GeoSpace your Rails Apps!

Peter Jackson, Intridea
@peteonrails
Although I love New Zealand, I am not Peter Jackson the filmmaker. If I were, I would not be taking your phone calls.

It’s pretty easy for me to get reservations in LA.

Though I’m not the famous filmmaker, I am still sorry about King Kong.
I am not affiliated with the brand of cigarettes from Australia.

In fact, I have nothing to do with the southern hemisphere at all.

I AM PETER JACKSON, THE TECHNOLOGIST. RIGHT?
Still not me

WRONG. I'm not this Peter Jackson either.

Again, if I were, I wouldn’t be taking your calls.
Ah. There I am. I am Peter Jackson, the DC based Rubyist, the lover of maps and mountains now hailing from New Hampshire.
I am here because:

- I love maps
- We are at a crossroads

I love maps. I always have. I have a map tattooed on my body. No Shit. When I became a programmer, I was naturally drawn to mapping applications. Before getting involved in mobile and web development, that meant I had to work on military and NASA software. As it happens, they like maps too.

So many untapped possibilities that the rubyist can take advantage of. I am here to help inspire you to go create them.

My goals for this talk are:

1. Show you why spatial programming is different from RDBMS or NOSQL applications
2. Inspire you to make more awesome location based applications
3. Give you just enough to be dangerous.
4. Inspire you to create the next level of Rails GIS tools
Agenda

• What is Spatial Programming?
• Important Spatial Terms
• Tools in the spatial stack
• How you get started
• What’s next?

And this is the agenda I will cover in order to give you those tools.
What is Spatial Programming?

• Exposing physical space as a first-order programming concept.

• Rich, built-in support for shapes, space, and the relationship of physical objects to one another.
Describing Objects Spatially

- Locations on the earth
- Geometric equations
- Shapes of buildings
- Parts in an assembly (a blueprint)
- Positions of your battleships

Locations on the earth: Most common spatial application. e.g. Google Earth. Pushpins on a map.

Geometric equations: we might use spatial programming to describe things we learned in geometry class, or even trigonometry class. Things like the areas of polygons, the surface areas and volumes of solids, and even vector and matrix mathematics can be described and stored as first order objects in a spatial system.

Shapes of Buildings: We’ve all seen this on Google Earth. 3 dimensional solids are stored with location information and superimposed on a map.

Parts in an Assembly: A non-location based spatial application, we might capture and store the entire contents of a blueprint into a spatial database and then ask it questions like “Show me all of the components within 12 inches of the fuel line in my vehicle that produce heat.”
Spatial Logic

We can use spatial techniques to answer questions that are difficult to answer with typical object relational data.
Spatial Logic

• Find components of a vehicle close to the fuel line that emit heat.

• List top 10 New York neighborhoods ordered by the proportion of people who have graduate degrees.

• Find competitors located within 10 miles of “Route 95” where there is an available billboard within 20 miles.
Spatial Logic

• Components of a vehicle close to the fuel line that emit heat.

```ruby
RGeo::Shapefile::Reader.open('car_blueprint.shp') do |file|
  file.each do |record|
    components << record.geometry
  end
end

fuel_line = components.select { |c| c.name == "fuel line" }
danger_zone = fuel_line.buffer(15) # 15 units away.
oh_noes = components.select { |c| c.intersects?(buffer) && c.hot? }
```
Spatial Logic

• New York Neighborhoods / Graduate Degrees.
Spatial Logic

• New York Neighborhoods / Graduate Degrees.

```
SELECT Round(100.0 * Sum(t.edu_graduate_dipl) / Sum(t.edu_total), 1) AS graduate_pct,
       n.name, n.boroname
FROM nyc_neighborhoods n
JOIN nyc_census_tracts t
ON ST_Contains(n.the_geom, ST_Centroid(t.the_geom)) # MAGIC!
WHERE t.edu_total > 0
GROUP BY n.name, n.boroname
ORDER BY graduate_pct DESC
LIMIT 10;
```
### Spatial Logic

