Федеральное агентство по образованию

ФГБОУ ВО Уфимский государственный авиационный технический

университет

Кафедра ВМиК

ЛАБОРАТОРНАЯ РАБОТА №1

По дисциплине: «Инженерная и компьютерная графика»

«Программирование графики с помощью GDI+»

Выполнил: студент группы ПРО-222

Сафронов И.А.

Проверил: Котельников В.А.

Уфа 2021

1. **Цель работы:** Ознакомиться с GDI+
2. **Задачи:**

Открыть файл Glava4. Перевести примеры с VB на С#.

1. **Ход** **работы**

Изображение выглядит как текст

Автоматически созданное описание

Рис. 1 Меню "Фигуры"

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2 Меню "Изображение"

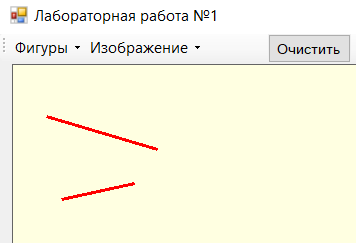


Рис. 3 Линии

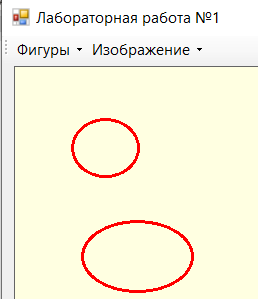


Рис. 4 Эллипс

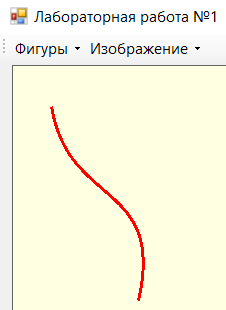


Рис. 5 Кривая Безье



Рис. 6 Сегмент Эллипса

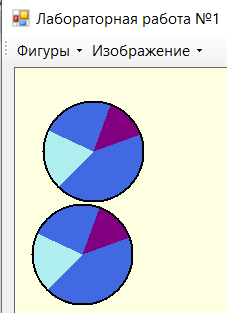


Рис. 7 Диаграммы

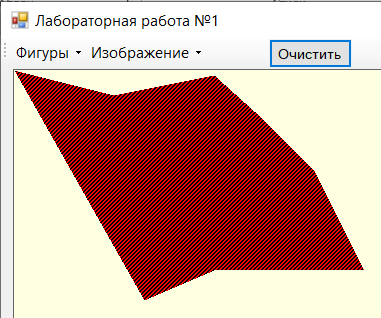


Рис. 8 Полигон



Рис. 9 Заливка пути из двух Полигонов

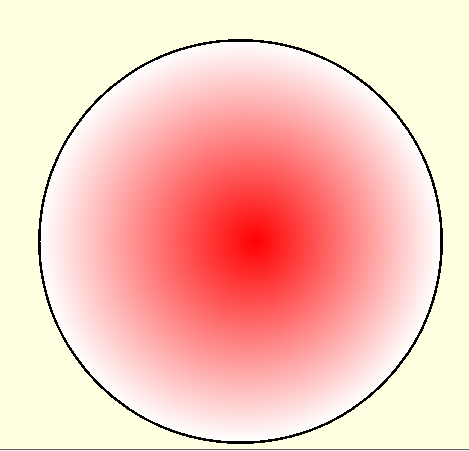


Рис. 10 Градиент



Рис. 11 Градиент из массива точек

Изображение выглядит как еда, шоколад, сладость, кирпич

Автоматически созданное описание

Рис. 12 Текстурная кисть

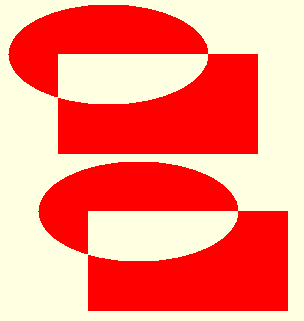


Рис. 13 Вычитания фигур

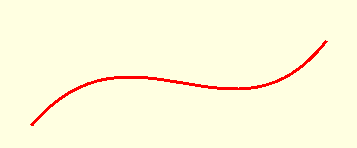


Рис. 14 Поворот Кривой Безье

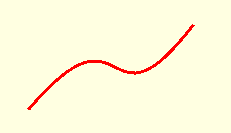


