**Process of preparing Atlas of Urban Expansion GIS data**

Notes:

* Highlighted text indicates steps that are performed automatically by the scripts
* When saving files, take care to place them in the appropriate directory, as different QGIS tools may default to different locations
* Save project frequently to minimize cost of QGIS crashing
* The scripts work with QGIS 2.x (2.18.13 recommended), but do not work with QGIS 3.x (current)

1. Download GIS data from AUE webpage
   1. Should comprise two separate zip files corresponding to Areas and Blocks links
2. Decompress and organize GIS files
   1. Within appropriate location, create folder with name of city (eg "AHVAZ")
   2. Place downloaded zip files within corresponding city folder
   3. Unzip areas zip file and name the uncompressed folder *City*\_Areas
      1. Should contain a number of files—and no folders—including *City*\_studyArea.xxx. If not, rearrange files to match this structure
   4. Unzip blocks zip file and name the uncompressed folder *City*\_Blocks
      1. Should contain three folders: *City*\_Arterial, *City*\_T0, and *City*\_T1\_T3. If this is not the case (eg another folder, possibly named *City*), rearrange contents to match this structure
3. Open QGIS and save new project as "*CITY*.qgs"
4. Apply setup script "0 variables.py"
   1. In File Explorer, use a code or text editor to open the script
      1. Can right click and open file with "IDLE" on my computer
   2. In QGIS open the Python console (Plugins -> Python Console) and then open the editor (in menu bar, Show Editor button)
   3. Copy contents into Python console editor
   4. Set values as appropriate
      1. *city* is the name of the city. *region* is the AUE region containing that city. *user* is your Windows username. These are used to construct the correct file path to the shared drive, amongst other things.
         1. eg "Belgaum", "South and Central Asia", "Peter.Kerins" -> "C:/Users/Peter.Kerins/World Resources Institute/Urban Land Use - Documents/AUE Data and Maps/City Data/South and Central Asia/Belgaum/"
   5. Run contents of console editor (Run Script button in editor menu bar)
      1. This will set variables for subsequent scripts
5. Apply first processing script "1 area-medians-boundary\_open\_script.py"—open file, copy contents into console editor, and execute
   1. Add city's **study area** (*City*\_studyArea.shp, in “Areas” folder)
      1. For cosmetic purposes, as well as visual verification that all locales are contained wholly within study area (not guaranteed, eg Shenzhen)
   2. Add city's **street medians** (*city\_*Master\_AR\_Medians.shp in “Arterial” folder)
      1. For cosmetic purposes; does not affect process output
      2. Okay if streets extend beyond study area
   3. Add city's **locale boundary** files:
      1. Create QGIS group (Layer Panel folder) named *City*\_Boundaries\_T0 and add all shapefiles in the *City*\_T0 folder that end in 0
         1. Recommend using Add Vector Layer (Control+Shift+V) so that you can filter to only .shp files
      2. Create QGIS group (Layer Panel folder) named *City*\_Boundaries\_T1\_T3 and add all shapefiles in the *City*\_T1\_T3 folder that end in 0
      3. Ensure attribute table for each contains the ID\_string with the unique locale ID
   4. Merge all locale boundary files [Vector - Data Management Tools - Merge Vector Layers]
      1. Confirm success by examining attribute table for *City*\_Locales\_Merged. It should have latitude and longitude fields, with values of the location of the centroid of each locale
   5. Use field calculator to add attribute "Locale\_No" to *City*\_Locales\_Merged; the expression is simply *@row\_number - 1* (minus one for zero-based index)
   6. Save merged locales layer to shapefile named *City*\_Locales\_Merged
      1. Place file in top-level cities folder (where zip files and folders for blocks and areas are located)
      2. Remove the old, temporary layer of merged locales
6. Convert merged locales layer into polygons [Vector - Geometry Tools - Lines to Polygons]
   1. Save to shapefile and name it *City*\_Locales\_Polygons. Place file in top-level cities folder (where zip files and folders for blocks and areas are located)
   2. Rename layer to match file
7. Apply second processing script "2 block\_open\_addID\_merge\_script.py"—open file, copy contents into console editor, and execute
   1. Add city's **blocks** file:
      1. Create a folder (“group”) in the QGIS table of contents, and add to it all T0 shapefiles that end in 1
      2. Similarly, create another folder and add to it all T1\_T3 shapefiles that end in 1
