Mapping the Tcl world:
using Tcl to curate OpenStreetMap

Kevin B. Kenny
5 November 2019
How’d we get here? I’m a Tcl geek and a map geek!
Timeline of geekinessness

- '60s: Kevin discovers maps
- '70s: Kevin makes maps of Earth and sky
- '80s: Kevin maps TV networks and transmission links
- '90s: Kevin first imports an external data set
- '00s: Kevin first edits OpenStreetMap
- '10s: Kevin first edits OpenStreetMap
- Future!

Kevin starts being a programmer
Tcl escapes the laboratory
OpenStreetMap founded
Kevin invents several bad scripting languages, uses several more
The 1970’s
Draw with a pen
High-resolution output
Took hours!

Draw with electrons
10 inch diagonal screen
“Instant” (well, minutes) gratification
Draw with a pen
High-resolution output
Took hours!

Draw with electrons
10 inch diagonal screen
“Instant” (well, minutes) gratification
The 1980’s
The 1990's
Hobby projects around year 2000

Prompted by Richard Suchenwirth-Bauersachs: “Mapping Colorado” on the Wiki

Lots of pieces, no really usable ecosystem.

- TclWorld
- Shapefile reader
- Tklib `map::slippy`
- Tklib `mapproj`
- … and so on

Andrey Shadura GSoC 2010

Tcl/Tk OpenStreetMap editor

Handler for the OSM-XML file format

Again, not integrated in the ecosystem

Trouble with multipolygons (Tk’s problem, not Andrey’s)
The 2010’s: OpenStreetMap

- Got back into hiking
- Appalled at the state of trail maps
- Only citizen-mappers can fix!
- Started contributing to OSM
Too much land, too few mappers!

- One example: Adirondack Park
  - Area: 24300 km² (not quite Belgium-sized)
  - Population: <130000

- Need external data sources
Motivation
Example: New York City recreational lands

<table>
<thead>
<tr>
<th>RECREATION AREA</th>
<th>TOWN</th>
<th>LOCATION</th>
<th>WMU</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Chosa Brook</td>
<td>Lexington</td>
<td>RT 13</td>
<td>4P</td>
</tr>
<tr>
<td>Kahrid Creek</td>
<td>Jewett</td>
<td>Shed Rd.</td>
<td>3A</td>
</tr>
<tr>
<td>Lake Heloise</td>
<td>Windham</td>
<td>Nauvee Road</td>
<td>4R</td>
</tr>
<tr>
<td>Lenox Brook</td>
<td>Hunter</td>
<td>BEECHER RD</td>
<td>3A</td>
</tr>
<tr>
<td>Lewis Creek</td>
<td>Ashland</td>
<td>Mall Rte Rd - County Route 5</td>
<td>4R</td>
</tr>
<tr>
<td>Little Woodkill</td>
<td>Lexington</td>
<td>Trussell Rd.</td>
<td>4R</td>
</tr>
<tr>
<td>Mellen Hill</td>
<td>Lexington</td>
<td>NYS Route 23A</td>
<td>4R</td>
</tr>
<tr>
<td>Macumber Road</td>
<td>Pratts ville</td>
<td>Macumber Rd., Pratts ville</td>
<td>4R</td>
</tr>
<tr>
<td>Maplecomit</td>
<td>Windham</td>
<td>County Route 40</td>
<td>3A</td>
</tr>
<tr>
<td>Mackrevelt Road</td>
<td>Windham</td>
<td>Windham</td>
<td>4R</td>
</tr>
<tr>
<td>Mill Street</td>
<td>Windham</td>
<td>Windham</td>
<td>4R</td>
</tr>
<tr>
<td>Mitchell Hollow*</td>
<td>Windham</td>
<td>Slam Rd &amp; Mitchell Hollow Rd</td>
<td>4R</td>
</tr>
<tr>
<td>North Ashland</td>
<td>Ashland</td>
<td>County Route 10 S &amp; SC</td>
<td>4R</td>
</tr>
<tr>
<td>North Lexington</td>
<td>Lexington</td>
<td>County Route 23C &amp; VanEten Rd</td>
<td>4R</td>
</tr>
<tr>
<td>Old Road</td>
<td>Windham</td>
<td>Dusty Rd.</td>
<td>4R</td>
</tr>
<tr>
<td>Partridge Road</td>
<td>Ashland</td>
<td>NYS Route 10 &amp; Partridge Rd</td>
<td>4R</td>
</tr>
<tr>
<td>Petersen Ridge</td>
<td>Ashland</td>
<td>NYS Route 23</td>
<td>4R</td>
</tr>
<tr>
<td>Pine Island Mountain</td>
<td>Lexington</td>
<td>Unknown</td>
<td>3A</td>
</tr>
<tr>
<td>Pinckill Meadows</td>
<td>Windham</td>
<td>Pinckill Meadows Rd.</td>
<td>3A</td>
</tr>
<tr>
<td>Platte Clove Road</td>
<td>Hunter</td>
<td>County Route 16</td>
<td>3A</td>
</tr>
<tr>
<td>Railroad Bend</td>
<td>Hunter</td>
<td>Ski Bowl Rd.</td>
<td>3A</td>
</tr>
<tr>
<td>Red Falls</td>
<td>Pratts ville</td>
<td>Route 23</td>
<td>4R</td>
</tr>
<tr>
<td>Red Kill Headwaters</td>
<td>Halcott</td>
<td>Trues Fautiener Rd.</td>
<td>3A</td>
</tr>
<tr>
<td>Richmond</td>
<td>Windham</td>
<td>County Route 10</td>
<td>4R</td>
</tr>
<tr>
<td>Slatebush Brook</td>
<td>Lexington</td>
<td>Beech Ridge Road</td>
<td>4R</td>
</tr>
<tr>
<td>Roaring Kill</td>
<td>Hunter</td>
<td>Mink Hollow Rd.</td>
<td>3A</td>
</tr>
<tr>
<td>Roundtop Mountain</td>
<td>Hunter</td>
<td>Grissie Rd.</td>
<td>3A</td>
</tr>
<tr>
<td>Rushcreek</td>
<td>Windham</td>
<td>DEMING RD</td>
<td>4R</td>
</tr>
<tr>
<td>Sohobridge Creek</td>
<td>Pratts ville</td>
<td>NYS Route 23A</td>
<td>4R</td>
</tr>
</tbody>
</table>