Рис. 15 Поворот Кривой Безье

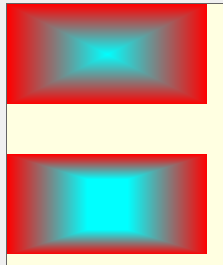


Рис. 16 Градиент вдоль пути

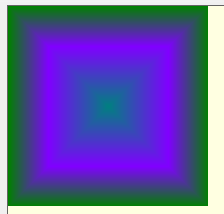


Рис. 17 Интерполяция цвета

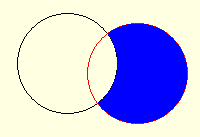


Рис. 18 Комплемент

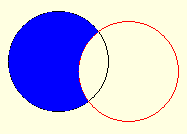


Рис. 19 Исключение

Изображение выглядит как текст, гора, небо, внешний

Автоматически созданное описание

Рис. 20 Изображение

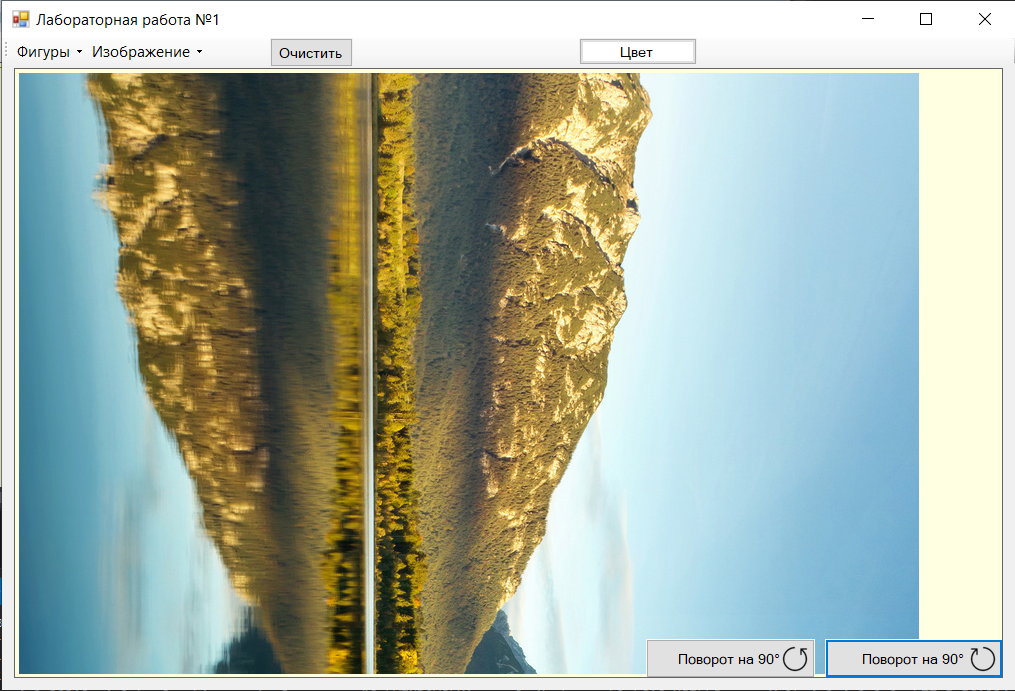


Рис. 21 Поворот изображение на 90 градусов

Изображение выглядит как текст, небо, гора, снимок экрана

Автоматически созданное описание

Рис. 22 Изменение масштаба

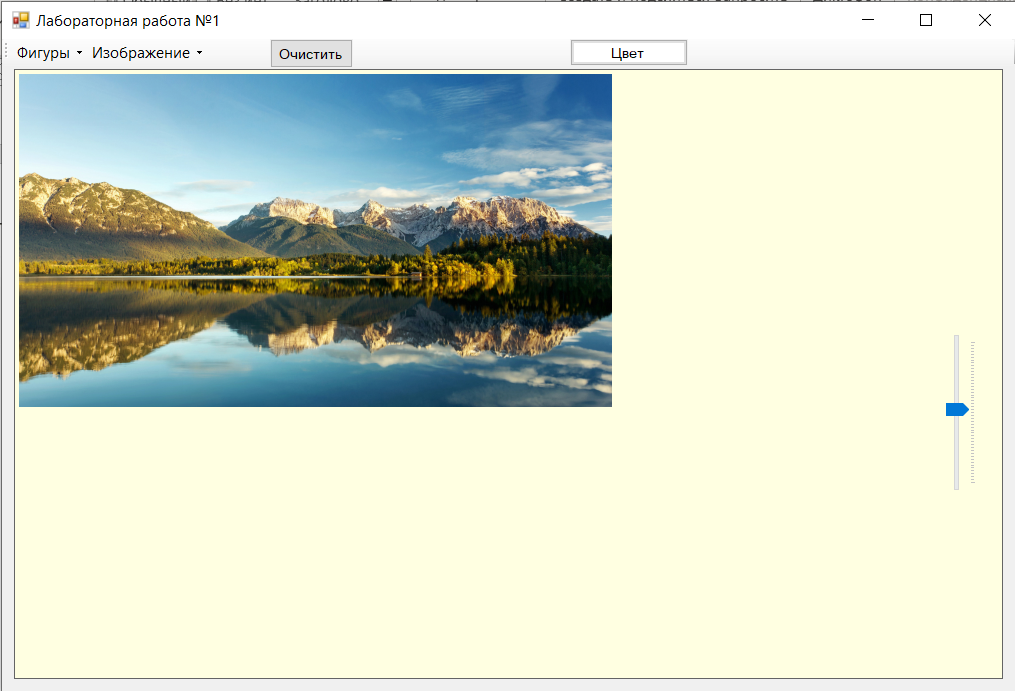


Рис. 23 Изменение масштаба

Изображение выглядит как теннис

Автоматически созданное описание

Рис. 24 Инвертирование цвета: До



Рис. 25 Инвертирования цвета: После

1. **Вывод:**

В рамках лабораторной работы я ознакомился со средствами GDI+ и на практике опробовал некоторые из них в работе.

Приложение 1. Листинг программы:

**MainForm.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace \_1\_laba\_Graphics

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

public int mode = 0;

public int count\_click = 0;

Color color = Color.Red;

List<Point> points = new List<Point>();

List<Shape> storage = new List<Shape>();

Bitmap bitmap1;

private void panel\_drawing\_Paint(object sender, PaintEventArgs e)

