GGS 650 Project Option 1

Find intersection of US major rivers and US freeways

JIAO MA

GEORGE MASON UNIVERSITY

Introduction

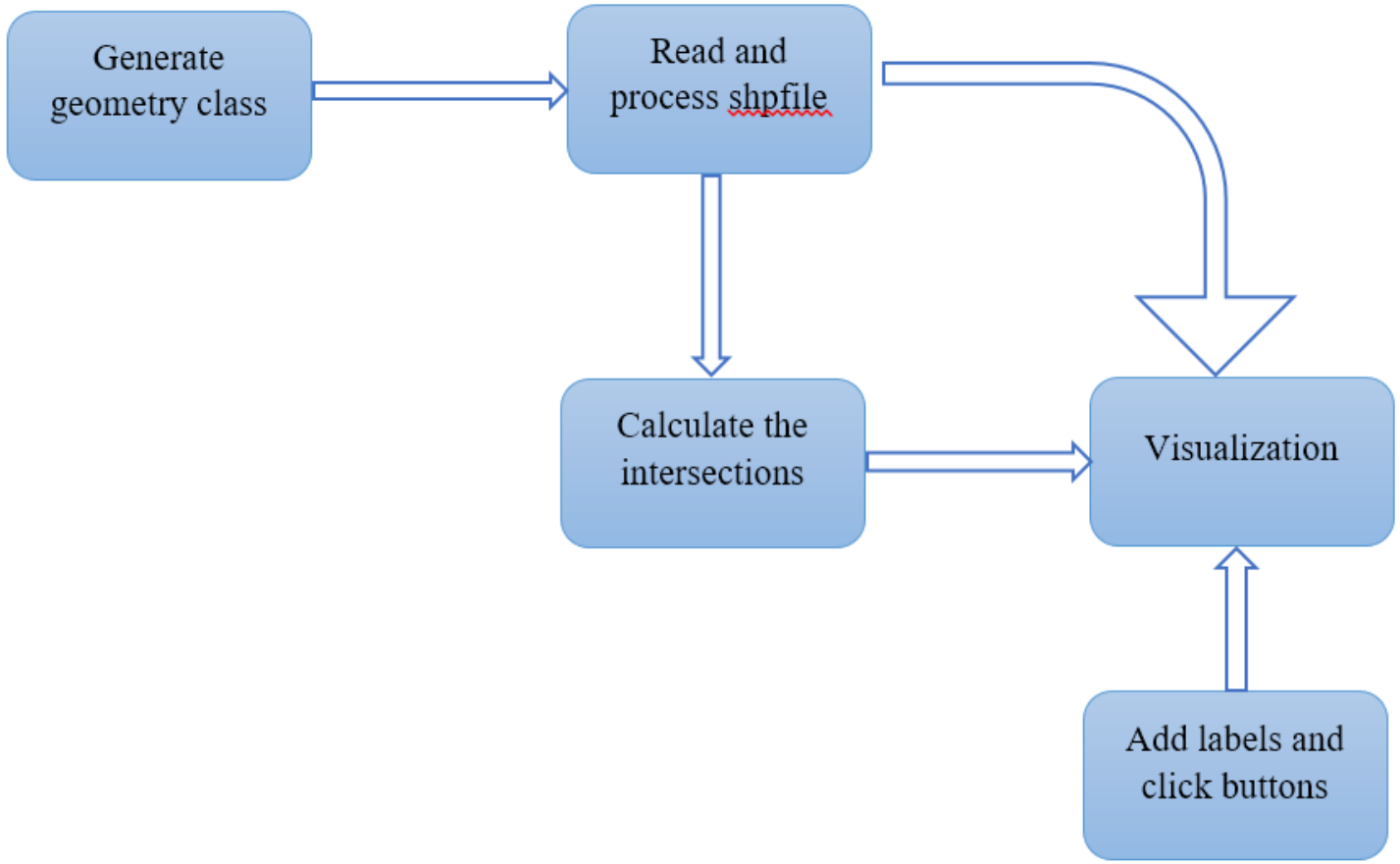
- Transportation has a pollution for the nearby rivers
- find intersection points and intersection segments for major rivers and highways
- Revised based on Mini GIS
- Tkinter is used to visualization

Introduction

Data Use

- US states polygon shapefile
- US major rivers polyline shapefile
- US freeways polyline shapefile

