Wydział Sztuki,Techniki i Komunikacji

Kierunek studiów: informatyka

Przedmiot: Programowanie Obiektowe

Dokumentacja

Projektu nr 2

Autor Piotr Skupiński

nr albumu: 59369 grupa 2

Warszawa 05.2021

Zawartość

[Klasa Prostokąt 3](#_Toc104371119)

[Klasa Kwadrat 3](#_Toc104371120)

[Klasa Koło 4](#_Toc104371121)

[Klasa FillElipse 4](#_Toc104371122)

[KlasaFillRectangle 5](#_Toc104371123)

[Klasa DrawPie 5](#_Toc104371124)

[Klasa FillPie 6](#_Toc104371125)

[Klasa DrawArc 6](#_Toc104371126)

[Klasa KrzywaKardynalna 7](#_Toc104371127)

[Klasa FillClosedCurve 9](#_Toc104371128)

[Klasa WielokątForemny 10](#_Toc104371129)

[Klasa WielokątForemnyWypełniony 11](#_Toc104371130)

[Dokumentacja Formularzy 11](#_Toc104371131)

[Formularz FormFiguryGeoInitial 11](#_Toc104371132)

[Formularz Prezentacja Losowa Ze Slajderem 12](#_Toc104371133)

[Formularz Kreślenie Figur Geometrycznych 13](#_Toc104371134)

[Samoocena 13](#_Toc104371135)

[Samoocena Labolatorium 13](#_Toc104371136)

[Samoocena Projekt 14](#_Toc104371137)

## Klasa Prostokąt

public class Prostokąt : Punkt

{

protected int psx, psy, psSzerokość, psWysokość;

public Prostokąt(int psx, int psy, int psSzerokość, int psWysokość) : base(psx, psy)

{

this.psx = psx;

this.psy = psx;

this.psSzerokość = psSzerokość;

this.psWysokość = psWysokość;

}

public override void Wykreśl(Graphics psRysownica)

{

Pen psPióro = new Pen(psKolor, this.psGrubośćLinii);

psPióro.DashStyle = psStylLinii;

psRysownica.DrawRectangle(psPióro, psX, psY, psSzerokość, psWysokość);

psWidoczny = true;

psPióro.Dispose();

}

public override void Wymaż(Control psKontrolka, Graphics psRysownica)

{

Pen psPióro = new Pen(psKontrolka.BackColor, this.psGrubośćLinii);

psPióro.DashStyle = psStylLinii;

psRysownica.DrawRectangle(psPióro, psX, psY, psSzerokość, psWysokość);

psWidoczny = true;

psPióro.Dispose();

}

}

## Klasa Kwadrat

public class Kwadrat:Prostokąt

{

public Kwadrat(int psx, int psy, int psSzerokość) : base(psx,psy,psSzerokość,psSzerokość)

{

this.psx = psx;

this.psy = psx;

this.psSzerokość = psSzerokość;

}

public override void Wykreśl(Graphics psRysownica)

{

Pen psPióro = new Pen(psKolor, this.psGrubośćLinii);

psPióro.DashStyle = psStylLinii;

psRysownica.DrawRectangle(psPióro, psX, psY, psSzerokość, psSzerokość);

psWidoczny = true;

psPióro.Dispose();

}

public override void Wymaż(Control psKontrolka, Graphics psRysownica)

{

Pen psPióro = new Pen(psKontrolka.BackColor, this.psGrubośćLinii);

psPióro.DashStyle = psStylLinii;

psRysownica.DrawRectangle(psPióro, psX, psY, psSzerokość, psSzerokość);

psWidoczny = true;

psPióro.Dispose();

}

}

## Klasa Koło

public class Koło:Okrąg

{

public Koło(int psx, int psy, int psPromień) : base(psx, psy,psPromień)

{

this.psPromień = psPromień;

}

public override void Wykreśl(Graphics psRysownica)

{

SolidBrush psPędzel = new SolidBrush(psKolor);

psRysownica.FillEllipse(psPędzel, psX, psY, psPromień, psPromień);

psWidoczny = true;

psPędzel.Dispose();

}

public override void Wymaż(Control psKontrolka, Graphics psRysownica)

{

SolidBrush psPędzel = new SolidBrush(psKontrolka.BackColor);

psRysownica.FillEllipse(psPędzel, psX, psY, psPromień, psPromień);

psWidoczny = false;

psPędzel.Dispose();