{ // Для отрисовки на панели

Graphics graphic = e.Graphics;

foreach(Shape i in storage)

i.Paint(graphic);

}

private void panel\_drawing\_MouseMove(object sender, MouseEventArgs e)

{ // Для отображении линий

if(mode == 1 && count\_click == 1)

{

if (storage.Any())

if (storage.Last() is Line)

storage.RemoveAt(storage.Count - 1);

points.Insert(1, new Point(e.X, e.Y));

storage.Add(new Line(points[0], points[1], color));

panel\_drawing.Refresh();

}

if(mode == 2 && count\_click == 1) // Для отображения кругов

{

if (storage.Any())

if (storage.Last() is Ellipse)

storage.RemoveAt(storage.Count - 1);

storage.Add(new Ellipse(points[0].X, points[0].Y, e.X - points[0].X, e.Y - points[0].Y, color));

panel\_drawing.Refresh();

}

}

private void panel\_drawing\_MouseClick(object sender, MouseEventArgs e)

{ // Добавляем фигуру в хранилище

switch (mode)

{

case 1:

if(count\_click == 0)

{

points.Add(new Point(e.X, e.Y));

count\_click = 1;

}

else

{

points.Add(new Point(e.X, e.Y));

storage.Add(new Line(points[0], points[1], color));

points.Clear();

count\_click = 0;

}

break;

case 2:

if (count\_click == 0)

{

points.Add(new Point(e.X, e.Y));

count\_click = 1;

}

else

{

storage.Add(new Ellipse(points[0].X, points[0].Y, e.X - points[0].X, e.Y - points[0].Y, color));

points.Clear();

count\_click = 0;

