計算機演算法

作業二

班級：資訊三丙

學號：D0683497

姓名：柯利韋

1. 作法
   1. 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
   2. 第二個畫面，列出亂數產的的點及把點標示在圖上
   3. 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
2. 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 hull of

        // the given set of points. The convex hull excludes collinear 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> points)

        {

            if (points.Count <= 1)

            {

                return new List<Point>(points);

            }

            // Andrew's monotone chain algorithm. Positive y coordinates correspond to "up"

            // as per the mathematical convention, instead of "down" as per the computer

            // graphics convention. This doesn't affect the correctness 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/presentation"

        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/presentation"

      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="Page1">

    <Grid>

        <StackPanel Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Center" Height="60">

            <Label x:Name="label" Content="輸入點數" Margin="0,0,20,0" FontSize="35" HorizontalAlignment="Center" VerticalAlignment="Center" />

            <TextBox x:Name="textBox" Text="" HorizontalContentAlignment="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" Margin="0,150,0,0" Visibility="Collapsed">

            <ProgressBar x:Name="progressBar" />

            <TextBlock HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="20">亂數產生中...</TextBlock>

        </Grid>

    </Grid>

</Page>

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(pointTotalNumber);

                // 換頁

                Page2 page2 = new Page2(points);

                this.NavigationService.Navigate(page2);

            }

        }

        /\* 產生點的亂數 \*/

        private List<Point> GenerateRandomPoints(int pointTotalNumber)

        {

            var result = new List<Point>();

            Random random = new Random();

            for (int i = 0; i < pointTotalNumber; i++)

            {

                // 進度條

                progressBar.Dispatcher.Invoke(() => progressBar.Value = 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/presentation"

      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" VerticalAlignment="Center">

        <Grid Height="540" Width="740" HorizontalAlignment="Left" VerticalAlignment="Center">

            <Border BorderBrush="Black" Width="510" Height="510" BorderThickness="2">

                <Canvas x:Name="Main\_Canvas" Width="500" Height="500"

                        HorizontalAlignment="Center" VerticalAlignment="Center"

                        Grid.Column="0" Grid.Row="0"></Canvas>

            </Border>

        </Grid>

        <Grid Height="540" Width="220" HorizontalAlignment="Right" VerticalAlignment="Center">

            <DataGrid x:Name="data" IsReadOnly="True" Height="450" Width="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" Content="下一步" HorizontalAlignment="Right" Margin="0,0,15,20"

                    VerticalAlignment="Bottom" Click="buttonNext\_Click"/>

            <Button x:Name="buttonReatsrt" Height="30" Width="80" Content="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, RoutedEventArgs e)

        {

            Page1 page1 = new Page1();