}

}

## Klasa FillElipse

public class FillElipse : Punkt

{

protected int psOśDuża, psOśMała;

public FillElipse(int psx, int psy, int psOśDuża, int psOśMała) : base(psx, psy)

{

this.psOśDuża = psOśDuża;

this.psOśMała = psOśMała;

}

public override void Wykreśl(Graphics psRysownica)

{

SolidBrush psPędzel = new SolidBrush(psKolor);

psRysownica.FillEllipse(psPędzel, psX , psY , psOśDuża, psOśMała);

psWidoczny = true;

psPędzel.Dispose();

}

public override void Wymaż(Control psKontrolka, Graphics psRysownica)

{

SolidBrush psPędzel = new SolidBrush(psKolor);

psRysownica.FillEllipse(psPędzel, psX , psY , psOśDuża, psOśMała);

psWidoczny = true;

psPędzel.Dispose();

}

}

## KlasaFillRectangle

public class FillRectangle : Punkt

{

protected int psx, psy, psSzerokość, psWysokość;

public FillRectangle(int psx, int psy, int psSzerokość, int psWysokość) : base(psx, psy)

{

this.psx = psx;

this.psy = psx;

this.psSzerokość = psSzerokość;

this.psWysokość = psWysokość;

}

public override void Wykreśl(Graphics psRysownica)

{

SolidBrush psPędzel = new SolidBrush(psKolor);

psRysownica.FillRectangle(psPędzel, psX, psY, psSzerokość, psWysokość);

psWidoczny = true;

psPędzel.Dispose();

}

public override void Wymaż(Control psKontrolka, Graphics psRysownica)

{

SolidBrush psPędzel = new SolidBrush(psKolor);

psRysownica.FillRectangle(psPędzel, psX, psY, psSzerokość, psWysokość);

psWidoczny = true;

psPędzel.Dispose();

}

}

## Klasa DrawPie

public class DrawPie : Elipsa

{

int psStartAngle;

int psSweepAngle;

protected int psPromień;

public DrawPie(int psx, int psy, int psPromień, int psStartAngle, int psSweepAngle) :

base(psx, psy, 2 \* psPromień, 2 \* psPromień)

{

this.psPromień = psPromień;

this.psStartAngle = psStartAngle;

this.psSweepAngle = psSweepAngle;

}

public override void Wykreśl(Graphics psRysownica)

{

psStylLinii = DashStyle.Solid;

Pen psPióro = new Pen(psKolor, this.psGrubośćLinii / 2);

psPióro.DashStyle = psStylLinii;

psRysownica.DrawPie(psPióro, psX, psY, psPromień, psPromień, psStartAngle, psSweepAngle);

psWidoczny = true;

psPióro.Dispose();

}

public override void Wymaż(Control psKontrolka, Graphics psRysownica)

{

psStylLinii = DashStyle.Solid;

Pen psPióro = new Pen(psKontrolka.BackColor, this.psGrubośćLinii / 2);

psPióro.DashStyle = psStylLinii;

psRysownica.DrawPie(psPióro, psX, psY, psPromień, psPromień, psStartAngle, psSweepAngle);

psWidoczny = true;

psPióro.Dispose();

}

}

## Klasa FillPie

public class FillPie : Elipsa

{

float psStartAngle;

float psSweepAngle;

protected int psPromień;

public FillPie(int psx, int psy, int psPromień, int psStartAngle, int psSweepAngle) : base(psx, psy, 2 \* psPromień, 2 \* psPromień)

{

this.psPromień = psPromień;

this.psStartAngle = psStartAngle;

this.psSweepAngle = psSweepAngle;

}

public override void Wykreśl(Graphics psRysownica)

{

SolidBrush psPędzel = new SolidBrush(psKolor);

psRysownica.FillPie(psPędzel, psX, psY, psPromień, psPromień, psStartAngle, psSweepAngle);

psWidoczny = true;

psPędzel.Dispose();

}

public override void Wymaż(Control psKontrolka, Graphics psRysownica)

{

SolidBrush psPędzel = new SolidBrush(psKolor);

psRysownica.FillPie(psPędzel, psX, psY, psPromień, psPromień, psStartAngle, psSweepAngle);

psWidoczny = true;

psPędzel.Dispose();

}

}

## Klasa DrawArc

public class DrawArc : Elipsa

{

float psStartAngle;

float psSweepAngle;

protected int psPromień;

public DrawArc(int psx, int psy, int psPromień, int psStartAngle, int psSweepAngle) :

base(psx, psy, 2 \* psPromień, 2 \* psPromień)