Work Flow


Generate geometry class
Read and process shpfile
Calculate the intersections
Visualization
Add labels and click buttons

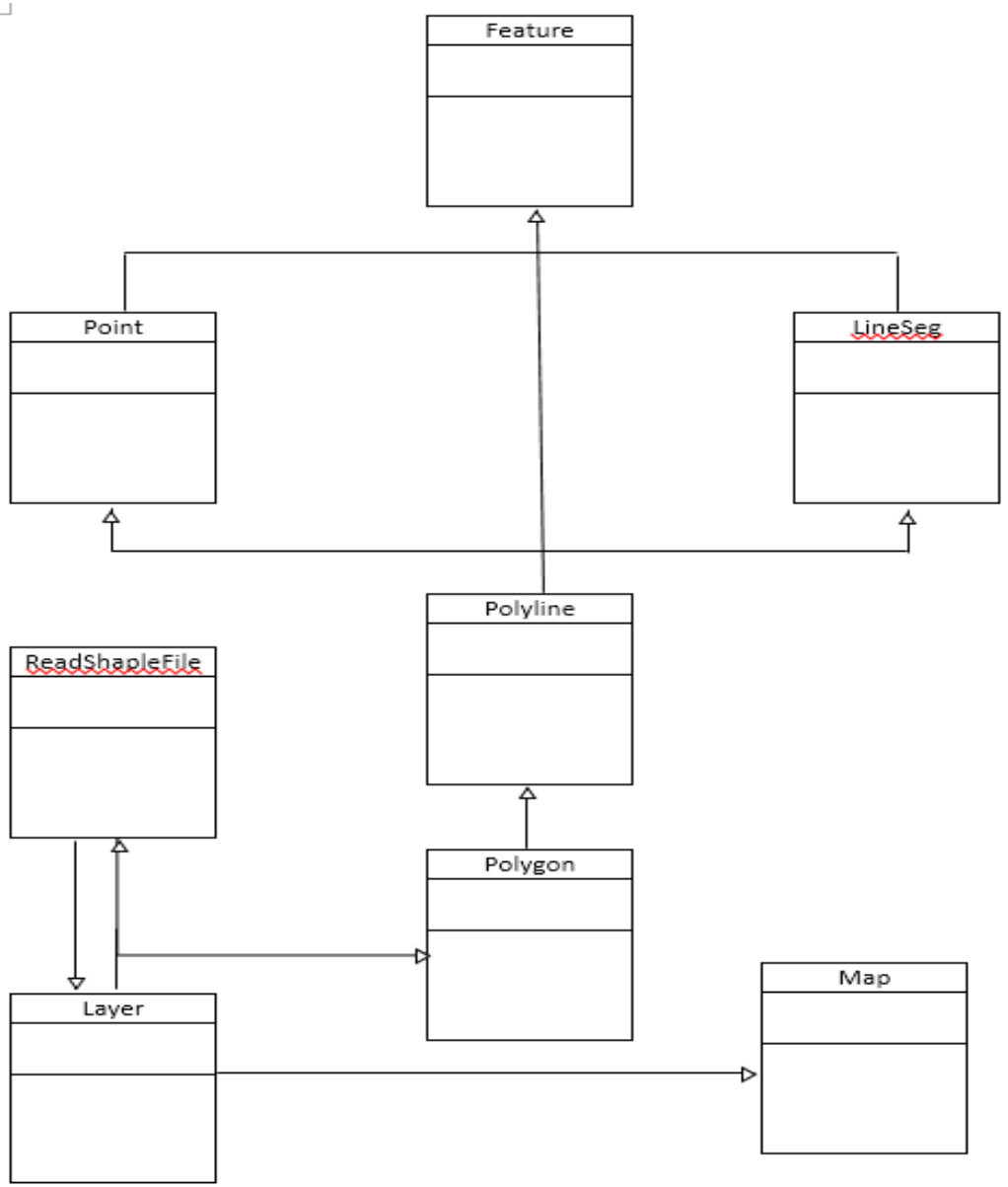
File Composition

nine python files

Except Main file, other files are written as class

Python File (9)

- Feature.py
- Layer.py
- LineSeg.py
- main.py
- Map.py
- Point.py
- Polygon.py
- Polyline.py
- ReadShapeFile.py



