Perfect Polygons in QGIS

An Introduction to the Polygonizer Plug-in

Introduction

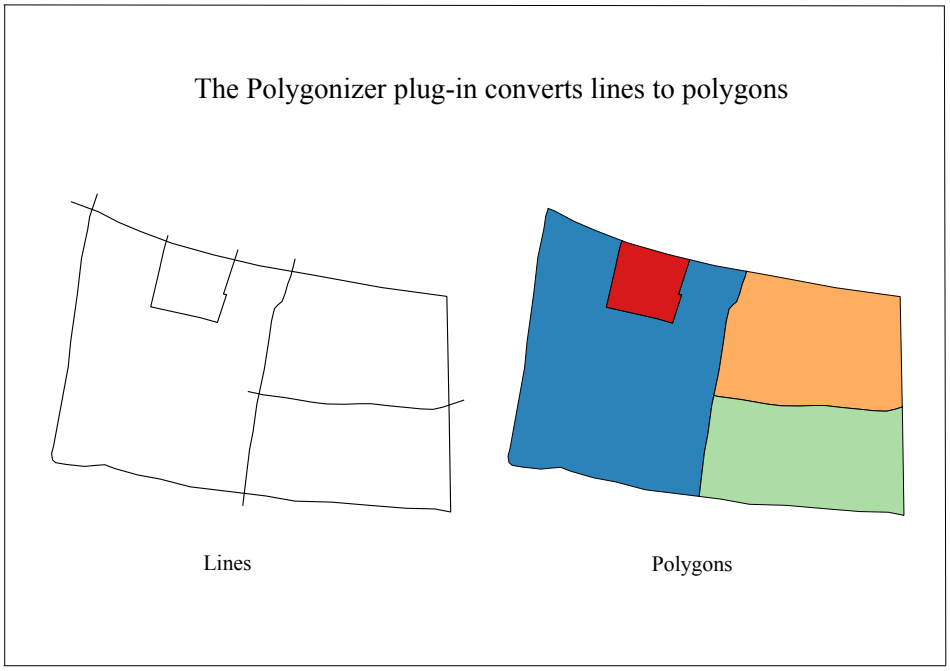
It is a requirement that polygons which share common boundaries should not contain overlaps or gaps.

In this note I describe a method that allows large polygons to be sliced into any number of smaller polygons, all perfectly formed without gaps or overlaps, utilizing the QGIS Polygonizer plug-in.

The Polygonizer plug-in can take a shapefile or a KML file containing lines and convert it to polygons, saving the result as a shapefile. This is an efficient way of digitising polygons because it eliminates double-tracing and avoids the need to snap abutting polygons together.

The Polygonizer plug-in

The illustration below shows, on the left lines that can be converted to polygons and on the right the result of polygonizing the lines. The Polygonizer extracts points where lines intersect and uses them, together with points from the original lines, to construct polygons.



The Polygonizer plug-in requires the python-shapely package to be installed. In Ubuntu this package may be installed using the Synaptic package manager. In Windows the package can be installed using OSGEO4W.

Tracing lines from Google Earth

The map on the front page of this note shows fields near to the village of Laxton, in the English county of Nottinghamshire. The map was made by tracing the field boundaries as lines in Google Earth and polygonizing them.

The traced lines were saved to this KML file:

`http://confound.me.uk/maps/Laxton.kml`

The file contains the lines used to make the polygons in the map. Load the file into Google Earth and expand the folder labelled “Laxton”. This contains fourteen line-strings, or *paths*, as they are known to Google Earth,

numbered 1 to 14, which is the order in which they were traced. When starting a project it is necessary to create a folder to hold the lines, right-click on “My Places”, select “Add” and then “Folder” to do this.

Adding lines to a new folder is straightforward, highlight the folder, click on the GE “path” icon and trace a line. The order in which lines are traced is not important, but in this case I started with a line, shown in red in Google Earth, that enclosed an area of interest. Note that the ends of the line cross each other at a point near the north-west corner of the map. This *pseudo-polygon* was then divided by three new lines. The second area was created in a similar way, a pseudo-polygon was traced around the area of interest and divided (and in this case further sub-divided) by lines.

The sequence in which the lines were drawn in Google Earth can be shown by hiding them all and then making them visible one after the other, in the order in which they were traced. Note how the lines intersect, creating “tails” that extend beyond the points of intersection.

