<Window x:Class="MyDigiMap.MainWindow"

        xmlns="<http://schemas.microsoft.com/winfx/2006/xaml/presentation>"

        xmlns:x="<http://schemas.microsoft.com/winfx/2006/xaml>"

        Title="My DigiMap" Height="440" Width="420" >

    <Grid>

        <Grid.ColumnDefinitions>

            <ColumnDefinition Width="406" />

        </Grid.ColumnDefinitions>

        <Canvas Grid.Column="0" Name="cnvMain">

            <Image Source="Resources/openStreetMap.jpg"></Image>

            <Path Name="patDirection" Stroke="Firebrick" StrokeThickness="3" ></Path>

            <Ellipse Name="elpBegin" Canvas.Left="-50" Width="16" Height="16" Fill="Lime" Stroke="Green" StrokeThickness="2"></Ellipse>

            <Ellipse Name="elpEnd" Canvas.Left="-50" Width="16" Height="16" Fill="Yellow" Stroke="OrangeRed" StrokeThickness="2"></Ellipse>

            <StackPanel Width="200" Canvas.Bottom="10" Canvas.Left="10" Background="Black" Opacity="0.7">

                <StackPanel Orientation="Vertical" Margin="5">

                    <StackPanel Orientation="Horizontal">

                        <Label Background="Lime" BorderThickness="1" BorderBrush="Green"></Label>

                        <Label Foreground="White">Từ  :</Label>

                    </StackPanel>

                    <ComboBox Name="cboBegin"></ComboBox>

                </StackPanel>

                <StackPanel Orientation="Vertical" Margin="5">

                    <StackPanel Orientation="Horizontal">

                        <Label Background="Yellow" BorderThickness="1" BorderBrush="OrangeRed"></Label>

                        <Label Foreground="White">Đến :</Label>

                    </StackPanel>

                    <ComboBox Name="cboEnd"></ComboBox>

                </StackPanel>

            </StackPanel>

        </Canvas>

    </Grid>

</Window>

private int \_vertexNumber; //số đỉnh

private string[] \_vertexName; //tên các đỉnh

private double[,] \_graphMatrix; //ma trận kề

private List<int>[] \_graphList; //danh sách kề

private Point[] \_vertexPosition; //tọa độ các đỉnh

private void InputGraphFromFile(string fileUrl)

{

    TextReader textReader = new StreamReader(fileUrl);

    \_vertexNumber = int.Parse(textReader.ReadLine());

    \_vertexName = new string[\_vertexNumber];

    \_graphList = new List<int>[\_vertexNumber];

    \_graphMatrix = new double[\_vertexNumber,\_vertexNumber];

    \_vertexPosition = new Point[\_vertexNumber];

    int i, j, k, n;

    double w;

    string[] buffStringArray;

    for (i = 0; i < \_vertexNumber; ++i)

        for(j=0;j<\_vertexNumber; ++j)

            if (i == j)

                \_graphMatrix[i,j] = 0;

            else

                \_graphMatrix[i,j] = MaxWeight;

    for (i = 0; i < \_vertexNumber; ++i)

    {

        buffStringArray = textReader.ReadLine().Split(' ');

        n = int.Parse(buffStringArray[0]);

        \_graphList[i] = new List<int>();

        for(j=0;j<n;++j)

        {

            k = int.Parse(buffStringArray[j \* 2 + 1]); //mỗi đỉnh kề của đỉnh i

            w = double.Parse(buffStringArray[j \* 2 + 2]); //trọng số đính kèm từ đỉnh i đến đỉnh k

            \_graphList[i].Add(k);

            \_graphMatrix[i, k] = w;

        }

        \_vertexName[i] = buffStringArray[n \* 2 + 1]; //tên đỉnh

        \_vertexPosition[i].X = double.Parse(buffStringArray[n \* 2 + 2]); //tọa độ x

        \_vertexPosition[i].Y = double.Parse(buffStringArray[n \* 2 + 3]); //tọa độ y

    }

    textReader.Close();

}

int[] \_parent; //kết quả dijkstra : parent[i] là đỉnh trước của đỉnh i

private const double MaxWeight = 100000;

private double Dijkstra(int start, int end, ref int[] parent)

