計算機演算法

作業二

班級：資訊三丙

學號：D0683497

姓名：柯利韋

一、作法

(一) 第一個畫面，輸入總點數後先亂數產生 0~500(受限畫面大小)的點

(二) 第二個畫面，列出亂數產的的點及把點標示在圖上

(三) 第三個畫面，點下一步後產生 Convex Hull 的點，列出 Convex Hull 的點，將點連起來

二、程式碼

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace HW2
{
    public sealed class ConvexHull
    {
        // Returns a new list of points representing the convex hul
l of
        // the given set of points. The convex hull excludes collin
ear points.
        // This algorithm runs in O(n log n) time.
        public static IList<Point> MakeHull(IList<Point> points)
        {
            List<Point> newPoints = new List<Point>(points);
            // 排序(小 到 大)
            newPoints.Sort();
            return MakeHullPresorted(newPoints);
        }
        // Returns the convex hull, assuming that each points[i] <=
 points[i + 1]. Runs in O(n) time.
        public static IList<Point> MakeHullPresorted(IList<Point> p
oints)
        {
            if (points.Count <= 1)
            {
                return new List<Point>(points);
            }

            // Andrew's monotone chain algorithm. Positive y coordi
nates correspond to "up"
            // as per the mathematical convention, instead of "down
" as per the computer
            // graphics convention. This doesn't affect the correct
ness of the result.

            List<Point> upperHull = new List<Point>();
            foreach (Point p in points)
            {
                while (upperHull.Count >= 2)
                {
                    Point q = upperHull[upperHull.Count - 1];
                    Point r = upperHull[upperHull.Count - 2];
                    if ((q.X - r.X) * (p.Y - r.Y) >= (q.Y - r.Y) *
(p.X - r.X))
                    {
                        upperHull.RemoveAt(upperHull.Count - 1);
                    }
                    else
                    {
                        break;
                    }
                }
                upperHull.Add(p);
            }
            upperHull.RemoveAt(upperHull.Count - 1);

            IList<Point> lowerHull = new List<Point>();
            for (int i = points.Count - 1; i >= 0; i--)
            {
                Point p = points[i];
                while (lowerHull.Count >= 2)
                {
                    Point q = lowerHull[lowerHull.Count - 1];
                    Point r = lowerHull[lowerHull.Count - 2];
                    if ((q.X - r.X) * (p.Y - r.Y) >= (q.Y - r.Y) *
(p.X - r.X))
                    {
                        lowerHull.RemoveAt(lowerHull.Count - 1);
                    }
                    else
                    {
                        break;
                    }
                }
                lowerHull.Add(p);
            }
            lowerHull.RemoveAt(lowerHull.Count - 1);

            if (!(upperHull.Count == 1 && Enumerable.SequenceEqual(
upperHull, lowerHull)))
            {
                upperHull.AddRange(lowerHull);
            }
            return upperHull;
        }
    }

    public class Point : IComparable<Point>
    {
        /// <summary>
        /// X 軸
        /// </summary>
        public double X { get; set; }

        /// <summary>
        /// Y 軸
        /// </summary>
        public double Y { get; set; }

        public Point(double x, double y)
        {
            this.X = x;
            this.Y = y;
        }

        public int CompareTo(Point other)
        {
            if (X < other.X)
            {
                return -1;
            }
            else if (X > other.X)
            {
                return +1;
            }
            else if (Y < other.Y)
            {
                return -1;
            }
            else if (Y > other.Y)
            {
                return +1;
            }
            else
            {
                return 0;
            }
        }

    }
}
```

```
<Window x:Class="HW2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presenta
tion"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:local="clr-namespace:HW2"
        mc:Ignorable="d"
        Title="MainWindow" Height="540" Width="960">
    <Grid>
        <Frame x:Name="Main" NavigationUIVisibility="Hidden" />
    </Grid>
</Window>
```

```
using System.Windows;

namespace HW2
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            Page1 page1 = new Page1();
            Main.Navigate(page1);
        }
    }
}
```

```
<Page x:Class="HW2.Page1"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentati
on"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:HW2"
      mc:Ignorable="d"
      d:DesignHeight="540" d:DesignWidth="960"
```

