COS20007: Object Oriented Programming

Pass Task 3.3: Drawing Program - A Drawing Class with your own attributes  
Show Wai Yan/105293041

# Drawing.cs

using SplashKitSDK;

namespace ShapeDrawer

{

public class Drawing

{

// Fields

private readonly List<Shape> \_shapes;

private Color \_background;

// Constructor

public Drawing(Color background)

{

\_shapes = new List<Shape>();

\_background = background;

}

public Drawing() : this(Color.White)

{

}

// Property

public List<Shape> SelectedShapes

{

// readonly property

get

{

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

foreach (Shape s in \_shapes)

{

if (s.Selected) selectedShapes.Add(s);

}

return selectedShapes;

}

}

public int ShapeCount

{

// readonly property

get { return this.\_shapes.Count; }

}

public Color Background

{

get { return this.\_background; }

set { this.\_background = value; }

}

// Methods

public void Draw()

{

SplashKit.ClearScreen(\_background);

foreach (Shape s in \_shapes)

{

s.Draw();

}

}

public void SelectShapesAt(Point2D pt)

{

foreach (Shape s in \_shapes)

{

s.Selected = s.IsAt(pt);

}

}

public void AddShape(Shape s)

{

\_shapes.Add(s);

}

public void RemoveShape(Shape s)

{

\_ = \_shapes.Remove(s);

}

}

}

# Shape.cs

using SplashKitSDK;

namespace ShapeDrawer

{

public class Shape

{

// Fields

private Color \_color;

private float \_x;

private float \_y;

private int \_width;

private int \_height;

private bool \_selected = false;

// Constructors

public Shape(int param)

{

\_color = Color.Chocolate;

\_x = 0.0f; \_y = 0.0f;

\_width = param; \_height = param;

}

// Properties

public Color Color

{

get { return \_color; }

set { \_color = value; }

}

public float X

{

get { return \_x; }

set { \_x = value; }

}

public float Y

{

get { return \_y; }

set { \_y = value; }

}

public int Width

{

get { return \_width; }

set { \_width = value; }

}

public int Height

{

get { return \_height; }

set { \_height = value; }

}

public bool Selected

{

get { return this.\_selected; }

set { this.\_selected = value; }

}

// Methods

public void Draw()

{

if (this.\_selected) this.DrawOutline();

SplashKit.FillRectangle(\_color, \_x, \_y, \_width, \_height);

}

public bool IsAt(Point2D pt)

{

return (pt.X >= \_x && pt.X <= \_x + \_width) && (pt.Y >= \_y && pt.Y <= \_y + \_height);

}

public void DrawOutline()

{

int outlineThickness = 6; //5+!

SplashKit.FillRectangle(Color.Black, \_x-outlineThickness, \_y-outlineThickness, \_width+2\*outlineThickness, \_height+2\*outlineThickness);

}

}

}

# Program.cs

using System;

using SplashKitSDK;

namespace ShapeDrawer

{

public class Program

{

public static void Main()

{

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

Drawing myDrawing = new Drawing();

do

{

SplashKit.ProcessEvents();

SplashKit.ClearScreen();

if (SplashKit.MouseClicked(MouseButton.LeftButton))

{

Shape myShape = new Shape(141);

myShape.X = SplashKit.MouseX();

myShape.Y = SplashKit.MouseY();

myDrawing.AddShape(myShape);

}

if (SplashKit.KeyTyped(KeyCode.SpaceKey))

{

myDrawing.Background = SplashKit.RandomColor();

}

if (SplashKit.MouseClicked(MouseButton.RightButton))

{

myDrawing.SelectShapesAt(SplashKit.MousePosition());

}

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

{

foreach (Shape s in myDrawing.SelectedShapes)

{

myDrawing.RemoveShape(s);

}

}

myDrawing.Draw();

SplashKit.RefreshScreen();

} while (!window.CloseRequested);

}

}

}

# Screenshot of the Splashkit Window showing your drawing

A screenshot of a computer

AI-generated content may be incorrect.

# 