Once traced the lines were saved in a KML file by right-clicking on the *folder* containing them and selecting “Save Place As...”. The lines could have been saved as a KMZ file, but this would have needed unzipping before it could be loaded into QGIS.[1]

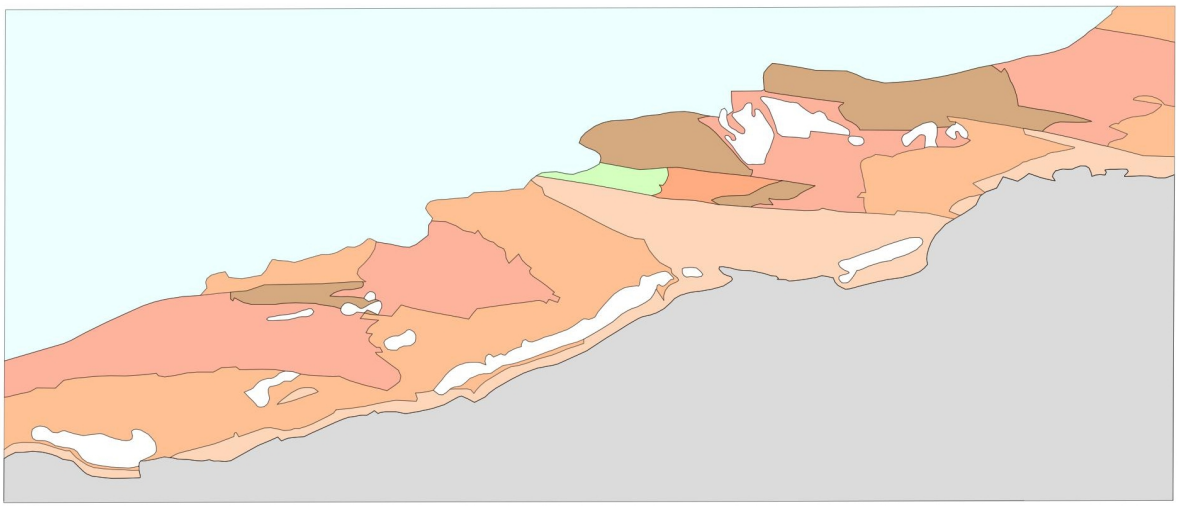
Using the Polygonizer plug-in

Load the KML file into QGIS (QGIS will recognise that its CRS is EPSG:4326). Set the project CRS and enable on-the-fly coordinate transformation. In this case, the CRS used for the project was EPSG:27700.

Start the Polygonizer, select the KML layer, supply a file name and location for the output shapefile and click “OK”. Then load the newly created shapefile into QGIS and examine the polygons.

Tracing from raster images

Load a georeferenced raster image into QGIS, set its CRS, set the project CRS and enable on-the-fly coordinate transformation. Create a new *line* shapefile[2] and trace lines from the image, ensuring that the lines intersect such as to permit their conversion to polygons. Save the shapefile, then use the Polygonizer plug-in to convert it to a polygon shapefile.



This map shows the ammonite zones exposed at low tide on the foreshore near Kilve, in the English county of Somerset. It has relatively complicated geometry and was traced from a scanned geological survey map as lines

[1] KMZ files are zipped KML files.
[2] When creating the new layer set its CRS to that of the raster layer.

that were subsequently polygonized. The first line drawn was a pseudo-polygon that enclosed the whole of the area of interest. This was then sliced into smaller areas using lines. Every object on the map is a polygon.

I had previously digitized exactly the same area using polygons and estimate that using the Polygonizer method halved the time spent tracing the raster image, with the additional benefit of creating perfect polygons.

Tracing from OpenLayers

With an OpenLayers layer displayed on the QGIS screen, zoom to an area of interest and create a new *line* shapefile. Set the CRS of the new layer to EPSG:900913 (the so-called *Google Projection*), trace the lines required and save the file. Convert the line shapefile to a polygon shapefile using the Polygonizer plug-in.[3]

Styling and labelling

The polygonizer does not create an attribute table with attributes suitable for labelling or styling but fortunately attributes for any purpose, including styling and labelling, can be added to a polygon shapefile quickly and simply.

Overlay the polygon shapefile with a new point shapefile created with the attributes required. Make the point layer editable, then click a point at (approximately) the centroid of each polygon, entering the attribute date for each point as it is created, as prompted. When all of the points required have been added to the point shapefile join it to the polygon shapefile using the 'Join attributes by location' tool (Vector → Data Management Tools → Join attributes by location). Select the polygon shapefile as the 'Target vector layer' and the point shapefile as the 'Join vector layer'. The resulting polygon shapefile will contain the attributes from the point shapefile, which can be used for styling and labelling.

Acknowledgements

The Polygonizer plug-in is the creation of Piotr Pociask and his efforts on behalf of the QGIS community are gratefully acknowledged.

Thanks are owed to Anita Graser who was kind enough to read through several drafts of this note and who suggested many helpful amendments to it.

About the author

The author of this note is Nick Hopton. Comments on the note may be addressed to *nhopton@gmail.com.*

[3] At the time of writing my personal preferance is to avoid where possible digitizing from OpenLayers, partly because of an irritating screen flash that accompanies the creation of each new object and partly because he has noticed that on occasions OpenLayers do not register correctly on the QGIS screen.