```
using MultipleShapeKinds;
using SplashKitSDK;
using System.ComponentModel;

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

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

        }

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

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

        public int ShapeCount
        {
            get
            {
                return _shapes.Count;
            }
        }

        public Color BackGround
        {
            get
```

```
50              {
51                  return _background;
52              }
53              set
54              {
55                  _background = value;
56              }
57          }
58
59          public void Draw()
60          {
61              SplashKit.ClearScreen(_background);
62              for (int i = 0; i < ShapeCount; i++)
63              {
64                  _shapes[i].Draw();
65              }
66          }
67
68          public void SelectShapeAt(Point2D pt)
69          {
70              foreach (Shape s in _shapes)
71              {
72                  if (s.IsAt(pt))
73                  {
74                      s.Selected = true;
75                  }
76                  else
77                  {
78                      s.Selected = false;
79                  }
80              }
81          }
82
83          public void AddShape(Shape s)
84          {
85              _shapes.Add(s);
86          }
87
88          public void RemoveShape(Shape s)
89          {
90              _ = _shapes.Remove(s);
91          }
92      }
93  }
94
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using SplashKitSDK;
7
8  namespace MultipleShapeKinds
9  {
10     public class MyCircle:Shape
11     {
12         private int _radius;
13
14         public MyCircle() : this(Color.Blue, 0.0f, 0.0f, 50+73)
15         {
16
17         }
18
19         public MyCircle(Color color, float x, float y, int radius) : base
             (color)
20         {
21             _radius = radius;
22         }
23
24         public int Radius
25         {
26             get
27             {
28                 return _radius;
29             }
30             set
31             {
32                 _radius = value;
33             }
34         }
35
36         public override void Draw()
37         {
38             if(Selected)
39             {
40                 DrawOutline();
41             }
42             SplashKit.FillCircle(Color, X, Y, _radius);
43         }
44
45         public override void DrawOutline()
46         {
47             SplashKit.FillCircle(Color.Black, X, Y, _radius + 2);
48         }
```

```
49
50          public override bool IsAt(Point2D pt)
51          {
52              if (Math.Abs(pt.X - X) < _radius && Math.Abs(pt.Y - Y) <
                  _radius)
53              {
54                  return true;
55              }
56              else return false;
57          }