}

break;

case 3:

if(count\_click == 3)

{

points.Add(new Point(e.X, e.Y));

storage.Add(new Bezier(points[0], points[1], points[2], points[3], color));

points.Clear();

count\_click = 0;

}

else

{

points.Add(new Point(e.X, e.Y));

count\_click++;

}

break;

case 4:

storage.Add(new Pie(e.X - 50, e.Y - 50, 100, 100, 0.0F, trackBar1.Value, color));

break;

case 5:

storage.Add(new Diagram(e.X - 50, e.Y - 50, 100, 100, 0.0F, 360.0F, color));

break;

case 6:

storage.Add(new Polygon(color));

break;

case 7:

if (count\_click == 7)

{

points.Add(new Point(e.X, e.Y));

storage.Add(new PolygonV2(points, color));

panel\_drawing.Refresh();

points.Clear();

count\_click = 0;

}

else

{

points.Add(new Point(e.X, e.Y));

count\_click++;

}

break;

case 8:

storage.Add(new Gradient(color));

break;

case 9:

storage.Add(new GradientV2());

break;

case 10:

storage.Add(new TextuteKist(e.X-30, e.Y-30));

break;

case 11:

storage.Add(new SubShape(e.X - 50, e.Y - 50, color));

break;

case 13:

storage.Add(new GradientAlongThePath());

break;

case 14:

storage.Add(new Interpolation());

break;

case 15:

storage.Add(new Complement());

break;

case 16:

storage.Add(new Exclude());

break;

}

panel\_drawing.Refresh();

}

private void btn\_select\_color\_Click(object sender, EventArgs e)

{ // Кнопка выбора цвета

if (colorDialog1.ShowDialog() == DialogResult.Cancel)

return;

btn\_select\_color.BackColor = colorDialog1.Color;

color = colorDialog1.Color;

}

private void Line\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Линии

mode = 1;

count\_click = 0;

}

private void Ellips\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Эллипсов

mode = 2;

count\_click = 0;

}

private void Beziers\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима кривая Безье

mode = 3;

count\_click = 0;

}

private void SegmentEllipse\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима сегмента Эллипса

mode = 4;

if (!trackBar1.Visible)

trackBar1.Visible = true;

else

trackBar1.Visible = false;

}

private void Diagram\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Диаграммы

mode = 5;

}

private void Polygon\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Полигон

mode = 6;

}

private void PolygonV2\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Заливка пути из двух полигонов

mode = 7;

count\_click = 0;

}

private void Gradient\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Градиент

mode = 8;

}

private void GradientV2\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Градиент из массива точек

mode = 9;

}

private void TextureKist\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима текстурная кисть

mode = 10;

}

private void SubShape\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Вычитания фигур

mode = 11;

}

private void Selectphoto\_Button\_Click(object sender, EventArgs e)

{ // Выбираем путь к изображению

if (openPicture.ShowDialog() == DialogResult.Cancel)

return;

pictureBox1.ImageLocation = openPicture.FileName;

bitmap1 = (Bitmap)Bitmap.FromFile(openPicture.FileName);

pictureBox1.Visible = true;

}

private void Flip\_btn\_Click(object sender, EventArgs e)

{ // Выбор режима Поворот изображения

if (!flip\_90l.Visible)

{

flip\_90l.Visible = true;

flip\_90r.Visible = true;

}

else

{

flip\_90l.Visible = false;

flip\_90r.Visible = false;

}

}

private void flip\_90l\_Click(object sender, EventArgs e)

{ // Поворачивает выбранное изображение на 90° влево

bitmap1.RotateFlip(RotateFlipType.Rotate90FlipXY);

pictureBox1.Image = bitmap1;

pictureBox1.Refresh();

}

private void flip\_90r\_Click(object sender, EventArgs e)

{ // Поворачивает выбранное изображение на 90° вправо

bitmap1.RotateFlip(RotateFlipType.Rotate90FlipNone);

pictureBox1.Image = bitmap1;

pictureBox1.Refresh();

}

private void Scale\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Масштабирования

if(!trackBar2.Visible)

trackBar2.Visible = true;

else

trackBar2.Visible = false;