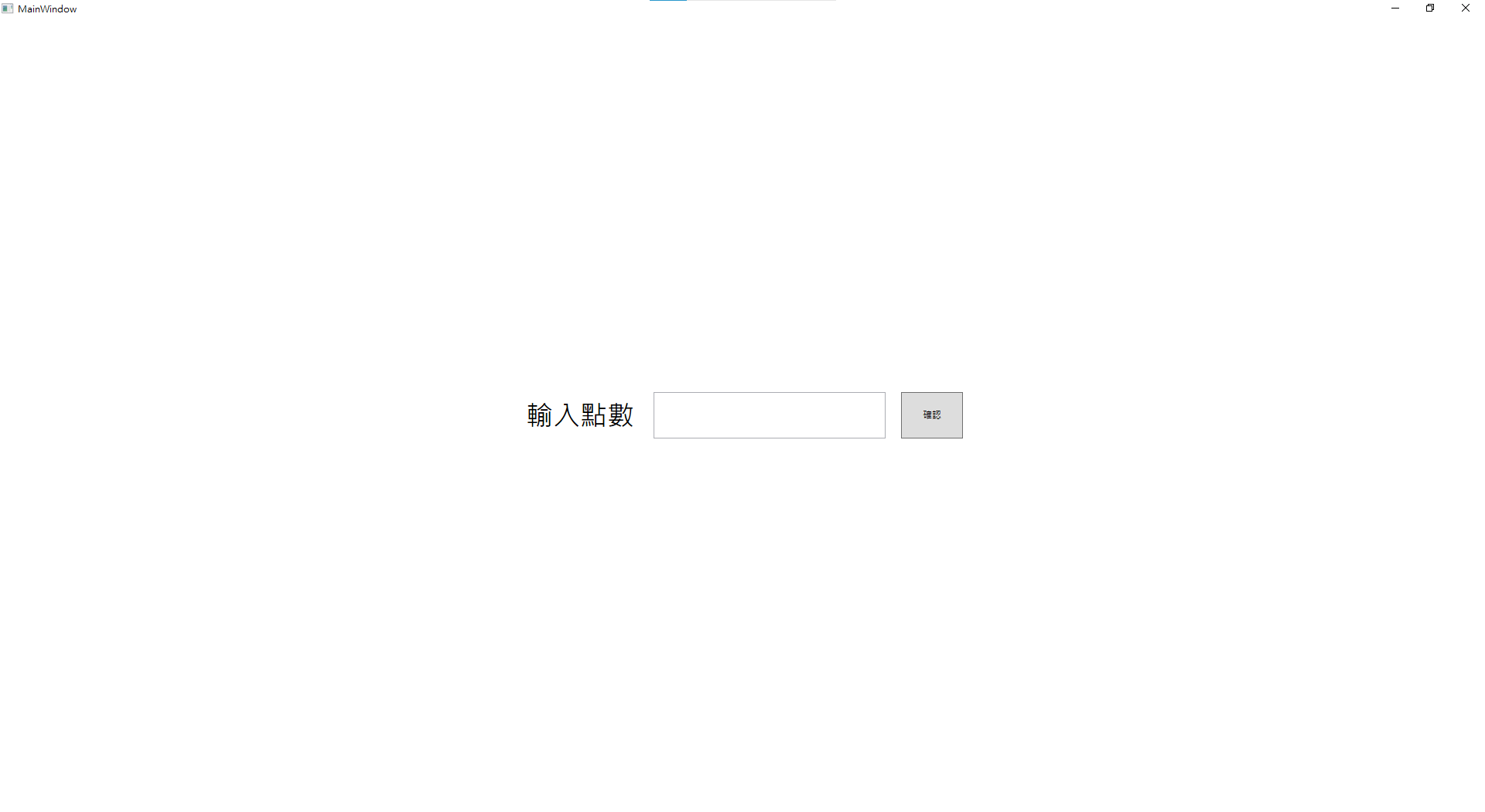
            this.NavigationService.Navigate(page1);

