3.3 ShapesDrawer2

source code
program.cs

```
using System;
using System.ComponentModel.Design;
using SplashKitSDK;

namespace shapedrawer
{
   public class Program //this program.cs main job is to draw the canvas and display the varibles
   {


      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))
            {
               int x_pos = (int)SplashKit.MouseX();
               int y_pos = (int)SplashKit.MouseY();
```

```
            Shape newShape = new Shape();
            newShape.X = x_pos;
            newShape.Y = y_pos;
            newShape.Width = 100; // Set default width
            newShape.Height = 100; // Set default height
            newShape.Color = SplashKit.RandomRGBColor(255);

            // Add the new shape to the Drawing object
            myDrawing.AddShape(newShape); //call addShape function from drawing.cs
            Console.WriteLine("added shape");



        }
        if (SplashKit.KeyTyped(KeyCode.SpaceKey)) //spacekey to change different
background color
        {
            myDrawing.Background = SplashKit.RandomRGBColor(255);


        }
        if (SplashKit.MouseClicked(MouseButton.RightButton)) //right click to highlight
outline of the shape
        {
            myDrawing.SelectShapesAt(SplashKit.MousePosition());


        }


        if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
SplashKit.KeyTyped(KeyCode.DeleteKey))//to delete shapes drawn
```

```
            {
                if (SplashKit.KeyTyped(KeyCode.BackspaceKey)) //backspce key to delete
                {
                    Console.WriteLine("Deleted shape");
                }
                if (SplashKit.KeyTyped(KeyCode.DeleteKey)) //delete key to delete
                {
                    Console.WriteLine("Deleted shape");
                }
                myDrawing.RemoveShape(); //call the remove shape function frim drawing.cs
            }

            myDrawing.Draw();

            SplashKit.RefreshScreen();
        } while (!window.CloseRequested);
    }
  }
}
```

//when declare a new function do NOT place it in the same function as prev or else the function wouldnt as its not the main father function

Drawing.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;
```

```
namespace shapedrawer
{
   public class Drawing
   {
      private readonly List<Shape> _shapes; //readonly list to store all shapes
      private Color _background;

      public Drawing(Color background)
      {
         _shapes = new List<Shape>();
         _background = background;
      }
      public Drawing() : this(Color.White) //"this" to avoid duplication
      {

      }

      public int ShapeCount //property to class Drawing that returns the Count from the
_shapes list collection object
      {
         get { return _shapes.Count; } //get and return from _shapes = new List<shape>
      }

      public void AddShape(Shape s) //adds shape into the list from shape
      {
         _shapes.Add(s);
      }
      public void RemoveShape()
```

```
{
    foreach (Shape s in _shapes.ToList())//each shape s is from AddShape
    {
        if (s.Selected) //if shape is selected
        {
            _shapes.Remove(s); //remove the shape
        }

    }
}
public void Draw()
{
    SplashKit.ClearScreen(_background);
    foreach (Shape s in _shapes)
    {
        s.Draw();
    }
}
public Color Background
{
    get
    {
        return _background;

    }
    set
    {
        _background = value;
```

```
    }



}
 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()
{
    List<Shape> _Selectedshapes = new List<Shape>();
    foreach (Shape s in _Selectedshapes)
    {
        if (s.Selected)
        {
            _Selectedshapes.Add(s);
        }
```