```xml
      Title="Page1">

    <Grid>
        <StackPanel Orientation="Horizontal" VerticalAlignment="Cent
er" HorizontalAlignment="Center" Height="60">
            <Label x:Name="label" Content="輸入點數
" Margin="0,0,20,0" FontSize="35" HorizontalAlignment="Center" Verti
calAlignment="Center" />
            <TextBox x:Name="textBox" Text="" HorizontalContentAlign
ment="Center" VerticalContentAlignment="Center" Width="300" FontSize
="30"/>
            <Button Content="確認
" Margin="20,0,0,0" Width="80" Click="Button_Click" />
        </StackPanel>
        <Grid x:Name="progressBarGrid" Width="500" Height="40" Margi
n="0,150,0,0" Visibility="Collapsed">
            <ProgressBar x:Name="progressBar" />
            <TextBlock HorizontalAlignment="Center" VerticalAlignmen
t="Center" FontSize="20">亂數產生中...</TextBlock>
        </Grid>
    </Grid>
</Page>
```

```csharp
using System;
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Threading;

namespace HW2
{
    /// <summary>
    /// Page1.xaml 的互動邏輯
    /// </summary>
    public partial class Page1 : Page
    {
        public Page1()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            if (textBox.Text != String.Empty)
            {
                int pointTotalNumber = Int32.Parse(textBox.Text);

                // 顯示進度條
                progressBarGrid.Visibility = Visibility.Visible;

                List<Point> points = GenerateRandomPoints(pointTotal
Number);

                // 換頁
                Page2 page2 = new Page2(points);
                this.NavigationService.Navigate(page2);
            }
        }

        /* 產生點的亂數 */
        private List<Point> GenerateRandomPoints(int pointTotalNumbe
r)
        {
            var result = new List<Point>();
            Random random = new Random();

            for (int i = 0; i < pointTotalNumber; i++)
            {
                // 進度條
                progressBar.Dispatcher.Invoke(() => progressBar.Valu
e = i / (pointTotalNumber / 100.0), DispatcherPriority.Background);

                // 根據 Canvas 的 Width 跟 Height
                Point p = new Point(random.Next(0, 500), random.Next
(0, 500));
                result.Add(p);
            }