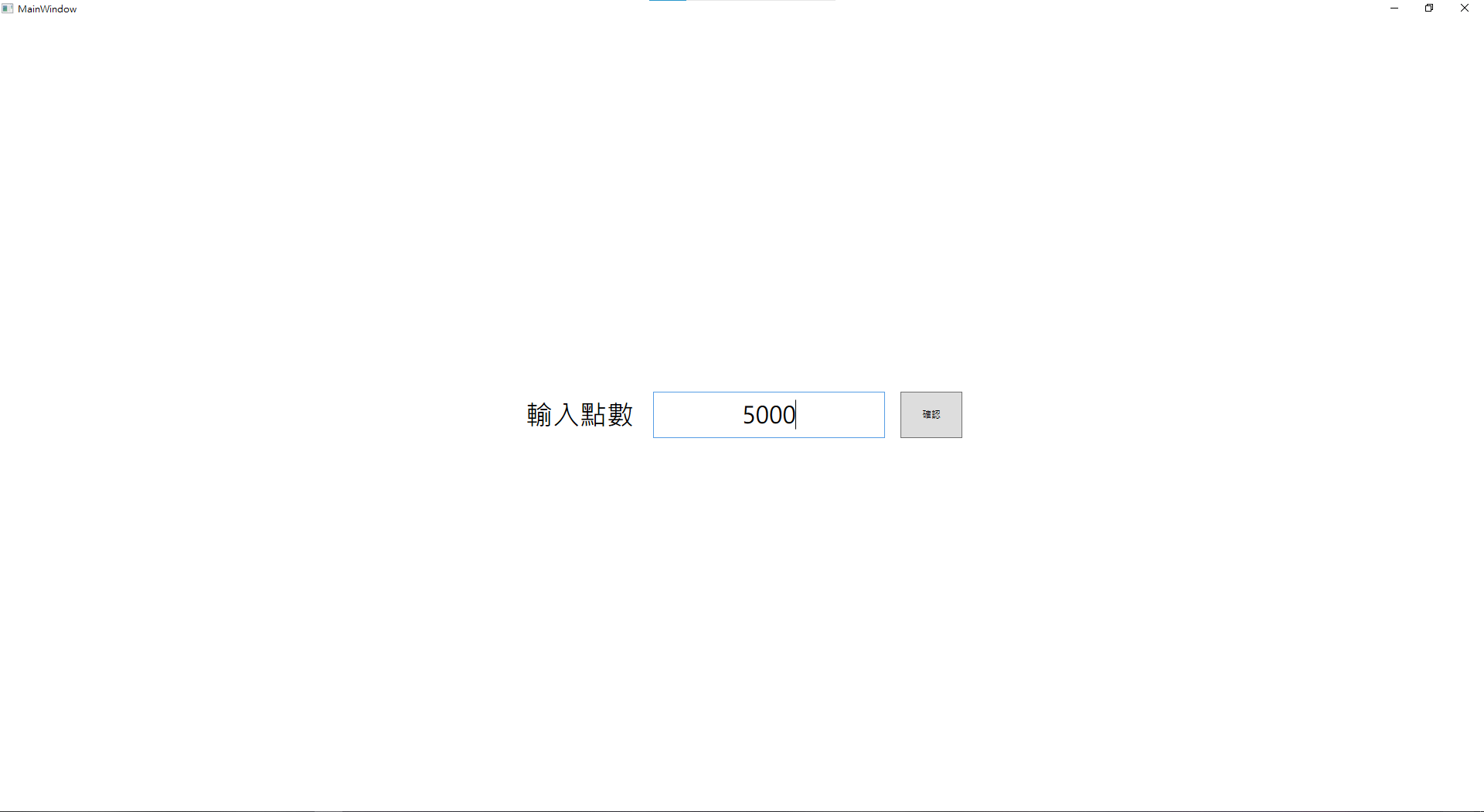
        }

    }

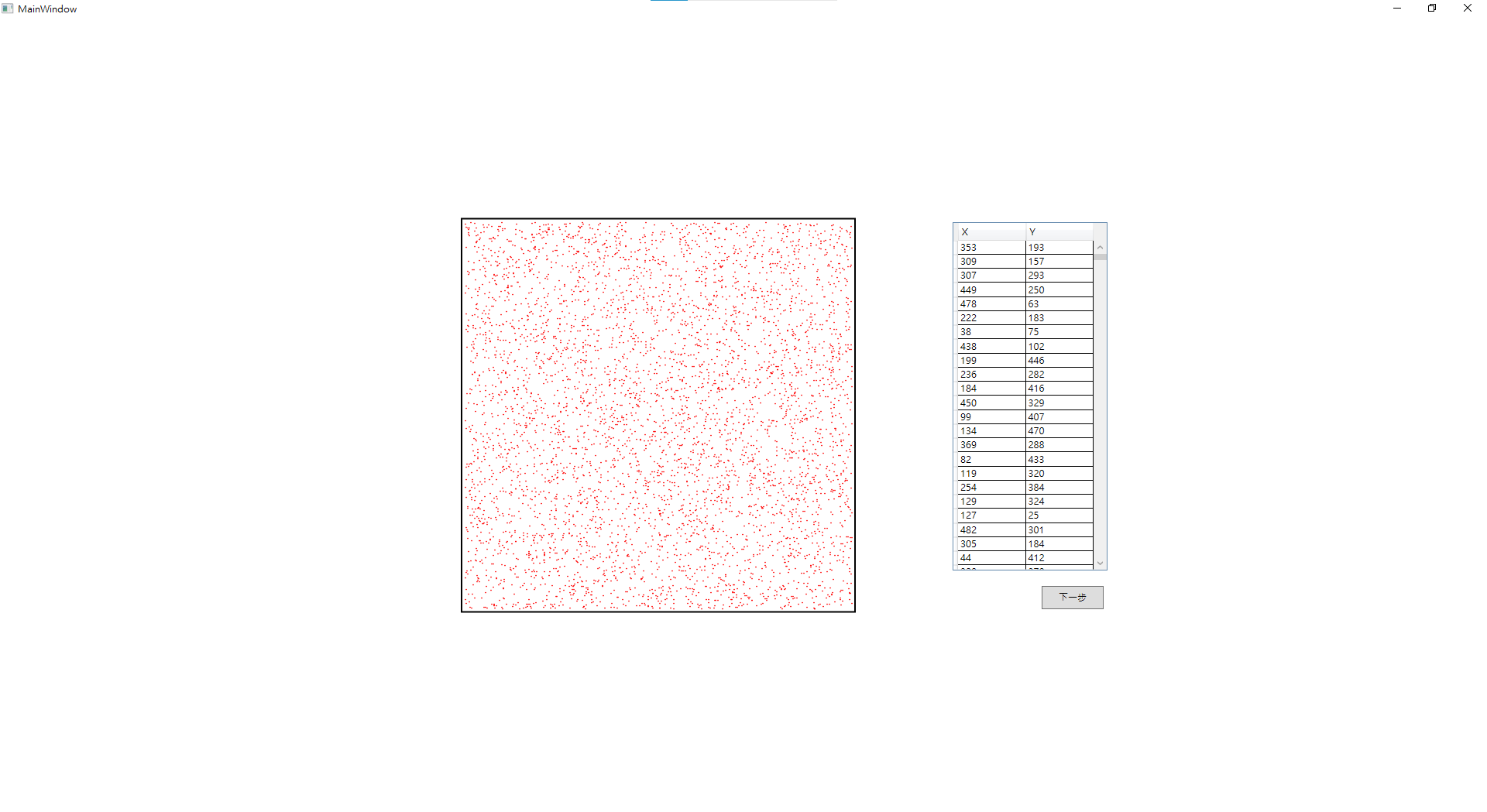
}

