos + dir[k];
                if (CheckBorder(width, nextPos))
                {
                    if (lee[nextPos.x, nextPos.y] == 0)
                    {
                        lee[nextPos.x, nextPos.y] = lee[pos.x, pos.y] + 1;
                        pos = nextPos;
                        break;
                    }
                }
            }
        }
    }
    start = MapIsFull(width);
    if (start != null)
    {
        for (int k = 0; k < 4; k ++)
        {
            Coords nextPos = start + dir[k];
            if (CheckBorder(width, nextPos))
            {
                if (lee[nextPos.x, nextPos.y] != 0)
                {
                    lee[start.x, start.y] = lee[nextPos.x, nextPos.y] + 1;
                    break;
                }
            }
        }
    }
} while (MapIsFull(width) != null);
```

```
            Console.WriteLine("Out");
            #endregion

            Coords end = Coords.Zero;
            start = new Coords(width - 1, width - 1);
            Coords before = Coords.Left;
            Coords after;
            string s = "";
            for (int i = 0; i < 8; i ++)
            {
               s = "";
               for (int j = 0; j < 8; j ++)
               {
                  s += " " + lee[i, j].ToString();
               }
               Console.WriteLine(s);
            }
            while (!Coords.Equal(end,start))
            {
               for (int k = 0; k < 4; k ++)
               {
                  Coords nextPos = end + dir[k];
                  if (CheckBorder(width, nextPos))
                  {
                     if (lee[nextPos.x, nextPos.y] == lee[end.x, end.y] - 1)
                     {
                        after = dir[k];
                        if (Coords.Equal(before + after, Coords.Zero))
                        {
                           before = after * -1;
                           end = nextPos;
                        }
                        else
                        {
                           mirrors[end.x, end.y] = 1;
                           before = after * -1;
                           end = nextPos;
                           Console.WriteLine("end: " + end.x + " " + end.y + " start: " + start.x + " " +
start.y);
                           break;
                        }

                     }
```

```
            }
        }
    }
    after = Coords.Right;
    if (!Coords.Equal(before + after, Coords.Zero))
    {
        mirrors[end.x, end.y] = 1;
    }

}

private void Form1_Load(object sender, EventArgs e)
{
    g = CreateGraphics();
    Random r = new Random();
    MakeRandomMap();
    for (int i = 0; i < 8; i ++)
    {
        for (int j = 0; j < 8; j ++)
        {
            if (mirrors[i,j] != 0)
                DrawHalfSquare(new Coords(i * tileWidth + 10, j * tileWidth + 10), 40,
mirrors[i, j], Color.Red);
        }
    }
    timer1.Start();
}

private void timer1_Tick(object sender, EventArgs e)
{
    g.DrawImage(draw, new Point(0, 0));
    timer1.Stop();
}

private void Form1_MouseDown(object sender, MouseEventArgs e)
{
    for (int i = 0; i < 8; i ++)
    {
        DrawLine(new Coords(0, i * tileWidth + 30), new Coords(490, i * tileWidth + 30),
BackColor);
        DrawLine(new Coords(i * tileWidth + 30, 0), new Coords(i * tileWidth + 30, 490),
BackColor);
    }
```

```
        for (int i = 0; i < 8; i ++)
        {
            for (int j = 0; j < 8; j ++)
            {
                if (mirrors[i,j] != 0)
                {
                    DrawHalfSquare(new Coords(i, j) * tileWidth + Coords.One * 10, 40, mirrors[i,
j], Color.Red);
                }
            }
        }
        Coords mousePos = (Coords)e.Location;
        mirrors[e.Location.X / tileWidth, e.Location.Y / tileWidth]++;
        if (mirrors[e.Location.X / tileWidth, e.Location.Y / tileWidth] == 5)
            mirrors[e.Location.X / tileWidth, e.Location.Y / tileWidth] = 1;
        Console.WriteLine("CurrentRot = " + mirrors[e.Location.X / tileWidth, e.Location.Y /
tileWidth]);