{

this.psPromień = psPromień;

this.psStartAngle = psStartAngle;

this.psSweepAngle = psSweepAngle;

}

public override void Wykreśl(Graphics psRysownica)

{

psStylLinii = DashStyle.Solid;

Pen psPióro = new Pen(psKolor, this.psGrubośćLinii / 2);

psPióro.DashStyle = psStylLinii;

psRysownica.DrawArc(psPióro, psX, psY, psPromień, psPromień, psStartAngle, psSweepAngle);

psWidoczny = true;

psPióro.Dispose();

}

public override void Wymaż(Control psKontrolka, Graphics psRysownica)

{

psStylLinii = DashStyle.Solid;

Pen psPióro = new Pen(psKontrolka.BackColor, this.psGrubośćLinii / 2);

psPióro.DashStyle = psStylLinii;

psRysownica.DrawArc(psPióro, psX, psY, psPromień, psPromień, psStartAngle, psSweepAngle);

psWidoczny = true;

psPióro.Dispose();

}

}

## Klasa KrzywaKardynalna

public class KrzywaKardynalna:Punkt

{

public List<Point> psPunktyKrzywejKardynalnej = new List<Point>();

protected int psPPK = 5;

public ushort LiczbaPunktówKontrolnych

{

get;

set;

}

protected Font psFontOpisuPunktów = new Font("Arial", 8, FontStyle.Italic);

public KrzywaKardynalna(Graphics psRysownica, Pen psPióro, Point psPunktXY)

: base(psPunktXY.X, psPunktXY.Y, psPióro.Color, psPióro.DashStyle, (int)psPióro.Width)

{

psPunktyKrzywejKardynalnej.Add(psPunktXY);

using (SolidBrush psPędzel = new SolidBrush(psKolor))

{

psRysownica.FillEllipse(psPędzel, psPunktXY.X - psPPK, psPunktXY.Y - psPPK, 2 \* psPPK, 2 \* psPPK);

psRysownica.DrawString("p" + (psPunktyKrzywejKardynalnej.Count - 1).ToString(), psFontOpisuPunktów, psPędzel,

psPunktyKrzywejKardynalnej[psPunktyKrzywejKardynalnej.Count - 1]);

}

}

public virtual void DodajNowyPunktKontrolny(Point psPunktXY, Graphics psRysownica)

{

psPunktyKrzywejKardynalnej.Add(psPunktXY);

using (SolidBrush psPędzel = new SolidBrush(Color.Red))

{

if (psPunktyKrzywejKardynalnej.Count == 1 || psPunktyKrzywejKardynalnej.Count == 6)

{

psPędzel.Color = psKolor;

}

psRysownica.FillEllipse(psPędzel, psPunktXY.X - psPPK, psPunktXY.Y - psPPK, 2 \* psPPK, 2 \* psPPK);

psRysownica.DrawString("p" + (psPunktyKrzywejKardynalnej.Count - 1).ToString(), psFontOpisuPunktów, psPędzel,

psPunktyKrzywejKardynalnej[psPunktyKrzywejKardynalnej.Count - 1]);

}

if (psPunktyKrzywejKardynalnej.Count == 6)

{

Wykreśl(psRysownica);

}

}

public override void Wykreśl(Graphics psRysownica)

{

using (Pen psPióro = new Pen(psKolor, psGrubośćLinii))

{

psPióro.DashStyle = psStylLinii;

Point[] psPunktyKontrolne = new Point[psPunktyKrzywejKardynalnej.Count];

for (ushort i = 0; i < psPunktyKrzywejKardynalnej.Count; i++)

{

psPunktyKontrolne[i] = new Point(psPunktyKrzywejKardynalnej[i].X, psPunktyKrzywejKardynalnej[i].Y);

}

psRysownica.DrawCurve(psPióro, psPunktyKontrolne);

psWidoczny = true;

}

}

public override void Wymaż(Control psKontrolka, Graphics psRysownica)

{

if (psWidoczny)

{

using (Pen psPióro = new Pen(psKontrolka.BackColor, psGrubośćLinii))

{

psPióro.DashStyle = psStylLinii;

Point[] psPunktyKontrolne = new Point[psPunktyKrzywejKardynalnej.Count];

for (ushort i = 0; i < psPunktyKrzywejKardynalnej.Count; i++)

{

psPunktyKontrolne[i] = new Point(psPunktyKrzywejKardynalnej[i].X, psPunktyKrzywejKardynalnej[i].Y);

}

psRysownica.DrawCurve(psPióro, psPunktyKontrolne);

psWidoczny = false;

