

五專資工三甲

# 視窗程式設計

---

SB060024

莊鈞凱

ZHUANG JUN KAI

## 題目

請各位同學依照上課內容，逐步完成小畫家視窗程式，功能要求如下：

- 使用工具列(ToolBar)、畫布(Canvas)、選單(Menu)與狀態列(StatusBar)元件。
- 使用第三方套件(Xceed extended.wpf.toolkit)的色彩選擇器(ColorPicker)來自行選擇筆刷色彩與填滿色彩，用以繪製線條(Line)、矩形(Rectangle)、橢圓形(Ellipse)與自由線(Polyline)。
- 狀態列呈現滑鼠點選之起點與終點功能，並呈現目前畫布所有各類形狀的個數。
- 將畫布內容存檔成 png 或 jpg 檔。
- 請將你的程式碼與程式執行畫面，輸出並彙整成一個 pdf 檔案來繳交，pdf 檔名請以學號+作業編號方式命名，列如第一個作業檔名為 A123456789ex1.pdf。請注意作業繳交要求、日期與格式。

```
1 using Microsoft.Win32;
2 using System;
3 using System.IO;
4 using System.Linq;
5 using System.Windows;
6 using System.Windows.Controls;
7 using System.Windows.Input;
8 using System.Windows.Media;
9 using System.Windows.Media.Imaging;
10 using System.Windows.Shapes;
11
12 namespace painter
13 {
14     /// <summary>
15     /// Interaction logic for MainWindow.xaml
16     /// </summary>
17     public partial class MainWindow : Window
18     {
19         string shapeType = "Line"; // 要有初始值
20         string actionType = "Draw";
21         int strokeThickness = 1;
22         Color strokeColor = Colors.Red;
23         Color fillColor = Colors.Yellow;
24
25         Point start, dest;
26         public MainWindow()
27         {
28             InitializeComponent();
29             strokeColorPicker.SelectedColor = strokeColor;
30             fillColorPicker.SelectedColor = fillColor;
31         }
32
33         private void ShapeButton_Click(object sender, RoutedEventArgs e)
34         {
35             var targetRadioButton = sender as RadioButton;
36             shapeType = targetRadioButton.Tag.ToString();
37             actionType = "Draw";
38             //MessageBox.Show(shapeType);
39         }
40
41         private void strokeThicknessSlider_ValueChanged(object sender,
42             RoutedPropertyChangedEventArgs<double> e)
43         {
44             strokeThickness = Convert.ToInt32
45                 (strokeThicknessSlider.Value);
46         }
47
48         private void myCanvas_MouseMove(object sender, MouseEventArgs e)
```

```
48     dest = e.GetPosition(myCanvas);
49     DisplayStatus();
50
51     switch (actionType)
52     {
53     case "Draw":
54         if (e.LeftButton == MouseButtonState.Pressed)
55         {
56             switch (shapeType)
57             {
58             case "Line":
59                 var line = myCanvas.Children.OfType<Line>
60                 ().LastOrDefault();
61                 line.X2 = dest.X;
62                 line.Y2 = dest.Y;
63                 break;
64             case "Rectangle":
65                 Point origin2 = new Point
66                 {
67                     X = Math.Min(start.X, dest.X),
68                     Y = Math.Min(start.Y, dest.Y)
69                 };
70                 var rect =
71                 myCanvas.Children.OfType<Rectangle>().LastOrDefault();
72                 rect.SetValue(Canvas.LeftProperty,
73                 origin2.X);
74                 rect.SetValue(Canvas.TopProperty,
75                 origin2.Y);
76                 rect.Width = Math.Abs(dest.X - start.X);
77                 rect.Height = Math.Abs(dest.Y - start.Y);
78                 break;
79             case "Ellipse":
80                 Point origin3 = new Point
81                 {
82                     X = Math.Min(start.X, dest.X),
83                     Y = Math.Min(start.Y, dest.Y)
84                 };
85                 var ellipse =
86                 myCanvas.Children.OfType<Ellipse>().LastOrDefault();