- **New York Neighborhoods / Graduate Degrees.**

<table>
<thead>
<tr>
<th>graduate_pct</th>
<th>name</th>
<th>boroname</th>
</tr>
</thead>
<tbody>
<tr>
<td>40.4</td>
<td>Carnegie Hill</td>
<td>Manhattan</td>
</tr>
<tr>
<td>40.2</td>
<td>Flatbush</td>
<td>Brooklyn</td>
</tr>
<tr>
<td>34.8</td>
<td>Battery Park</td>
<td>Manhattan</td>
</tr>
<tr>
<td>33.9</td>
<td>North Sutton Area</td>
<td>Manhattan</td>
</tr>
<tr>
<td>33.4</td>
<td>Upper West Side</td>
<td>Manhattan</td>
</tr>
<tr>
<td>33.3</td>
<td>Upper East Side</td>
<td>Manhattan</td>
</tr>
<tr>
<td>32.0</td>
<td>Tribeca</td>
<td>Manhattan</td>
</tr>
<tr>
<td>31.8</td>
<td>Greenwich Village</td>
<td>Manhattan</td>
</tr>
<tr>
<td>29.8</td>
<td>West Village</td>
<td>Manhattan</td>
</tr>
<tr>
<td>29.7</td>
<td>Central Park</td>
<td>Manhattan</td>
</tr>
</tbody>
</table>
Important terms
Important terms

- **GIS**: Geographical Information System
- **Layers**: just like the layers in Photoshop, but they're georeferenced.
- **Projection**: algorithm for flatting the globe
- **Geometry**: the core data type in a Geospatial application
GIS

A Geographical Information System visually represents data about geography.

Top Left: GRASS
Top Right: ESRI Arc of iOS (iPad version)
Bottom Left: GRASS (open source GIS)
Bottom Center: ARCGis Desktop
Bottom Right: Nasa WorldWind - The most awesome GIS API that nobody uses, because you have to be a rocket scientist to use it.

I'd like to challenge you at this point. Imagine a web application where you can grab a map, tilt the plane of view, rotate that view to interact with a spatial data set, and then manipulate, calculate, superimpose, and transform that data.

Google Earth just gave us that capability late last year. NASA WorldWind has allowed application developers to do that for many years.

WE HAVE A LOT TO LEARN FROM GIS SYSTEMS.

IF YOU WANT TO RULE THE LOCATION AWARE APPLICATION SPACE, USE A GIS FOR INSPIRATION!!!
Most widely used GIS

The most widely used GIS is Google earth. It's free, it's easy to use, and the KML specification has made it wildly popular.

It's not just about getting a map with a pinpoint of the DCRUG meeting on the screen,

You can use it to perform analysis of building on a college campus

Or to map out the Wireless LAN infrastructure of an area.

Google Earth and Google Maps represent a valuable tool for Rubyists to use for visualization.
The most widely used GIS is Google earth. It's free, it's easy to use, and the KML specification has made it wildly popular.

It's not just about getting a map with a pinpoint of the DCRUG meeting on the screen,

You can use it to perform analysis of building on a college campus

Or to map out the Wireless LAN infrastructure of an area.

Google Earth and Google Maps represent a valuable tool for Rubyists to use for visualization.
The most widely used GIS is Google earth. It's free, it's easy to use, and the KML specification has made it wildly popular.

It's not just about getting a map with a pinpoint of the DCRUG meeting on the screen,

You can use it to perform analysis of building on a college campus

Or to map out the Wireless LAN infrastructure of an area.

Google Earth and Google Maps represent a valuable tool for Rubyists to use for visualization.
Layers

Just like in Photoshop, Spatial tools let you layer RASTER and VECTOR data on the screen to tell a story.
Layers
Layers

Illustration 4: A towns layer added to the map view.
Layers

Illustration 5: A schools layer added to the map view.
Illustration 6: A railways layer added to the map view.
Illustration 7: A rivers layer added to the map view.
Layers
Projection

- Method for representing the spheroid earth on a flat surface, such as a screen or paper map. They can be:
Projection

Planar

Polar
Equatorial
Oblique
Projection

Conic
Projection

Cylindrical

Normal

Transverse

Oblique
Projections are important for Layering

Mercator Projection
Satellite Imagery

+ 

Braun Projection
Boundary Lines
Projections are important for Layering

= FAIL
In this example I’ve drawn a straight line from Vancouver to Frankfurt.