GIS Data Model designed example

Feature

+ none

+ vis(self, map, color): pass
+ intersect(self, Feature): pass

LineSeg

+ x1: float
+ y1: float
+ x2: float
+ y2: float

+ getLength(self) : Int
+ bboxcheck(self, lineseg) : Boolean
+ overlap(self, lineseg): Boolean
+ intersect(self, lineseg) : Boolean or List

Polyline

+ none

+ getLength(self): float
+ vis(self, map, color) : draw List
+ visInterLine(self, map, color): List
+ bboxcheck(self,polyline): Bollean
+ intersect(self,polyline): List
+ intersectLine1(self,polyline): List
+ intersectLine1(self,polyline): List
+ transform(self, map): List
+ findBoundingbox(self): Tuple

Points

+ x: float
+ y: float

+ distance(self, point): Int
+ vis(self, map, color) : draw List
+ transform(self, map): List

Polyline.py

```python
def intersectLine1(self,polyline):
    interLine1 = []
    for k in range(self.numParts):
        if (k == self.numParts-1):
            endPointIndex = self.numPoints
        else:
            endPointIndex = self.partsIndex[k+1]
            tempXYlist = []
        for m in range(self.partsIndex[k], endPointIndex-1):
            for l in range(polyline.numParts):
                if (l == polyline.numParts-1):
                    endPointIndexl = polyline.numPoints
                else:
                    endPointIndexl = polyline.partsIndex[l+1]
                for n in range(polyline.partsIndex[l], endPointIndexl-1):
                    ls1= LineSeg(self.x[m],self.y[m],self.x[m+1],self.y[m+1])
                    ls2= LineSeg(polyline.x[n],polyline.y[n],polyline.x[n+1],polyline.y[n+1])
                    if ls1.bboxcheck(ls2):
                        interp = ls1.intersect(ls2)
                        if interp:
                            interLine1.append(ls1)

    return interLine1
```