87                 ellipse.SetValue(Canvas.LeftProperty,
88                 origin3.X);
89                 ellipse.SetValue(Canvas.TopProperty,
90                 origin3.Y);
91                 ellipse.Width = Math.Abs(dest.X -
92                 start.X);
93                 ellipse.Height = Math.Abs(dest.Y -
94                 start.Y);
```

```
88             break;
89             case "Polyline":
90                 var polyline =
myCanvas.Children.OfType<Polyline>().LastOrDefault();
91                 polyline.Points.Add(dest);
92                 break;
93             }
94         }
95         break;
96     case "Erase":
97         var shape = e.OriginalSource as Shape;
98         myCanvas.Children.Remove(shape);
99         if (myCanvas.Children.Count == 0) myCanvas.Cursor =
Cursors.Arrow;
100         break;
101     }
102
103
104 }
105
106 private void myCanvas_MouseLeftButtonDown(object sender,
MouseButtonEventArgs e)
107 {
108     start = e.GetPosition(myCanvas);
109     myCanvas.Cursor = Cursors.Cross;
110
111     if(actionType == "Draw")
112     {
113         switch (shapeType)
114         {
115             case "Line":
116                 DrawLine(Colors.Gray, 1);
117                 break;
118             case "Rectangle":
119                 var rect = new Rectangle
120                 {
121                     Stroke = Brushes.Gray,
122                     StrokeThickness = 1,
123                     Fill = Brushes.LightGray
124                 };
125                 myCanvas.Children.Add(rect);
126                 rect.SetValue(Canvas.LeftProperty, start.X);
127                 rect.SetValue(Canvas.TopProperty, start.Y);
128                 break;
129             case "Ellipse":
130                 var ellipse = new Ellipse
131                 {
132                     Stroke = Brushes.Gray,
133                     StrokeThickness = 1,
```

```

134         Fill = Brushes.LightGray
135     };
136     myCanvas.Children.Add(ellipse);
137     ellipse.SetValue(Canvas.LeftProperty, start.X);
138     ellipse.SetValue(Canvas.TopProperty, start.Y);
139     break;
140     case "Polyline":
141         var polyline = new Polyline
142         {
143             Stroke = Brushes.Gray,
144             StrokeThickness = 1,
145             Fill = Brushes.LightGray
146         };
147         myCanvas.Children.Add(polyline);
148         break;
149     }
150 }
151 DisplayStatus();
152 }
153
154 private void DisplayStatus()
155 {
156     int LineCount = myCanvas.Children.OfType<Line>().Count();
157     int rectCount = myCanvas.Children.OfType<Rectangle>().Count();
158     int ElliCount = myCanvas.Children.OfType<Ellipse>().Count();
159     int polylineCount = myCanvas.Children.OfType<Polyline>().Count();
160     coordinateLabel.Content = $"座標點:({Math.Round(start.X)} , {Math.Round(start.Y)}) : ({Math.Round(dest.X)} , {Math.Round(dest.Y)})";
161     shapeLabel.Content = $"Line : {LineCount} Rectangle : {rectCount} Ellipse : {ElliCount}, Polyline: {polylineCount}";
162 }
163
164 private void strokeColorPicker_SelectedColorChanged(object sender, RoutedPropertyChangedEventArgs<Color?> e)
165 {
166     strokeColor = (Color)strokeColorPicker.SelectedColor;
167 }
168
169 private void fillColorPicker_SelectedColorChanged(object sender, RoutedPropertyChangedEventArgs<Color?> e)
170 {
171     fillColor = (Color)fillColorPicker.SelectedColor;
172 }
173
174 private void myCanvas_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