In this projection, it represents the “shortest distance”, or a line across the 50th latitude.

But that is not reality. It’s a projected coordinate system. You cannot rely on what you see.
Projections are important for Calculation

The reality is, the shortest distance from Vancouver to Frankfurt “arches” up toward the North Pole. But it’s not really an arch. Look at the picture on the right to see why.

In a spherical coordinate system, the shortest distance between 2 points is still a straight line. But we can’t travel through the crust of the earth.

The shortest surface distance between two points is an arc of a Great Circle.

So: Calculation. We define a line from Vancouver to Frankfurt. Calculate the distance from New York to the line in question. In the planar coordinate system, the intersection is somewhere in Quebec. In the spheric coordinate system, the intersection is in VANCOUVER!!

This is important when you are representing data internally in your Ruby apps. ALWAYS KNOW WHAT PROJECTION AND COORDINATE SYSTEM YOU ARE USING!!!
Projection Hints

- Roughly equivalent to SRID (System Reference ID) in the various spatial databases

- When in doubt, try EPSG:4326 or EPSG:900913 (Google). These are the most widely used, but there are many more available.

- EPSG:4326 and EPSG:4236 are similar, but have different origin points. Dyslexia FTW.

If you really want to know more: http://spatialreference.org/
Projections

Pick the one that tells your story.
Geometry

- The basic spatial data type
- Point
- Line
- Polygon
- Curve
- Multiline
- Multipolygon
- Geometry Collection
Geography

Just like the Geometry type, but uses a curved coordinate set.
Spatial Stack

- Spatial DBMS
  - PostGIS
  - Oracle Spatial
  - MySQL Spatial
- spatial_adapter
- GeoRuby
- RGeo
- OpenLayers / YM4R / Google / Bing
Spatial DBMS

- Stores GEOMETRY in your tables
- Provides spatial functions like DISTANCE()
- Implements spatial indexing, so you can forget your High School math.
Spatial DBMS

- **PostGIS** -> The best, IMHO.
- **Oracle Spatial** -> Powerful, but moody
- **MySQL Spatial** -> OK, but use PostGIS if you can.
- **SpatialLite**: Pretty good if you need a lightweight database.
<table>
<thead>
<tr>
<th>id</th>
<th>abbrev</th>
<th>name</th>
<th>geom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MD</td>
<td>Maryland</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PA</td>
<td>Pennsylvania</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>VA</td>
<td>Virginia</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>WV</td>
<td>West Virginia</td>
<td></td>
</tr>
</tbody>
</table>

Tuesday, June 7, 2011
<table>
<thead>
<tr>
<th>:id</th>
<th>:first_name</th>
<th>:last_name</th>
<th>:location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Peter</td>
<td>Jackson</td>
<td>38.91094, -77.0327</td>
</tr>
<tr>
<td>2</td>
<td>Henry</td>
<td>Jackson</td>
<td>38.91094, -77.0327</td>
</tr>
<tr>
<td>3</td>
<td>Charlotte</td>
<td>Jackson</td>
<td>41.22, -79.01</td>
</tr>
</tbody>
</table>
GeoRuby

- Exposes the Geometry data types in Ruby.
- Converts ESRI Shapefile data
- Handles other interchange formats
- Does a limited set of geometric magic
GeoRuby

• Also, it is:
  • Slow
  • Incomplete
spatial_adapter

• Translates GEOMETRY columns in your DB into GeoRuby::Geometry types.

• Supports :geometry columns in migrations
If you are using PostGIS or MySQL, you can use the stock `spatial_adapter` maintained by “fragility”.

If you are using Oracle Spatial, you need my fork of `spatial_adapter`. My fork will not be pulled into the main stream.

`github.com/peteonrails/spatial_adapter`
spatial_adapter

• Also, it is:
  • Hard to extend and add adapters
  • A monkeypatched mess (no offense!)
  • Long in the tooth
RGeo

• Use it instead of GeoRuby and SpatialAdapter
• It supports geometric calculations in memory.
• It has a nice adapter framework for ActiveRecord
• Sophisticated Arel support is in the works
RGeo

RGeo has made life much better.

