PostGIS 2.0

Released in **April 2012**
after 26 months of development!

Fast! Solid!
tremendous work by all

Outstanding Documentation
note Chapter 13 for 2.0 diffs

Installation & Requirements

Required Libraries

GEOS – C++ port of the Java Topology Suite
PROJ4 – projection library
LibXML2 – for GML support, etc
GDAL – for raster support

PostgreSQL 8.4 for PostGIS 2.0
PostgreSQL 9.0+ for PostGIS 2.1

http://postgis.refractions.net/documentation/manual-2.0/postgis_installation.html

Arch: x86_64 / AMD64 PPC armv61 Sparc64 x86 i686 i386
OS: OSX Linux BSD Solaris HP-UX AIX Windows7 WindowsXP WindowsServer

Installation & Requirements

Create Extension

CREATE EXTENSION postgis;
CREATE EXTENSION postgis_topology;

PostgreSQL 9.1+

http://postgis.refractions.net/documentation/manual-2.0/postgis_installation.html

Things You May Need to Know

- The binary format of data has changed. You *must* dump PostGIS 1.5 databases and then reload. However, most common loading methods handle the binary conversion for you

```
pg_dump my_15_db -t table1 -t table2 | psql -q new_20_db
```

- geometry_columns is now a VIEW
- What is GDAL ?! It was decided to include raster support by default. PostGIS Raster requires GDAL, a general purpose raster manipulation and interchange library plus tool suite. So you will get GDAL when you install PostGIS 2.0. *see* www.gdal.org
- You cannot make a PostGIS 1.5 database in PostgreSQL 9.2

  version 1.5 is being maintained – *send feedback to the steering committee*

- For Ubuntu, we use UBUNTUGIS-UNSTABLE to build OSGeo LIVE.

  (not as unstable as it sounds)

Introduction

from the README:

PostGIS

Geographic Information Systems Extensions to PostgreSQL

This distribution contains a module which implements GIS simple features, ties the features to R-tree indexing, and provides many spatial functions for accessing and analyzing geographic data.

Spatial Data – Types

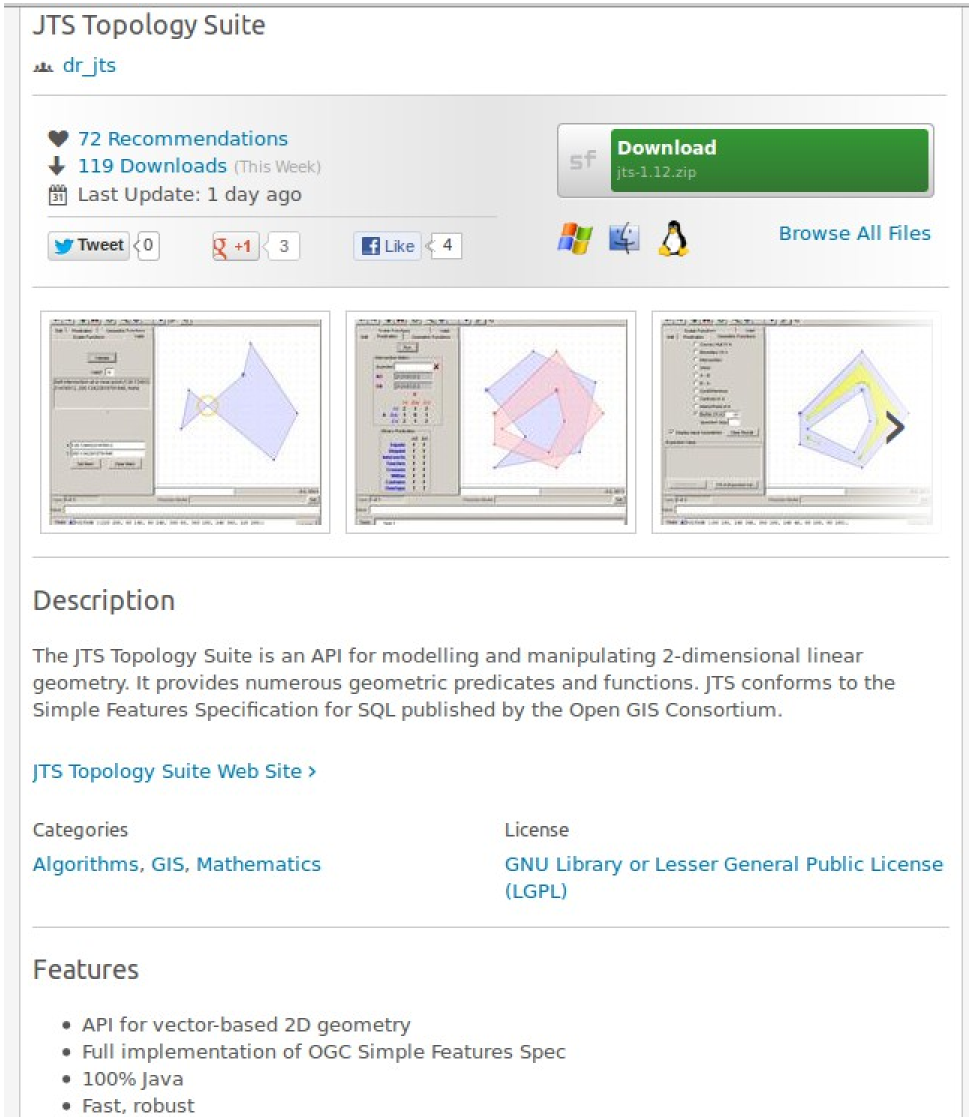
```
         List of data types
 Schema |      Name       | Description
--------+-----------------+-------------
 public | box2d           |
 public | box2df          |
 public | box3d           |
 public | geography       |
 public | geometry        |
 public | geometry_dump   |
 public | geomval         |
 public | gidx            |
 public | histogram       |
 public | pgis_abs        |
 public | quantile        |
 public | raster          |
 public | reclassarg      |
 public | spheroid        |
 public | summarystats    |
 public | valid_detail    |
 public | valuecount      |
```

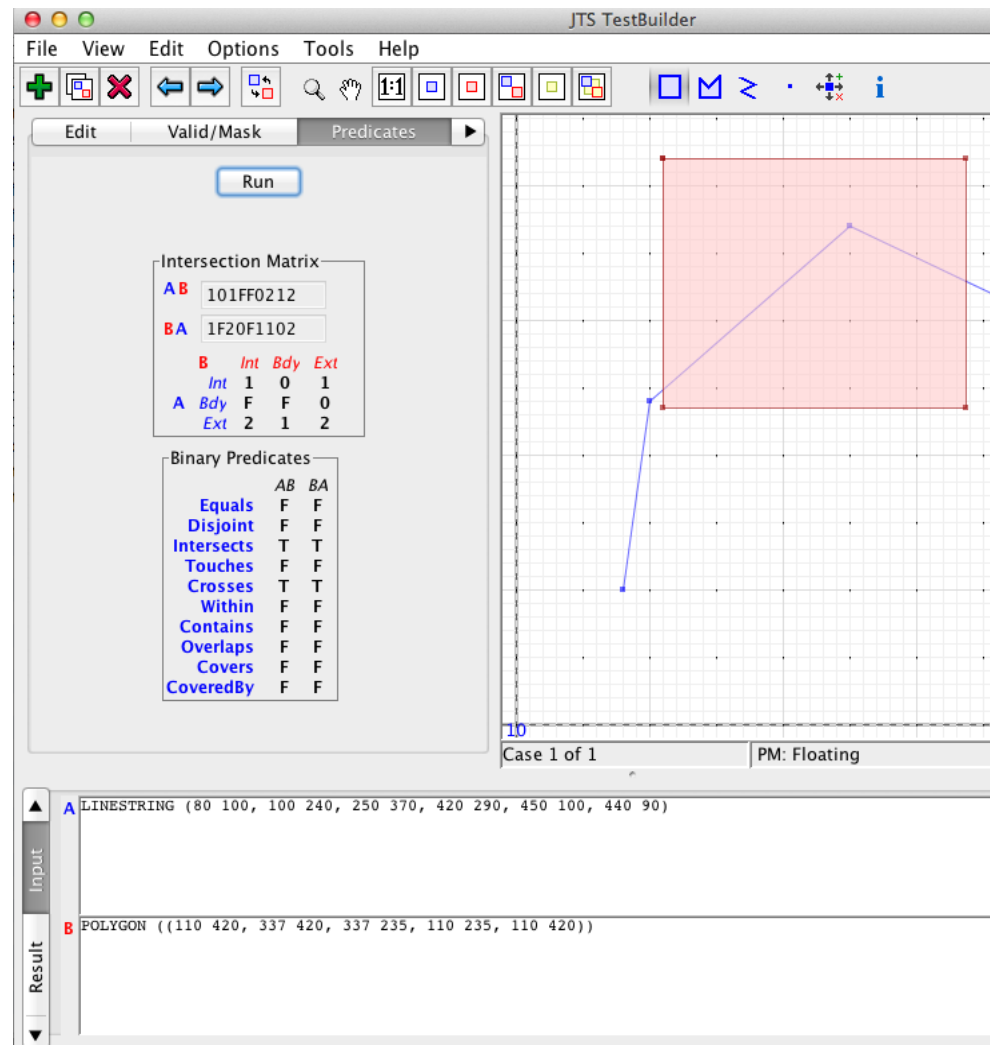
Spatial Data – Vector

Vector Features
POINT, LINESTRING, POLYGON
MULTIPOINT, MULTILINESTRING, MULTIPOLYGON
X Y Z M, *so called* 2.5D
GEOMETRYCOLLECTION

New or better in 2.0
CIRCULARSTRING,
CURVEDPOLYGON,
TRIANGLE
TIN (Triangulated Irregular Network, a kind of triangle collection)
POLYHEDRALSURFACE,
X Y Z M, *actual* 3D *in development*

Spatial Data – Vector





Spatial Data – Vector

GEOS Geometry Engine Open Source

Search

Login | Help/Guide | About Trac | Preferences

Wiki | Timeline | Roadmap | Browse Source | View Tickets | Search

Start Page | Index | History | Last Change

GEOS - Geometry Engine, Open Source

GEOS (Geometry Engine - Open Source) is a C++ port of the Java Topology Suite (JTS). As such, it aims to contain the complete functionality of JTS in C++. This includes all the OpenGIS Simple Features for SQL spatial predicate functions and spatial operators, as well as specific JTS enhanced topology functions.

GEOS is available under the terms of GNU Lesser General Public License (LGPL), and is a project of OSGeo.

