33 lines (29 sloc) | 913 Bytes

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.IO;
7  using SplashKitSDK;
8
9  namespace _5._2C_Not_Complete
10 {
11     public static class ExtensionMethods
12     {
13         public static int ReadInteger(this StreamReader reader)
14         {
15             return Convert.ToInt32(reader.ReadLine());
16         }
17
18         public static float ReadSingle(this StreamReader reader)
19         {
20             return Convert.ToSingle(reader.ReadLine());
21         }
22
23         public static Color ReadColor(this StreamReader reader)
24         {
25             return Color.RGBColor(reader.ReadSingle(), reader.ReadSingle(), reader.ReadSingle(
26         }
27
28         public static void WriteColor(this StreamWriter writer, Color color)
29         {
30             writer.WriteLine("{0}\n{1}\n{2}", color.R, color.G, color.B);
31         }
32     }
```

```
33  }
```

84 lines (69 sloc) | 2.09 KB

```
1  using SplashKitSDK;
2  using System.IO;
3
4  namespace _5._2C_Not_Complete
5  {
6      public abstract class Shape
7      {
8          //! Field Declerations
9          private Color _color;
10         private float _x, _y;
11         private bool _selected;
12
13         //! Constructors
14         public Shape()
15             : this(Color.Red, false)
16         {
17
18         }
19
20         public Shape(Color color, bool selected)
21         {
22             this._color = color;
23             this._selected = selected;
24         }
25
26         public Shape(Color color, float x, float y, bool selected)
27             : this(color, selected)
28         {
29             this._x = x;
30             this._y = x;
31         }
32
```

```
33
34          //! Properties
35          public Color Color
36          {
37              get { return this._color; }
38              set {this._color = value; }
39          }
40
41          public float X
42          {
43              get { return this._x; }
44              set { this._x = value; }
45          }
46
47          public float Y
48          {
49              get { return this._y; }
50              set { this._y = value; }
51          }
52
53          public bool Selected
54          {
55              get { return this._selected; }
56              set { this._selected = value; }
57          }
58
59          //! Methods
60          //? Takes a point and determines if said point lies within our shape
61          public abstract bool IsAt(Point2D mouseLocation);
62
63          //? Draws the outline of a rectangle
64          public abstract void DrawOutline();
65
66          //? Draws a Rectangle based on parameters and outlines the rectangle if the shape is s
67          public abstract void Draw();
68
69          //! 5.2C Save and Load functionality
70          public virtual void SaveTo(StreamWriter writer)
71          {
72              writer.WriteColor(Color);
73              writer.WriteLine(X);
74              writer.WriteLine(Y); //? possible to add selected if you wanted to load that aswel
75          }
76
77          public virtual void LoadFrom(StreamReader reader)
78          {
79              Color = reader.ReadColor();
80              X = reader.ReadInteger();
81              Y = reader.ReadInteger();
82          }
83      }
84  }
```

80 lines (68 sloc) | 1.97 KB

```
1  using SplashKitSDK;
2  using System.IO;
3
4  namespace _5._2C_Not_Complete
5  {
6      public class MyRectangle : Shape
7      {
8          //! Fields
9          private int _width, _height;
10
11         //! Constructor(s)
12         public MyRectangle()
13         {
14
15         }
16
17         public MyRectangle(Color color, float x, float y, bool selected, int width, int height
18         {
19             _width = width;
20             _height = height;
21         }
22
23         //! Properties
24         public int Width
25         {
26             get { return _width; }
27             set { _width = value; }
28         }
29
30
31         public int Height
32         {
```