```

```
175     {
176         if (actionType == "Draw")
177         {
178             switch (shapeType)
179             {
180                 case "Line":
181                     var line = myCanvas.Children.OfType<Line>
182                     ().LastOrDefault();
183                     line.Stroke = new SolidColorBrush(strokeColor);
184                     line.StrokeThickness = strokeThickness;
185                     break;
186                 case "Rectangle":
187                     var rect = myCanvas.Children.OfType<Rectangle>
188                     ().LastOrDefault();
189                     rect.Stroke = new SolidColorBrush(strokeColor);
190                     rect.Fill = new SolidColorBrush(fillColor);
191                     rect.StrokeThickness = strokeThickness;
192                     break;
193                 case "Ellipse":
194                     var elli = myCanvas.Children.OfType<Ellipse>
195                     ().LastOrDefault();
196                     elli.Stroke = new SolidColorBrush(strokeColor);
197                     elli.StrokeThickness = strokeThickness;
198                     elli.Fill = new SolidColorBrush(fillColor);
199                     break;
200                 case "Polyline":
201                     var poly = myCanvas.Children.OfType<Polyline>
202                     ().LastOrDefault();
203                     poly.Stroke = new SolidColorBrush(strokeColor);
204                     poly.Fill = new SolidColorBrush(fillColor);
205                     poly.StrokeThickness = strokeThickness;
206                     break;
207             }
208             myCanvas.Cursor = Cursors.Arrow;
209         }
210     }
211
212     private void clear_canva(object sender, RoutedEventArgs e)
213     {
214         myCanvas.Children.Clear();
215     }
216
217     private void eraseButton_Click(object sender, RoutedEventArgs e)
218     {
219         actionType = "Erase";
220         myCanvas.Cursor = Cursors.Hand;
221         DisplayStatus();
222     }
223 }
```

```
220     private void saveCanvas(object sender, RoutedEventArgs e)
221     {
222         SaveFileDialog saveFileDialog = new SaveFileDialog();
223         saveFileDialog.Title = "儲存畫布";
224         saveFileDialog.Filter = "Png檔案|*.png|所有檔案|*.*";
225
226         if (saveFileDialog.ShowDialog() == true)
227         {
228             // Create a RenderTargetBitmap to capture the canvas content
229             RenderTargetBitmap renderBitmap = new RenderTargetBitmap(
230                 (int)myCanvas.ActualWidth,
231                 (int)myCanvas.ActualHeight,
232                 64d, 64d, PixelFormats.Default);
233
234             // Render the canvas to the RenderTargetBitmap
235             renderBitmap.Render(myCanvas);
236
237             // Create a BitmapEncoder (e.g., PNGEncoder) to save the image
238             PngBitmapEncoder encoder = new PngBitmapEncoder();
239             encoder.Frames.Add(BitmapFrame.Create(renderBitmap));
240
241             // Create a file stream using the user-selected file name
242             string fileName = saveFileDialog.FileName;
243             using (FileStream fs = new FileStream(fileName, FileMode.Create))
244             {
245                 encoder.Save(fs);
246             }
247
248             //MessageBox.Show($"Canvas content saved as {fileName}");
249         }
250     }
251
252     private void DrawLine(Color color, int thickness)
253     {
254         Brush stroke = new SolidColorBrush(color);
255         Line line = new Line
256         {
257             Stroke = stroke,
258             X1 = start.X,
259             Y1 = start.Y,
260             X2 = dest.X,
261             Y2 = dest.Y
262         };
263         myCanvas.Children.Add(line);
264     }
265 }
```



