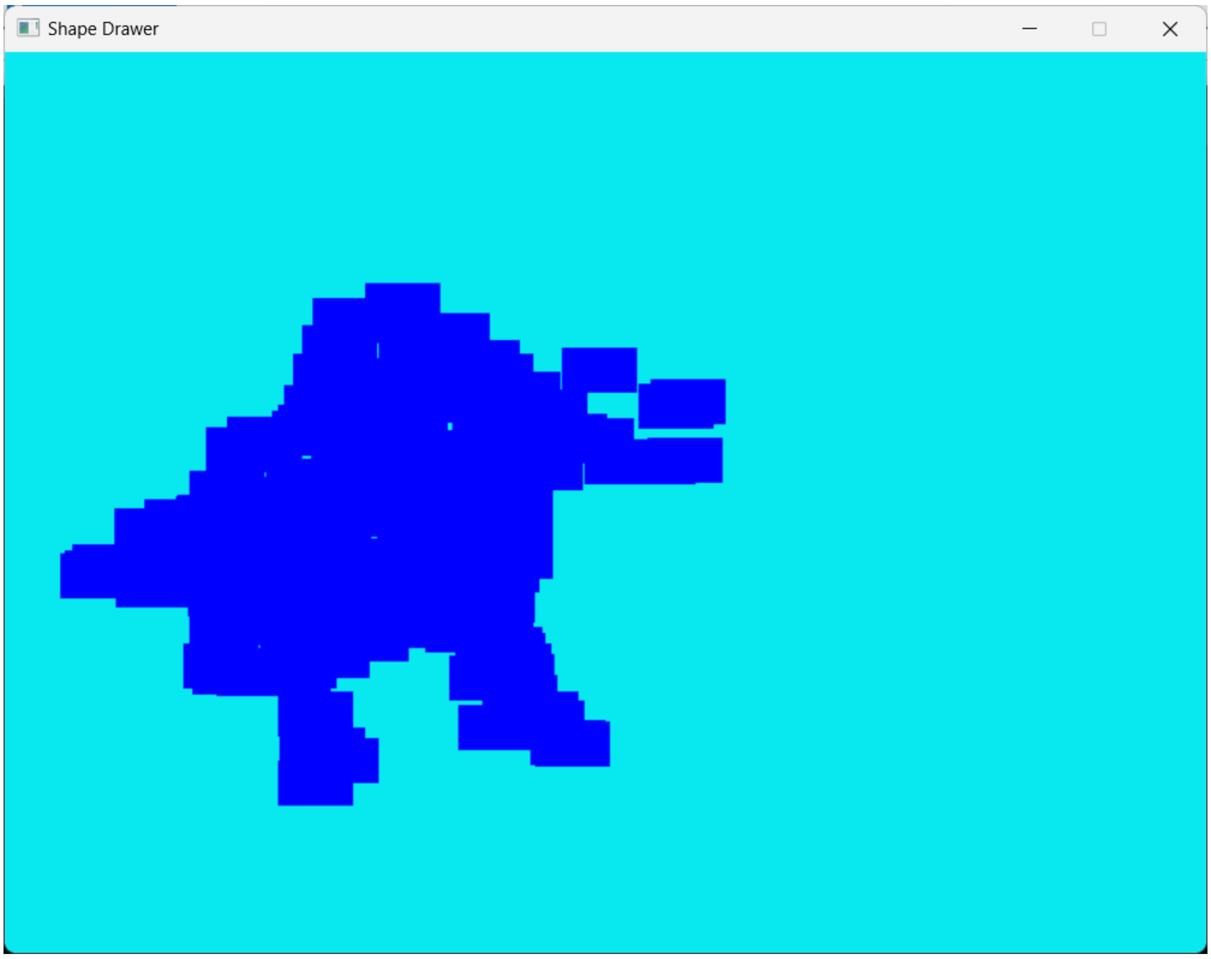
Output of splashkit:



Drawing.cs:

```
using System.Collections.Generic;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;

        // Public property to access the background color
        public Color Background
```

```
{
   get { return _background; }
   set { _background = value; }
}

public Drawing(Color background)
{
   _shapes = new List<Shape>();
   _background = background;
}

public Drawing() : this(Color.White) { } // Default constructor

public void AddShape(Shape shape)
{
   _shapes.Add(shape);
}

public void RemoveShape(Shape shape)
{
   _shapes.Remove(shape);
}

public void Draw()
{
   SplashKit.ClearScreen(_background);
   foreach (Shape shape in _shapes)
   {
      shape.Draw();
   }
}
```

```
public void SelectShapesAt(Point2D pt)
{
   foreach (Shape s in _shapes)
   {
      if (s.IsAt(pt))
      {
         s.Selected = true;
      }
      else
      {
         s.Selected = false;
      }
   }
}

public List<Shape> SelectedShapes
{
   get
   {
      List<Shape> result = new List<Shape>();
      foreach (Shape s in _shapes)
      {
         if (s.Selected)
         {
            result.Add(s);
         }
      }
      return result;
   }
}
```

```
    }
}
```

Shape.cs:

```
using SplashKitSDK;

namespace ShapeDrawer
{
    public class Shape
    {
        private double _x, _y;
        private bool _selected;

        public double X
        {
            get { return _x; }
            set { _x = value; }
        }

        public double Y
        {
            get { return _y; }
            set { _y = value; }
        }

        public bool Selected
        {
            get { return _selected; }
            set { _selected = value; }
        }
```

```
public Shape()
{
    _x = 0;
    _y = 0;
    _selected = false;
}

public void Draw()
{
    // Example: Draw a blue rectangle
    SplashKit.FillRectangle(Color.Blue, (float)_x, (float)_y, 50, 30);

    // Draw outline if selected
    if (_selected)
    {
        DrawOutline();
    }
}

public void DrawOutline()
{
    const int outlineThickness = 2;
    SplashKit.DrawRectangle(Color.Black,
        (float)(_x - outlineThickness),
        (float)(_y - outlineThickness),
        50 + outlineThickness * 2,
        30 + outlineThickness * 2);
}

public bool IsAt(Point2D pt)
{
```

```
            return pt.X >= _x && pt.X <= _x + 50 && pt.Y >= _y && pt.Y <= _y + 30; // Adjust based on
shape size
        }
    }
}
```

Program.cs:

```
using System;
using SplashKitSDK;

namespace ShapeDrawer
{
    public class Program
    {
        public static void Main()
        {
            Drawing myDrawing = new Drawing(); // Create a Drawing object using the default constructor

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

            do
            {
                SplashKit.ProcessEvents();
                window.Clear(Color.White); // Clear with a temporary white color

                myDrawing.Draw(); // Draw all shapes

                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    // Create a new shape at the mouse position
                    Shape newShape = new Shape();
            newShape.X = SplashKit.MouseX();
            newShape.Y = SplashKit.MouseY();
            myDrawing.AddShape(newShape); // Add the new shape to the drawing
        }

        if (SplashKit.KeyTyped(KeyCode.SpaceKey))
        {
            // Change the background color to a random color
            myDrawing.Background = SplashKit.RandomColor();
        }

        if (SplashKit.MouseClicked(MouseButton.RightButton))
        {
            // Select shapes at the current mouse pointer position
            myDrawing.SelectShapesAt(SplashKit.MousePosition());
        }

        // Remove selected shapes if the delete or backspace key is pressed
        if (SplashKit.KeyTyped(KeyCode.DeleteKey) || SplashKit.KeyTyped(KeyCode.BackspaceKey))
        {
            foreach (var shape in myDrawing.SelectedShapes)
            {
                myDrawing.RemoveShape(shape);
            }
        }

        window.Refresh(60);
    } while (!window.CloseRequested);

    window.Close();
}
```

```
    }
}
```