58      }
59  }
60
```

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

namespace MultipleShapeKinds
{
    public class MyLine:Shape
    {
        private float _endX;
        private float _endY;

        public MyLine() : this(Color.Red, 0.0f, 0.0f, 10, 10)
        {

        }

        public MyLine(Color color, float startX, float startY, float endX,
          float endY) : base(color)
        {
            X = startX;
            Y = startY;
        }

        public float EndX
        {
            get
            {
                return _endX;
            }
            set
            {
                _endX = value;
            }
        }

        public float EndY
        {
            get
            {
                return _endY;
            }
            set
            {
                _endY = value;
            }
        }
```

```
49
50          public override void Draw()
51          {
52              if (Selected)
53              {
54                  DrawOutline();
55              }
56              SplashKit.DrawLine(Color, X, Y, _endX, _endY);
57          }
58
59          public override void DrawOutline()
60          {
61              SplashKit.FillCircle(Color.Black, X, Y, 2);
62              SplashKit.FillCircle(Color.Black, _endX, _endY, 2);
63          }
64
65          public override bool IsAt(Point2D pt)
66          {
67              if (SplashKit.PointOnLine(pt, SplashKit.LineFrom(X, Y, _endX,
                  _endY)))
68              {
69                  return true;
70              }
71              else return false;
72          }
73      }
74  }
75
```

```
1  using SplashKitSDK;
2  using System;
3
4  namespace MultipleShapeKinds
5  {
6      public class MyRectangle:Shape
7      {
8          private int _width;
9          private int _height;
10         public MyRectangle() : this(Color.Green, 0.0f, 0.0f, 173, 173)
11         {
12
13         }
14
15         public MyRectangle(Color color, float x, float y, int width, int
             height) : base(color)
16         {
17             X = x;
18             Y = y;
19             _width = width;
20             _height = height;
21         }
22
23         public int Width
24         {
25             get
26             {
27                 return _width;
28             }
29             set
30             {
31                 _width = value;
32             }
33         }
34
35         public int Height
36         {
37             get
38             {
39                 return _height;
40             }
41             set
42             {
43                 _height = value;
44             }
45         }
46
47         public override void Draw()
48         {
```

```
49              if (Selected)
50              {
51                  DrawOutline();
52              }
53              SplashKit.FillRectangle(Color, X, Y, _width, _height);
54          }
55
56          public override void DrawOutline()