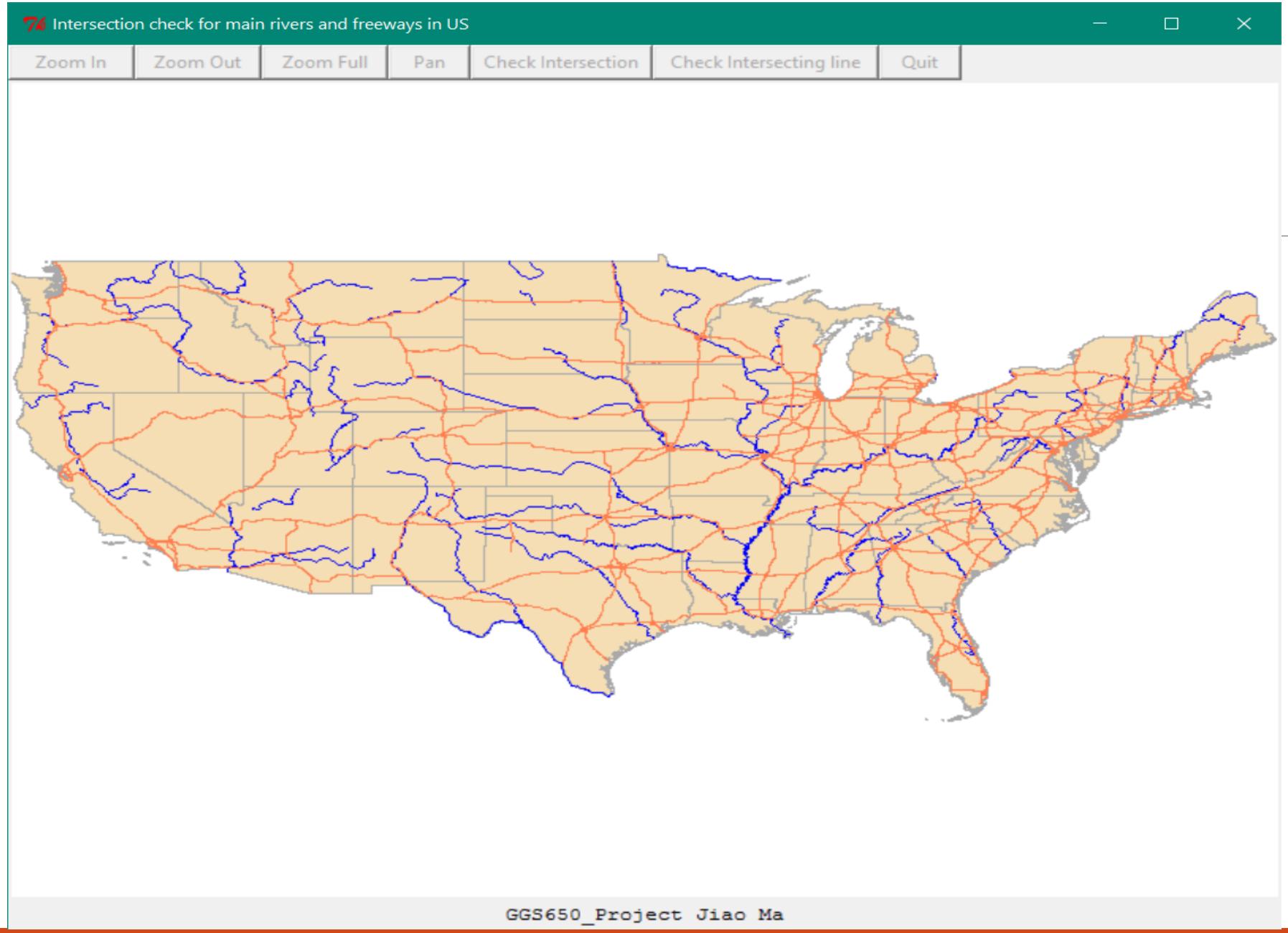
```python
def intersectLine2(self,polyline):
    interLine2 = []
    for k in range(self.numParts):
        if (k == self.numParts-1):
            endPointIndex = self.numPoints
        else:
            endPointIndex = self.partsIndex[k+1]
        for m in range(self.partsIndex[k], endPointIndex-1):
            for l in range(polyline.numParts):
                if (l == polyline.numParts-1):
                    endPointIndexl = polyline.numPoints
                else:
                    endPointIndexl = polyline.partsIndex[l+1]
                for n in range(polyline.partsIndex[l], endPointIndexl-1):
                    ls1= LineSeg(self.x[m],self.y[m],self.x[m+1],self.y[m+1])
                    ls2= LineSeg(polyline.x[n],polyline.y[n],polyline.x[n+1],polyline.y[n+1])
                    if ls1.bboxcheck(ls2):
                        interp = ls1.intersect(ls2)
                        if interp:
                            interLine2.append(ls2)

    return interLine2
```