```
        }
        return _Selectedshapes;
      }
    }
}
```

Shape.cs

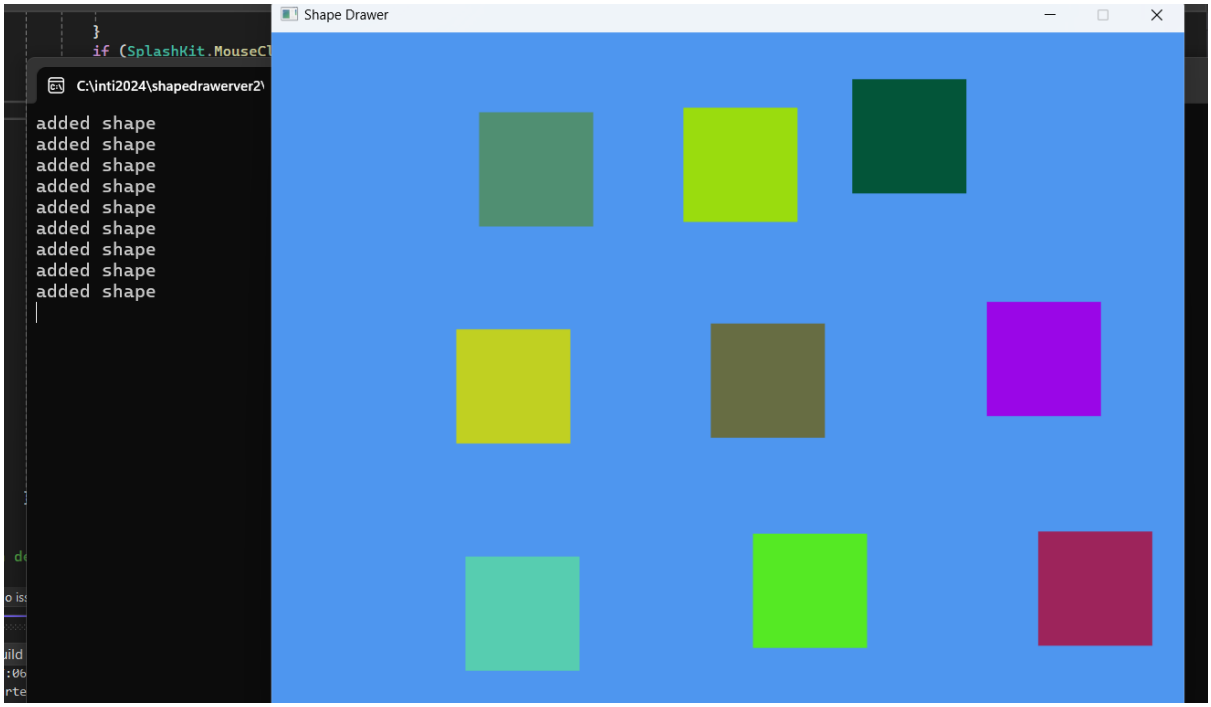
```
using SplashKitSDK;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace shapedrawer
{
    public class Shape
    {
        // Private fields
        private Color _color;
        private float _x, _y;
        private float _width, _height;
        private bool _selected;
        private int x_pos;
        private int y_pos;

        public Shape(int x_pos, int y_pos)
        {
            this.x_pos = x_pos;
```

```
    this.y_pos = y_pos;
}

public Shape()
{

}

// Properties
public Color Color //call and intialize the variable
{
    get { return _color; }
    set { _color = value; }
}

public float X //call and intialize the variable
{
    get { return _x; } //get and store the x value
    set { _x = value; }
}

public float Y //call and intialize the variable
{
    get { return _y; } //get and store the y value
    set { _y = value; }
}

public float Width //call and intialize the variable
```

```
    {
        get { return _width; } //get and store the width
        set { _width = value; }
    }

    public float Height //call and intialize the variable
    {
        get { return _height; } //get and store the height
        set { _height = value; }
    }

    // Method to draw the shape
    public void Draw()
    {
        if (_selected)
        {
            DrawOutline();
        }
        SplashKit.FillRectangle(_color, _x, _y, _width, _height);

    }

    // Method to check if a point is within the shape's area
    public bool IsAt(Point2D point)
    {
        return (point.X >= _x && point.X <= _x + _width) &&
               (point.Y >= _y && point.Y <= _y + _height); // to find the coord after or below a
specify point
```

```
    }//if x is more than equal to x and x is less than equal to x + width is to find the whole width dimension of the box


    public bool Selected //determine selected shapes value and return
    {
      get
      {
        return _selected;
      }
      set
      {
        _selected = value;
      }
    }
    public void DrawOutline() //draw outline to show that its highlighted
    {
      SplashKit.FillRectangle(Color.Black,X - 2, Y - 2, _width + 4, _height  + 4);
    }
  }
}
```

Output for the shapedrawer



and for additional deleted shapes