```
33              get { return _height; }
34              set { _height = value; }
35          }
36
37          //! Method(s)
38          public override bool IsAt(Point2D mouseLocation)
39          {
40              if (X < mouseLocation.X && mouseLocation.X < (X + Width) && Y < mouseLocation.Y &&
41              {
42                  return true;
43              }
44              else
45              {
46                  return false;
47              }
48          }
49          public override void DrawOutline()
50          {
51              SplashKit.DrawRectangle(Color.Black, (X - 2), (Y - 2), (Width + 4), (Height + 4));
52          }
53
54          public override void Draw()
55          {
56              SplashKit.FillRectangle(Color, X, Y, Width, Height);
57
58              if (Selected)
59              {
60                  DrawOutline();
61              }
62          }
63
64          //! 5.2C Saving and Loading functionality
65          public override void SaveTo(StreamWriter writer)
66          {
67              writer.WriteLine("Rectangle");
68              base.SaveTo(writer);
69              writer.WriteLine(Width);
70              writer.WriteLine(Height);
71          }
72
73          public override void LoadFrom(StreamReader reader)
74          {
75              base.LoadFrom(reader);
76              Width = reader.ReadInteger();
77              Height = reader.ReadInteger();
78          }
79      }
80  }
```

71 lines (60 sloc) | 1.76 KB

```
1  using SplashKitSDK;
2  using System.IO;
3
4  namespace _5._2C_Not_Complete
5  {
6      public class MyCircle : Shape
7      {
8          //! Fields
9          private int _radius;
10
11         //! Constructor(s)
12         public MyCircle()
13         {
14
15         }
16
17         public MyCircle(Color color, float x, float y, bool selected, int radius) : base(color
18         {
19             this._radius = radius;
20         }
21
22         //! Properties
23         public int Radius
24         {
25             get { return _radius; }
26             set { _radius = value; }
27         }
28
29         //! Method(s)
30         public override bool IsAt(Point2D mouseLocation)
31         {
32             Point2D origin = new()
```

```
33              {
34                  //Had to set the points or they kept changing with the mouse location, real pr
35                  X = X,
36                  Y = Y
37              };
38
39              Circle circle = SplashKit.CircleAt(origin, _radius);
40              return SplashKit.PointInCircle(mouseLocation, circle);
41          }
42
43          public override void DrawOutline()
44          {
45              SplashKit.DrawCircle(Color.Black, X, Y, (Radius + 2));
46          }
47
48          public override void Draw()
49          {
50              SplashKit.FillCircle(Color, X, Y, Radius);
51              if (Selected)
52              {
53                  DrawOutline();
54              }
55          }
56
57          //! 5.2C Saving and Loading functionality
58          public override void SaveTo(StreamWriter writer)
59          {
60              writer.WriteLine("Circle");
61              base.SaveTo(writer);
62              writer.WriteLine(Radius);
63          }
64
65          public override void LoadFrom(StreamReader reader)
66          {
67              base.LoadFrom(reader);
68              Radius = reader.ReadInteger();
69          }
70      }
71  }
```

81 lines (67 sloc) | 1.92 KB

```
1   using SplashKitSDK;
2   using System;
3   using System.IO;
4
5   namespace _5._2C_Not_Complete
6   {
7       public class MyLine : Shape
8       {
9           //! Fields
10          private float _length;
11
12          //! Constructors
13          public MyLine()
14              : this(Color.BlueViolet, 0, 0, false, 150)
15          {
16
17          }
18
19          public MyLine(Color color, float x, float y, bool selected, float length) : base(color
20          {
21              this._length = length;
22          }
23
24          //! Properties
25          public float Length
26          {
27              get { return _length; }
28              set { _length = value; }
29          }
30
31
32          //! Methods
```

```
33          public override bool IsAt(Point2D mouseLocation)
34          {
35              Point2D initialPoint = new()
36              {
37                  X = X,
38                  Y = Y
39              };
40
41              Point2D finalPoint = new()
42              {
43                  X = X + Length,
44                  Y = Y
45              };
46
47               Line line = SplashKit.LineFrom(initialPoint, finalPoint);
48              return SplashKit.PointOnLine(mouseLocation, line);
49          }
50
51          public override void DrawOutline()
52          {
53              SplashKit.DrawCircle(Color.GhostWhite, X, Y, 2);
54               SplashKit.DrawCircle(Color.GhostWhite, X + Length, Y, 2);
55          }
56
57          public override void Draw()
58          {
59              SplashKit.DrawLine(Color, X, Y, (X + Length), Y);
60
61              if (Selected)
62              {
63                  DrawOutline();
64              }
65          }
66
67          //! 5.2C Saving and Loading functionality
68          public override void SaveTo(StreamWriter writer)
69          {
70              writer.WriteLine("Line");
71              base.SaveTo(writer);
72              writer.WriteLine(Length);
73          }
74
75          public override void LoadFrom(StreamReader reader)
76          {
77              base.LoadFrom(reader);
78              Length = reader.ReadInteger();
79          }
80      }
81  }
```

154 lines (128 sloc) | 3.94 KB