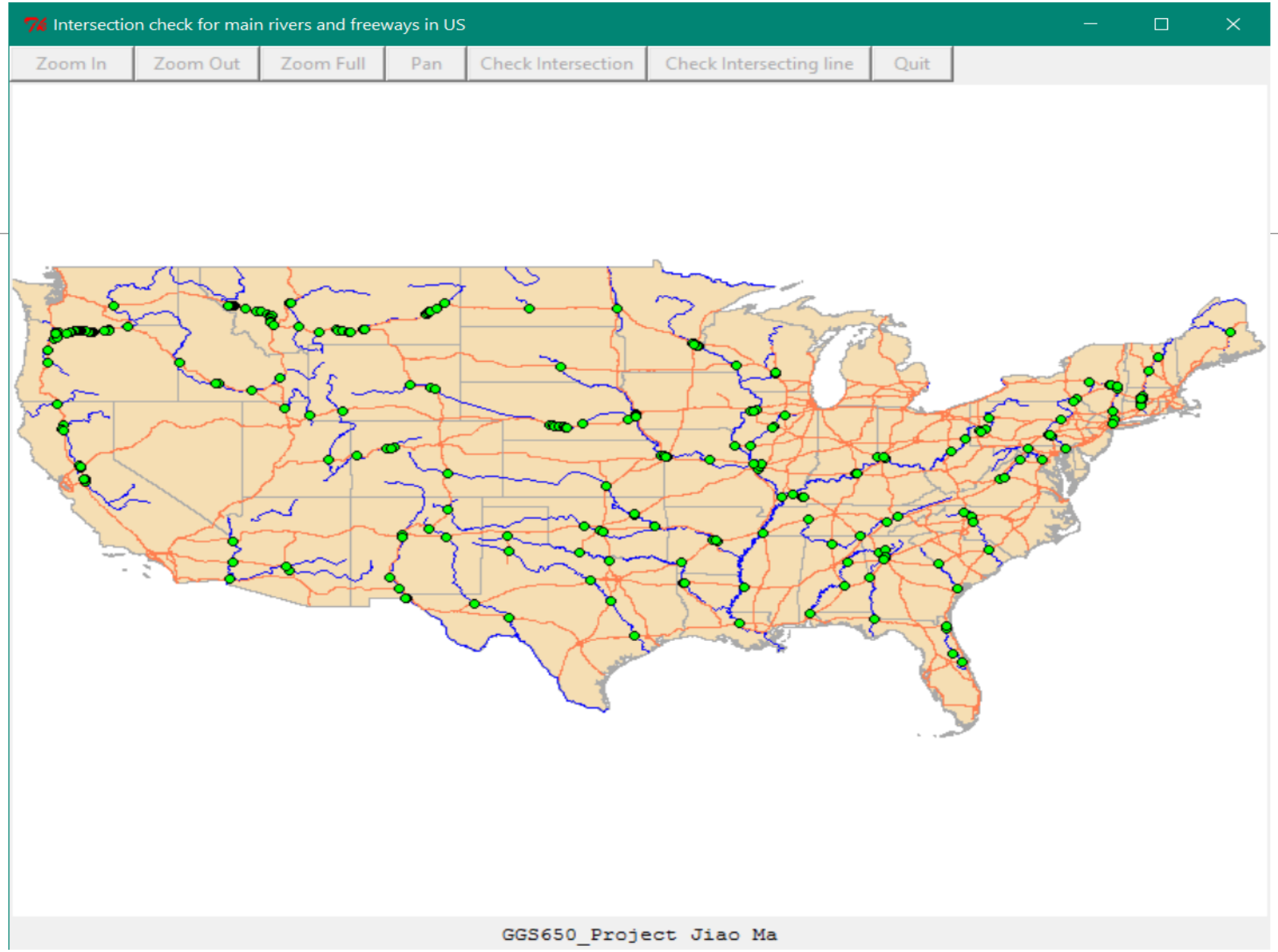
Layer.py

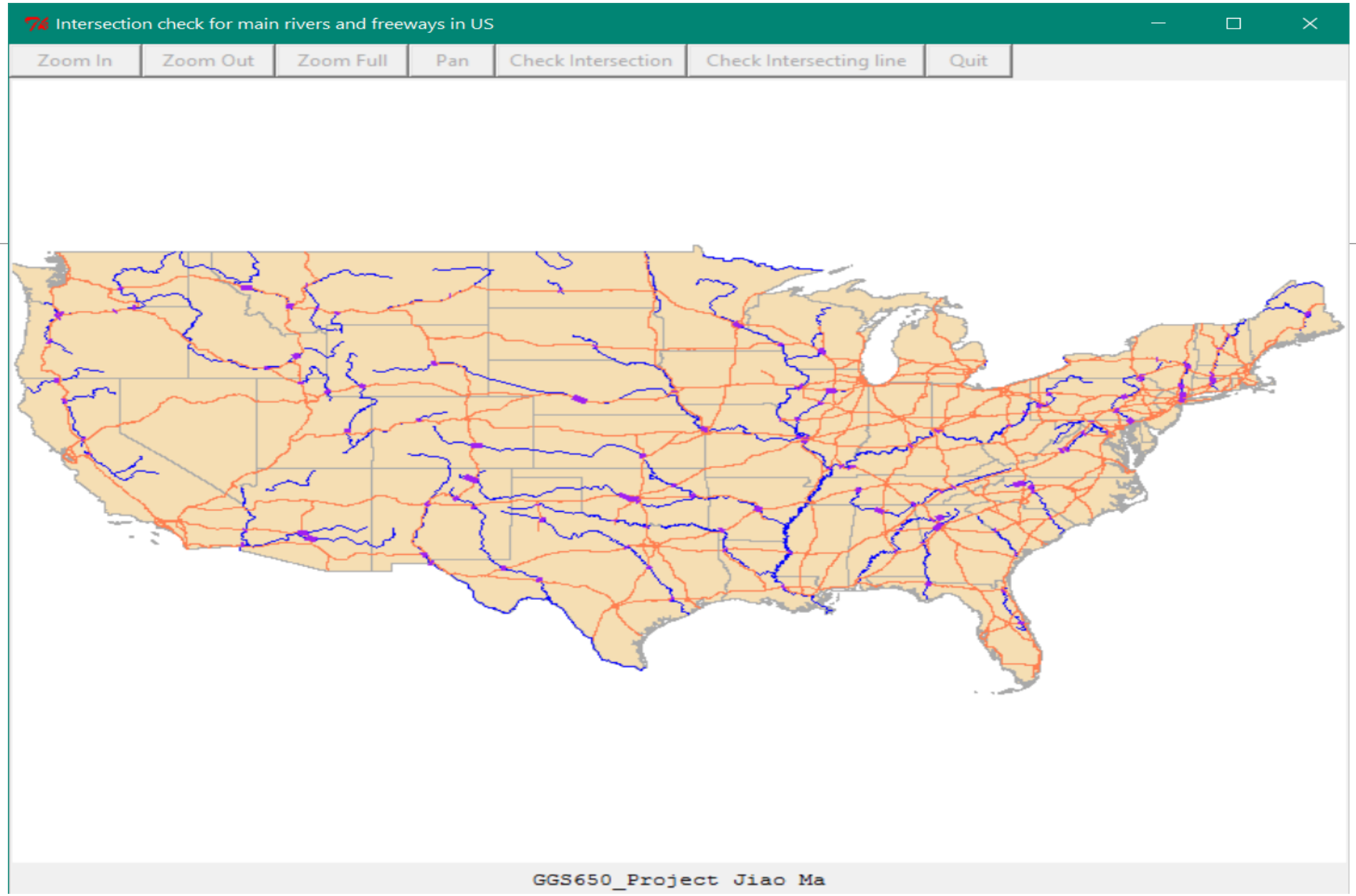
```
def intersectLine1(self,layer):
    interLine1 = []
    for feature1 in self.features:
            for feature2 in layer.features:
                if feature1.bboxcheck(feature2):
                    retLine1 = feature1.intersectLine1(feature2)
                    if retLine1:
                            interLine1.append(retLine1[0])
    print "There is %d intersecting lines" %(len(interLine1))
    print interLine1
    return interLine1

def intersectLine2(self,layer):
    interLine2 = []
    for feature1 in self.features:
            for feature2 in layer.features:
                if feature1.bboxcheck(feature2):
                    retLine2 = feature1.intersectLine2(feature2)
                    if retLine2:
                            interLine2.append(retLine2[0])
    print "There is %d intersecting lines" %(len(interLine2))
    print interLine2
    return interLine2
```

Map.py

```
def vis(self):
    self.can.delete('all')
    self.calculate()
    for layer in self.layers:
        for feature in layer.features:
            feature.vis(self, layer.color)
    for point in self.intersectPoints:
        xy = self.transform(point)
        self.can.create_oval(xy[0]-3, xy[1]-3, xy[0]+3, xy[1]+3, fill='green')