Thank you, Daniel Azuma and GeoPage, Inc.

Kudos!
OpenLayers

One of many javascript visualization libraries

- Bing Maps
- Google Maps
- Yahoo Maps
- NASA Worldwind
- KML Overlays
- Your own custom imagery
OpenLayers
You can also use

- YM4R
- Google Maps
- Bing Maps
- OpenStreetMap
- Any number of new mapping visualization toolkits: GeoCommons, GeoPage, etc.
Spatial Stack

Visualization
Map viewer (JS Library) + Awesome Imagery + Your Data = WOW

Spatial Logic Components
your awesome biz logic here
GeoRuby
spatial_adapter

Spatial Data Sets
KML
ESRI Shapefiles
PostGIS
Oracle Spatial
MySQL Spatial

Tuesday, June 7, 2011
Spatial Stack

Visualization

Map viewer (JS Library) + Awesome Imagery + Your Data = WOW

Spatial Logic Components

your awesome biz logic here

R GEO is Awesomer!

Spatial Data Sets

KML
ESRI Shapefiles
PostGIS
Oracle Spatial
MySQL Spatial
Before Getting Started

• Install Proj and Geos (for speed and features)
• Get familiar with Geography and Geometry
• Review the READMEs!
• Handles changes between coordinate systems

`brew install proj`
• C/C++ library that implements complex geometry and geography calculations

`brew install geos`
Getting Started
Getting Started

tar xvfz postgis-1.5.1.tar.gz
cd postgis-1.5.1
./configure
make
make install
createdb yourdatabase
creatlang plpgsql yourdatabase
psql -d yourdatabase -f postgis.sql
psql -d yourdatabase -f postgis_comments.sql
psql -d yourdatabase -f spatial_ref_sys.sql
Getting Started

$ gem install
  rgeo  rgeo-shapefile
  activerecord-postgis-adapter
  rgeo-geojson
Getting Started

Now fire up IRB any try this:
# Verify that your install works and can do point geometry

```
factory = ::RGeo::Cartesian.preferred_factory
=> #<RGeo::Cartesian::Factory:0x101cd0eb8 @has_z=false, @proj4=nil, @srid=0, @coord_sys=nil, @has_m=false>

?> p00 = factory.point(0, 0)
=> Point(0.0 0.0)

>> p01 = factory.point(0, 1)
=> Point(0.0 1.0)

>> p11 = factory.point(1, 1)
=> Point(1.0 1.0)

>> p10 = factory.point(1, 0)
=> Point(1.0 0.0)
```
> zigzag_line = factory.line_string([p00, p10, p01, p11])
> #=> #<RGeo::Cartesian::LineStringImpl:0x80e5ba5c "LineString(0.0 0.0,1.0 0.0,0.0 1.0,1.0 1.0)">

>> zigzag_line.is_simple?
=> true

> self_crossing_line = factory.line_string([p00, p11, p01, p10])
> #=> #<RGeo::Cartesian::LineStringImpl:0x80e55b20 "LineString(0.0 0.0,1.0 1.0,0.0 1.0,1.0 0.0)">

>> self_crossing_line.is_simple?  # returns false
=> false

>> p00.distance(p11)
=> 1.4142135623731
# Other useful stuff you can do

### Calculations

```ruby
p1.distance(p2)  # Distance
polygon.envelope  # Find a bounding box
surface1.area    # Area
```

### Aggregation

```ruby
multi = p1 + p2
```

### Useful Checks

```ruby
line1.crosses?(line2)   # intersection
line1.touches?(line2)   # adjacency
multi.contains?(p1)     # containment
```

### Create a buffer around something

```ruby
line_string.buffer(256)
```
More than calculating distance:

- Find Bounding Boxes and Envelopes
- Convert to GeoJSON
- Serialization of Geometry / Geography
- Transformation between coordinate systems
What ELSE can we do?

A lot. Let’s look:
Spatial Queries

More than just “Dots on a map”

We can answer **really difficult, meaningful** questions.
Spatial Queries

Q: How many freshwater wells are located within 5 miles of a chemical plant?