        if (mirrors[7, 7] == 3 || mirrors[7, 7] == 2)
        {
            DrawHalfSquare(new Coords(e.Location.X / tileWidth, e.Location.Y / tileWidth) *
tileWidth + Coords.One * 10, 40, mirrors[e.Location.X / tileWidth, e.Location.Y / tileWidth],
Color.Red);
            DrawLine(new Coords(7 * tileWidth + 30, 7 * tileWidth + 30), new Coords(8 *
tileWidth, 7 * tileWidth + 30), Color.Green);
        }
        else
        {
            DrawLine(new Coords(7 * tileWidth + 30, 7 * tileWidth + 30), new Coords(8 *
tileWidth, 7 * tileWidth + 30), BackColor);
            DrawHalfSquare(new Coords(e.Location.X / tileWidth, e.Location.Y / tileWidth) *
tileWidth + Coords.One * 10, 40, mirrors[e.Location.X / tileWidth, e.Location.Y / tileWidth],
Color.Red);
        }
        Coords pos = new Coords(7, 7);
        Coords dir = Coords.Left;
        while (CheckBorder(8, pos) && !Coords.Equal(dir, Coords.Zero))
        {
            if (mirrors[pos.x,pos.y] != 0)
            {
                switch (mirrors[pos.x, pos.y])
                {
```

```
                case 1:
                    if (dir == Coords.Right)
                        dir = Coords.Down;
                    else if (dir == Coords.Up)
                        dir = Coords.Left;
                    else
                        dir = Coords.Zero;
                    break;
                case 2:
                    if (dir == Coords.Right)
                        dir = Coords.Up;
                    else if (dir == Coords.Down)
                        dir = Coords.Left;
                    else
                        dir = Coords.Zero;
                    break;
                case 3:
                    if (dir == Coords.Left)
                        dir = Coords.Up;
                    else if (dir == Coords.Down)
                        dir = Coords.Right;
                    else
                        dir = Coords.Zero;
                    break;
                case 4:
                    if (dir == Coords.Left)
                        dir = Coords.Down;
                    else if (dir == Coords.Up)
                        dir = Coords.Right;
                    else
                        dir = Coords.Zero;
                    break;
            }
        }
        Coords nextPos = pos + dir;
        if (CheckBorder(8, nextPos))
        {
            DrawLine(new Coords(pos.x * tileWidth + 30, pos.y * tileWidth + 30), new
Coords(nextPos.x * tileWidth + 30, nextPos.y * tileWidth + 30), Color.Green);
        }
        if (Coords.Equal(pos, Coords.Zero))
        {
```

```
                Console.WriteLine("GameOver");
            }
            pos = nextPos;
        }
        if (Coords.Equal(dir, Coords.Zero))
        {
            DrawHalfSquare(pos * tileWidth + Coords.One * 10, 40, mirrors[pos.x, pos.y],
Color.Red);
        }

        g.DrawImage(draw, new Point(0, 0));
    }

    public void DrawSquare(Coords pos, Size size, Color col)
    {
        for (int i = pos.x; i <= pos.x + size.Width; i ++)
        {
            for (int j = pos.y; j < pos.y + size.Height; j ++)
            {
                draw.SetPixel(i, j, col);
            }
        }
    }

