**Program.cs**

using SplashKitSDK;

using ShapeDrawer3;

namespace ShapeDrawer3

{

public class Program

{

private enum ShapeKind

{

Rectangle,

Circle,

Line

}

public static void Main()

{

Window window = new("Shape Drawer", 800, 600);

Drawing myDrawing = new();

ShapeKind kindtoAdd = ShapeKind.Rectangle;

do

{

SplashKit.ProcessEvents();

if (SplashKit.KeyDown(KeyCode.SpaceKey))

{

myDrawing.Background = Color.Random();

}

if (SplashKit.KeyTyped(KeyCode.RKey))

{

kindtoAdd = ShapeKind.Rectangle;

}

else if (SplashKit.KeyTyped(KeyCode.CKey))

{

kindtoAdd = ShapeKind.Circle;

}

else if (SplashKit.KeyTyped(KeyCode.LKey))

{

kindtoAdd = ShapeKind.Line;

}

if (SplashKit.MouseClicked(MouseButton.LeftButton))

{

Shape newShape;

switch (kindtoAdd)

{

case ShapeKind.Circle:

newShape = new MyCircle

{

X = SplashKit.MouseX(),

Y = SplashKit.MouseY()

};

break;

case ShapeKind.Line:

newShape = new MyLine

{

X = SplashKit.MouseX(),

Y = SplashKit.MouseY()

};

break;

default:

newShape = new MyRectangle

{

X = SplashKit.MouseX(),

Y = SplashKit.MouseY()

};

break;

}

myDrawing.AddShape(newShape);

}

if (SplashKit.KeyDown(KeyCode.DeleteKey) || SplashKit.KeyDown(KeyCode.BackspaceKey))

{

myDrawing.RemoveShapes();

}

if (SplashKit.MouseClicked(MouseButton.RightButton))

{

myDrawing.SelectShapeAt(SplashKit.MousePosition());

}

myDrawing.Draw();

SplashKit.RefreshScreen();

}

while (!window.CloseRequested);

}

}

}

**Shape.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Reflection;

using SplashKitSDK;

namespace ShapeDrawer3

{

public abstract class Shape(Color color)

{

private Color \_color = color;

private float \_x, \_y;

private bool \_selected;

public Shape() : this(Color.Yellow)

{

}

public float X

{

get

{

return \_x;

}

set

{

\_x = value;

}

}

public float Y

{

get

{

return \_y;

}

set

{

\_y = value;

}

}

public Color Color

{

get

{

return \_color;

}

set

{

\_color = value;

}

}

public bool Selected

{

get

{

return \_selected;

}

set

{

\_selected = value;

}

}

public abstract void Draw();

public abstract void DrawOutLine();

public abstract bool IsAt(Point2D pt);

}

}

**Drawing.cs**

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Reflection;

using ShapeDrawer3;

using SplashKitSDK;

namespace ShapeDrawer3

{

public class Drawing(SplashKitSDK.Color background)

{

public Drawing() : this(SplashKitSDK.Color.White)

{

}

private readonly List<Shape> \_shapes = [];

public List<Shape> SelectedShape()

{

List<Shape> \_selectedShapes = [];

foreach (Shape s in \_shapes)

{

if (s.Selected)

{

\_selectedShapes.Add(s);

}

}

return \_selectedShapes;

}

public int ShapeCount

{

get { return \_shapes.Count; }

}

private SplashKitSDK.Color \_background = background;

public SplashKitSDK.Color Background

{

get

{

return \_background;

}

set

{

\_background = value;

}

}

public void Draw()

{

SplashKit.ClearScreen(\_background);

foreach (Shape shape in \_shapes)

{

shape.Draw();

}

}

public void SelectShapeAt(Point2D pt)

{

foreach (Shape s in \_shapes)

{

if (s.IsAt(pt))

s.Selected = true;

else

s.Selected = false;

}

}

public void AddShape(Shape shape)

{

\_shapes.Add(shape);

}

public void RemoveShapes()

{

foreach (Shape s in \_shapes.ToList())

{

if (s.Selected)

{

\_shapes.Remove(s);

}

}

}

}

}

**MyRectangle.cs**

using ShapeDrawer3;

using SplashKitSDK;

namespace ShapeDrawer3

{

public class MyRectangle : Shape

{

private int \_width, \_height;

public MyRectangle() : this(Color.Green, 0.0f, 0.0f, 100, 100)

{

}

public MyRectangle(Color color, float x, float y, int width, int height) : base(color)

{

X = x;

Y = y;

Width = width;

Height = height;

}

public int Width

{

get

{

return \_width;

}

set

{

\_width = value;

}

}

public int Height

{

get

{

return \_height;

}

set

{

\_height = value;

}

}

public override void Draw()

{

if (Selected)

{

DrawOutLine();

}

SplashKit.FillRectangle(Color, X, Y, \_width, \_height);

}

public override void DrawOutLine()

{

SplashKit.DrawRectangle(Color.Black, X - 2, Y - 2, \_width + 4, \_height + 4);

}

public override bool IsAt(Point2D pt)

{

return pt.X >= X && pt.X < X + \_width && pt.Y >= Y && pt.Y <= Y + \_height;

}

}

}

**MyCircle.cs**

using ShapeDrawer3;

using SplashKitSDK;

namespace ShapeDrawer3

{

public class MyCircle : Shape

{

private int \_radius;

public MyCircle() : this(Color.Blue, 0.0f, 0.0f, 50)

{

}

public MyCircle(Color color, float x, float y, int radius) : base(color)

{

X = x;

Y = y;

Radius = radius;

}

public int Radius

{

get

{

return \_radius;

}

set

{

\_radius = value;

}

}

public override void Draw()

{

if (Selected)

{

DrawOutLine();

}

SplashKit.FillCircle(Color, X, Y, \_radius);

}

public override void DrawOutLine()

{

SplashKit.DrawCircle(Color.Black, X, Y, \_radius + 2);

}

public override bool IsAt(Point2D pt)

{

return SplashKit.PointInCircle(pt, SplashKit.CircleAt(X, Y, \_radius));

}

}

}

**MyLine.cs**

using SplashKitSDK;

namespace ShapeDrawer3

{

public class MyLine : Shape

{

private float \_endX;

private float \_endY;

public MyLine() : this(Color.Red, 0, 0, 100, 0)

{

}

public MyLine(Color color, float startX, float startY, float endX, float endY)

{

Color = color;

X = startX;

Y = startY;

EndX = endX;

EndY = endY;

}

public float EndX

{

get

{

return \_endX;

}

set

{

\_endX = value;

}

}

public float EndY

{

get

{

return \_endY;

}

set

{

\_endY = value;

}

}

public override void Draw()

{

if (Selected)

{

DrawOutLine();

}

SplashKit.DrawLine(Color, X, Y, X + \_endX, Y + \_endY);

}

public override void DrawOutLine()

{

SplashKit.FillCircle(Color.Black, X, Y, 5);

SplashKit.FillCircle(Color.Black, X + \_endX, Y + \_endY, 5);

}

public override bool IsAt(Point2D pt)

{

return SplashKit.PointOnLine(pt, SplashKit.LineFrom(X, Y, X + \_endX, Y + \_endY));

}

}

}

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động