57          {
58              SplashKit.FillRectangle(Color.Black , X - (5 + 3), Y - (5 + 3),
                   _width + 2 * (5 + 3), _height + 2 * (5 + 3));
59          }
60
61          public override bool IsAt(Point2D pt)
62          {
63              if (pt.X > X && pt.X < X + _width && pt.Y > Y && pt.Y < Y +
                  _height)
64              {
65                  return true;
66              }
67              else
68              {
69                  return false;
70              }
71          }
72      }
73  }
74
```

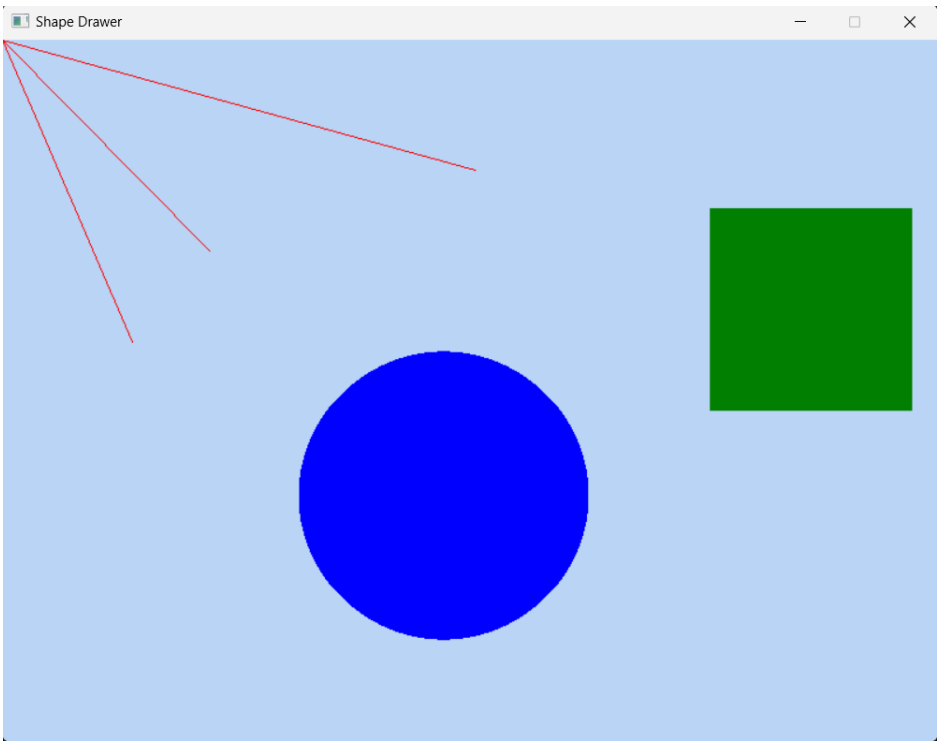
```
1  using System;
2  using MultipleShapeKinds;
3  using SplashKitSDK;
4
5  namespace MultipleShapeKinds
6  {
7      public class Program
8      {
9          private enum ShapeKind
10         {
11             Rectangle,
12             Circle,
13             Line
14         }
15         public static void Main()
16         {
17             ShapeKind kindToAdd = ShapeKind.Circle;
18
19             Window window = new Window("Shape Drawer", 800, 600);
20
21             Drawing myDrawing = new Drawing();
22
23             int count=0;
24             do
25             {
26                 SplashKit.ProcessEvents();
27                 SplashKit.ClearScreen();
28
29                 if(SplashKit.KeyTyped(KeyCode.RKey))
30                 {
31                     kindToAdd = ShapeKind.Rectangle;
32                     Console.WriteLine("rectangle");
33                     count = 0;