   2. Use a script to automatically update the attribute table for every block file. Script will add a column named “ID\_string” (text) and populate it with the layer name, ie the unique locale ID.
      1. Open the Python Console (Plugins - Python Console)
      2. Open the Editor (in Python Console toolbar, click button Show Editor)
      3. In the Editor, load contents of “block\_processing\_script.py”
         1. Either load file using “Open Script…” button and selecting \World Resources Institute\Urban Land Use - Documents\AUE Data and Maps\block\_processing\_script.py
         2. Or copy and paste contents of that file into editor
      4. Select folder in QGIS table of contents containing block shapefiles to modify
      5. Press play button (“Run script”) in Editor toolbar to execute script
      6. Repeat previous two steps for any other folders (eg select T0 folder in table of contents and run script, then do the same for T1\_T3 folder)
      7. Open the attribute tables for some of the blocks shapefiles to confirm that they now contain the “ID\_string” field, and that the field contains the locale ID
   3. Merge all blocks files into one shapefile [Vector - Data Management Tools - Merge Vector Layers]
   4. Save merged blocks layer to shapefile named *City*\_Blocks\_Merged
      1. Place file in top-level cities folder (where zip files and folders for blocks and areas are located)
      2. Remove the old, temporary layer of merged blocks
8. Convert merged layer into polygons [Vector - Geometry Tools - Lines to Polygons]
   1. Safe to file and name it *City*\_Blocks\_Polygons
   2. Rename layer to match file
9. Create the **roads** file by subtracting the **blocks** file from the **locale boundary** file [Vector - Geoprocessing - Difference]
   1. Input layer = *City*\_Locales\_Polygons, Difference layer = *City*\_Blocks\_Polygons, check “ignore invalid input features”
      1. Save to file and name it *City*\_Roads
   2. If you get an error message while trying to complete this process, that means that one or more of the polygons is wonky. This is a result from the original polyline file. To fix this, run a validity test on the *City*\_Blocks\_Polygons layer [Vector – Geometry – Check Validity] and identify the specific error spots (noted with a dot). Once you see the error, you can edit the *nodes* of the polygon to properly align.
      1. TIP: Settings – Snapping Options – Snap: to Segment, Tolerance: 10 pixels, Check both “Enable topological editing” and “Enable snapping on intersection”
   3. In new **roads** shapefile, edit the attribute table: add a column entitled "Land\_use" (must be Text) and put "6" for every entry
   4. Append "ID\_string" with "6", so that road feature IDs are 15 characters like the others
      1. In field calculator, update existing field "ID\_string" and use expression *"ID\_string" || '6'*
10. Create the **complete** file:
    1. Merge the **blocks** file and **roads** file into a **complete** file
       1. Save to file and name it *City*\_Complete
    2. Ensure the resulting attribute table contains all 7 land use categories (0 to 6) and an "ID\_string" noting the locale identifier
       1. Note that at this juncture, the attribute table may show values of "0.0" or "NULL" for some attributes such as Locale\_No, Latitude, or Longitude. These will be corrected in the next step.
11. Apply final processing script "3 update\_complete.py"—open file, copy contents into console editor, and execute
    1. Every block polygon belongs to a locale. We want to make that locale identifier on a feature-by-feature (ie polygon-by-polygon) basis in the Complete file
    2. Script uses *City*\_Locales\_Polygons as the reference point, and uses the Locale\_No associated with each ID\_string in that file to update the Locale\_No field in the Complete file, by looking for features whose ID\_string starts with the the locale identifier
    3. Confirm success: spot-check one or two features (ie blocks), and confirm that center of containing locale matches the coordinates in the attribute table
12. Export **complete** file to GeoJSON (right click, Save As, Format: GeoJSON)
    1. Save file to folder "QGIS and GEOJSON" (with .qgs project file)

Current style and colors:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | Open Space | #b2df8a | #b2df8a |
| 1 | Non-Residential | #bc7aa8 | #fb9a99 |
| 2 | Residential Atomistic | #b0d2e8 |  |
| 3 | Residential Informal Subdivision | #73b3d8 |  |
| 4 | Residential Formal Subdivision | #3e8ec4 | #1f78b4 |
| 5 | Residential Housing Project | #1563aa |  |
| 6 | Roads | #777777 | #e31a1c |