Capabilities Include

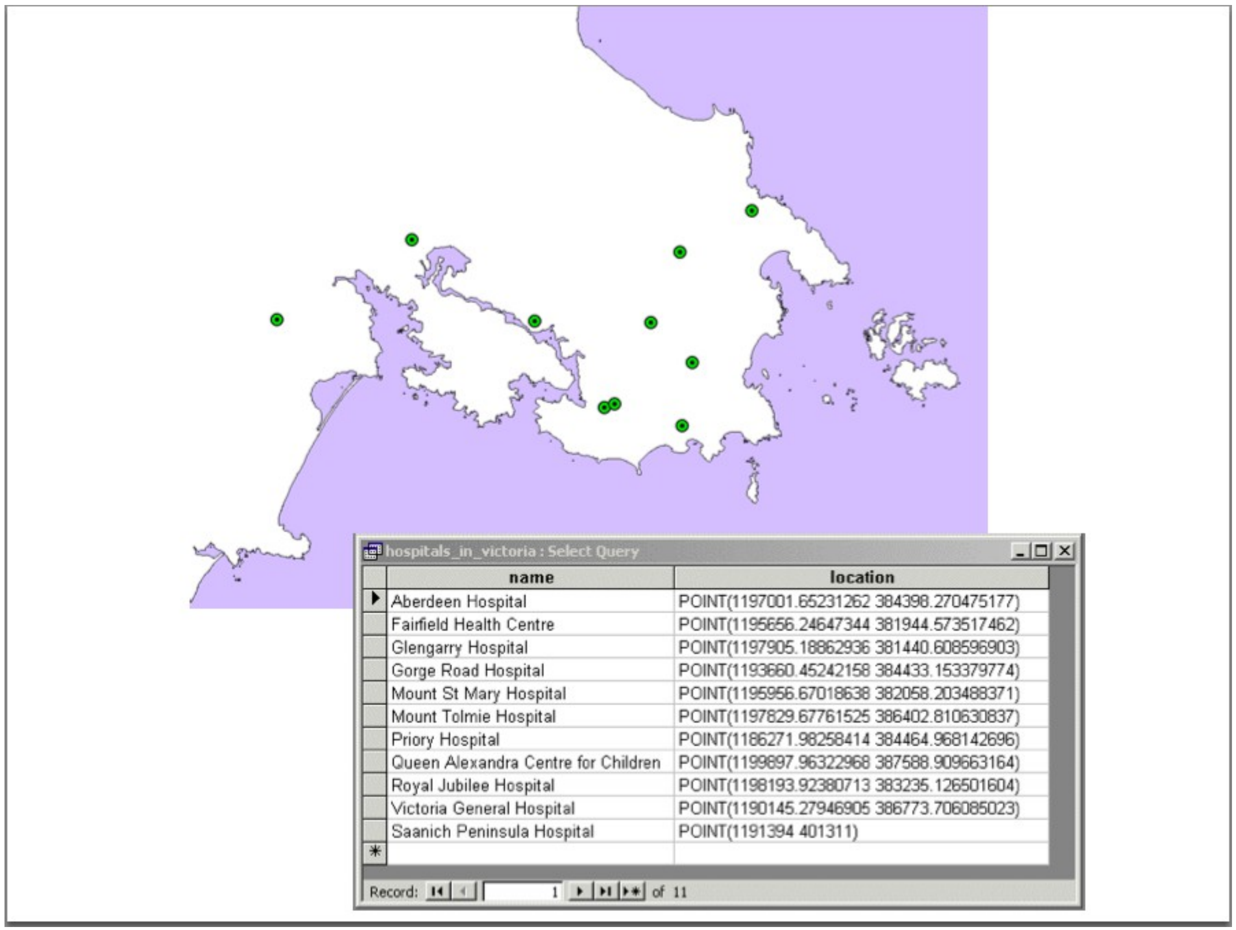
- Geometries: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeometryCollection
- Predicates: Intersects, Touches, Disjoint, Crosses, Within, Contains, Overlaps, Equals, Covers
- Operations: Union, Distance, Intersection, Symmetric Difference, Convex Hull, Envelope, Buffer, Simplify, Polygon Assembly, Valid, Area, Length,
- Prepared geometries (pre-spatially indexed)
- STR spatial index
- OGC Well Known Text (WKT) and Well Known Binary (WKB) encoders and decoders.
- C and C++ API (C API gives long term ABI stability)
- Thread safe (using the reentrant API)

Download

Example Use Case

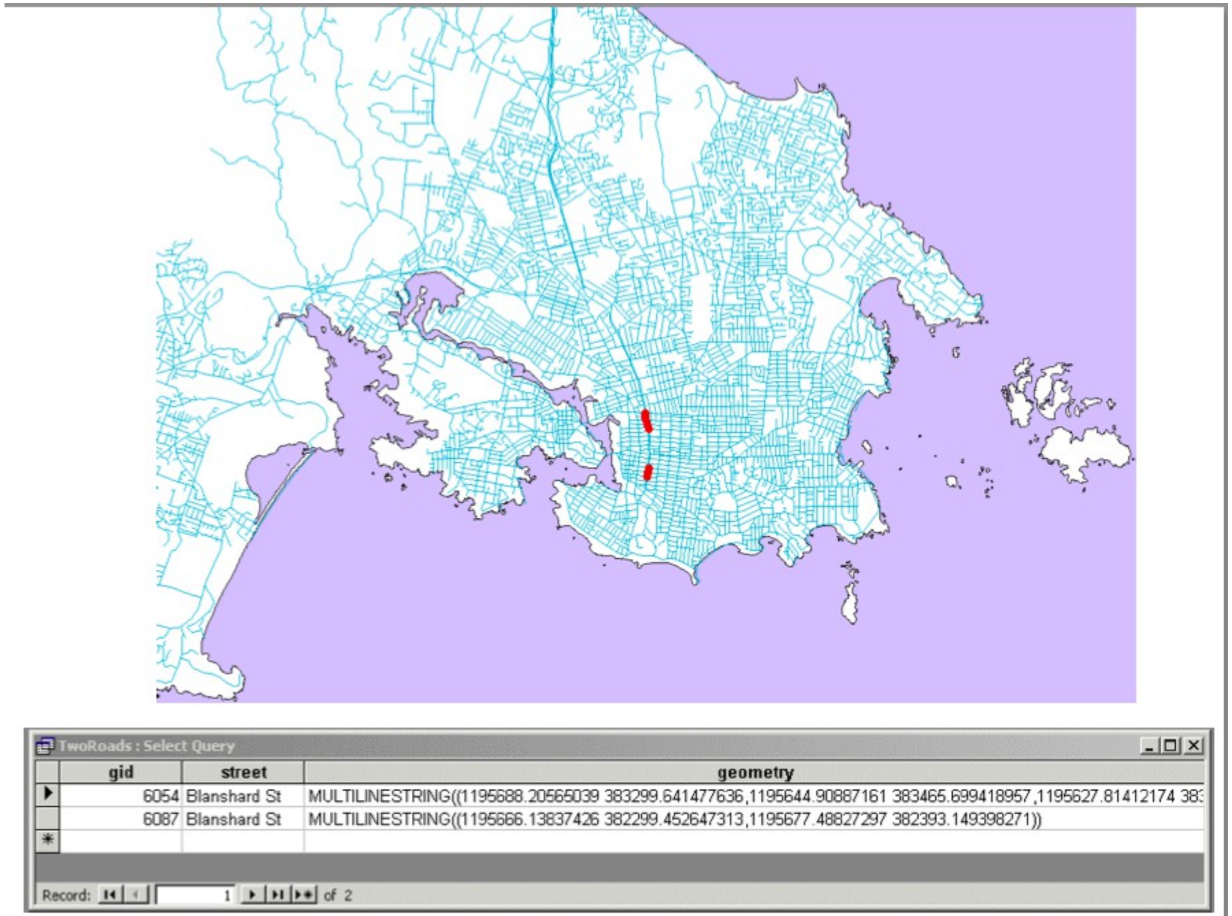
For a simple example of using a PostGIS database, let's go back to **FOSS4G 2003** and an example by David Blasby, founding member of the Open Source Geospatial Foundation **OSGeo** and **PostGIS** pioneer

Example Use Case

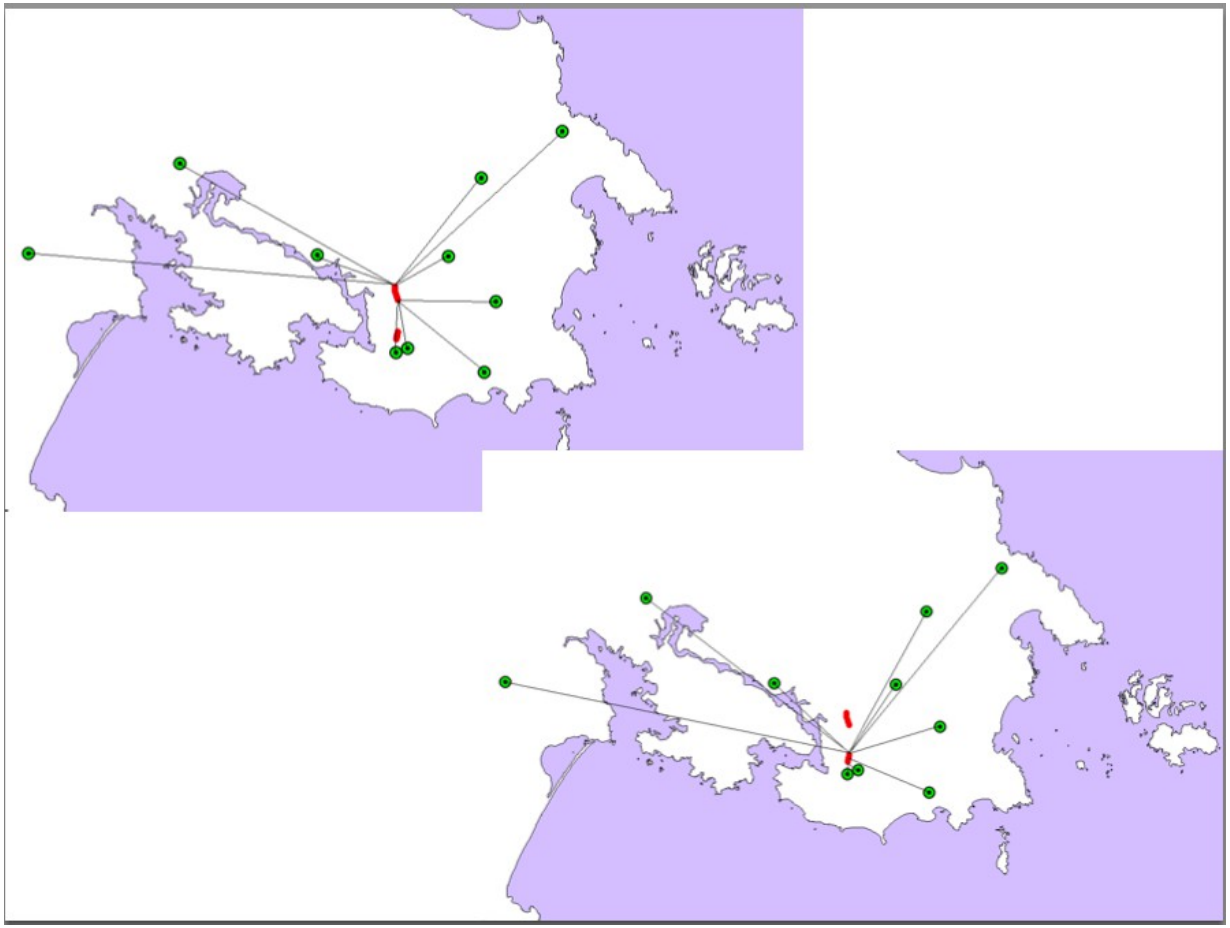


name	location
Aberdeen Hospital	POINT(1197001.65231262 384398.270475177)
Fairfield Health Centre	POINT(1195656.24647344 381944.573517462)
Glengarry Hospital	POINT(1197905.18862936 381440.608596903)
Gorge Road Hospital	POINT(1193660.45242158 384433.153379774)
Mount St Mary Hospital	POINT(1195956.67018638 382058.203488371)
Mount Tolmie Hospital	POINT(1197829.67761525 386402.810630837)
Priory Hospital	POINT(1186271.98258414 384464.968142696)
Queen Alexandra Centre for Children	POINT(1199897.96322968 387588.909663164)
Royal Jubilee Hospital	POINT(1198193.92380713 383235.126501604)
Victoria General Hospital	POINT(1190145.27946905 386773.706085023)
Saanich Peninsula Hospital	POINT(1191394 401311)