    for line1 in self.interSegLine1:
        transLine1 = self.transformSeg(line1)
        self.can.create_line(transLine1[0], transLine1[1], transLine1[2], transLine1[3], fill='purple', width = '4')
    for line2 in self.interSegLine2:
        transLine2 = self.transformSeg(line2)
        self.can.create_line(transLine2[0], transLine2[1],transLine2[2], transLine2[3], fill='purple', width = '4')
    self.can.pack()
    self.lab.pack()
```


Intersection check for main rivers and freeways in US
Zoom In
Zoom Out
Zoom Full
Pan
Check Intersection
Check Intersecting line
Quit
GGS650_Project Jiao Ma


Intersection check for main rivers and freeways in US
Zoom In
Zoom Out
Zoom Full
Pan
Check Intersection
Check Intersecting line
Quit
GGS650_Project Jiao Ma


Intersection check for main rivers and freeways in US
Zoom In
Zoom Out
Zoom Full
Pan
Check Intersection
Check Intersecting line
Quit
GGS650_Project Jiao Ma

Result

Python 2.7.8 Shell

File Edit Shell Debug Options Windows Help

```
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
US_states-49
 2.384067 seconds
US_major_rivers-49
 0.003194 seconds
US_freeways-49
 0.625500 seconds
24.5679574262 seconds is needed for visulization
There is 565 intersections
824.094230563 seconds were spent on checking intersections
```

Thank you!