Ryan Bravo
Research Technical Paper
Summer Research
August 17, 2006

This summer I utilized Geographic Information Systems to work on the WIPER project. Most of my work involved generating visualizations of the cell phone activity data to facilitate comprehension of how the cell phone activity varies over a given time period in a certain area.

The visualization of cell phone data is crucial to the WIPER project. In an emergency response situation, time is of the essence. The visualizations would allow the user to quickly ascertain how traffic is flowing. This also makes abnormalities more apparent. The representations in my visualizations used color to represent activity. It ranged from green for low activity, yellow, orange, and then red for high activity. If a normally yellow area turns red, something might be wrong.

For someone just using my research and deliverables, they are in the /home/rbravo directory on apocalypse.cse.nd.edu. The final version of my script is also in Tim's CVS directory. Comprehensive instructions are in the User Manual section. Most of my initial work was done on sim6.cse.nd.edu, which I believe has been taken down. All the important data and programs had already been moved to apocalypse.

If another undergraduate were to continue my work, I would recommend that they read the GRASS book that Dr. Madey has. We didn't order it until later in the summer, after I had become familiar with GRASS. The book was still a tremendous resource and would help a new GRASS user quite a bit. I would also encourage a new REU to just play with the software. Especially with GRASS, I discovered many new functions by accident. These functions were very useful later on.

User Manual

Associating Cell Activity with Voronoi Cells

To create a shapefile with attributes take the shape file and change to postgres commands:

```
shp2pgsql -g geometrycolumn shapefile tablename > file.sql
```

Load commands into postgres to create table:

```
psql -U user < file.sql
```

Combine shapefile's table with attributes' file:

```
insert into table (column1, column2) select table1.column table2.column
from table1, table2 where conditions
```

Convert new table to shp file:

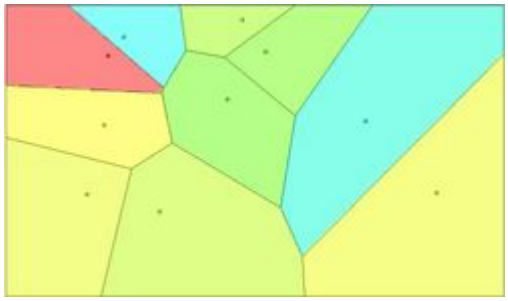
```
pgsql2shp -f filename -u pguser -P password -g geometrycolumn database table
```

In GRASS, load shapefile:

```
v.in.ogr -o dsn=/path/to/vector/file output=grass_vector_name
```

Convert from vector to raster:

```
v.to.rast input=inputvector output=outputraster use=attr column=attribute
```



Raster Cells

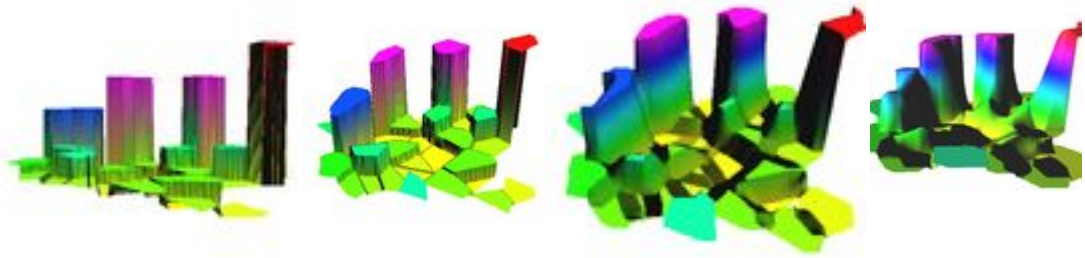
Attribute is whatever attribute in your database that you want the raster value to be. You can check what columns you have available by using

```
db.describe vectorname
```

The raster's resolution depends on the region's current resolution.

```
Raster resolution can be set using g.region.
```

3D Rasters



Nice View Smoother Sideview Smoothed

Creating 3D rasters is surprisingly easy. Take any raster file and view it in a "Map Display" window. In that window click the NVIZ button at the top left. You might need to use g.region to increase the height boundaries, but from what I can tell it shouldn't be

necessary. You will also probably move the height, zexag, and perspective bars around to get the view you want.

Ryan Bravo's Magical Scripts

To facilitate speedy and easy generation of the 2D raster images and videos, I wrote a series of scripts. One bash script governs the whole system, setting environment variables and calling the other scripts.

The first sub-whatscript takes a call data file and generates the activity data. The data file must be sorted in chronological order. The data file says a call was made at a certain time at a certain location with a certain tower number and the script aggragates this, replacing the tower number with a hashed number based on location. There are variables at the top of the script that the user should change to determine what the script is doing: the time frame to look at, the number of divisions to make, etc. Using these predefined settings, the script then generates a SQL file and another bash script.

The SQL file is then loaded into postgreSQL. Using postGIS, the geometries for voronoi cells have already been loaded into the database. The activity data is combined with the voronoi cells based on the hashed tower number and output to a shape file.

The bash script that was generated previously is then called. This loads the new shapefiles into GRASS as vector data with the activity snapshots as attributes. This vector is then converted into X snapshots of raster data. Each of these raster files are output as jpg and into an mpeg animation. I have overlayed a road vector on the jpg output by redirecting the output from the computer monitor to a picture file.

To run the script you need the main bash scripts, the first script it calls, and the call activity file. No other files are actually necessary. The script itself creates the SQL file, the second bash script, the activity data file, the jpg pictures, and the mpeg.