{

    double[] distance = new double[\_vertexNumber + 1];

    parent = new int[\_vertexNumber];

    bool[] visited = new bool[\_vertexNumber];

    int i, j, k;

    for(i=0;i<\_vertexNumber;++i)

    {

        distance[i] = \_graphMatrix[start,i];

        parent[i] = start;

        visited[i] = false;

    }

    visited[start] = true;

    distance[start] = MaxWeight;

    distance[\_vertexNumber] = MaxWeight;

    while (true)

    {

        var min = \_vertexNumber;

        for (i = \_vertexNumber-1; i >= 0; --i)

            if (visited[i] == false && distance[i] < distance[min])

                min = i;

        if (min == \_vertexNumber)

            break;

        if (min == end)

            break;

        var v = min;

        visited[v] = true;

        foreach (var u in \_graphList[v])

        {

            var sum =  distance[v] + \_graphMatrix[v,u];

            if (visited[u] == false && distance[u] > sum)

            {

                distance[u] = sum;

                parent[u] = v;

            }

        }

    }

    return distance[end];

}

private List<int> GetVertexPath(int start, int end, int[] parent)

{

    List<int> result = new List<int>();

    int temp = end;

    while (temp != start)

    {

        result.Add(temp);

        temp = parent[temp];

    }

    result.Add(temp);

    result.Reverse();

    return result;

}

int \_startVertex; //đỉnh đi

int \_endVertex; //đỉnh đến

private void InitInputComboboxSourceAndEvent()

{

    var userVertex = \_vertexName

        .Select((name, index) => new { VertexIndex = index, Name = name})

        .Where(p => p.Name != "\_")

        .Select(name => new { VertexIndex = name.VertexIndex, Name = name.Name.Replace('\_', ' ') });

    cboBegin.ItemsSource = userVertex;

    cboBegin.DisplayMemberPath = "Name";

    cboEnd.ItemsSource = userVertex;

    cboEnd.DisplayMemberPath = "Name";

    var descriptor = DependencyPropertyDescriptor.FromProperty(ComboBox.TextProperty, typeof(ComboBox));

    //khi chọn 1 item sẽ cập nhật lại \_startVertex

    descriptor.AddValueChanged(cboBegin, delegate

    {

        if (cboBegin.SelectedIndex != -1)

        {

            \_startVertex = ((dynamic)cboBegin.SelectedItem).VertexIndex;

            DrawVertex(elpBegin, \_startVertex);

            DrawPath();

        }

    });

    //khi chọn 1 item sẽ cập nhật lại \_endVertex

    descriptor.AddValueChanged(cboEnd, delegate

    {

        if (cboEnd.SelectedIndex != -1)

        {

            \_endVertex = ((dynamic)cboEnd.SelectedItem).VertexIndex;

            DrawVertex(elpEnd, \_endVertex);

            DrawPath();

        }

    });

}

private void DrawVertex(Ellipse element, int v)

{

    Canvas.SetLeft(element, \_vertexPosition[v].X - element.Width/2);

    Canvas.SetTop(element, \_vertexPosition[v].Y - element.Height/2);

}

private void DrawPath()

{

    if (cboBegin.SelectedIndex != -1 && cboEnd.SelectedIndex != -1)

    {

        var minDistance = Dijkstra(\_startVertex, \_endVertex);

        var minPath = GetVertexPath(\_startVertex, \_endVertex);

        var pathSegmentCollection = new PathSegmentCollection();

        var pathFigure = new PathFigure() {

            StartPoint = \_vertexPosition[minPath[0]],

            Segments = pathSegmentCollection

        };

        for (int i = 0; i < minPath.Count; i++)

            pathSegmentCollection.Add(new LineSegment(\_vertexPosition[minPath[i]], true));

        var pathFigureCollection = new PathFigureCollection();

        pathFigureCollection.Add(pathFigure);

        var pathGeo = new PathGeometry(pathFigureCollection);

        patDirection.Data = pathGeo;

    }

}

private const string \_fileUrl = "Resources/mapdata.txt";

public MainWindow()

{

    InitializeComponent();

    InputGraphFromFile(\_fileUrl);

    InitInputComboboxSourceAndEvent();

}