            return result;
        }
    }
}
```

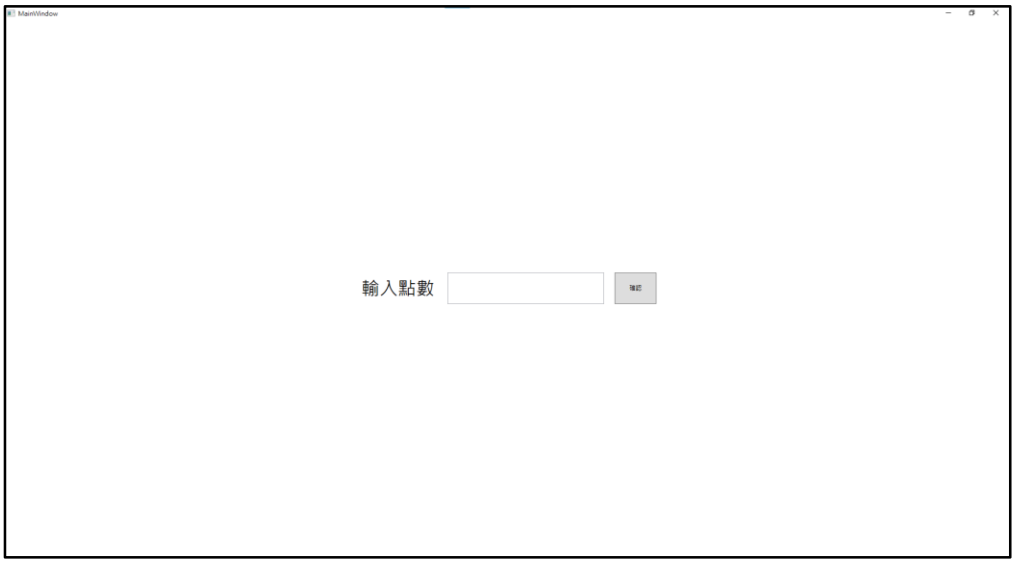
```
<Page x:Class="HW2.Page2"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentati
on"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:HW2"
      mc:Ignorable="d"
      d:DesignHeight="540" d:DesignWidth="960"
      Title="Page2">
    <Grid Height="540" Width="960" HorizontalAlignment="Center" Vert
icalAlignment="Center">
        <Grid Height="540" Width="740" HorizontalAlignment="Left" Ve
rticalAlignment="Center">
            <Border BorderBrush="Black" Width="510" Height="510" Bor
derThickness="2">
                <Canvas x:Name="Main_Canvas" Width="500" Height="500
"
                        HorizontalAlignment="Center" VerticalAlignme
nt="Center"
                        Grid.Column="0" Grid.Row="0"></Canvas>
            </Border>
        </Grid>
        <Grid Height="540" Width="220" HorizontalAlignment="Right" V
erticalAlignment="Center">
            <DataGrid x:Name="data" IsReadOnly="True" Height="450" W
idth="200" Margin="0,20,0,0" AutoGenerateColumns="False"
```

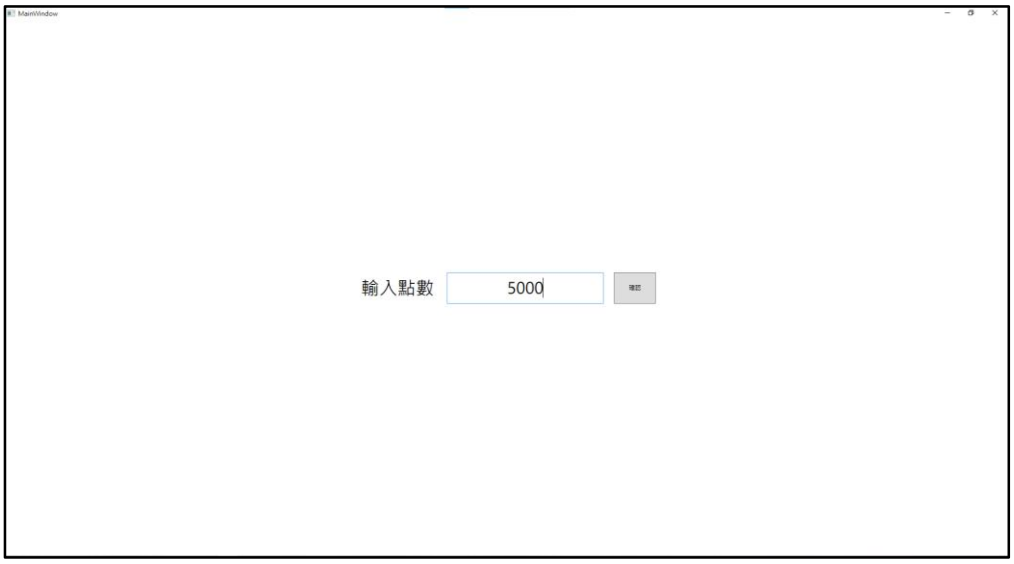
```
                      HorizontalAlignment="Center" VerticalAlignment
="Top">
                <DataGrid.Columns>
                    <DataGridTextColumn Binding="{Binding X}" Header
="X" Width="*" />
                    <DataGridTextColumn Binding="{Binding Y}" Header
="Y" Width="*" />
                </DataGrid.Columns>
            </DataGrid>
            <Button x:Name="buttonNext" Height="30" Width="80" Conte
nt="下一步" HorizontalAlignment="Right" Margin="0,0,15,20"
                    VerticalAlignment="Bottom" Click="buttonNext_Cli
ck"/>
            <Button x:Name="buttonReatsrt" Height="30" Width="80" Co
ntent="重新開始" HorizontalAlignment="Right" Margin="0,0,15,20"
                    VerticalAlignment="Bottom" Visibility="Collapsed
" Click="buttonReatsrt_Click" />
        </Grid>
    </Grid>
</Page>
```

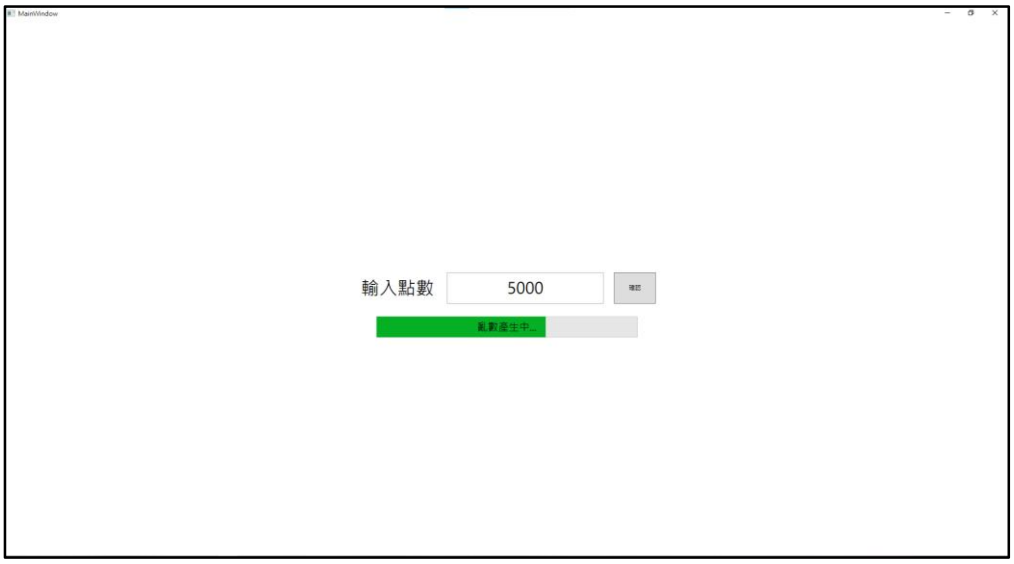
```
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Shapes;

