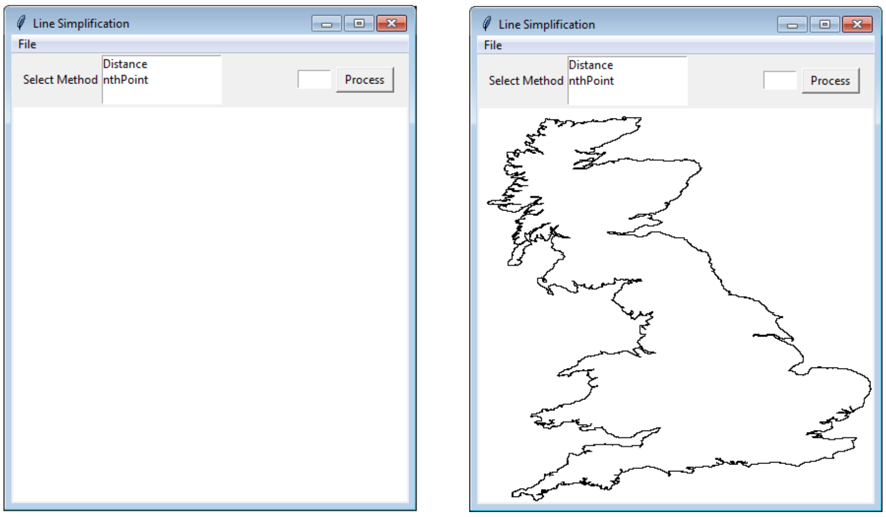
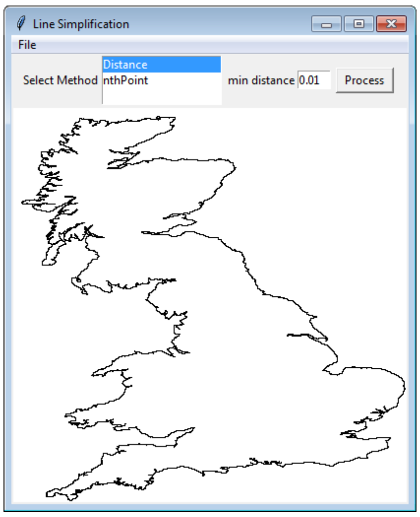
The Exercise

Line simplification is a process used by cartographers to reduce the detail in geographic features, such as a coastal outline. The object of this coursework is for you to write an extensible Graphical User Interface (GUI) that can allow a user to load line data, perform line simplification and save the results. To successfully complete this coursework you will need to investigate the functionality available in tkinter.

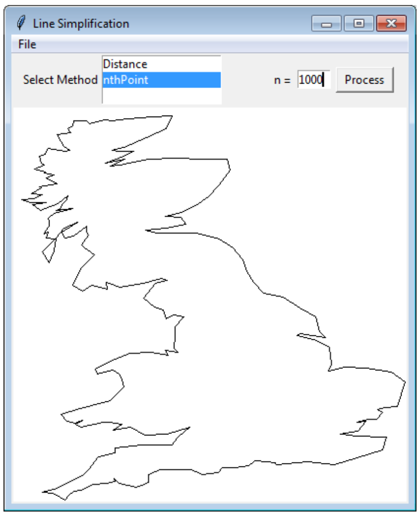


(a) Display immediately at start-up (b) Display immediately after loading data

Figure 1: User interface in action



(a) Display immediately before processing using Distance with *min distance* = 0.01



(b) Display immediately after processing using $n$thpoint with $n = 1000$

Figure 2: Line Simplification Methods

1 The Data

The file format of the data to load is as follows.
Each line contains information about one point.
It has the form "unique identifier",x,y.
Below is the first 4 lines of the file `MainlandUKoutline.csv`.
"1",-4.88527679443359,55.7298622131348
"2",-4.88527679443359,55.7295837402344
"3",-4.88361120223993,55.7295837402344
"4",-4.88361120223993,55.7290267944338

The unique identifier is always a number (represented as a string) and can be used to order the data. The `MainlandUKoutline.csv` file has all the points ordered, however your code should be able to accept data that is unordered and place them in numeric order with respect to the unique identifier.

2 Line Simplification Methods

There are many methods for line simplification, you shall implement the 2 simplest, called the $n$thpoint algorithm and the distance algorithm. Consider the ordered points:
"1",0,0
"2",1.5,0

“3”,2.5,0
“4”,3.5,0
“5”,4.5,0

The $n$thpoint algorithm

The $n$thpoint algorithm has one user defined parameter which is an integer denoted by n. The algorithm is as follows:

Keep the first datapoint, i.e. point with id “1”, then keep every nth point. So if $n = 1$ we keep all the points.
If $n = 2$, keep points “1”,“3”, “5”,
if $n = 3$, keep points with id “1”,“4”,
etc.

The distance algorithm

The distance algorithm has one user defined parameter which is a float denoted by the phrase *min distance*. The algorithm is as follows:
Keep the first datapoint, i.e. point with id “1”. Call this point the *Last Kept Point*. Sequentially step through the datapoints. If the Euclidean distance between the current point and the *Last Kept Point* is less than *min distance* then remove it (a method for calculating distances is supplied in the class Pt). Once the current point is a distance that is greater than or equal to the *min distance* from the *Last Kept Point*, then this point is kept and we make this the new *Last Kept Point* and continue to step through the data sequentially.

For example if we set the *min distance* to 2, then:
To begin “1”,0,0 is the *Last Kept Point*.
“2” is distance 1.5 from the *Last Kept Point* so we delete it.
“3” is a distance 2.5 from *Last Kept Point*, hence we keep it and make it the *Last Kept Point*.

“4”” is a distance 1 from the *Last Kept Point*, so it is deleted, however “5” is a distance 2 from the *Last Kept Point* hence it is kept. So for *min distance* = 2, the output should be the points with id “1”,“3”, “5”.

Display

For the display use a `canvas` widget and use its `create_polygon` method to display the outline. See Figure 1. Your program should automatically scale the data so that it is as large as possible within the viewing area (and reflect the data such that the higher the y value the higher up the screen).

Above the canvas have a `label` that states ”Select Method” followed by a `listbox` then another `label`, an `entry box` and finally a `button`. The `listbox` can show 3 lines of text. This `listbox` will list all the available line simplification algorithms. A user selects an algorithm (hint `bind` this `listbox` with `<<ListboxSelect>>`). This will change the `label` to the left of the `entry box` to show the correct parameter name.

Have a `menu` item called `File` which has sub-items `load` and `save`, each link to the relevant file manager window.

Extensibility and Polymorphism

The Graphical User Interface should not be too tightly coded to the line simplification algorithms. In fact we aim to have a situation that if a user wishes to write their own line simplification approach they could include it with minimal effort.

To achieve this, your line simplification algorithms should be implemented as classes that inherit from the abstract class `GUIconnection` (an abstract class has method definitions but with no usable method bodies). Your Graphical User Interface should only communicate with your line simplification classes through the methods defined in the abstract class `GUIconnection`. At start-up, your program should read a text file called `plugin.txt`, which will be located in the same directory as `LineSimplification.py`, each line of this file contains the class name of a line simplification approach, these classes will populate the `listbox`.

Within your Graphical User Interface use polymorphism to make the inclusion of another line simplification class as simple as possible.