A:

SELECT c.chemical_plant_name
FROM well_table a, chemical_plants b
WHERE sdo_within_distance (b.geom, a.geom, 'distance=5 unit=mile') = 'TRUE'
Using PostGIS and RGeo, a Rails app could do this:
class CreateWells< ActiveRecord::Migration

  def self.up
    create_table :wells do |t|
      t.references :customer
      t.geometry :location       # <= spatial_adapter / rgeo magic.
    end
  end

  def self.down
    drop_table :wells
  end

end

# More spatial adapter magic
add_index :wells, :location, :spatial => true
class Well < ActiveRecord::Base
  
def self.find_by_proximity(km)
    # This example uses Oracle syntax
    find(:all, :joins => :chemical_plant,
         :conditions => ['sdo_within_distance (chemical_plants.geom, wells.geom, 'distance=? unit=km') = 'TRUE', km])
  end
end

class WellController < ApplicationController
  
def index
    @wells = Well.find_by_proximity(params[:buffer])
  end
end
<script src="./OpenLayers.js"></script>

<script type="text/javascript">
    var map, layer;

    function init(){
        map = new OpenLayers.Map('map', {maxResolution:'auto'});
        map.addControl(new OpenLayers.Control.LayerSwitcher());
        layer = new OpenLayers.Layer.WMS( "OpenLayers WMS", "http://labs.metacarta.com/wms/vmap0", {layers: 'basic'} );
        map.addLayer(layer);
        map.setCenter(new OpenLayers.LonLat(0, 0), 0);
        var newl = new OpenLayers.Layer.Markers( 'POIs');
        map.addLayer(newl);
    }
</script>
Inspiration
Inspiration

Mainly Nice People, but a few dickheads

Scale 1:134,300,000
Robinson Projection
standard parallels 38°N and 38°S
Inspiration

✧ Find some interesting data
✧ Use it to tell a compelling story
✧ Visualize it creatively
Finding Data

naturalearthdata.com

Raster Imagery
Vector Layers
Demographics
Boundaries
Waterways
Coastlines
Finding Data

data.gov

Demographics
Census
Agency Research
Environmental Data
NASA
NOAA
Finding Data

GeoCommons.com

Community-driven datasets and maps

Ruby API

INSPIRATION!!

https://github.com/fortiusone/geoiq-ruby
Inspiration
Inspiration
Inspiration

What if all you have is “Dots on a Map”?

1. Go Temporal

2. Use Scale

3. Integrate Real-Time data
Inspiration
Inspiration

How do we finish the IRAQ Casualty Map story?

Overlay an Infrastructure Layer
Add news stories
Aggregate the data into Hot Spots
Show Enemy Strongholds
Troop Positions And Operations
Correlate with Political Events
Integrating Real Time Data

Make your map more valuable by being timely.
Inspiration

• Start Thinking Like A Geographer
  • Aggregate

• Where do things overlap?

• Tell a compelling story about People
The Oil Spill
Getting Started Summary

• Install Spatial DBMS
• Install RGeo and its dependencies
• Use OpenLayers or some other JS library
• Get Inspired and Start Thinking spatially!
What’s Next for the Stack?

- View Layer integration sucks: fix it!
- Geocoding, location databases
- Better support for Tiles and Custom Base Layers
- GeoDjango is great, GeoRails can be better!
Want to know more?

GeoSpatial Rails Summit

Wednesday, May 18th, 2011

8PM - Ballroom IV

Hint: It’s in this room, y’all.
Agenda

• What is Spatial Programming?
• Important Spatial Terms
• Tools in the spatial stack
• How you get started
• What’s next?
I hope you learned alot.

Please rate this talk:  http://spkr8.com/t/7100
License
Creative Commons Attribution 3.0 Unported

You are free:
1. to Share — to copy, distribute and transmit the work
2. to Remix — to adapt the work

Attribution — You must attribute this work to Peter Jackson (with link). If you want to remix or redistribute the work, including for your own personal or commercial use with no redistribution, please be sure that the prior sentence remains intact.

THIS IS NOT A SHARE-ALIKE LICENSE. YOU MAY MAKE CHANGES TO THIS WORK AND KEEP THEM FOR YOUR OWN USE, COMMERCIAL OR PERSONAL.

http://creativecommons.org/licenses/by/3.0/