}

private void trackBar2\_Scroll(object sender, EventArgs e)

{ // Масштабирование изображения

int Width = Convert.ToInt32(bitmap1.Width / (trackBar2.Value \* 0.1));

int Height = Convert.ToInt32(bitmap1.Height / (trackBar2.Value \* 0.1));

Bitmap temp = new Bitmap(bitmap1, new Size(Width, Height));

pictureBox1.Image = temp;

pictureBox1.Refresh();

}

private void Inver\_Button\_Click(object sender, EventArgs e)

{ // Инвертирование цветов изображения

int x;

int y;

for (x = 0; x < bitmap1.Width; x++)

{

for (y = 0; y < bitmap1.Height; y++)

{

Color oldColor = bitmap1.GetPixel(x, y);

Color newColor;

newColor = Color.FromArgb(oldColor.A, 255 - oldColor.R, 255 - oldColor.G, 255 - oldColor.B);

bitmap1.SetPixel(x, y, newColor);

pictureBox1.Image = bitmap1;

pictureBox1.Refresh();

}

}

}

private void Clear\_Button\_Click(object sender, EventArgs e)

{ // Кнопка очистки

storage.Clear();

panel\_drawing.Refresh();

pictureBox1.Visible = false;

flip\_90l.Visible = false;

flip\_90r.Visible = false;

trackBar1.Visible = false;

trackBar2.Visible = false;

trackBar\_X.Visible = false;

trackBar\_Y.Visible = false;

}

private void FlipBezier\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Поворот кривой Безье

if (!trackBar\_X.Visible)

{

trackBar\_X.Visible = true;

trackBar\_Y.Visible = true;

}

else

{

trackBar\_X.Visible = false;

trackBar\_Y.Visible = false;

}

mode = 12;

points.Add(new Point(233, 271));

points.Add(new Point(334, 152));

points.Add(new Point(497, 371));

points.Add(new Point(588, 252));

storage.Add(new Bezier(points[0], points[1], points[2], points[3], color));

points.Clear();

}

private void GradientAlongThePath\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Градиент вдоль пути

mode = 13;

}

private void Interpolation\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Интерполяции

mode = 14;

}

private void Complement\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Комлемент

mode = 15;

}

private void Exclude\_Button\_Click(object sender, EventArgs e)

{ // Выбор режима Исключения

mode = 16;

}

int cX = 0;

int cY = 0;

int x1, x2, x3, x4;

int y1, y2, y3, y4;

private void trackBar\_X\_Scroll(object sender, EventArgs e)

{ // Поворот по Оси X

if (mode == 12)

{

if (storage.Any())

if (storage.Last() is Bezier)

{

int k = trackBar\_X.Value;

Bezier last = storage.Last() as Bezier;

if (cX == 0)

{

x1 = last.pt1.X;

x2 = last.pt2.X;

x3 = last.pt3.X;

x4 = last.pt4.X;

cX = 1;

cY = 0;

}

points.Add(new Point(x1 + k, last.pt1.Y));

points.Add(new Point(x2 + k, last.pt2.Y));

points.Add(new Point(x3 - k, last.pt3.Y));

points.Add(new Point(x4 - k, last.pt4.Y));

storage.RemoveAt(storage.Count - 1);

storage.Add(new Bezier(points[0], points[1], points[2], points[3], color));

}

panel\_drawing.Refresh();

points.Clear();

}

}

private void trackBar\_Y\_Scroll(object sender, EventArgs e)

{ // Поворот по Оси Y

if (mode == 12)

{

if (storage.Any())

if (storage.Last() is Bezier)

{

int k = trackBar\_Y.Value;

Bezier last = storage.Last() as Bezier;

if (cY == 0)

{

y1 = last.pt1.Y;

y2 = last.pt2.Y;

y3 = last.pt3.Y;

y4 = last.pt4.Y;

cY = 1;

cX = 0;