}

}

}

public override void UaktualnijXY(int psX, int psY)

{

int psPrzyrostX = psPunktyKrzywejKardynalnej[0].X - psX;

int psPrzyrostY = psPunktyKrzywejKardynalnej[0].Y - psY;

for (int i = 0; i < psPunktyKrzywejKardynalnej.Count; i++)

{

psPunktyKrzywejKardynalnej[i] = new Point(psPunktyKrzywejKardynalnej[i].X - psPrzyrostX,

psPunktyKrzywejKardynalnej[i].Y - psPrzyrostY);

}

}

}

## Klasa FillClosedCurve

public class FillClosedCurve : KrzywaKardynalna

{

public FillClosedCurve(Graphics psRysownica, Pen psPióro, Point psPunktXY)

: base(psRysownica, psPióro, psPunktXY)

{

psPunktyKrzywejKardynalnej.Add(psPunktXY);

using (SolidBrush psPędzel = new SolidBrush(psKolor))

{

psRysownica.FillEllipse(psPędzel, psPunktXY.X - psPPK, psPunktXY.Y - psPPK, 2 \* psPPK, 2 \* psPPK);

psRysownica.DrawString("p" + (psPunktyKrzywejKardynalnej.Count - 1).ToString(), psFontOpisuPunktów, psPędzel,

psPunktyKrzywejKardynalnej[psPunktyKrzywejKardynalnej.Count - 1]);

}

}

public override void Wykreśl(Graphics psRysownica)

{

using (SolidBrush psPędzel = new SolidBrush(psKolor))

{

Point[] psPunktyKontrolne = new Point[psPunktyKrzywejKardynalnej.Count];

for (ushort i = 0; i < psPunktyKrzywejKardynalnej.Count; i++)

{

psPunktyKontrolne[i] = new Point(psPunktyKrzywejKardynalnej[i].X, psPunktyKrzywejKardynalnej[i].Y);

}

psRysownica.FillClosedCurve(psPędzel, psPunktyKontrolne);

psWidoczny = true;

}

}

public override void Wymaż(Control psKontrolka, Graphics psRysownica)

{

if (psWidoczny)

{

using (SolidBrush psPędzel = new SolidBrush(psKontrolka.BackColor))

{

Point[] psPunktyKontrolne = new Point[psPunktyKrzywejKardynalnej.Count];

for (ushort i = 0; i < psPunktyKrzywejKardynalnej.Count; i++)

{

psPunktyKontrolne[i] = new Point(psPunktyKrzywejKardynalnej[i].X, psPunktyKrzywejKardynalnej[i].Y);

}

psRysownica.FillClosedCurve(psPędzel, psPunktyKontrolne);

psWidoczny = false;

}

}

}

}

## Klasa WielokątForemny

public class WielokątForemny : Kwadrat

{

static Random rnd = new Random();

static int N = rnd.Next(3, 12);//liczba ścian

public Point[] psPunktyWielokąta = new Point[N];

public WielokątForemny(int psx, int psy, int psSzerokość) : base(psx, psy, psSzerokość)

{

this.psx = psx;

this.psy = psx;

this.psSzerokość = psSzerokość;

}

public override void Wykreśl(Graphics psRysownica)

{

Pen psPióro = new Pen(psKolor, this.psGrubośćLinii);

psPióro.DashStyle = psStylLinii;

//N = rnd.Next(3, 12);

for(int n=0;n<N ;n++)

{

int X = (int)(psSzerokość \* Math.Cos(2 \* Math.PI \* n / N) + psx);

int Y = (int)(psSzerokość \* Math.Sin(2 \* Math.PI \* n / N) + psy);

psPunktyWielokąta[n]=(new Point(X, Y));

}

psRysownica.DrawPolygon(psPióro, psPunktyWielokąta);

psWidoczny = true;

psPióro.Dispose();

}

public override void Wymaż(Control psKontrolka, Graphics psRysownica)

{

Pen psPióro = new Pen(psKontrolka.BackColor, this.psGrubośćLinii);

psPióro.DashStyle = psStylLinii;

for (int n = 0; n < N; n++)

{

int X = (int)(psSzerokość \* Math.Cos(2 \* Math.PI \* n / N) + psx);

int Y = (int)(psSzerokość \* Math.Sin(2 \* Math.PI \* n / N) + psy);

psPunktyWielokąta[n] = (new Point(X, Y));

}

psRysownica.DrawPolygon(psPióro, psPunktyWielokąta);

psWidoczny = false;

psPióro.Dispose();