```
1   using System;
2   using System.Collections.Generic;
3   using System.IO;
4   using SplashKitSDK;
5
6   namespace _5._2C_Not_Complete
7   {
8       class Drawing
9       {
10          //! Fields
11          private readonly List<Shape> _shapes;
12          private Color _background;
13
14
15          //! Constructors
16          //? Default constructor, should draw a white background when initialised.
17          public Drawing(Color background)
18          {
19              _background = background;
20              _shapes = new();
21          }
22
23          public Drawing()
24              : this(Color.White)
25          {
26
27          }
28
29
30          //! Properties
31          public Color Background
32          {
```

```
33              get { return _background; }
34              set { _background = value; }
35          }
36
37          //? Readonly
38          public int ShapeCount
39          {
40              get { return _shapes.Count; }
41          }
42
43          //? Readonly, adds a selected shape to the selectedShapes array
44          public List<Shape> SelectedShapes
45          {
46              get
47              {
48                  List<Shape> selectedShapes = new();
49
50                  foreach(Shape genericShape in _shapes)
51                  {
52                      if (genericShape.Selected)
53                      {
54                          selectedShapes.Add(genericShape);
55                      }
56                  }
57                  return selectedShapes;
58              }
59          }
60
61          //! Methods and Fields
62          public void AddShape(Shape genericShape)
63          {
64              _shapes.Add(genericShape);
65          }
66
67          public void RemoveShape(Shape genericShape)
68          {
69              _shapes.Remove(genericShape);
70          }
71
72          //? Turns selected to true if shape is at mouselocation
73          public void SelectShapesAt(Point2D mouseLocation)
74          {
75              foreach(Shape genericShape in _shapes)
76              {
77                  if (!genericShape.Selected)
78                  {
79                      genericShape.Selected = genericShape.IsAt(mouseLocation);
80                  }
81              }
82          }
83
84          //? Draw da shapes
```

```
85          public void Draw()
86          {
87              SplashKit.ClearScreen(Background);
88
89              foreach (Shape genericShape in _shapes)
90              {
91                  genericShape.Draw();
92              }
93          }
94
95          //! 5.2C Code Relating to saving and loading functionality
96          public void Save(string filename)
97          {
98              StreamWriter writer = new(filename);
99
100             try
101             {
102                 writer.WriteColor(Background);
103                 writer.WriteLine(ShapeCount);
104
105                 foreach (Shape genericShape in _shapes)
106                 {
107                     genericShape.SaveTo(writer);
108                 }
109             }
110             finally
111             {
112                 writer.Close();
113             }
114         }
115
116         public void Load(string filename)
117         {
118             StreamReader reader = new(filename); //TODO create exception to handle opening non
119             try
120             {
121                 Shape genericShape;
122                 int count;
123                 string kind;
124
125                 Background = reader.ReadColor();
126                 count = reader.ReadInteger();
127
128                 _shapes.Clear();
129
130                 for (int i = 0; i < count; i++)
131                 {
132                     kind = reader.ReadLine();
133
134                     genericShape = kind switch
135                     {
136                         "Rectangle" => new MyRectangle(),
```

```
137                         "Circle" => new MyCircle(),
138                         "Line" => new MyLine(),
139                         _ => throw new Exception(kind + "is not a valid ShapeKind"),
140                     };
141
142                     genericShape.LoadFrom(reader);
143                     AddShape(genericShape);
144                 }
145             }
146
147             finally
148             {
149                 reader.Close();
150             }
151         }
152     }
153 }
154
```

127 lines (106 sloc) | 4.33 KB