Example Use Case



Example Use Case



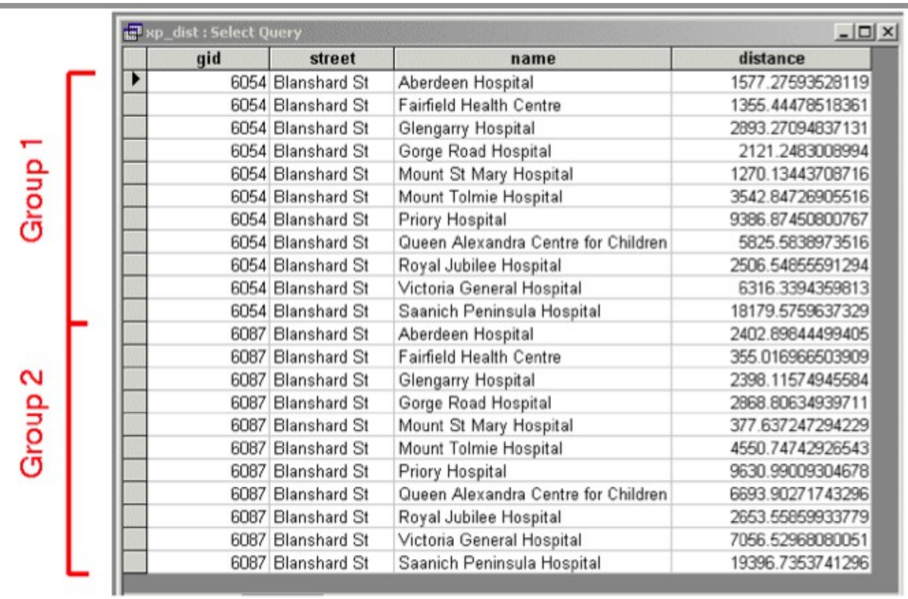
Example Use Case

xp_dist : Select Query

gid	street	name	distance
6054	Blanshard St	Aberdeen Hospital	1577.27593528119
6054	Blanshard St	Fairfield Health Centre	1355.44478518361
6054	Blanshard St	Glengarry Hospital	2893.27094837131
6054	Blanshard St	Gorge Road Hospital	2121.2483008994
6054	Blanshard St	Mount St Mary Hospital	1270.13443708716
6054	Blanshard St	Mount Tolmie Hospital	3542.84726905516
6054	Blanshard St	Priory Hospital	9386.87450800767
6054	Blanshard St	Queen Alexandra Centre for Children	5825.5838973516
6054	Blanshard St	Royal Jubilee Hospital	2506.54855591294
6054	Blanshard St	Victoria General Hospital	6316.3394359813
6054	Blanshard St	Saanich Peninsula Hospital	18179.5759637329
6087	Blanshard St	Aberdeen Hospital	2402.89844499405
6087	Blanshard St	Fairfield Health Centre	355.016966503909
6087	Blanshard St	Glengarry Hospital	2398.11574945584
6087	Blanshard St	Gorge Road Hospital	2868.80634939711
6087	Blanshard St	Mount St Mary Hospital	377.637247294229
6087	Blanshard St	Mount Tolmie Hospital	4550.74742926543
6087	Blanshard St	Priory Hospital	9630.99009304678
6087	Blanshard St	Queen Alexandra Centre for Children	6693.90271743296
6087	Blanshard St	Royal Jubilee Hospital	2653.55859933779
6087	Blanshard St	Victoria General Hospital	7056.52968080051
6087	Blanshard St	Saanich Peninsula Hospital	19396.7353741296

```
SELECT
    distance(hospitals.the_geom,roads.the_geom)
FROM hospitals, roads;
```

Example Use Case



gid	street	name	distance
6054	Blanshard St	Aberdeen Hospital	1577.27593528119
6054	Blanshard St	Fairfield Health Centre	1355.44478518361
6054	Blanshard St	Glengarry Hospital	2893.27094837131
6054	Blanshard St	Gorge Road Hospital	2121.2483008994
6054	Blanshard St	Mount St Mary Hospital	1270.13443708716
6054	Blanshard St	Mount Tolmie Hospital	3542.84726905516
6054	Blanshard St	Priory Hospital	9386.87450800767
6054	Blanshard St	Queen Alexandra Centre for Children	5825.5838973516
6054	Blanshard St	Royal Jubilee Hospital	2506.54855591294
6054	Blanshard St	Victoria General Hospital	6316.3394359813
6054	Blanshard St	Saanich Peninsula Hospital	18179.5759637329
6087	Blanshard St	Aberdeen Hospital	2402.89844499405
6087	Blanshard St	Fairfield Health Centre	355.016966503909
6087	Blanshard St	Glengarry Hospital	2398.11574945584
6087	Blanshard St	Gorge Road Hospital	2868.80634939711
6087	Blanshard St	Mount St Mary Hospital	377.637247294229
6087	Blanshard St	Mount Tolmie Hospital	4550.74742926543
6087	Blanshard St	Priory Hospital	9630.99009304678
6087	Blanshard St	Queen Alexandra Centre for Children	6693.90271743296
6087	Blanshard St	Royal Jubilee Hospital	2653.55859933779
6087	Blanshard St	Victoria General Hospital	7056.52968080051
6087	Blanshard St	Saanich Peninsula Hospital	19396.7353741296

The groups are defined by gid, street.

Example Use Case

xp_dist : Select Query

gid	street	name	distance
6054	Blanshard St	Aberdeen Hospital	1577.27593528119
6054	Blanshard St	Fairfield Health Centre	1355.44478518361
6054	Blanshard St	Glengarry Hospital	2893.27094837131
6054	Blanshard St	Gorge Road Hospital	2121.2483008994
6054	Blanshard St	Mount St Mary Hospital	1270.13443708716
6054	Blanshard St	Mount Tolmie Hospital	3542.84726905516
6054	Blanshard St	Priory Hospital	9386.87450800767
6054	Blanshard St	Queen Alexandra Centre for Children	5825.5838973516
6054	Blanshard St	Royal Jubilee Hospital	2506.54855691294
6054	Blanshard St	Victoria General Hospital	6316.3394359813
6054	Blanshard St	Saanich Peninsula Hospital	18179.5759637329
6087	Blanshard St	Aberdeen Hospital	2402.89844499405
6087	Blanshard St	Fairfield Health Centre	355.016966503909
6087	Blanshard St	Glengarry Hospital	2398.11574945584
6087	Blanshard St	Gorge Road Hospital	2868.80634939711
6087	Blanshard St	Mount St Mary Hospital	377.637247294229
6087	Blanshard St	Mount Tolmie Hospital	4550.7474292654 3
6087	Blanshard St	Priory Hospital	9630.99009304678
6087	Blanshard St	Queen Alexandra Centre for Children	6693.90271743296
6087	Blanshard St	Royal Jubilee Hospital	2653.55869933779
6087	Blanshard St	Victoria General Hospital	7056.52968080051
6087	Blanshard St	Saanich Peninsula Hospital	19396.7353741296

Group 1 (gid 6054 rows); Group 2 (gid 6087 rows)

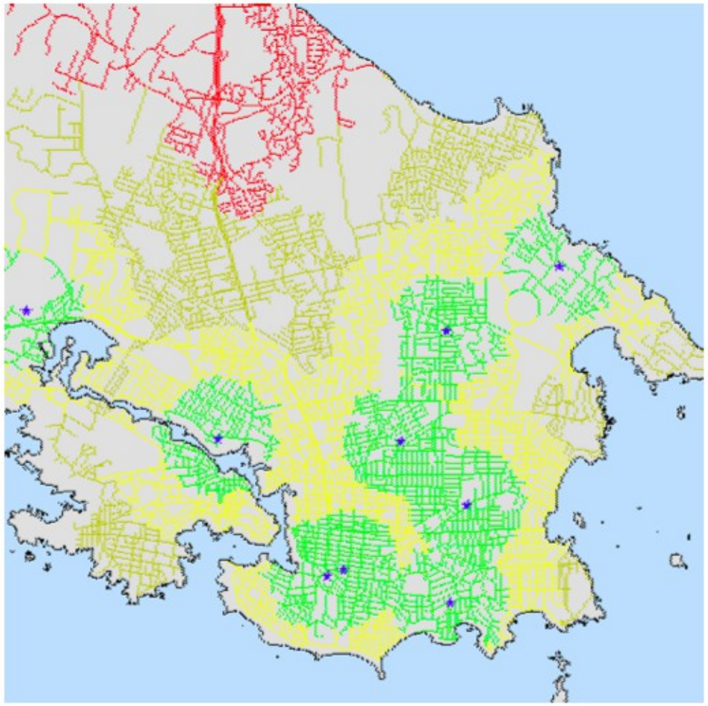
```
SELECT
      road.gid,
      road.street,
      min(distance(hospitals.the_geom,r.the_geom)) as min_distance
FROM roads,hospitals
GROUP BY  road.gid, street;
```

Query1 : Select Query

gid	street	min_distance
6054	Blanshard St	1270.13443709
6087	Blanshard St	355.016966504

Record: 1 of 2

Example Use Case



Spatial Data All

PROJ.4

Search

Login | Help/Guide | About Trac | Preferences

Wiki | Timeline | Roadmap | Browse Source | View Tickets | Search

Start Page | Index | History | Last Change

PROJ.4 - Cartographic Projections Library

This web page relates to the PROJ.4 Cartographic Projections library originally written by Gerald Evenden then of the USGS.

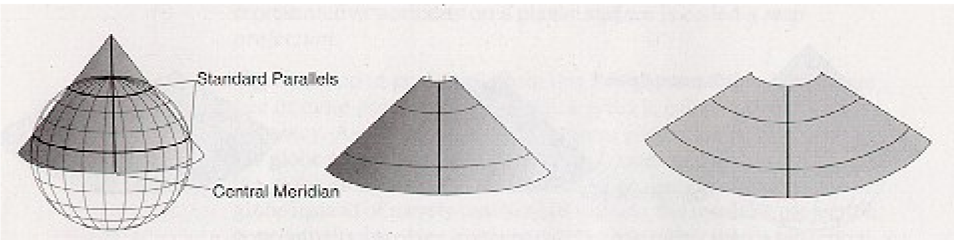
Download

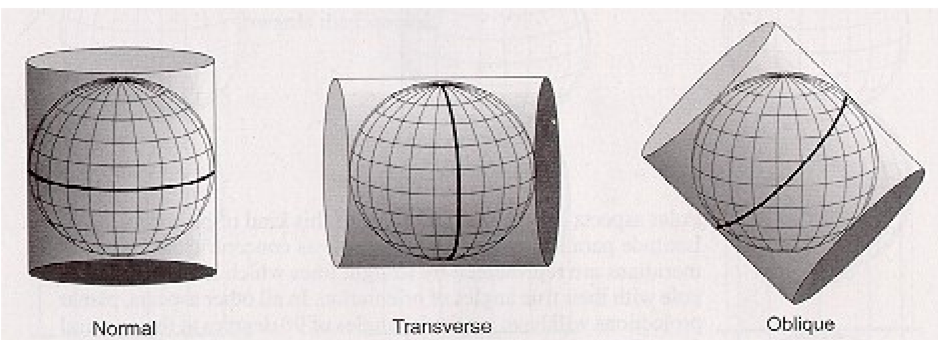
The current development source is available by anonymous SVN from http://svn.osgeo.org/metacrs/proj/trunk/proj.

The following files are available from the proj ftp directory or http mirror.

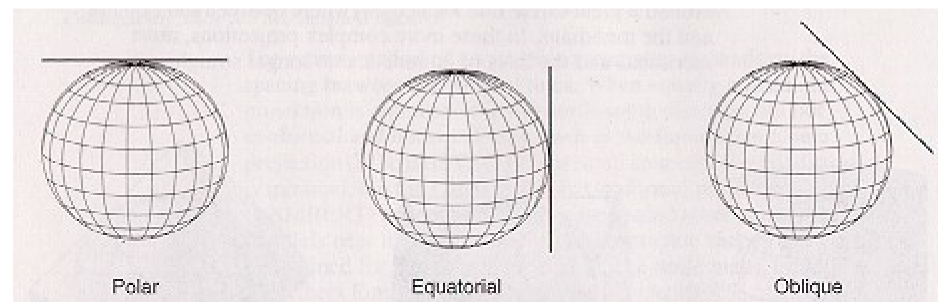
- Source Code:
  - http://download.osgeo.org/proj/proj-4.8.0.tar.gz or http://download.osgeo.org/proj/proj-4.8.0.zip: Current source release.
- Datum shift grids:
  - proj-datumgrid-1.5.zip: US, Canadian, French and New Zealand datum shift grids - unzip in the nad directory before configuring to add NAD27/NAD83 and NZGD49 datum conversion.
  - SwissGrid
  - NonFreeGrids
  - HarnGrids
  - HTDPGrids - NAD83/WGS84 conversion grids based on NOAA/NGS HTDP Model.
  - Geocentric Datum of Australia AGD66/AGD84
  - Canadian grid for NAD27.
  - German BeTA2007 DHDN GK3 => ETRS89/UTM
  - UK's OSTN02_NTv2: OSGB 1936 => ETRS89
  - Austrian Grid for MGI.
  - Spanish grids for ED50.
  - Portuguese grids for ED50, Lisbon 1890, Lisbon 1937 and Datum 73.
  - Brazilian grids for datums Corrego Alegre 1961, Corrego Alegre 1970-72, SAD69 and SAD69(96).
  - South African grid (Cape to Hartebeesthoek94 or WGS84).
- Binaries:
  - PostgreSQL RPM Repository: Up2date Proj (and other GIS) RPMs
  - proj446_win32_bin.zip: Prebuilt Win32 executables, DLL including NAD27 grid shift files.
  - The openSUSE Application:Geo Repository offers current PROJ.4 RPMs for SuSe.
  - PROJ.4 in a Debian Package
  - PROJ.4 as a pkgsrc package.
  - PROJ.4 Ported to the Delphi (Borland C++) environment.