namespace HW2
{
    /// <summary>
    /// Page2.xaml 的互動邏輯
    /// </summary>
    public partial class Page2 : Page
    {
        public Page2(List<Point> points)
        {
            InitializeComponent();
            data.ItemsSource = points;
            /* Canvas 左上為原點，往右為 x 軸正向，往下為 y 軸正向 */
            // 畫點
            foreach (var i in points)
            {
                // 沒有畫點的功能，使用橢圓產生點
                Ellipse el = new Ellipse();
                el.Height = 2.0;
                el.Width = 2.0;
                el.Fill = System.Windows.Media.Brushes.Red;
                el.Stroke = System.Windows.Media.Brushes.Red;
                el.StrokeThickness = 1;
                Canvas.SetLeft(el, i.X);
                Canvas.SetTop(el, i.Y);

                Main_Canvas.Children.Add(el);
            }
        }

        private void buttonNext_Click(object sender, RoutedEventArgs
      e)
        {
            List<Point> points = (List<Point>)data.ItemsSource;

            // ConvexHull 的點
            IList<Point> actual = ConvexHull.MakeHull(points);
            data.ItemsSource = actual;

            // 畫線
            for (int i = 0; i < actual.Count; i++)
            {
                if (i == actual.Count - 1)
                {
                    Line l = new Line();
                    l.Stroke = System.Windows.Media.Brushes.Green;
                    l.X1 = actual[i].X;
                    l.Y1 = actual[i].Y;
                    l.X2 = actual[0].X;
```

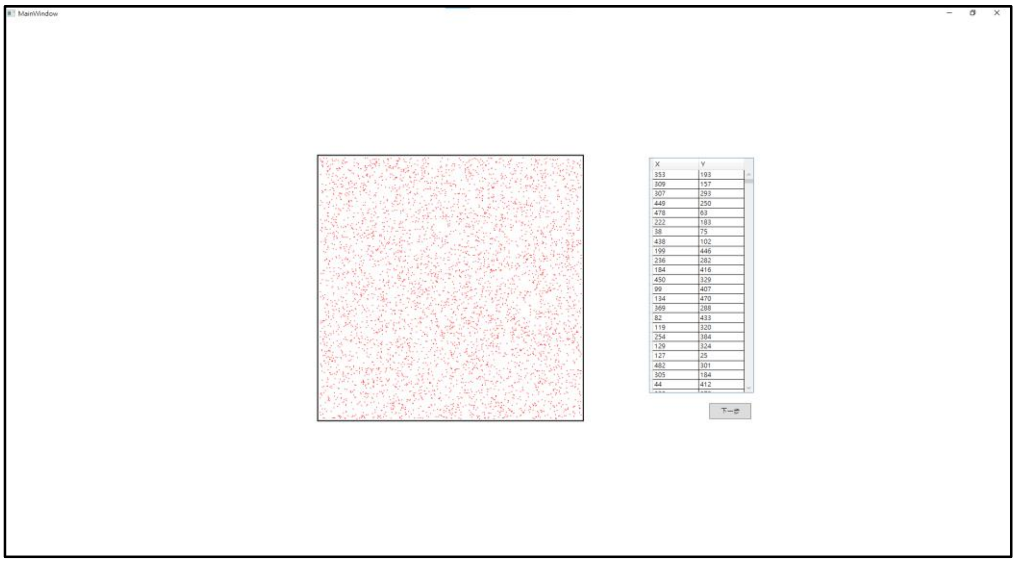
```
                    l.Y2 = actual[0].Y;
                    Main_Canvas.Children.Add(l);
                }
                else
                {
                    Line l = new Line();
                    l.Stroke = System.Windows.Media.Brushes.Green;
                    l.X1 = actual[i].X;
                    l.Y1 = actual[i].Y;
                    l.X2 = actual[i + 1].X;
                    l.Y2 = actual[i + 1].Y;
                    Main_Canvas.Children.Add(l);
                }
            }