34                 }
35
36                 if(SplashKit.KeyTyped(KeyCode.CKey))
37                 {
38                     kindToAdd = ShapeKind.Circle;
39                     Console.WriteLine("circle");
40                     count = 0;
41                 }
42
43                 if (SplashKit.KeyTyped(KeyCode.LKey))
44                 {
45                     kindToAdd = ShapeKind.Line;
46                     Console.WriteLine("line");
47                     count = 0;
48                 }
49
```

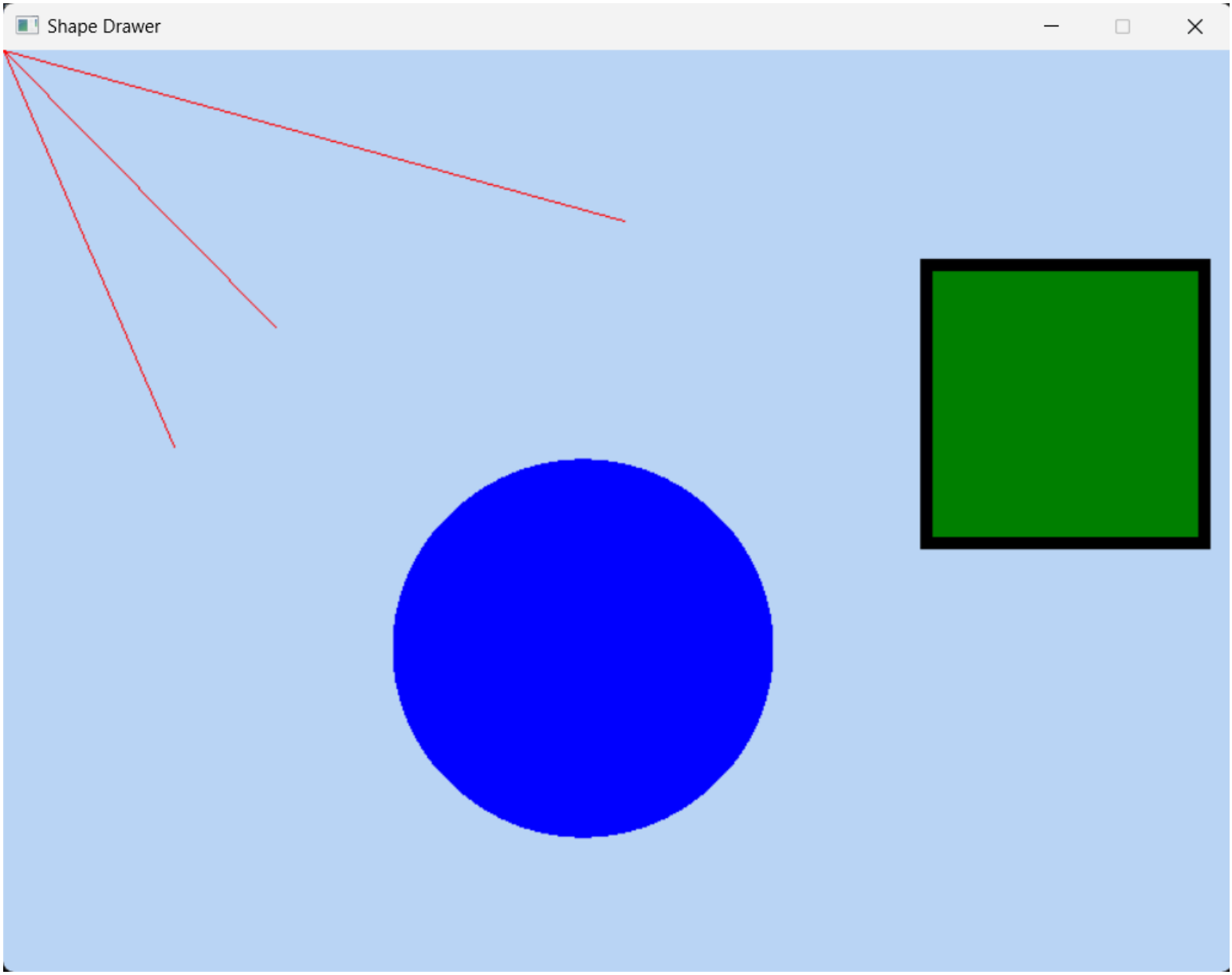
```
50                  if (SplashKit.MouseClicked(MouseButton.LeftButton) &&
                      count<3)
51                  {
52                      Shape newShape;
53
54                      switch(kindToAdd)
55                      {
56                          case ShapeKind.Circle:
57                              newShape = new MyCircle();
58                              break;
59
60                          case ShapeKind.Line:
61                              newShape = new MyLine();
62                              count++;
63                              break;
64
65                          default:
66                              newShape = new MyRectangle();
67                              break;
68
69                      }
70
71
72                      newShape.X = SplashKit.MouseX();
73                      newShape.Y = SplashKit.MouseY();
74                      myDrawing.AddShape(newShape);