}

}

## Klasa WielokątForemnyWypełniony

public class WielokątForemnyWypełniony: WielokątForemny

{

static Random rnd = new Random();

static int N = rnd.Next(3, 12);//liczba ścian

public WielokątForemnyWypełniony(int psx, int psy, int psSzerokość) : base(psx, psy, psSzerokość)

{

this.psx = psx;

this.psy = psx;

this.psSzerokość = psSzerokość;

}

public override void Wykreśl(Graphics psRysownica)

{

SolidBrush psPędzel = new SolidBrush(psKolor);

for (int n = 0; n < N; n++)

{

int X = (int)(psSzerokość \* Math.Cos(2 \* Math.PI \* n / N) + psx);

int Y = (int)(psSzerokość \* Math.Sin(2 \* Math.PI \* n / N) + psy);

psPunktyWielokąta[n] = (new Point(X, Y));

}

psRysownica.FillPolygon(psPędzel, psPunktyWielokąta);

psWidoczny = true;

psPędzel.Dispose();

}

public override void Wymaż(Control psKontrolka, Graphics psRysownica)

{

SolidBrush psPędzel = new SolidBrush(psKontrolka.BackColor);

for (int n = 0; n < N; n++)

{

int X = (int)(psSzerokość \* Math.Cos(2 \* Math.PI \* n / N) + psx);

int Y = (int)(psSzerokość \* Math.Sin(2 \* Math.PI \* n / N) + psy);

psPunktyWielokąta[n] = (new Point(X, Y));

}

psRysownica.FillPolygon(psPędzel, psPunktyWielokąta);

psWidoczny = false;

psPędzel.Dispose();

}

}

# Dokumentacja Formularzy

## Formularz FormFiguryGeoInitial

Formularz po witający pozwala na poruszanie się między dwoma funkcjonalnymi formularzami.

Graphical user interface, application, website

Description automatically generated

## Formularz Prezentacja Losowa Ze Slajderem

Graphical user interface, text, application, Word

Description automatically generated

Aby uruchomić pokaz figur należy zaznaczyć figury do losowania podać ich liczbę, następnie kliknąć przycisk start. Udostępnione zostaną wtedy przyciski do losowania atrybutów figur geometrycznych, można też w tym celu użyć paska menu. Po zakończeniu zmian atrybutów Należy kliknąć przycisk stop co udostępnia Pokaz slajdów. Pokaz slajdów posiada 2 tryby manualny i automatyczny. W trybie automatycznym kolejne figury będą pokazywane przez kilka sekund, w trybie manualnym pokaz kontrolowany jest poprzez przyciski w grupie Manualna Kontrola Pokazu. Przycisk Przywróć stan początkowy pozawala działa zgodnie z swoją nazwą a w menu plik znajdują się opcje powrót i wyjdź, pozwalające na opuszczenie formularza.

## Formularz Kreślenie Figur Geometrycznych

Graphical user interface, application, Word

Description automatically generated

Aby kreślić figury na formularzu należy wybrać żądaną figurę a następnie rysować za pomocą myszy na powierzchni graficznej.

Należy pamiętać że do narysowania Krzywej Beziera należy podać 4 punkty, a Pozostałych krzywych arbitralną liczbę 6 punktów. Program pozwala na usuwanie narysowanych figur za pomocą przycisku cofnij i losowania ich pozycji za pomocą przycisku losuj. Formularz posiada znaną z poprzedniego formularza funkcje pokazu narysowanych wcześniej figur.

Grupa kontrolek atrybuty graficzne pozwala na kontrolę kolorów figur, a także stylu i grubości linii którą figury będą rysowane. Menu kolor posiada funkcjonalności znane z poprzedniego formularza. Menu Plik również jest niezmienione względem poprzedniego formularza.

# Samoocena

## Samoocena Labolatorium

Samoocena: 4

Wszystkie kontrolki zostały wykonane chociaż wielokąt foremny posiada nierozwiązane problemy. Obniżam sobie oceną ponieważ przez własną nieostrożność wysyłam ten projekt po terminie.

## Samoocena Projekt

Samoocena: 4

Projekt posiada więcej funkcjonalności niż wymagane ale nie są one doskonałe DrawArc,DrawPie i FillPie posiadają niezindedyfikowane błędy a krzywe nie pozwalają na wybór liczby punktów przez użytkownika. Ponadto przez własną nieostrożność (niezapisywanie pracy na laptopie tylko go usypianie) nieostrożność wysyłam ten projekt po terminie.