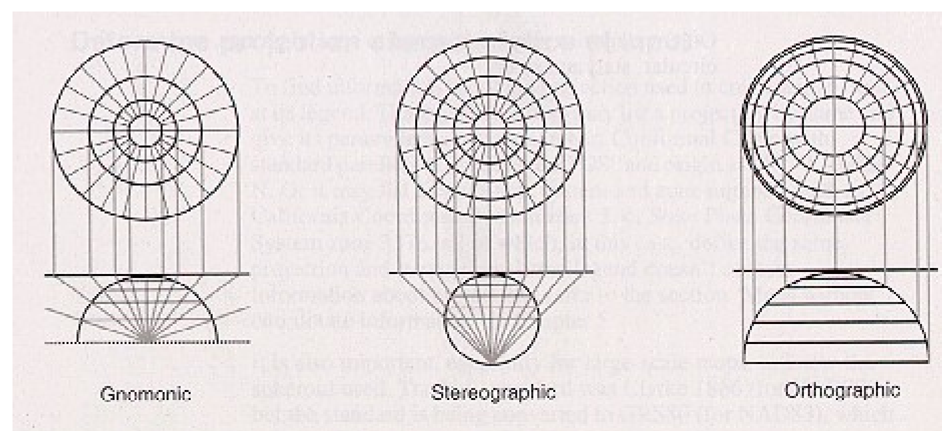
Projections





Images via ***University of Washington***
Projections_Coordinate_Systems_ESRM_250_UW.html





Briefly !!

Projections

There is a fundamental concept in geo data that is new to most database builders, and that is **Projections**

Map projection
From Wikipedia, the free encyclopedia

"A map projection is any method of representing the surface of a sphere or other three-dimensional body on a plane. Map projections are necessary for creating maps. All map projections distort the surface in some fashion. Depending on the purpose of the map, some distortions are acceptable and others are not; therefore different map projections exist in order to preserve some properties of the sphere-like body at the expense of other properties. There is no limit to the number of possible map projections."

http://courses.washington.edu/gis250/lessons/projection/

Projections

Why care about projections?

From the OGC Simple Features Specification:

Every geometry column is associated with a Spatial Reference System. The Spatial Reference System (SRS) identifies the coordinate system for all geometries stored in the column, and gives meaning to the numeric coordinate values for any geometry instance stored in the column. Examples of commonly used Spatial Reference Systems include 'Latitude Longitude', and 'UTM Zone 10'.

Projections Lat/Long

Latitude / Longitude

- May use the **GEOGRAPHY** type

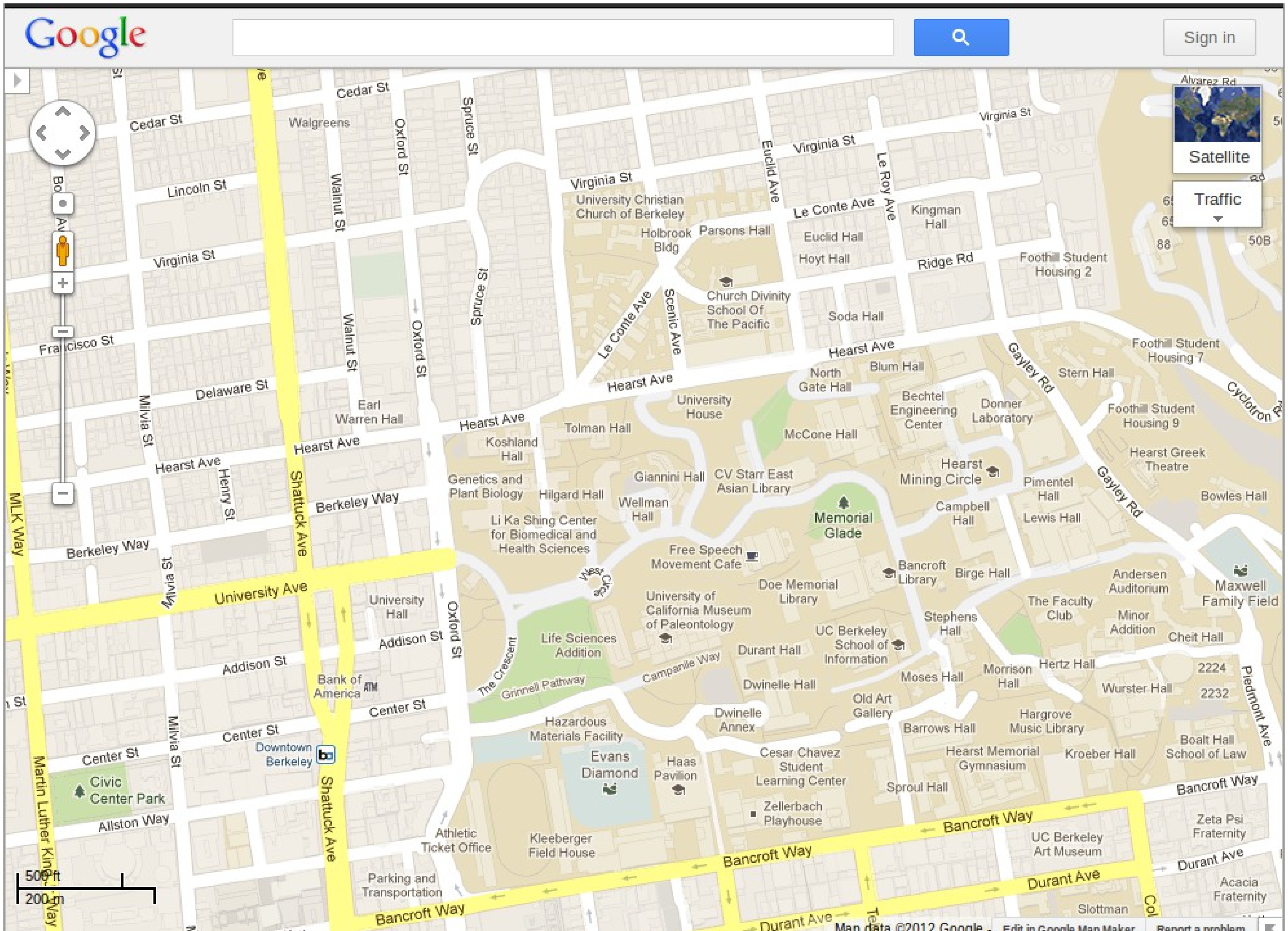
EPSG:4326

WGS 84 (Google it)

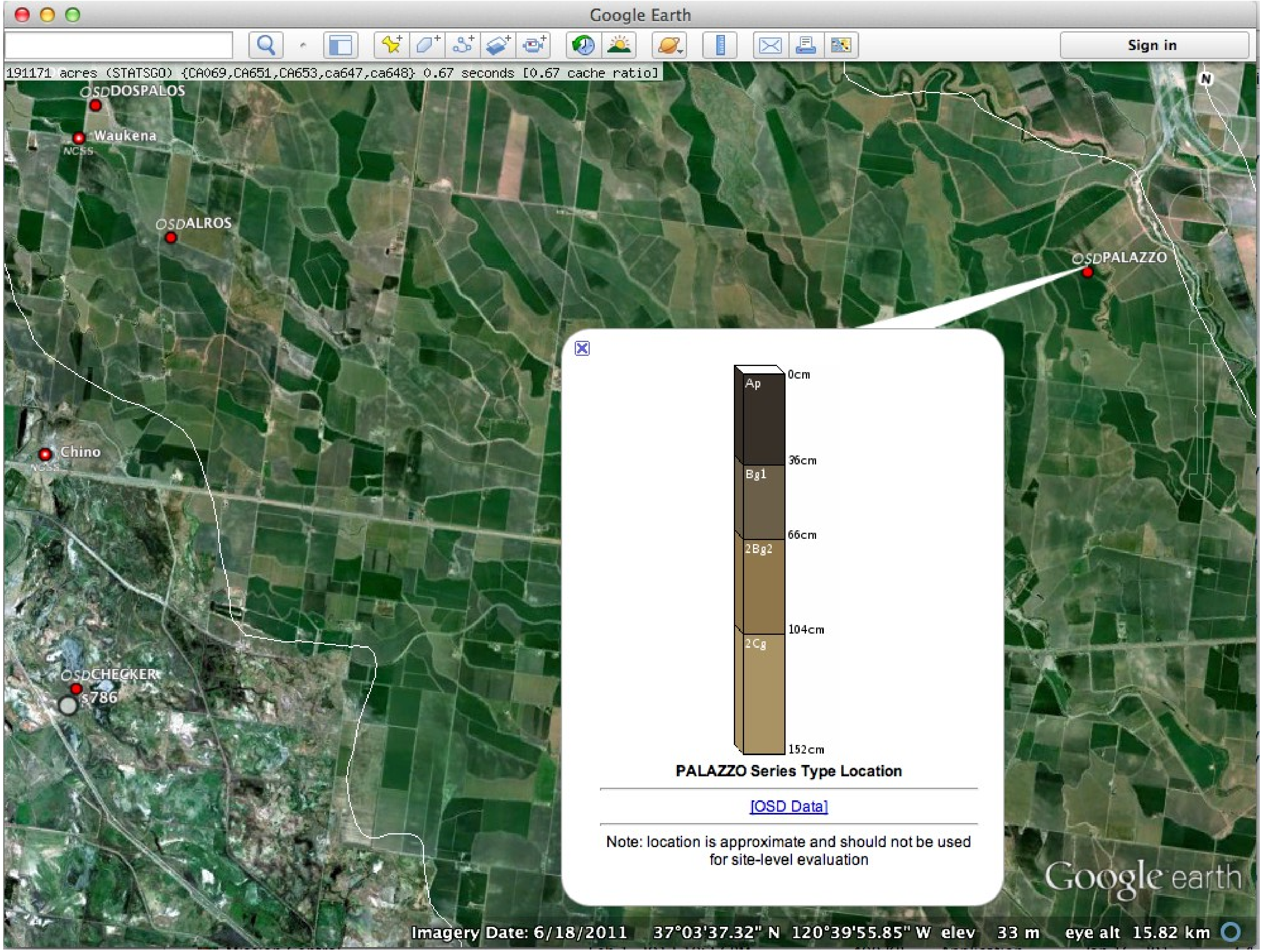
- **WGS84 Bounds**: -180.0000, -90.0000, 180.0000, 90.0000
- **Projected Bounds**: -180.0000, -90.0000, 180.0000, 90.0000
- **Scope**: Horizontal component of 3D system. Used by the GPS satellite navigation system and for NATO military geodetic surveying.
- **Last Revised**: 2007-08-27
- **Area**: World

http://postgis.refractions.net/documentation/manual-2.0/using_postgis_dbmanagement.html#PostGIS_Geography