75                  }
76
77                  if(SplashKit.KeyTyped(KeyCode.SpaceKey))
78                  {
79                      myDrawing.BackGround = SplashKit.RandomColor();
80                  }
81
82                  if(SplashKit.MouseClicked(MouseButton.RightButton))
83                  {
84                      myDrawing.SelectShapeAt(SplashKit.MousePosition());
85                  }
86
87                  if(SplashKit.KeyTyped(KeyCode.DeleteKey) ||
                      SplashKit.KeyTyped(KeyCode.BackspaceKey))
88                  {
89                      foreach(Shape s in myDrawing.SelectedShapes)
90                      {
91                          myDrawing.RemoveShape(s);
92                      }
93                  }
94
95                  myDrawing.Draw();
96                  SplashKit.RefreshScreen();
```

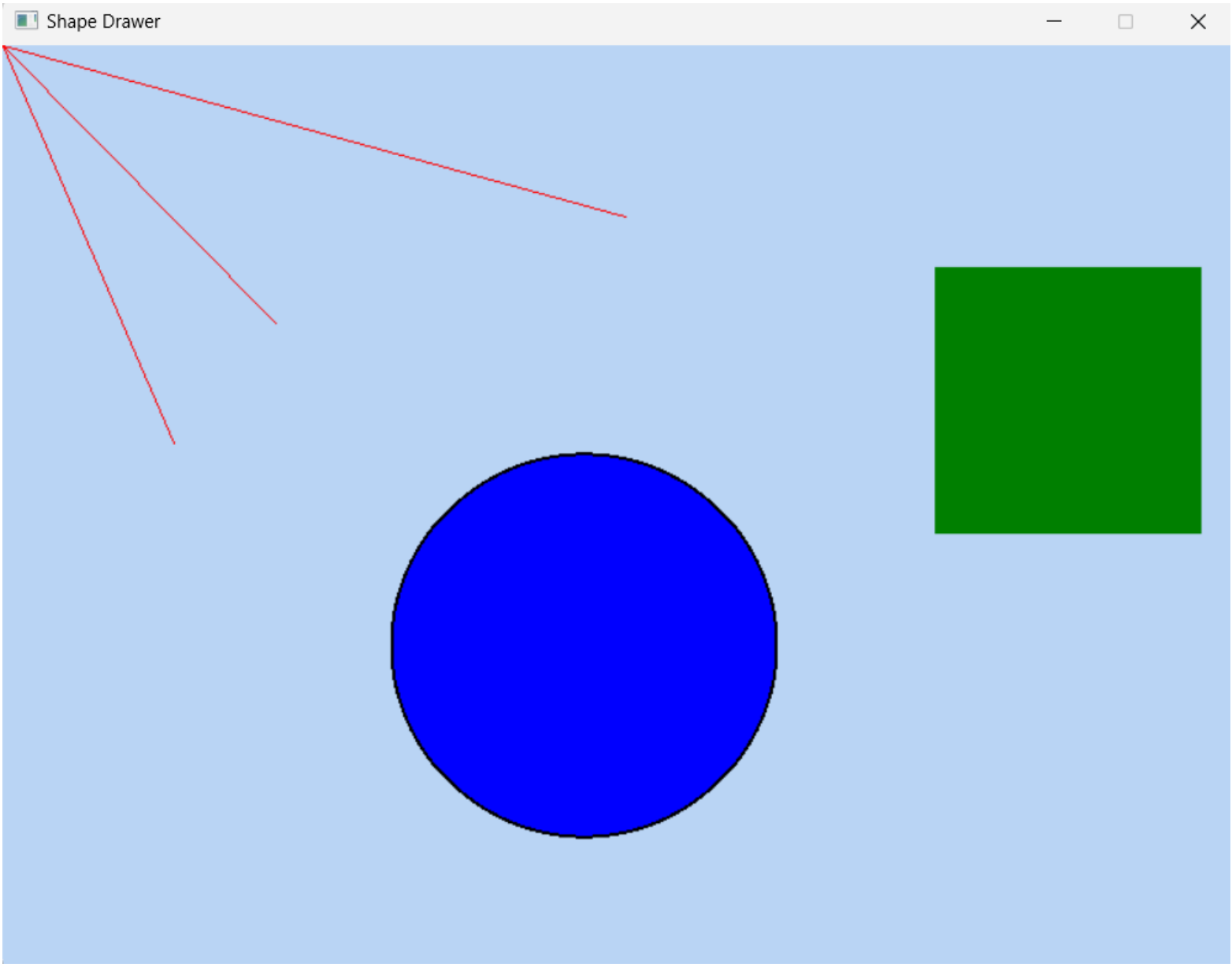
```
 97              } while (!window.CloseRequested);
 98          }
 99
100      }
101  }
102
```

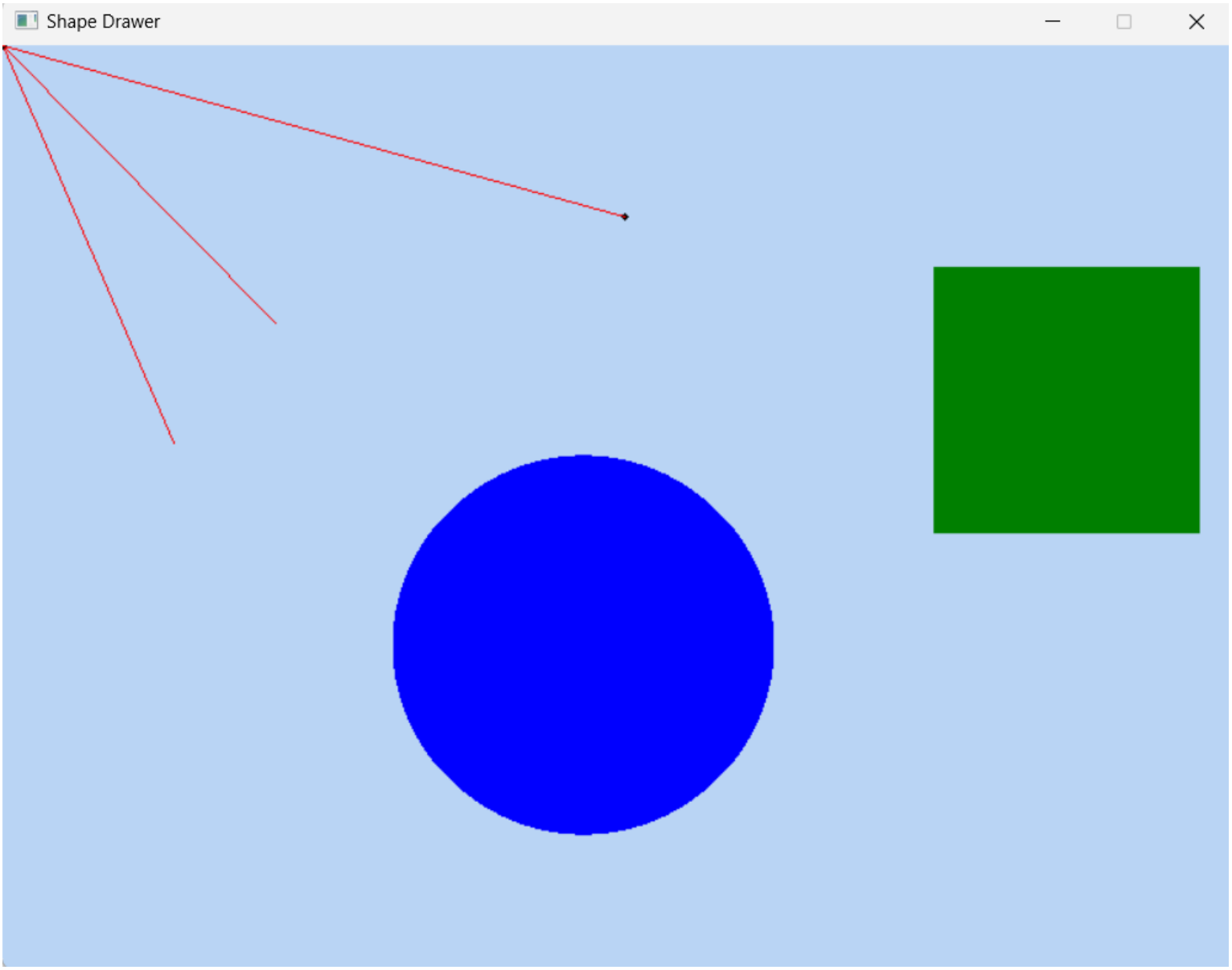
```
1  using SplashKitSDK;
2  using System;
3
4  namespace MultipleShapeKinds
5  {
6      public abstract class Shape
7      {
8          private Color _color;
9          private float _x;
10         private float _y;
11         private bool _selected;
12
13         public Shape() : this(Color.Yellow)
14         {
15
16         }
17
18         public Shape(Color color)
19         {
20             _color = color;