266 }

267

```

1 <Window x:Class="painter.MainWindow"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5     xmlns:mc="http://schemas.openxmlformats.org/markup-
        compatibility/2006"
6     xmlns:xctk="http://schemas.xceed.com/wpf/xaml/toolkit"
7     mc:Ignorable="d"
8     Title="2023 wpf painter" Height="551" Width="800">
9     <Grid>
10         <DockPanel LastChildFill="False">
11             <Menu Height="25" DockPanel.Dock="Top">
12                 <MenuItem Header="檔案">
13                     <MenuItem Header="新增畫布" />
14                     <MenuItem Header="儲存畫布" Click="saveCanvas" />
15                 </MenuItem>
16                 <MenuItem Header="工具">
17                     <MenuItem Header="清除畫布" Click="clear_canva" />
18                 </MenuItem>
19             </Menu>
20             <ToolBarTray DockPanel.Dock="Top" Height="45">
21                 <ToolBar Width="489" Height="30"
22                     HorizontalAlignment="Center">
23                     <Label Content="筆刷色彩" />
24                     <xctk:ColorPicker x:Name="strokeColorPicker"
25                         Width="100" DisplayColorAndName="True"
26                         SelectedColorChanged="strokeColorPicker_SelectedColorCha
27                         nged" />
28                     <Slider x:Name="strokeThicknessSlider" Width="100"
29                         VerticalAlignment="Center" Value="1"
30                         IsSnapToTickEnabled="True"
31                         ValueChanged="strokeThicknessSlider_ValueChanged" />
32                     <Label Content="{Binding Value,
33                         ElementName=strokeThicknessSlider}" Width="30"
34                         HorizontalContentAlignment="Center" />
35                     <Label Content="筆刷色彩" />
36                     <xctk:ColorPicker x:Name="fillColorPicker" Width="100"
37                         DisplayColorAndName="True"
38                         SelectedColorChanged="fillColorPicker_SelectedColorChang
39                         ed" />
40                 </ToolBar>
41                 <ToolBar Margin="0,0,-10,0">
42                     <RadioButton Tag="Line" VerticalAlignment="Center"
43                         IsChecked="True" Click="ShapeButton_Click" Width="50">
44                         <Image Height="32" Width="40" Source="/line.png" />
45                     </RadioButton>
46                     <RadioButton Tag="Rectangle" VerticalAlignment="Center"
47                         Click="ShapeButton_Click" Width="50">
48                         <Image Height="32" Width="40" Source="/

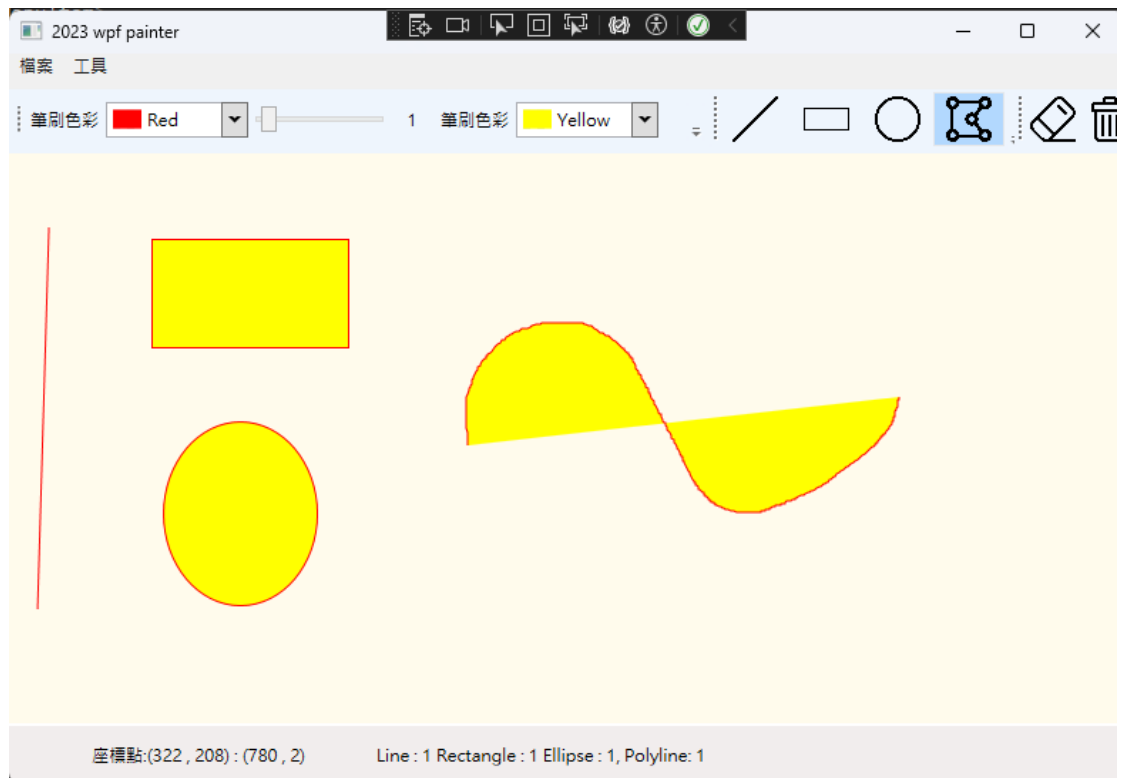
```

```

rectangle.png" />
35     </RadioButton>
36     <RadioButton Tag="Ellipse" VerticalAlignment="Center"
Click="ShapeButton_Click" Width="50">
37         <Image Height="32" Width="40" Source="/circle.png" />
>
38     </RadioButton>
39     <RadioButton Tag="Polyline" VerticalAlignment="Center"
Click="ShapeButton_Click" Width="50">
40         <Image Height="32" Width="40" Source="/
polyline.png" />
41     </RadioButton>
42 </ToolBar>
43 <ToolBar Width="110">
44     <Button FontSize="14" Width="40"
VerticalAlignment="Center" Click="eraseButton_Click">
45         <Image x:Name="erase_button" Height="32" Width="40"
Source="/eraser.png" />
46     </Button>
47     <Button FontSize="14" Width="40"
VerticalAlignment="Center" Click="clear_canva">
48         <Image Height="32" Width="40" Source="/trash.png" />
49     </Button>
50 </ToolBar>
51 </ToolBarTray>
52
53 <Canvas x:Name="myCanvas" DockPanel.Dock="Top" Height="400"
Background=#FFFFFFBEC"
MouseLeftButtonDown="myCanvas_MouseLeftButtonDown"
MouseMove="myCanvas_MouseMove"
MouseLeftButtonUp="myCanvas_MouseLeftButtonUp" />
54 <StatusBar DockPanel.Dock="Bottom" Height="40">
55     <StackPanel Orientation="Horizontal" Margin="50,0,0,0">
56         <Label x:Name="coordinateLabel" Content="座標"
Width="200" Height="25"></Label>
57         <Label x:Name="shapeLabel" Width="250" Height="25" />
58     </StackPanel>
59 </StatusBar>
60 </DockPanel>
61
62 </Grid>
63 </Window>

```

輸出結果：



存檔後的 png 檔案：