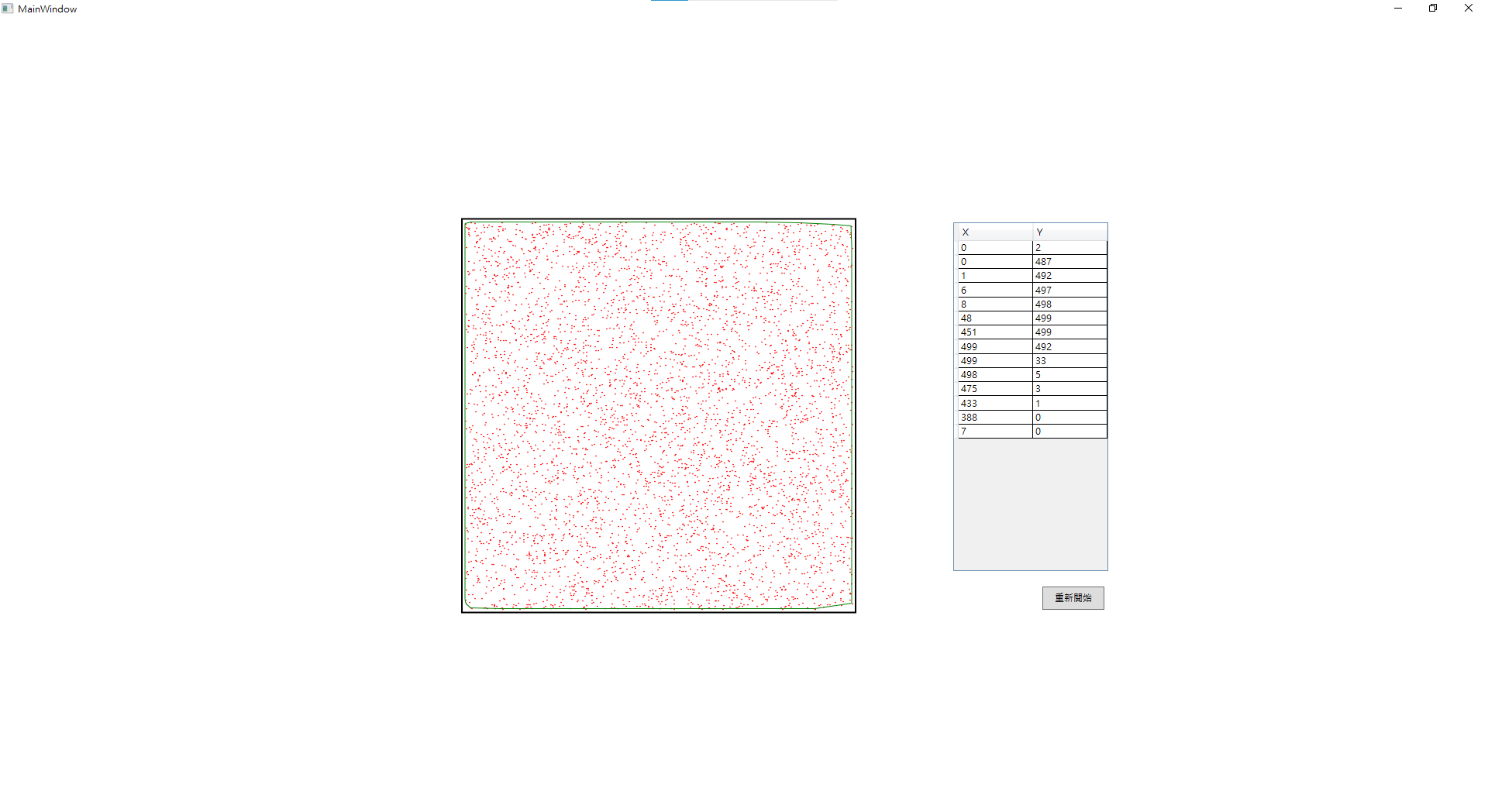
1. 執行結果











1. 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。