21             _x = 0.0f;
22             _y = 0.0f;
23         }
24
25         public Color Color
26         {
27             get
28             {
29                 return _color;
30             }
31             set
32             {
33                 _color = value;
34             }
35         }
36
37         public float X
38         {
39             get
40             {
41                 return _x;
42             }
43             set
44             {
45                 _x = value;
46             }
47         }
48
49         public float Y
```

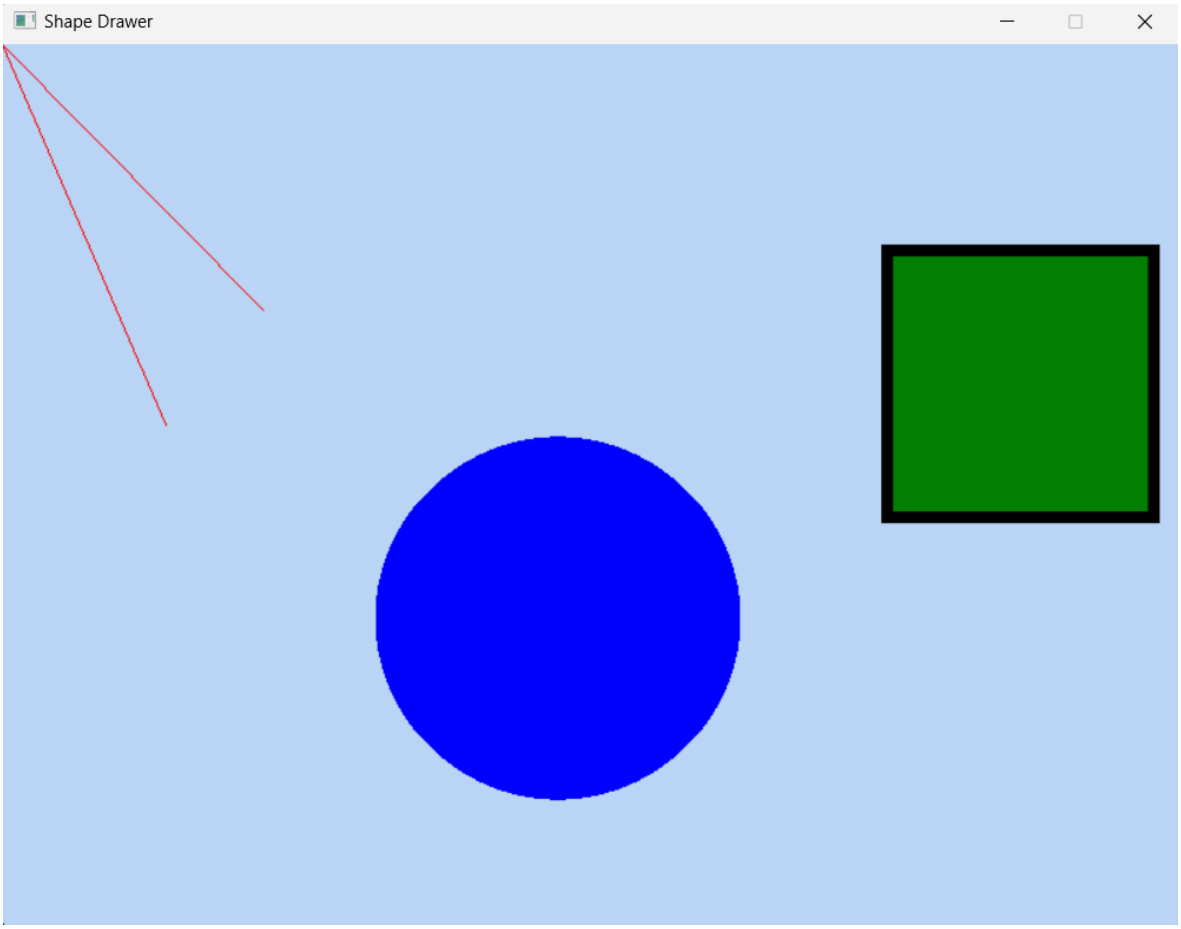
```
50          {
51              get
52              {
53                  return _y;
54              }
55              set
56              {
57                  _y = value;
58              }
59          }
60
61          public bool Selected
62          {
63              get
64              {
65                  return _selected;
66              }
67              set
68              {
69                  _selected = value;
70              }
71          }
72
73          public abstract void Draw();
74
75          public abstract Boolean IsAt(Point2D pt);
76
77          public abstract void DrawOutline();
78      }
79  }
80
```

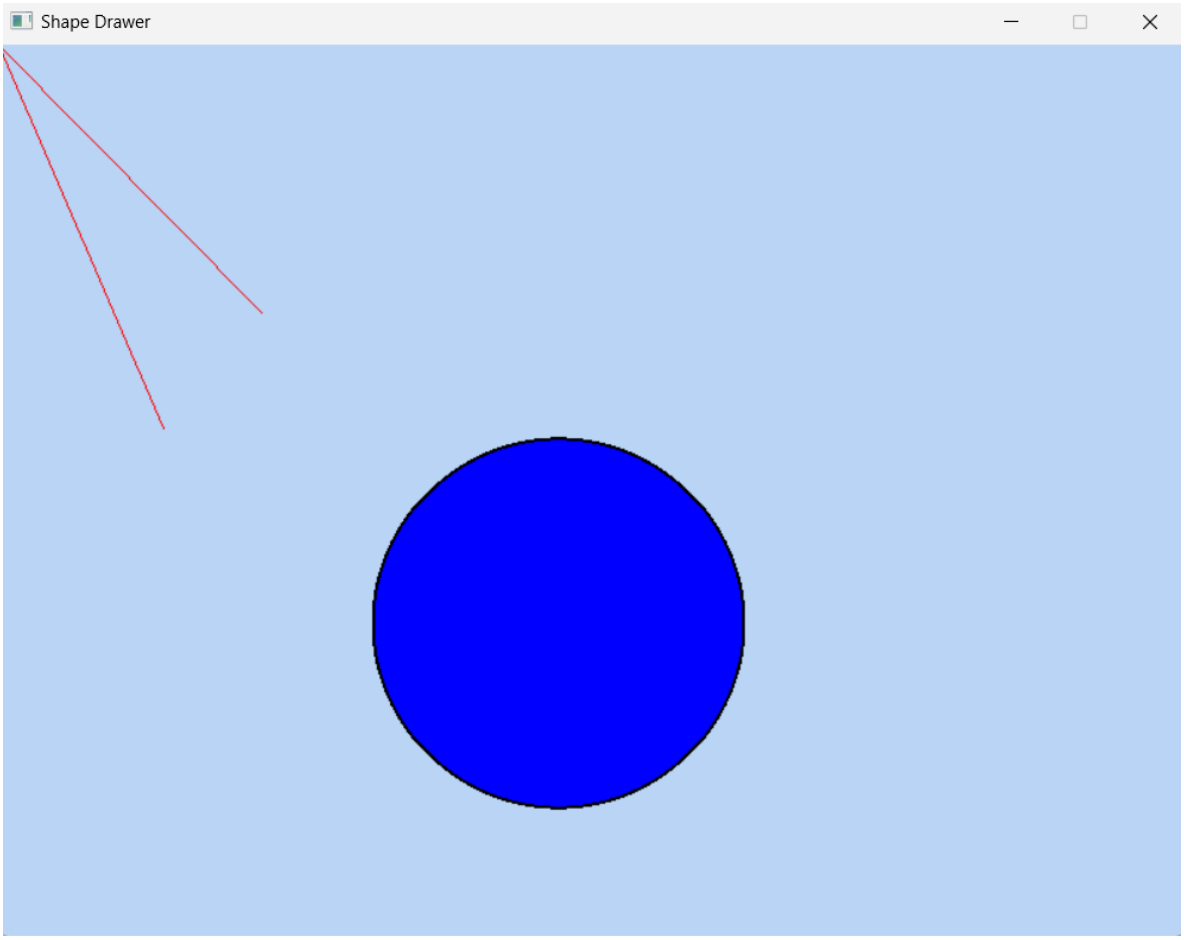


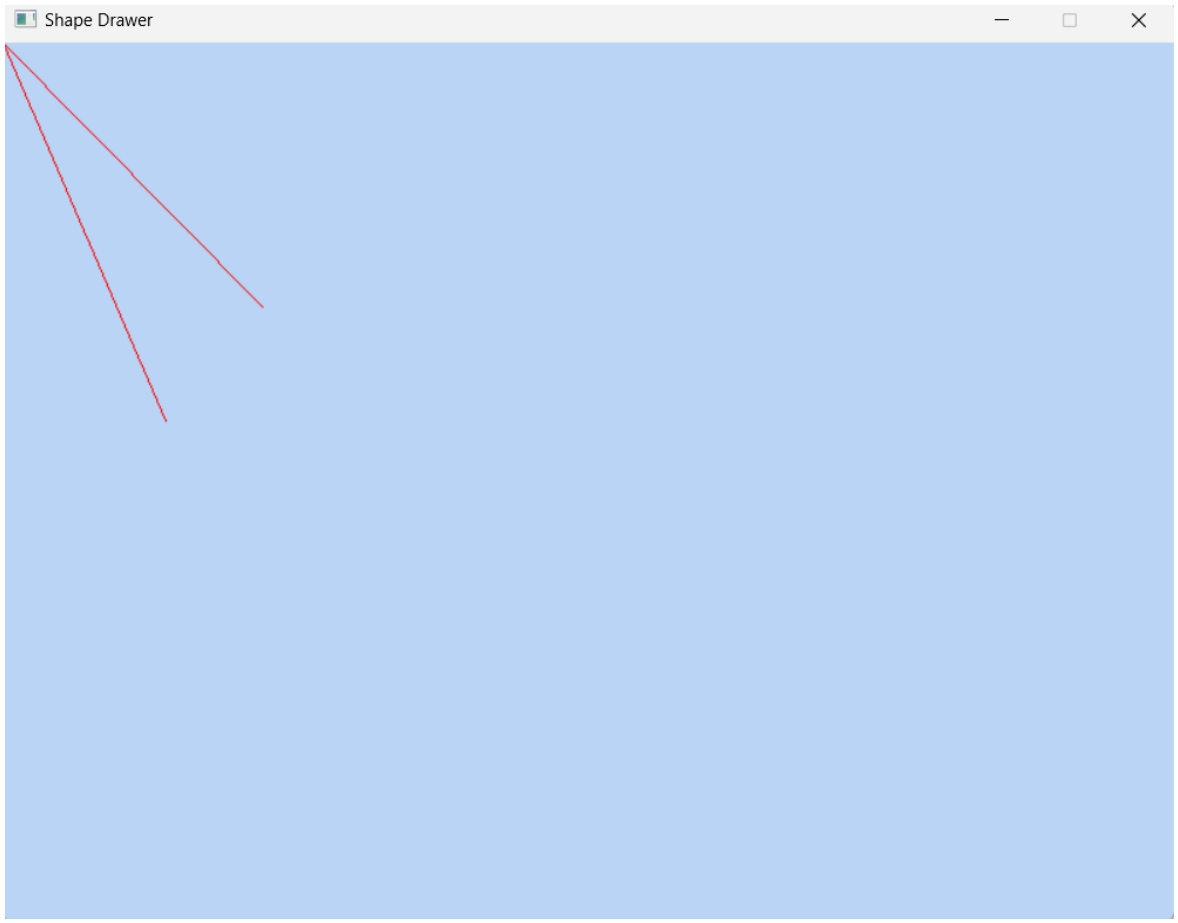

Shape Drawer

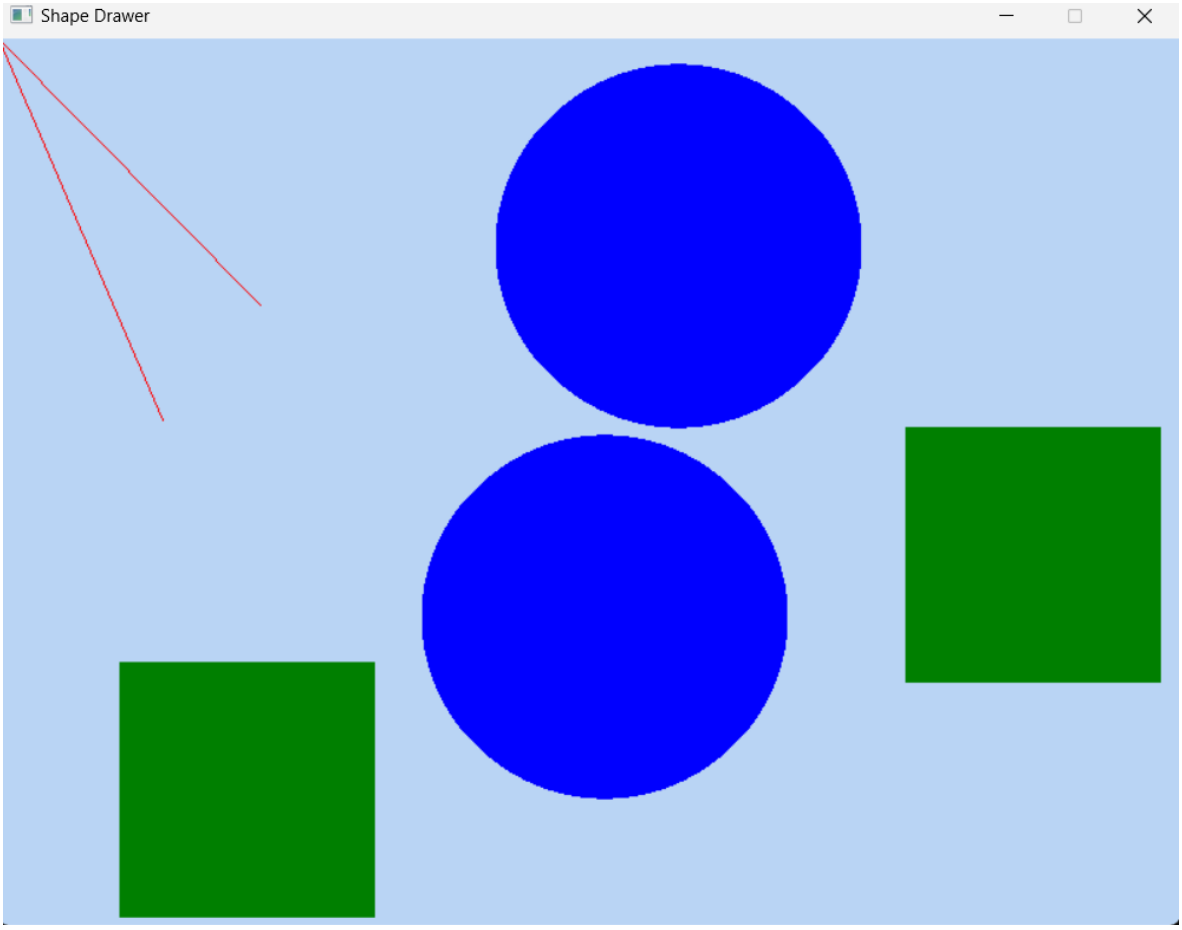

Shape Drawer


Shape Drawer


Shape Drawer


Shape Drawer


Shape Drawer


Shape Drawer