**NYCDEP Public Access Area**

- *PAA - Public Access Area*
- *DUA - Day Use Area*

**Roundtop Mountain Unit**

- **330 Acres**
- Greene County, Town of Hunter
- **Long:** 74°6'49.975"W  Lat: 42°10'17.294"N
- **Uses:** Hiking, Hunting & Trapping
Step 1: Scarf down all the data

Can we make sense of the list?

```bash
exec pdftohtml open_rec_areas.pdf
```

Looking at the result, we can extract this mess:

```html
Hunter<br/> Gillespie Rd.<br/> 3A<br/> <b>Y</b><br/> <b>Y</b><br/> N<br/> Y<br/> <b>Y</b><br/>
330<br/>
```

Horrible looking HTML, but tdom can surely parse it.

A few hours later: there’s a script to download the list and all the maps and tag them with metadata.
Step 2: Make sense of PDF maps

(This was actually the first step… the alternative would have been a Freedom of Information demand!)

Would be extremely challenging to georeference the PDF maps for tracing. (Too little context).

Maybe they were printed from ArcGIS? Let’s see if they’re GeoPDF. A command line tool from GDAL (Geospatial Data Abstraction Library) will inspect them:

$ ogrinfo pdfs/Roundtop_Mountain.pdf

(drum roll please...)
Step 2: Make sense of PDF maps

Yes, GDAL can post these as GeoPDF:

$ ogrinfo pdfs/Roundtop_Mountain.pdf

Metadata:
   CREATION_DATE=D:20160428103334-05
   CREATOR=Esri ArcGIS
1: Other_2
2: Layers_Other
3: Layers_Labels_100_Ft_Elevation_Contours__Default
4: Layers_PAA
5: Layers_Roads
6: Layers_Streams
7: Layers_Rivers__Ponds__Lakes__and_Reservoirs
8: Layers_100_Ft_Elevation_Contours
9: Layers_Buildings_EOH

Most of these layer names make sense in terms of map features.

‘PAA’ turns out to be ‘Public Access Area,’ which is the boundary we want.

No Freedom of Information demand needed! (Whew!)
Step 3: Get the map data where we can work with it.

PostgreSQL.

- Much of the existing OpenStreetMap infrastructure already uses it.
- Very strong, GDAL-based, functions and index infrastructure for dealing with geospatial data.
- SpatiaLite (at least when I did this project) not nearly as well developed.

So, one at a time, we pour an individual map into a PostgreSQL table:

```
exec ogr2ogr -append -t_srs EPSG:3857 -f PostgreSQL \\
    PG:dbname=gis $fileName \\
    -nln intake -nlt MULTILINESTRING \\
    Layers_PAA
```
Step 4: Whoops! Topology!

- Input data are just boundary lines, not polygons.
- Lines broken into short segments
- Some lines look like noisy GPS tracks of someone walking a boundary
- Some adjacent parcels overlap
- And so on...

Tcl doesn’t have computational geometry facilities to clean this up. Do it in PostgreSQL, command it with TDBC. A couple of pages of Tcl (took a few days to design) take care of it.
Step 5: Review and conflation

This is the hard part – requiring human analysis.

Needs an editor for OSM data.

Andrey Shadura (Andrew Shadoura) wrote one in Tcl as a GSoC project

- No longer maintained
- An OSM editor is actually a huge ecosystem. Better to use an existing one.

Several OSM editors support an HTTP-based API to command them. The http and tls packages are already in the mix.

So, dump the data into XML (using an external ogr2osm.py program), and command an OSM editor to import it as a new layer, then do the rest by hand in the editor.
For a big, complex import, (the New York City recreation data wasn’t that big), developed a Tk GUI for managing conflation.

Select an object – loads it into the editor and downloads the surrounding region from OSM

Creates an additional layer with differences between the selected object and the best matching object in current data

Chooses keyword=value tags to apply to the selected object

Other actions – visit the area’s web site, apply the keyword=value tags to the object, copy the tags to the clipboard, mark the object as ‘done’ in the database, end the session.
Another project: render North American numbered highways

- 4 or more numbering systems overlaid
- Sign shape is important
- Many route concurrences
- Tcl script to handle data changes, generate SVG graphics.
- Concurrency sets calculated at render time in horrible PostgreSQL query.
- Serviceable for me, much work remains to deploy at scale

https://github.com/kennykb/osm-shields
Whither Tcl/Tk?

Tcl/Tk has played a tiny role in all this.

No more than a couple of thousand lines of code in any import project.

All glue – it doesn’t really do much itself, it orchestrates the big applications that do the heavy lifting.

We won’t rule the world this way!

But isn’t this what Tcl/Tk is for? It’s very, very sticky glue, and good at connecting things together.
Thank you!