            buttonNext.Visibility = Visibility.Collapsed;
            buttonReatsrt.Visibility = Visibility.Visible;
        }

        private void buttonReatsrt_Click(object sender, RoutedEventA
     rgs e)
        {
            Page1 page1 = new Page1();
            this.NavigationService.Navigate(page1);
        }
    }
}
```

三、執行結果

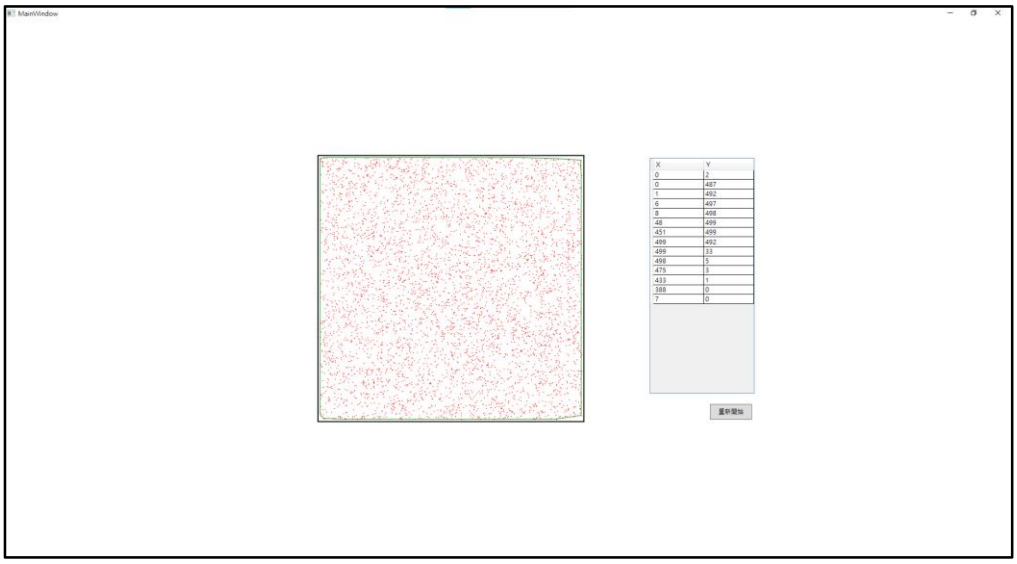

MainWindow
輸入點數
確認


MainWindow
輸入點數
5000
確認


MainWindow
輸入點數
5000
確認
亂數產生中...



X	Y
353	193
309	157
307	293
449	250
478	63
222	183
38	75
438	102
199	446
236	282
184	416
450	329
99	407
134	470
369	288
82	433
119	320
254	384
129	324
127	25
482	301
305	184
44	412



X	Y
0	2
0	487
1	492
6	497
8	498
48	499
451	499
499	492
499	33
498	5
475	3
433	1
388	0
7	0

四、心得

從結果來看，當點越多，圍出來的形狀越接近正方形，而且計算出 Convex Hull 的點所需時間非常的短，甚至比產生亂數的時間還要短。

因為 WPF 的 Canvas 沒有提供畫點的功能，所以用畫圓的方式代替，且 Canvas 是以左上角為原點，跟一般的認知不太一樣，所以剛開始畫圖的時候點跟線對不起來。