}

points.Add(new Point(last.pt1.X, y1 - k));

points.Add(new Point(last.pt2.X, y2 - k));

points.Add(new Point(last.pt3.X, y3 + k));

points.Add(new Point(last.pt4.X, y4 + k));

storage.RemoveAt(storage.Count - 1);

storage.Add(new Bezier(points[0], points[1], points[2], points[3], color));

}

panel\_drawing.Refresh();

points.Clear();

}

}

}

}

**Shape.cs:**

using System.Drawing;

namespace \_1\_laba\_Graphics

{

public abstract class Shape

{

public int x;

public int y;

public Color color;

public abstract void Paint(Graphics graphic);

}

}

**Bezier.cs:**

using System.Drawing;

namespace \_1\_laba\_Graphics

{

public class Bezier : Shape

{

public Point pt1;

public Point pt2;

public Point pt3;

public Point pt4;

public Bezier(Point p1, Point p2, Point p3, Point p4, Color color)

{

pt1 = p1;

pt2 = p2;

pt3 = p3;

pt4 = p4;

this.color = color;

}

public override void Paint(Graphics graphic)

{

graphic.DrawBezier(new Pen(this.color, 3), pt1, pt2, pt3, pt4);

}

}

}

**Complement.cs:**

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_1\_laba\_Graphics

{

public class Complement : Shape

{

public override void Paint(Graphics graphic)

{

GraphicsPath FirstPath = new GraphicsPath();

GraphicsPath SecondPath = new GraphicsPath();

Rectangle regionRect = new Rectangle(20, 20, 100, 100);

FirstPath.AddEllipse(regionRect);

graphic.DrawPath(Pens.Black, FirstPath);

Rectangle complementRect = new Rectangle(90, 30, 100, 100);

SecondPath.AddEllipse(complementRect);

graphic.DrawPath(Pens.Red, SecondPath);

Region myRegion = new Region(FirstPath);

Region complementRegion = new Region(SecondPath);

myRegion.Complement(SecondPath);

SolidBrush myBrush = new SolidBrush(Color.Blue);

graphic.FillRegion(myBrush, myRegion);

}

}

}

**Diagram.cs:**

using System.Drawing;

namespace \_1\_laba\_Graphics

{

public class Diagram: Pie

{

public Diagram(int x, int y, int Width, int Height, float startAngle, float endAngle, Color color) : base(x, y, Width, Height, startAngle, endAngle, color)

{

}

public override void Paint(Graphics graphic)

{

Rectangle rect = new Rectangle(x, y, Width, Height);

graphic.FillPie(new SolidBrush(Color.LightGoldenrodYellow), rect, 0, 130);

graphic.FillPie(new SolidBrush(Color.PaleTurquoise), rect, 130, 205);

graphic.FillPie(new SolidBrush(Color.RoyalBlue), rect, 205, 290);

graphic.FillPie(new SolidBrush(Color.Purple), rect, 290, 50);

graphic.DrawEllipse(new Pen(Color.Black, 2), rect);

}

}

}

**Ellipse.cs:**

using System.Drawing;

namespace \_1\_laba\_Graphics

{

public class Ellipse : Shape

{

public int Width;

public int Height;

public Ellipse(int x, int y, int width, int height, Color color)

{

this.x = x;

this.y = y;

this.Width = width;

this.Height = height;

this.color = color;

}

public override void Paint(Graphics graphic)

{

graphic.DrawEllipse(new Pen(this.color, 3), x, y, Width, Height);

}

}

}

**Exclude.cs:**

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_1\_laba\_Graphics

{

public class Exclude : Shape

{

public override void Paint(Graphics graphic)

{

GraphicsPath FirstPath = new GraphicsPath();

GraphicsPath SecondPath = new GraphicsPath();

Rectangle regionRect = new Rectangle(20, 20, 100, 100);

FirstPath.AddEllipse(regionRect);

graphic.DrawPath(Pens.Black, FirstPath);

RectangleF complementRect = new RectangleF(90, 30, 100, 100);

SecondPath.AddEllipse(complementRect);

graphic.DrawPath(Pens.Red, SecondPath);

Region myRegion = new Region(FirstPath);

myRegion.Exclude(SecondPath);

SolidBrush myBrush = new SolidBrush(Color.Blue);

graphic.FillRegion(myBrush, myRegion);