```
1   using SplashKitSDK;
2   using System;
3
4   namespace _5._2C_Not_Complete
5   {
6       public class Program
7       {
8           private enum ShapeKind
9           {
10              Rectangle,
11              Circle,
12              Line
13          }
14
15          public static void Main()
16          {
17              Window window = new("5.2C: Thomas Horsley - 103071494", 800, 600);
18              Drawing drawing = new();
19
20              ShapeKind kindToAdd = ShapeKind.Circle;
21
22              do
23              {
24                  SplashKit.ProcessEvents();
25                  SplashKit.ClearScreen();
26
27                  Point2D mouseLocation = SplashKit.MousePosition();
28
29                  //! Mouse Functionality
30                  if (SplashKit.MouseClicked(MouseButton.LeftButton))
31                  {
32                      if (kindToAdd == ShapeKind.Rectangle)
```

```
33                      {
34                          Shape rectangleShape = new MyRectangle(Color.Green, 0, 0, false, 100,
35                          {
36                              X = (float)mouseLocation.X,
37                              Y = (float)mouseLocation.Y
38                          };
39
40                          drawing.AddShape(rectangleShape);
41                      }
42                      if (kindToAdd == ShapeKind.Circle)
43                      {
44                          Shape circleShape = new MyCircle(Color.Red, 0, 0, false, 50)
45                          {
46                              X = (float)mouseLocation.X,
47                              Y = (float)mouseLocation.Y
48                          };
49
50                          drawing.AddShape(circleShape);
51                      }
52                      if(kindToAdd == ShapeKind.Line)
53                      {
54                          Shape lineShape = new MyLine(Color.GreenYellow, 0, 0, false, 50)
55                          {
56                              X = (float)mouseLocation.X,
57                              Y = (float)mouseLocation.Y,
58                          };
59
60                          drawing.AddShape(lineShape);
61                      }
62                  }
63
64                  //! Keystroke Functionality
65                  //? Relates keys pressed to shape kind
66                  if (SplashKit.KeyReleased(KeyCode.RKey))
67                  {
68                      kindToAdd = ShapeKind.Rectangle;
69                  }
70                  else if (SplashKit.KeyReleased(KeyCode.CKey))
71                  {
72                      kindToAdd = ShapeKind.Circle;
73                  }
74                  else if (SplashKit.KeyReleased(KeyCode.LKey))
75                  {
76                      kindToAdd = ShapeKind.Line;
77                  } //TODO <--- can i use cases instead?
78
79
80                  //? Checks if shape is selected
81                  if (SplashKit.MouseClicked(MouseButton.RightButton))
82                  {
83                      drawing.SelectShapesAt(mouseLocation);
84                  }
```

```
85
86
87                  //? Changes background color when user presses space
88                  if (SplashKit.KeyReleased(KeyCode.SpaceKey))
89                  {
90                      drawing.Background = SplashKit.RandomRGBColor(255);
91                  }
92
93                  if (SplashKit.KeyReleased(KeyCode.DeleteKey) || SplashKit.KeyReleased(KeyCode.
94                  {
95                      foreach(Shape genericShape in drawing.SelectedShapes)
96                      {
97                          drawing.RemoveShape(genericShape);
98                      }
99                  }
100
101                 //? Saves the data in a text file if Keydown S and Loads on Keydown O
102                 if (SplashKit.KeyReleased(KeyCode.SKey))
103                 {
104                     drawing.Save(Environment.GetFolderPath(Environment.SpecialFolder.Desktop)
105                 }
106
107                 if (SplashKit.KeyReleased(KeyCode.OKey))
108                 {
109                     try
110                     {
111                         drawing.Load(Environment.GetFolderPath(Environment.SpecialFolder.Deskt
112                     }
113                     catch (Exception loadException)
114                     {
115                         Console.Error.WriteLine("Error loading file {0}", loadException.Messag
116                     }
117
118                 }
119
120                 drawing.Draw();
121                 SplashKit.RefreshScreen(60);
122             } while (!window.CloseRequested);
123         }
124     }
125 }
126
127
```