    public void UpdateLaser()
    {

    }

    public void DrawHalfSquare(Coords pos, int size, int dir, Color col)
    {
        switch (dir)
        {
            case 1:
                for (int i = pos.x; i <= pos.x + size; i ++)
                {
                    for (int j = pos.y; j <= pos.y + size; j ++)
                    {
                        if (j <= pos.y + i - pos.x)
                            draw.SetPixel(i, j, col);
                        else
                            draw.SetPixel(i, j, BackColor);
                    }
```

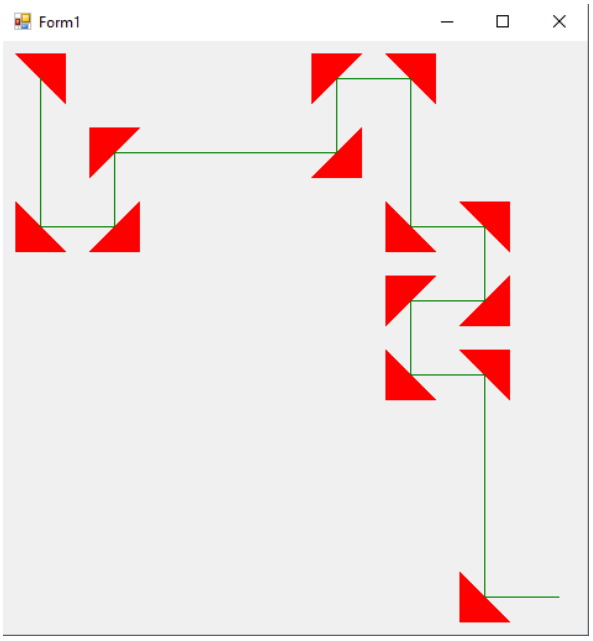
```
    }
    break;
case 3:
    for (int i = pos.x; i <= pos.x + size; i ++)
    {
        for (int j = pos.y + i - pos.x; j <= pos.y + size; j ++)
        {
            draw.SetPixel(i, j, col);
        }
        for (int j = pos.y; j <= pos.y + size; j ++)
        {
            if (j < pos.y + i - pos.x)
                draw.SetPixel(i, j, BackColor);
            else
                draw.SetPixel(i, j, col);
        }
    }
    break;
case 2:
    for (int j = pos.y; j <= pos.y + size; j ++)
    {
        for (int i = pos.x; i <= pos.x + size; i ++)
        {
            if (i + j - pos.x - pos.y >= size)
                draw.SetPixel(i, j, col);
            else
                draw.SetPixel(i, j, BackColor);
        }
    }
    break;
case 4:
    for (int j = pos.y; j <= pos.y + size; j ++)
    {
        for (int i = pos.x; i <= pos.x + size; i ++)
        {
            if (i + j - pos.x - pos.y <= size)
                draw.SetPixel(i, j, col);
            else
                draw.SetPixel(i, j, BackColor);
        }
    }
    break;
default:
```

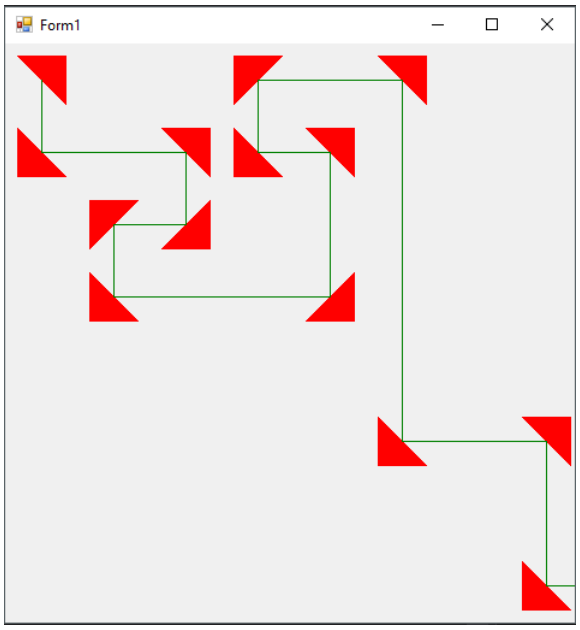
```
            break;
        }

    }
    public void DrawLine(Coords start, Coords stop, Color col)
    {
        double m = 0;
        if (start.x != stop.x)
        {
            m = (double)(start.y - stop.y) / (double)(start.x - stop.x);
        }
        else
        {
            m = 0;
        }
        if (Math.Abs(start.x - stop.x) > Math.Abs(start.y - stop.y))
        {
            for (int i = Math.Min(start.x, stop.x); i <= Math.Max(start.x, stop.x); i++)
            {
                double j = m * (double)(i - start.x) + start.y;
                draw.SetPixel(i, (int)j, col);
            }
        }
        else
        {
            for (int j = Math.Min(start.y, stop.y); j <= Math.Max(start.y, stop.y); j ++)
            {
                double i;
                if (m != 0)
                    i = (j - start.y) / m + start.x;
                else
                    i = start.x;
                draw.SetPixel((int)i, j, col);
            }
        }