}

}

}

**Gradient.cs:**

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_1\_laba\_Graphics

{

public class Gradient : Shape

{

Color Main\_color;

public Gradient(Color color)

{

Main\_color = color;

}

public override void Paint(Graphics graphic)

{

Rectangle MyRectangle = new Rectangle(200, 200, 400, 400);

Pen MyPen = new Pen(Color.Black , 5);

GraphicsPath path = new GraphicsPath();

path.AddEllipse(MyRectangle);

PathGradientBrush pthGrBrush = new PathGradientBrush(path);

// Центр пути будет красного цвета

pthGrBrush.CenterColor = Main\_color;

graphic.DrawEllipse(MyPen, MyRectangle);

graphic.FillPath(pthGrBrush, path);

}

}

}

**GradientAlongThePath.cs:**

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_1\_laba\_Graphics

{

public class GradientAlongThePath : Shape

{

public override void Paint(Graphics graphic)

{

GraphicsPath MyPath = new GraphicsPath();

MyPath.AddRectangle(new Rectangle(0, 0, 200, 100));

PathGradientBrush myBrush = new PathGradientBrush(MyPath);

Color[] redColor = new[] { Color.Red };

myBrush.SurroundColors = redColor;

myBrush.CenterColor = Color.Aqua;

graphic.FillPath(myBrush, MyPath);

myBrush.FocusScales = new PointF(0.2F, 0.5F);

graphic.TranslateTransform(0.0F, 150.0F);

graphic.FillPath(myBrush, MyPath);

}

}

}

**GradientV2:**

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_1\_laba\_Graphics

{

public class GradientV2 : Shape

{

public override void Paint(Graphics graphic)

{

PointF[] myPoints = new[] {

new PointF(200, 100),

new PointF(400, 100),

new PointF(500, 200),

new PointF(500, 400),

new PointF(400, 500),

new PointF(200, 500),

new PointF(100, 400),

new PointF(100, 200) };

PathGradientBrush myBrush = new PathGradientBrush(myPoints);

Color[] colors = new[] {

Color.FromArgb(255, 255, 255, 255),

Color.FromArgb(255, 255, 255, 0),

Color.FromArgb(255, 255, 0, 255),

Color.FromArgb(255, 0, 255, 255),

Color.FromArgb(255, 255, 0, 0),

Color.FromArgb(255, 0, 255, 0),

Color.FromArgb(255, 0, 0, 255),

Color.FromArgb(255, 0, 0, 0) };

myBrush.SurroundColors = colors;

myBrush.CenterColor = Color.White;

graphic.FillRectangle(myBrush, new Rectangle(100, 100, 500, 500));

}

}

}

**Interpolation:**

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_1\_laba\_Graphics

{

public class Interpolation : Shape

{

public override void Paint(Graphics graphic)

{

Point[] myPoints = new[] { new Point(0, 0), new Point(200, 0), new Point(200, 200), new Point(0, 200) };

PathGradientBrush myBrush = new PathGradientBrush(myPoints);

Color[] colors = new[] { Color.FromArgb(255, 0, 128, 0), Color.FromArgb(255, 128, 0, 255), Color.FromArgb(255, 0, 128, 128) };

float[] relativePositions = new[] { 0.0F, 0.4F, 1.0F };

ColorBlend colorBlend = new ColorBlend();

colorBlend.Colors = colors;

colorBlend.Positions = relativePositions;

myBrush.InterpolationColors = colorBlend;

// Заливаем прямоугольник, больший по размерам, чем квадрат

graphic.FillRectangle(myBrush, 0, 0, 200, 200);

}

}

}

**Line.cs:**

using System.Drawing;

namespace \_1\_laba\_Graphics

{

public class Line : Shape

{

Point point1;

Point point2;

public Line(Point p1, Point p2, Color color)

{

point1 = p1;

point2 = p2;

this.color = color;

}

public override void Paint(Graphics graphic)

{

Point[] points = { point1, point2 };

graphic.DrawPolygon(new Pen(this.color, 3), points);