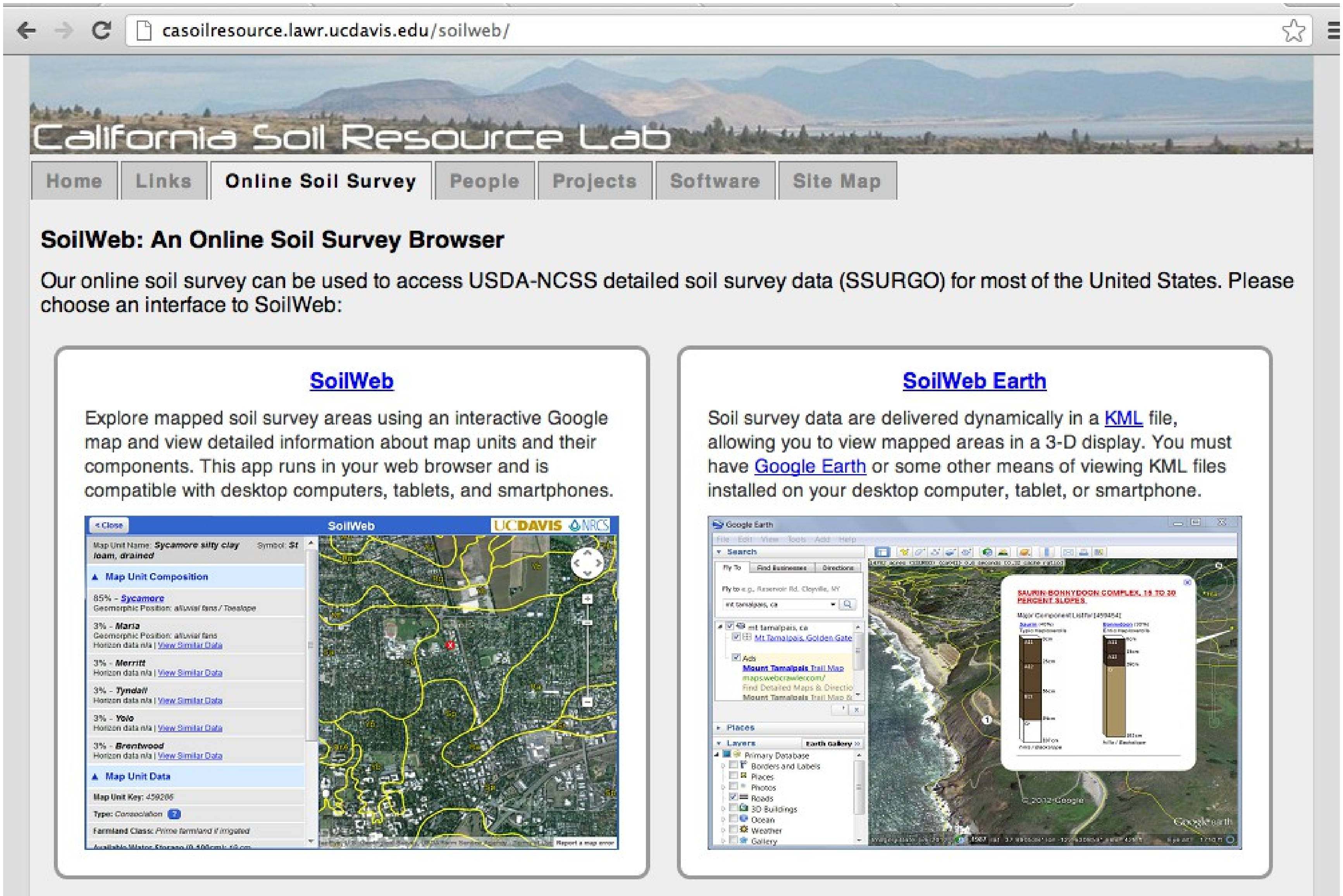
Projections Lat/Long



Projections Lat/Long



Projections Lat/Long



Projections

Lambert Equal Area

MODIS is an extensive program using sensors on two satellites that each provide complete daily coverage of the earth.
spectral, spatial and temporal

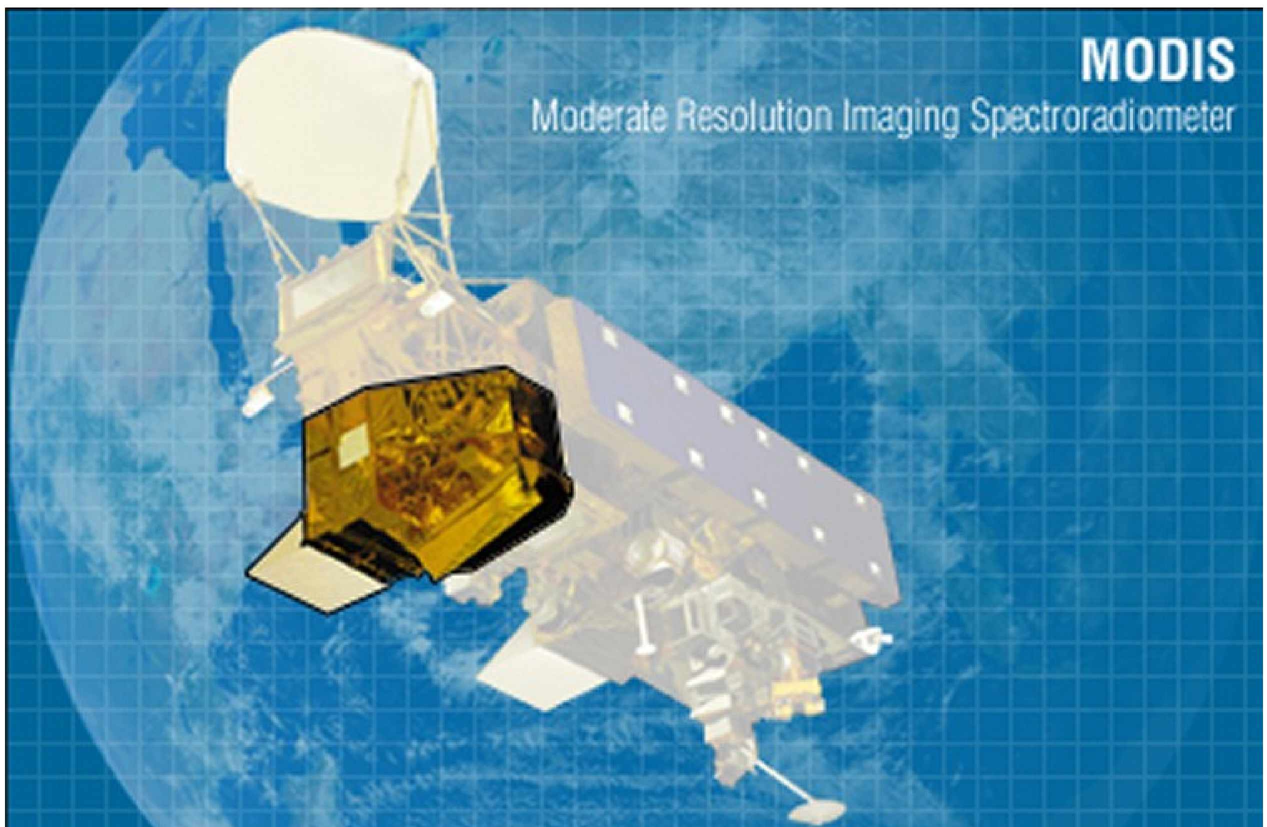
Some data products use Lambert Equal Area

SR-ORG:28

lambert azimutha equal area (Google it)

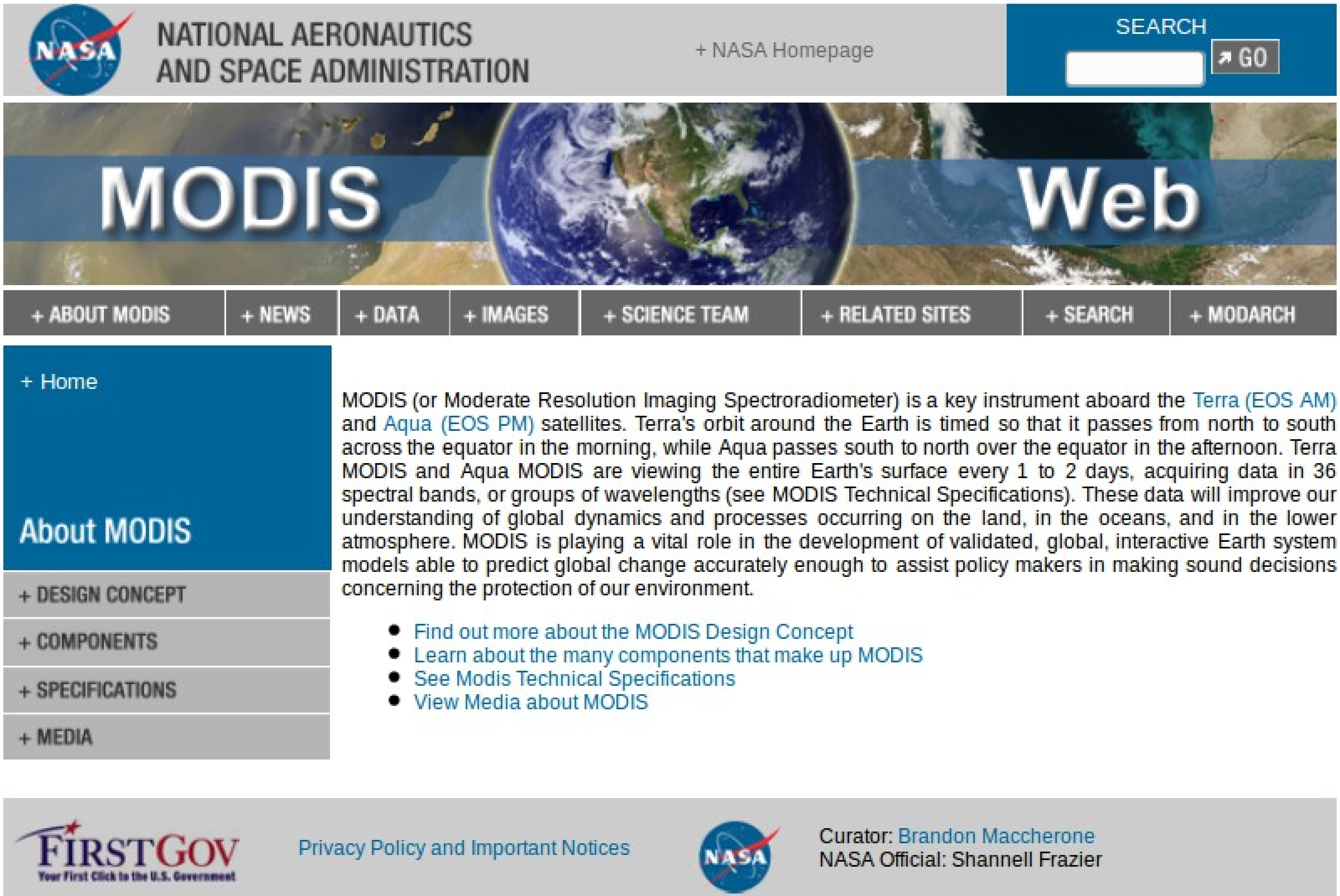
Lambert Azimuthal Equal Area definition used for MODIS rasters by The Remote Sensing application Center

- Well Known Text as HTML



Projections

Lambert Equal Area



Projections EPSG:900913

- Why care about Spherical-Mercator Projection?

from the OpenLayers Manual:

SphericalMercator and EPSG aliases

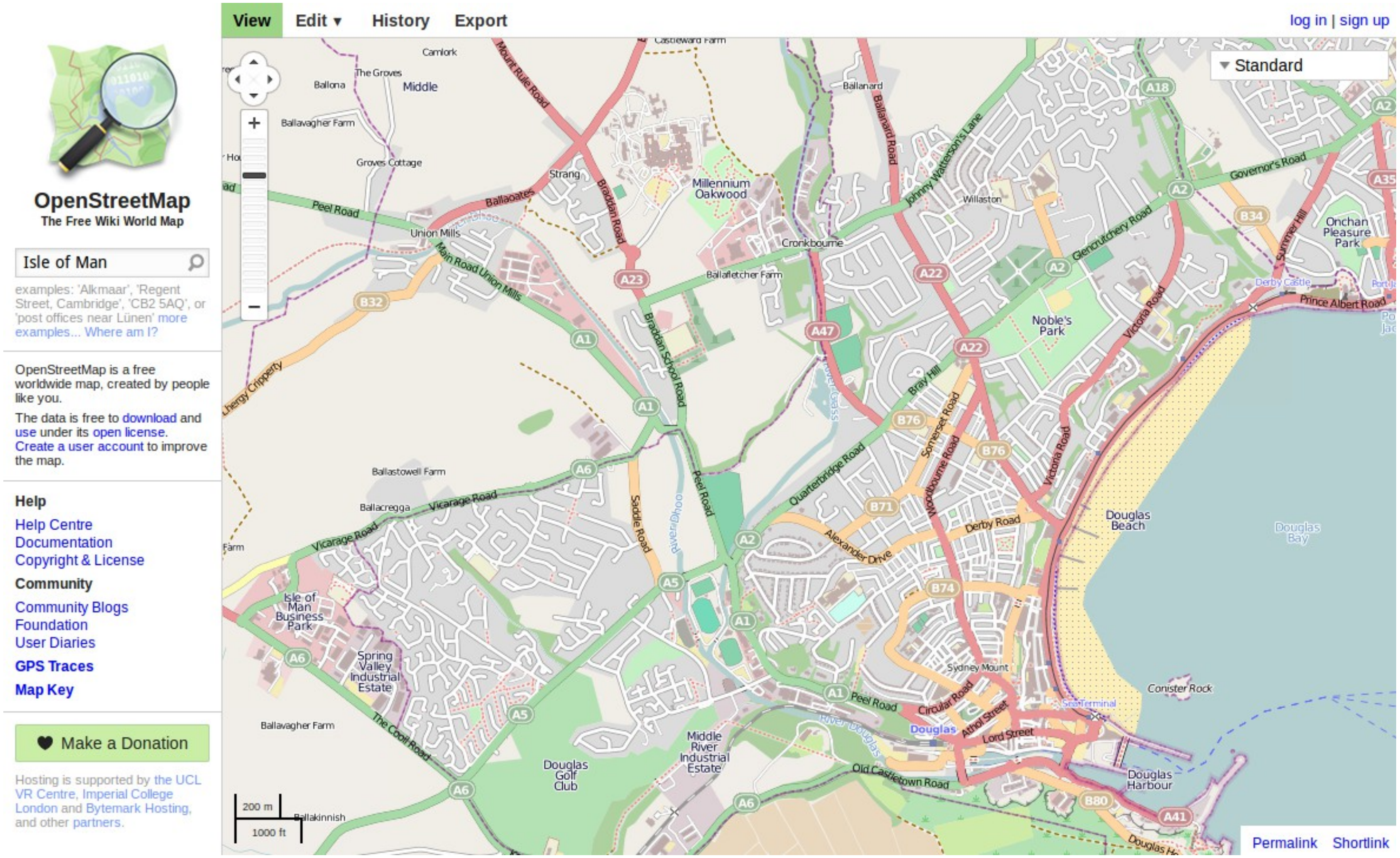
The SphericalMercator projection in OpenLayers uses code EPSG:900913. Many other services, such as OpenStreetMap, Bing and Yahoo are now also using the same projection, but are not necessarily supporting the use of code EPSG:900913. Other codes, such as EPSG:3857 and EPSG:102113 were invented. Today, there is an officially registered EPSG code 3857 whose projection is identical to EPSG:900913. (http://www.epsg-registry.org /export.htm?gml=urn:ogc:def:crs:EPSG::3857). So, if you need to combine overlay layers that are using either an alias or the official EPSG code with an OpenLayers SphericalMercator layer, you have to make sure that OpenLayers requests EPSG:3857 or other alias in stead of EPSG:900913. You can accomplish this by overriding the layer projection before adding the layer to the map. For example:

```
// create sphericalmercator layers
var googleLayer = new OpenLayers.Layer.Google("Google", {"spherical
var osmLayer = new OpenLayers.Layer.OSM("OpenStreetMap");

// override default epsg code
aliasproj = new OpenLayers.Projection("EPSG:3857");
googleLayer.projection = osmLayer.projection = aliasproj;


//add baselayers to map
map.addLayers([googleLayer, osmLayer]);
```

At this point, overlays (such as WMS layers) will be requested using the 3857 code; transformations will work between 4326 and 3857 as expected.

Projections EPSG:900913

- Why care about Spherical-Mercator Projection?



Google Maps · Bing · MapQuest · OpenStreetMap · Yahoo Maps

KNN

Indexed Nearest Neighbour Search in PostGIS

September 28th, 2011

An always popular question on the PostGIS users mailing list has been
"**how do I find the N nearest things to this point?**".
…
PostgreSQL has the ability to return ordered information where an index exists, but the ability has been restricted to B-Tree indexes until recently. Thanks to one of our clients, we were able to directly fund PostgreSQL developers **Oleg Bartunov** and **Teodor Sigaev** in adding the ability to return sorted results from a **GiST index**. And since PostGIS indexes use GiST, that means that now we can also return sorted results from our indexes. Which is a very long way of saying that PostGIS (the development code in the source repository) now has the ability to do index-assisted nearest neighbour searching.

This feature (the PostGIS side of it) was funded by Vizzuality, and hopefully it comes in useful in their CartoDB work.

requires: PostgreSQL 9.1 and PostGIS 2.0

Paul Ramsey, OpenGeo Blog

KNN

- http://blog.light42.com/wordpress/?p=897
- http://blog.light42.com/wordpress/?p=102

Topology

- http://postgis.refractions.net/docs/Topology.html
- http://blog.light42.com/wordpress/?p=209
- http://blog.light42.com/wordpress/?p=484

TIGER

- http://blog.light42.com/wordpress/?p=514

- **TIGER** data, published by the **Federal Census Bureau**, is a street network basis for the US Census.

- Crowd-sourced mapping and commercial GPS methods are now beating traditional paper methods, by a wide margin, for completeness and accuracy. The Census Bureau knows this and are evolving.

TIGER Geocoder

- http://www.postgis.org/documentation/manual-svn/Geocode.html

- Geocode and Reverse-Geocode functions
- http://blog.light42.com/wordpress/?p=453

- Improved speed and accuracy
- Uses PostgreSQL table inheritance for US States
- Supports 2010 now, 2012 soon

Tools

- ST_Snap()
- ST_Split()
- ST_SharedPaths()
- ST_UnaryUnion()

  ...
- **ST_MakeValid()**

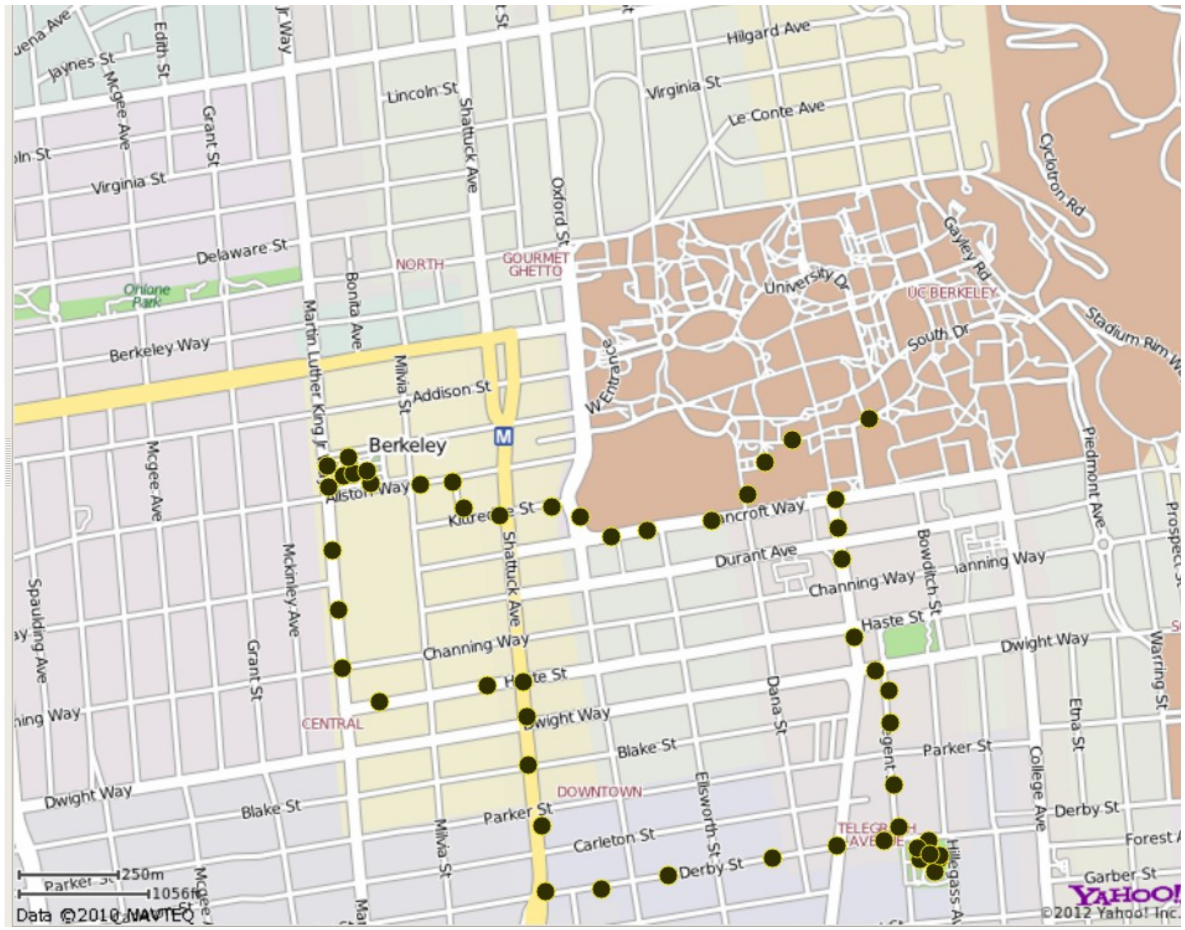
  http://blog.light42.com/wordpress/?p=869

Tools **ST_ConcaveHull**

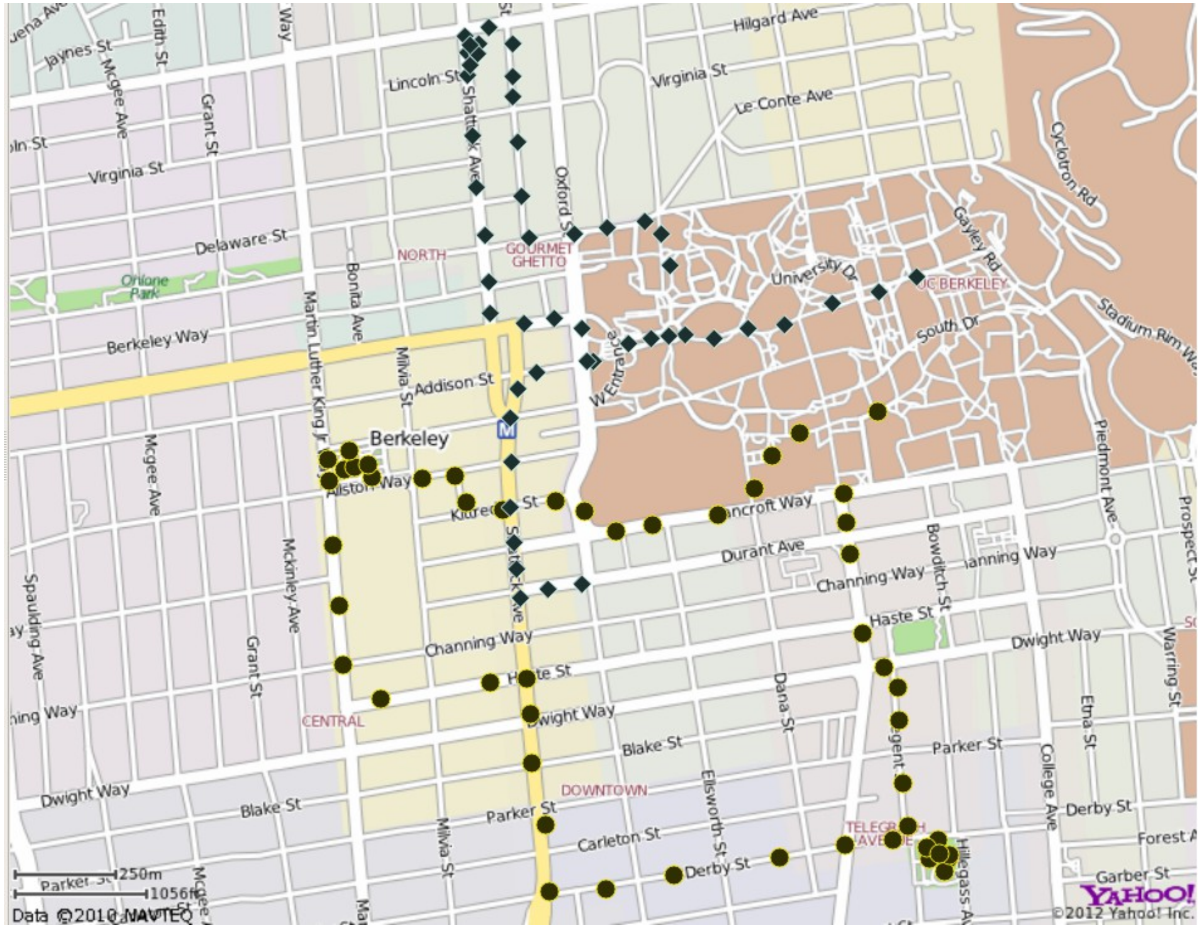
A marketing startup, Zed Masters, with an unfortunate lack of regard for personal privacy of users, has hacked access to the iPhone GPS unit in its free iPhone app "ZedMasters Hockey Face-off". When asked ZedMasters say that GPS Access is to enable users to find the nearest hockey game at any time, but in fact, they sell the location data gathered to other marketing companies, and use it themselves for other purposes. Here in Berkeley, ZedMasters wants to decide where to locate their ice cream and t-shirt sales station in a place where a lot of their users walk. To do so, they take tracks from the **iPhone GPS** data and perform an analysis to find out where the tracks overlap. <slides a,b,c,d,e> One approach is to feed some points to **ST_ConvexHull()** and then find the intersection of the resultant polygons, like so.