}


}
public class Coords
{
    public int x;
```

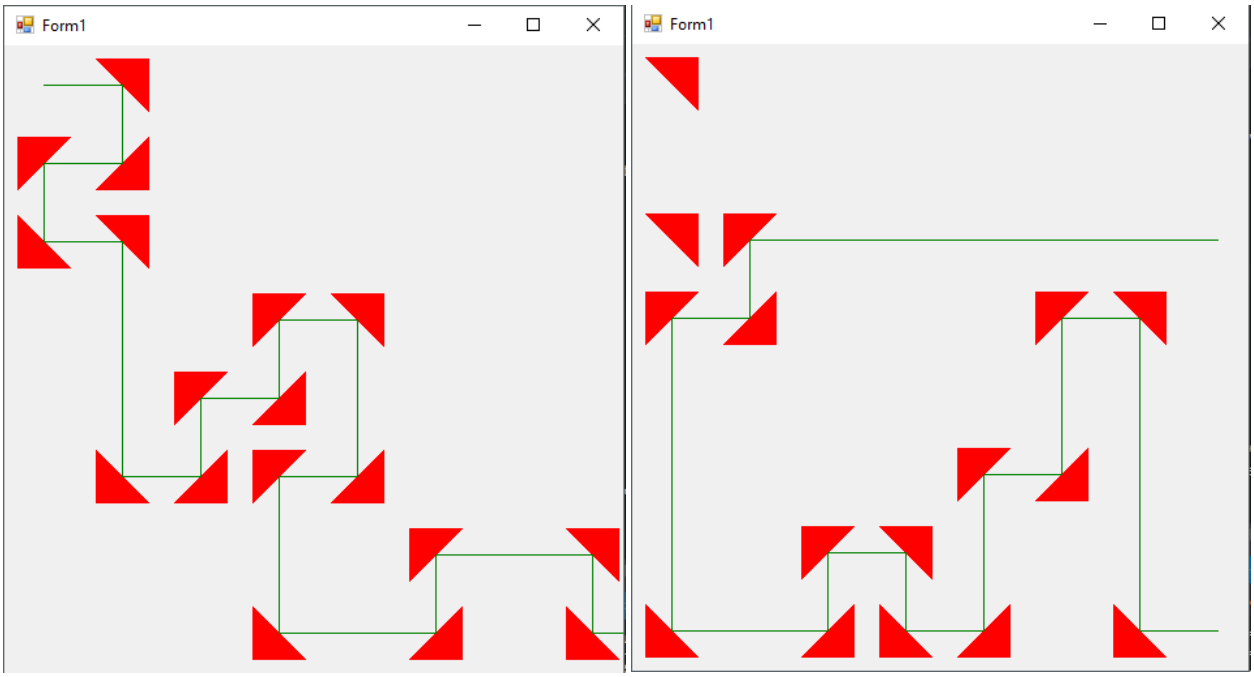
```
public int y;

public static bool Equal(Coords a, Coords b)
{
   if (a != null && b != null)
   {
      if (a.x == b.x && a.y == b.y)
         return true;
      else
         return false;
   }
   else
      return false;
}

public static Coords operator +(Coords a, Coords b)
{
   return new Coords(a.x + b.x, a.y + b.y);
}
public static Coords operator *(Coords p, int scale)
{
   return new Coords(p.x * scale, p.y * scale);
}
public static Coords operator /(Coords p, int scale)
{
   return new Coords(p.x / scale, p.y / scale);
}
public static explicit operator Coords(Point p)
{
   Coords c = new Coords(p.X, p.Y);
   return c;
}
public static Coords Left = new Coords(-1, 0);
public static Coords Right = new Coords(1, 0);
public static Coords Down = new Coords(0, 1);
public static Coords Up = new Coords(0, -1);
public static Coords Zero = new Coords(0, 0);
public static Coords One = new Coords(1, 1);

public Coords(int _x, int _y)
{
   this.x = _x;
   this.y = _y;
```

```
    }
    public Coords(double _x, double _y)
    {
        this.x = (int)_x;
        this.y = (int)_y;
    }

    public static int DistanceBetween(Coords a, Coords b)
    {
        return (int)Math.Sqrt(Math.Pow(a.x - b.x, 2) + Math.Pow(a.y - b.y, 2));
    }
    public Point ToPoint()
    {
        return new Point(this.x, this.y);
    }
}


}
```

Rularea aplicatiei






Form1
Form1