**Practice "256 shades of gray"**

Someone wants to use your geometry library for drawing. To do this, it needs a color for your Segment class. However, it seems to you that it is a bad idea to squeeze colors into a purely geometric entity.

Download the project, set the reference to your library in it, and after that make the GetColor and SetColor methods appear in your Segment class.

If no color is specified, GetColor returns Color.Black.

// Paste the final contents of the SegmentExtensions.cs file here

**Code:**

**Content of SegmentExtensions.cs file:**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using GeometryTasks;

namespace GeometryPainting

{

public static class SegmentExtensions

{

private static Dictionary<Segment, Color> cvet = new Dictionary<Segment, Color>();

public static void SetColor(this Segment s, Color color)

{

if (cvet.ContainsKey(s))

cvet[s] = color;

else

cvet.Add(s, color);

}

public static Color GetColor(this Segment s)

{

if (cvet.ContainsKey(s))

return cvet[s];

else

return Color.Black;

}

}

}

**Content of Program.cs file:**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

using GeometryTasks;

namespace GeometryPainting

{

internal static class Program

{

private static List<Segment> CreateSegments()

{

var result = new List<Segment>();

for (var i = 0; i <= 255; i++)

{

var segment = new Segment

{

Begin = new Vector {X = 0, Y = i},

End = new Vector {X = 255, Y = i}

};

if (i != 0) segment.SetColor(Color.FromArgb(i, i, i));

result.Add(segment);

}

return result;

}

private static void DrawSegments(object sender, PaintEventArgs e)

{

var segments = CreateSegments();

foreach (var segment in segments)

{

Pen pen = null;

pen = new Pen(segment.GetColor());

e.Graphics.DrawLine(pen, (float) segment.Begin.X, (float) segment.Begin.Y, (float) segment.End.X,

(float) segment.End.Y);

}

}

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

private static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

var form = new Form();

form.ClientSize = new Size(255, 255);

form.Paint += DrawSegments;

form.FormBorderStyle = FormBorderStyle.FixedDialog;

form.MaximizeBox = false;

Application.Run(form);

}

}

}

**Content of VectorTask.cs file:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using GeometryTasks;

namespace GeometryTasks

{

public class Vector

{

public double X;

public double Y;

public double GetLength()

{

return Geometry.GetLength(this);

}

public Vector Add(Vector v)

{

return Geometry.Add(this, v);

}

public bool Belongs(Segment s)

{

return Geometry.IsVectorInSegment(this, s);

}

}

public class Segment

{

public Vector Begin;

public Vector End;

public double GetLength()

{

return Geometry.GetLength(this);

}

public bool Contains(Vector v)

{

return Geometry.IsVectorInSegment(v, this);

}

}

public static class Geometry

{

public static Vector

VSum = new Vector { X = 0, Y = 0 };

public static Segment

Sg = new Segment

{

Begin = new Vector { X = 0, Y = 0 },

End = new Vector { X = 1, Y = 1 }

};

public static double DlinaV, S, S1;

public static bool B = false;

public static double GetLength(Vector v1)

{

S = v1.X \* v1.X + v1.Y \* v1.Y;

if (S > 0)

{

return DlinaV = Math.Sqrt(S);

}

else return 0;

}

public static Vector Add(Vector v1, Vector v2)

{

VSum.X = v1.X + v2.X;

VSum.Y = v1.Y + v2.Y;

return VSum;

}

public static double GetLength(Segment sgm)

{

S1 = (sgm.End.X - sgm.Begin.X) \* (sgm.End.X - sgm.Begin.X) +

(sgm.End.Y - sgm.Begin.Y) \* (sgm.End.Y - sgm.Begin.Y);

return Math.Sqrt(S1);

}

public static bool IsVectorInSegment(Vector v, Segment sg)

{

B = ((v.X - sg.Begin.X) \* (v.X - sg.End.X) <= 0) && ((v.Y - sg.Begin.Y) \* (v.Y - sg.End.Y) < 0);

if (((v.X == sg.Begin.X) || (v.X == sg.End.X)) && ((v.Y == sg.End.Y) || (v.Y == sg.Begin.Y)))

return true;

else

return B;

}

}

}