However, a young engineering hire from **UC Berkeley** noticed that **PostGIS 2.0** has a new function, called **ST_ConcaveHull()**, which gives a "shrink wrap" polygon around the points with a user defined tightness threshold. It turns out that this gives a far better picture of where the tracks overlap, here <slides d,e,f>

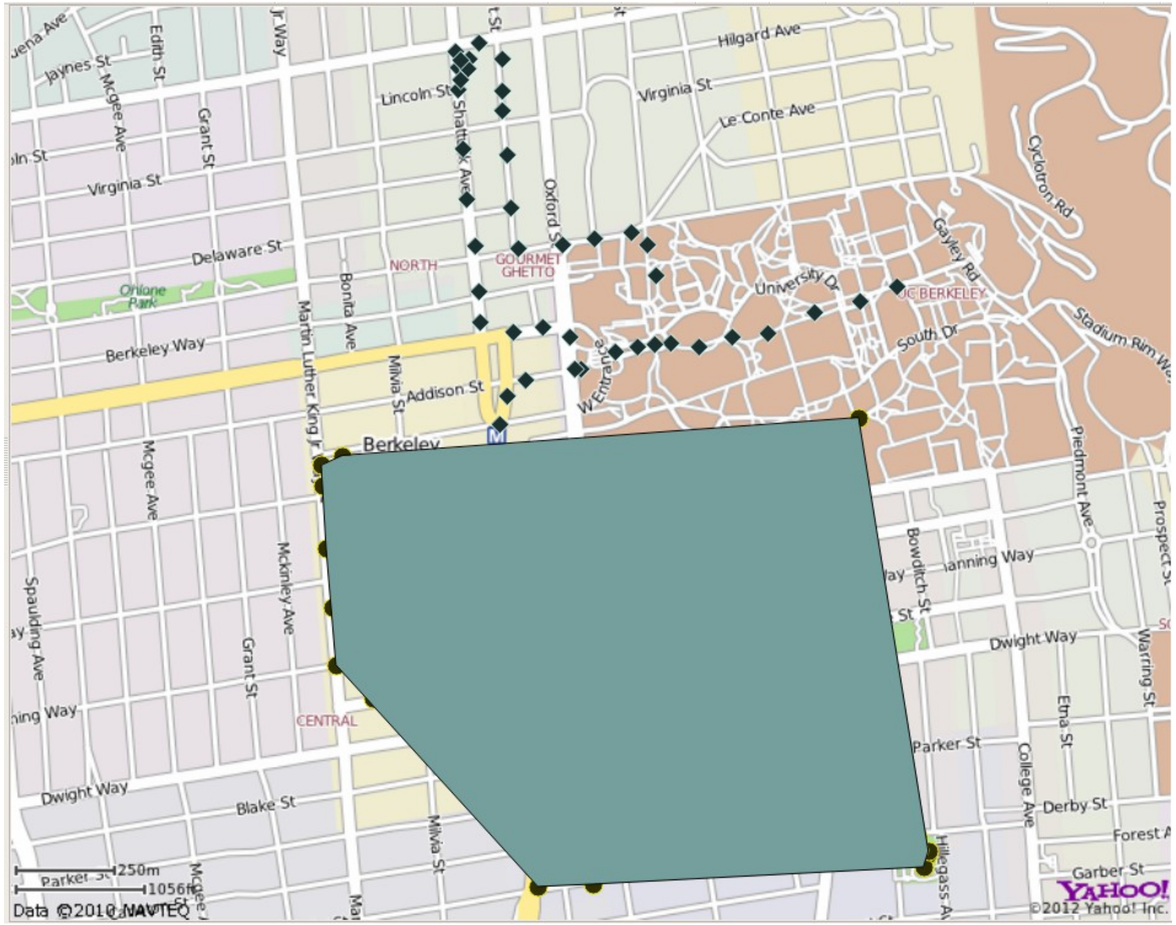
Tools ST_ConcaveHull



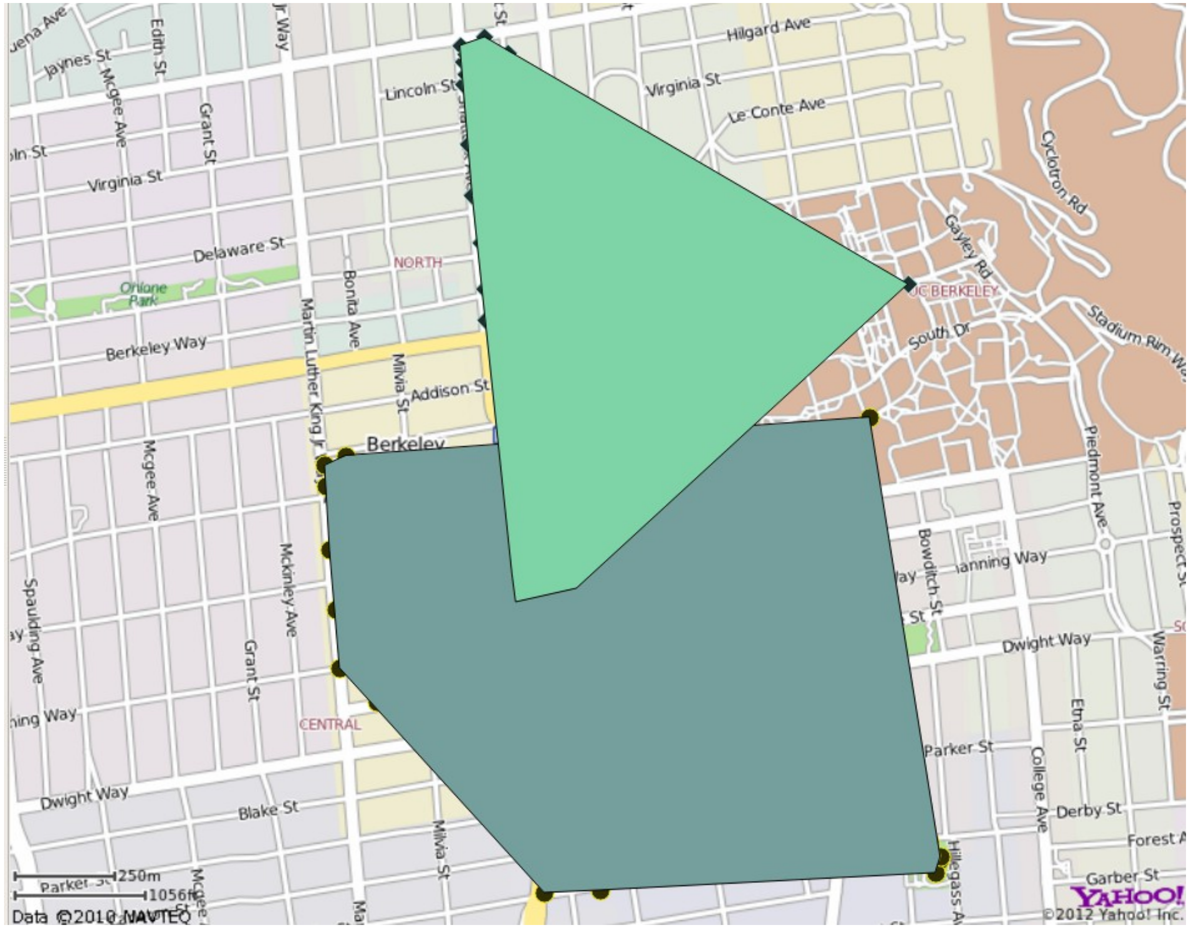
Tools ST_ConcaveHull



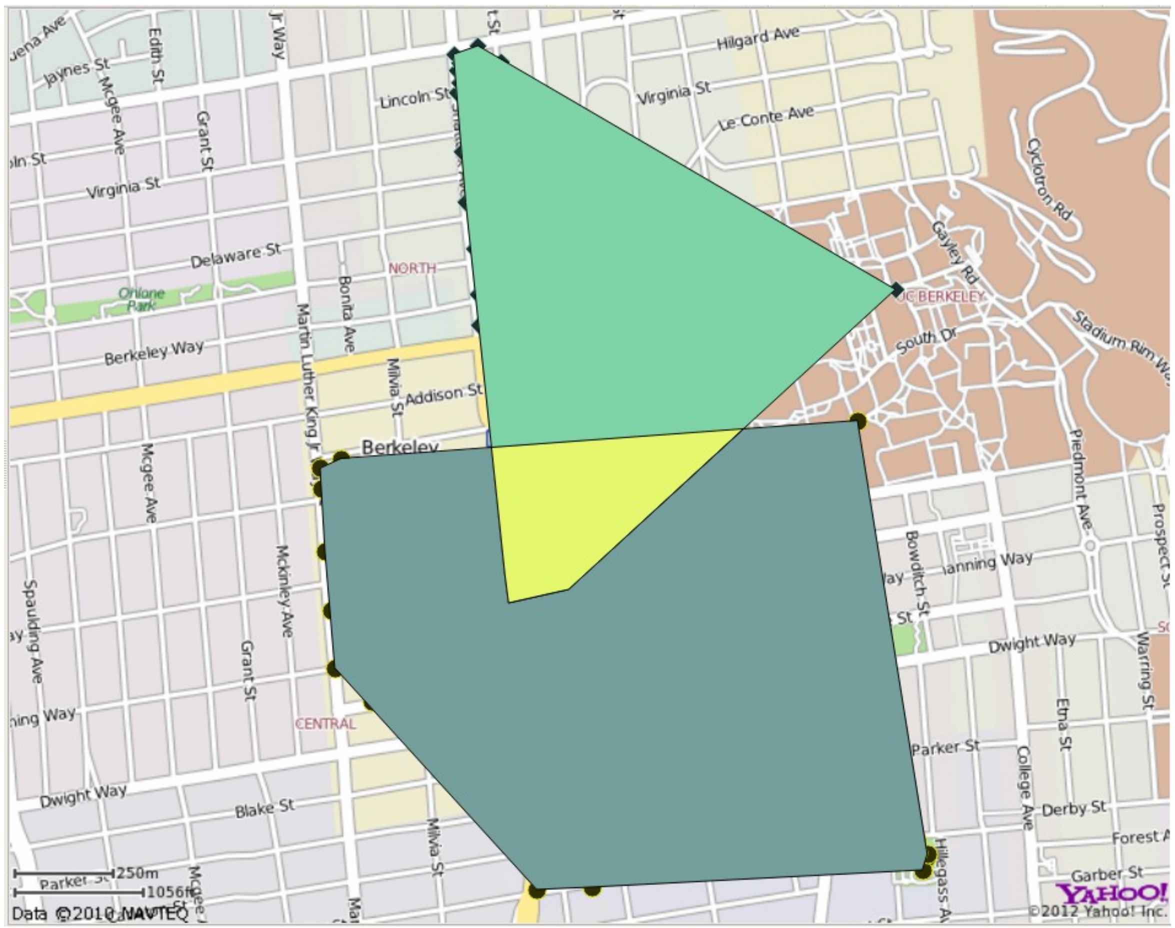
Tools ST_ConvexHull



Tools ST_ConvexHull

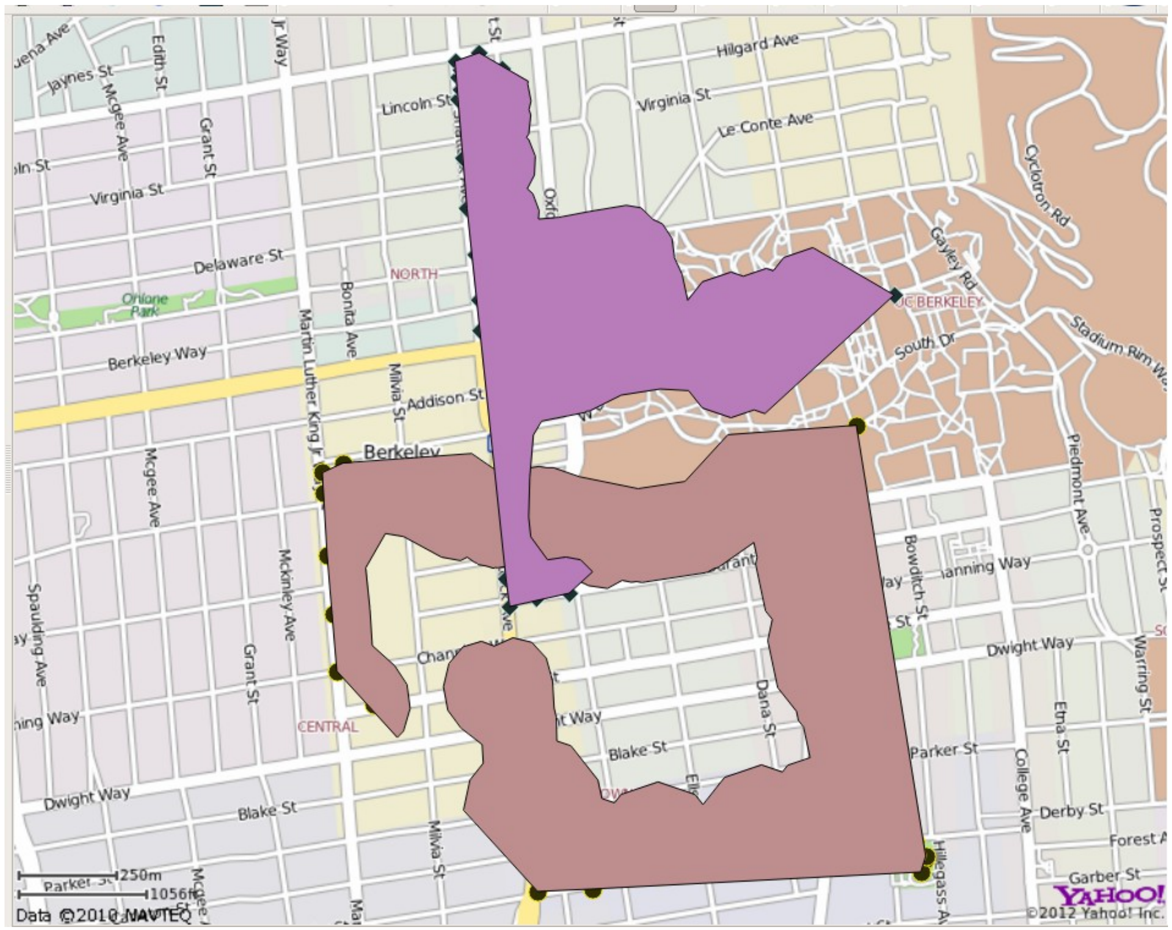


Tools ST_ConvexHull



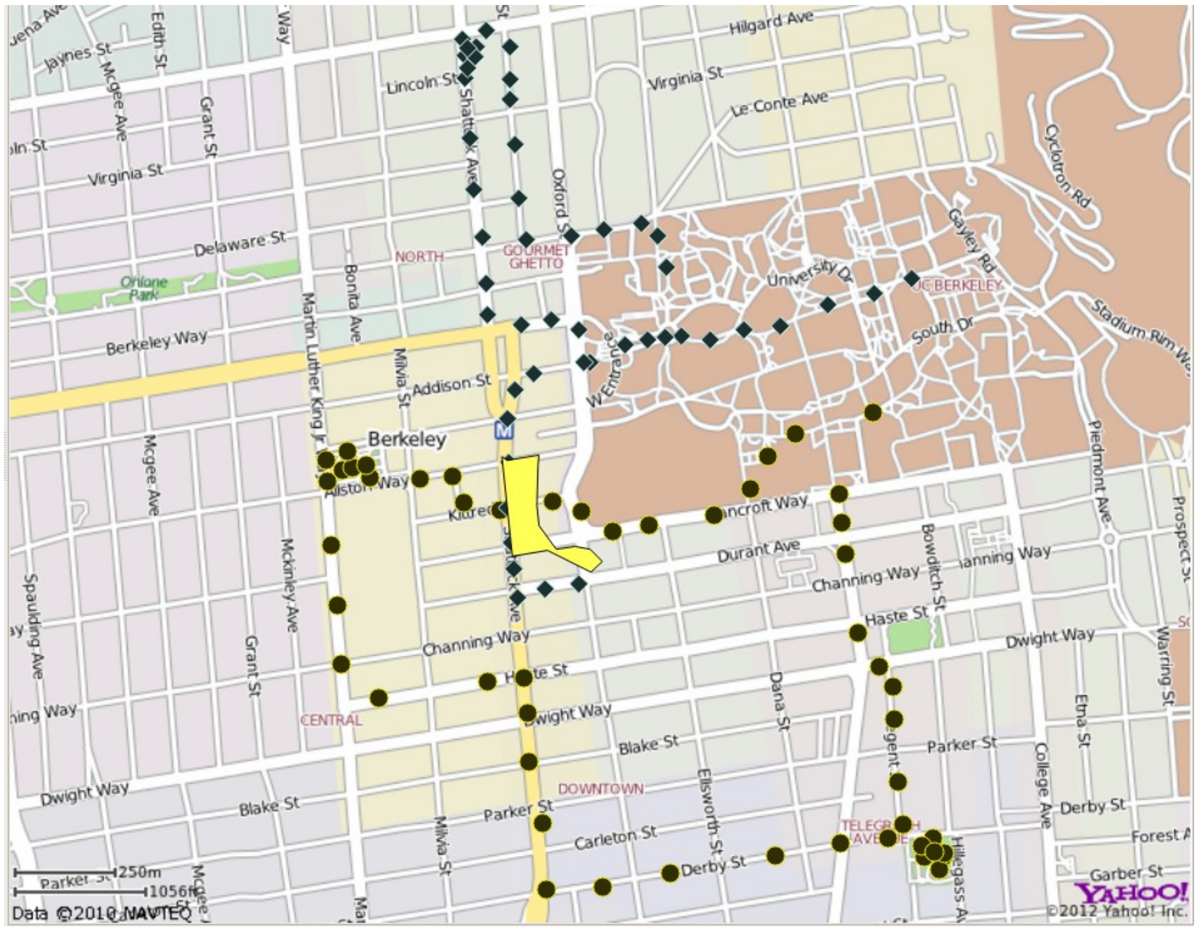
Tools ST_ConcaveHull

Tools **ST_ConcaveHull**

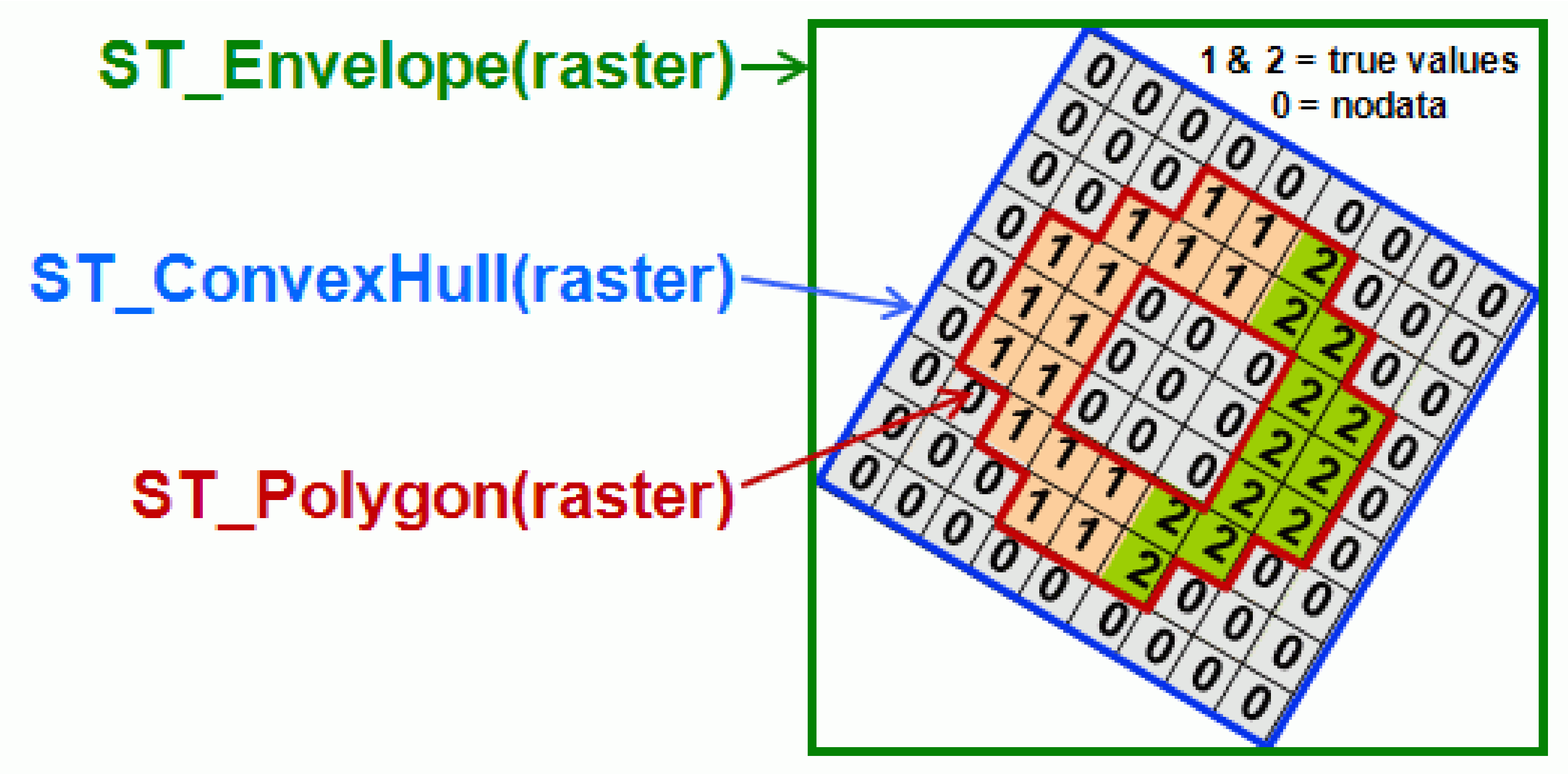


Tools ST_ConcaveHull

Tools ST_ConcaveHull



PostGIS Raster



Slides by "dustymugs"

Spatial Data – 3D

&&&
Returns TRUE if A's 3D bounding box intersects B's 3D bounding box, supports TINs

nd-index
New index type

ST_3DClosestPoint
Returns the 3-dimensional point on g1 closest to g2. This is the first point of the 3D shortest line.

ST_3DDFullyWithin
Returns true if all of the 3D geometries are within the specified distance of one another.

ST_3DDWithin
For 3d (z) geometry type Returns true if two geometries 3d distance is within number of units.

ST_3DDistance
For geometry type Returns the 3-dimensional cartesian minimum distance (based on spatial ref) between two geometries in projected units.

ST_3DExtent
an aggregate function that returns the box3D bounding box that bounds rows of geometries.

ST_3DIntersects
Returns TRUE if the Geometries "spatially intersect" in 3d - only for points and linestrings

Look into X3D for more details

sfcgal_viewer_demo.avi

More Resources

- **OpenGeo** – Intro to PostGIS

  http://workshops.opengeo.org/postgis-intro/index.html

- **BostonGIS dot com**

- *the fine online manual, wiki, mailing lists*

  and don't forget, the source code