}

}

}

**Pie.cs:**

using System.Drawing;

namespace \_1\_laba\_Graphics

{

public class Pie: Shape

{

public int Width;

public int Height;

public float startAngle;

public float endAngle;

public Pie(int x, int y, int Width, int Height, float startAngle, float endAngle, Color color)

{

this.x = x;

this.y = y;

this.Width = Width;

this.Height = Height;

this.startAngle = startAngle;

this.endAngle = endAngle;

this.color = color;

}

public override void Paint(Graphics graphic)

{

Rectangle rect = new Rectangle(x, y, Width, Height);

graphic.FillPie(new SolidBrush(color), rect, startAngle, endAngle);

graphic.DrawEllipse(new Pen(Color.Black, 2), rect);

}

}

}

**Polygon.cs:**

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_1\_laba\_Graphics

{

public class Polygon : Shape

{

PointF point1, point2, point3, point4, point5, point6, point7, point8;

public Polygon(Color color)

{

point1 = new PointF(0.0F, 0.0F);

point2 = new PointF(100.0F, 25.0F);

point3 = new PointF(200.0F, 5.0F);

point4 = new PointF(250.0F, 50.0F);

point5 = new PointF(300.0F, 100.0F);

point6 = new PointF(350.0F, 200.0F);

point7 = new PointF(200.0F, 200.0F);

point8 = new PointF(130.0F, 230.0F);

this.color = color;

}

public override void Paint(Graphics graphic)

{

PointF[] pointF = new[] { point1, point2, point3, point4, point5, point6, point7, point8 };

HatchBrush hatBrush = new HatchBrush(HatchStyle.DarkUpwardDiagonal, Color.Black, this.color);

FillMode newFillMode = FillMode.Winding;

graphic.FillPolygon(hatBrush, pointF, newFillMode);

}

}

}

**PolygonV2.cs:**

using System.Collections.Generic;

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_1\_laba\_Graphics

{

public class PolygonV2: Shape

{

List<Point> \_points;

public PolygonV2(List<Point> points, Color color)

{

\_points = new List<Point>(points);

this.color = color;

}

public override void Paint(Graphics graphic)

{

GraphicsPath GPath = new GraphicsPath();

HatchBrush hatBrush = new HatchBrush(HatchStyle.DarkUpwardDiagonal, Color.Black, this.color);

GPath.AddPolygon(\_points.GetRange(0, 4).ToArray());

GPath.AddPolygon(\_points.GetRange(4, 4).ToArray());

graphic.FillPath(hatBrush, GPath);

}

}

}

**SubShape.cs:**

using System.Drawing;

using System.Drawing.Drawing2D;

namespace \_1\_laba\_Graphics

{

public class SubShape : Shape

{

public int Width;

public int Height;

public SubShape(int x, int y, Color color)

{

this.x = x;

this.y = y;

this.color = color;

}

public override void Paint(Graphics graphic)

{

GraphicsPath FirstPath = new GraphicsPath();

GraphicsPath SecondPath = new GraphicsPath();

FirstPath.AddEllipse(new Rectangle(x, y, 200, 100));

SecondPath.AddRectangle(new Rectangle(x + 50, y + 50, 200, 100));

Region myRegion = new Region(FirstPath);

myRegion.Xor(SecondPath);

graphic.FillRegion(new SolidBrush(this.color), myRegion);

}

}

}

**TextureKist.cs:**

using System.Drawing;

namespace \_1\_laba\_Graphics

{

public class TextuteKist : Shape

{

Image texture;

public TextuteKist(int x, int y)

{

texture = Image.FromFile(@"C:\Users\Пользователь\Downloads\kirpich.jpg");

this.x = x;

this.y = y;

}

public override void Paint(Graphics graphic)

{

TextureBrush tBrush = new TextureBrush(texture);

Pen tPen = new Pen(tBrush, 2);

graphic.FillEllipse(tBrush, x, y, 60, 60);

}

}

}

Приложение 2. Скриншот коммитов

Изображение выглядит как текст

Автоматически созданное описание

Рис. 26 Скриншот коммитов

Приложение 3. Ссылка на GitHub

<https://github.com/MenikUG/1_laba_Graphics>