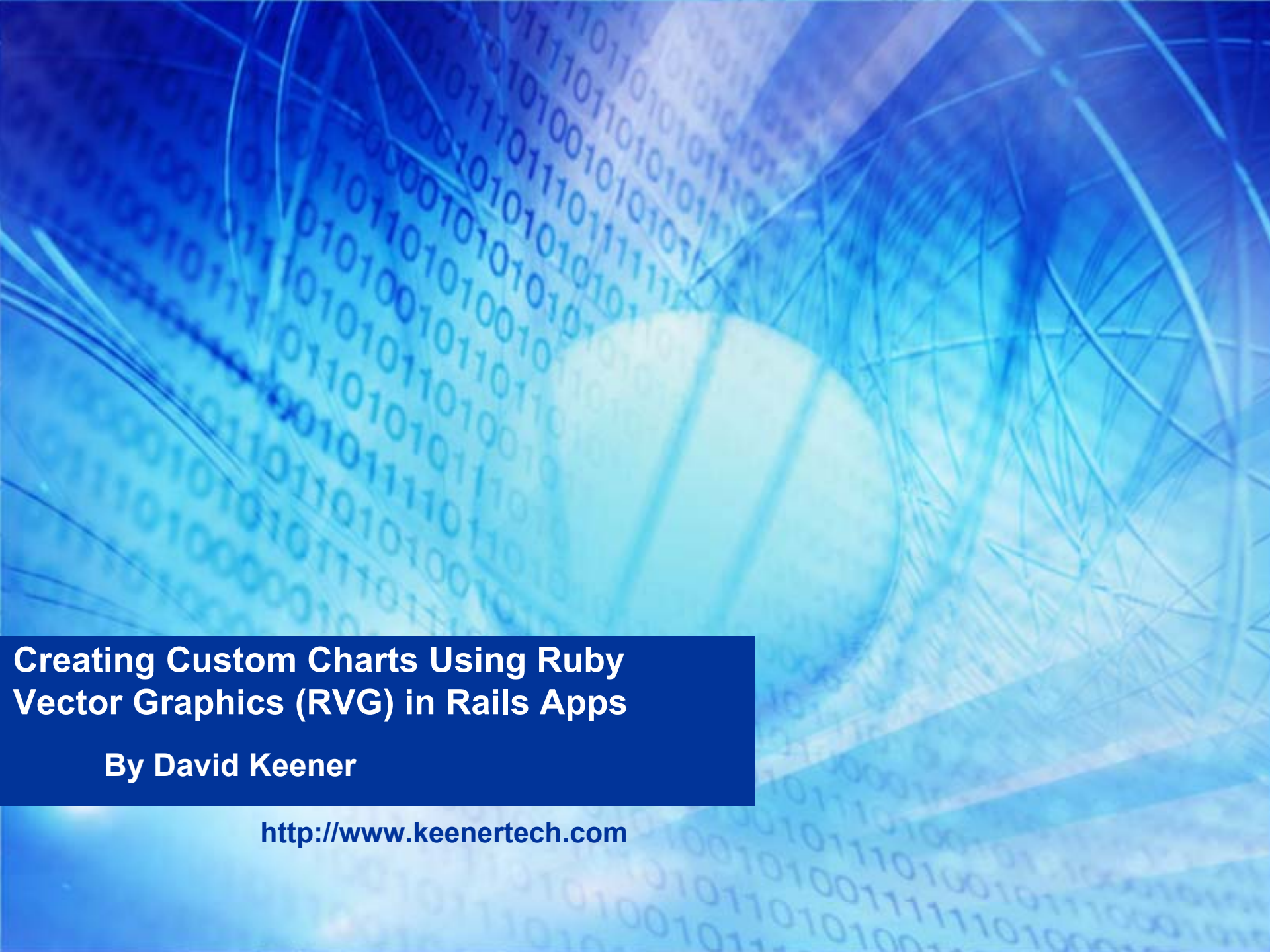


Creating Custom Charts Using Ruby Vector Graphics (RVG) in Rails Apps

By David Keener

http://www.keenertech.com

Overview

The Ruby Vector Graphics (RVG) API expands the capabilities of Ruby on Rails to include the dynamic creation of custom charts, drawings and graphics. By the end of this presentation, you will be able to answer the following questions about RVG and the astronomy-related demo application that uses it:

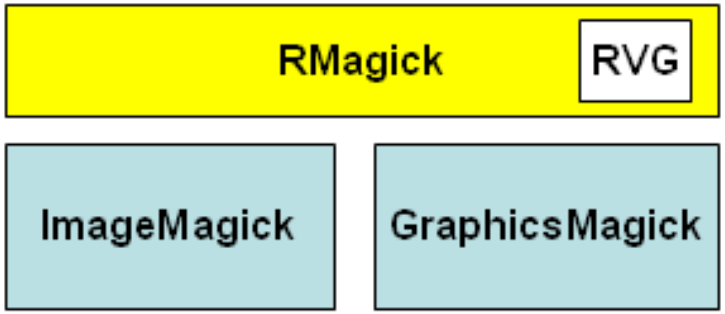
- What is RVG?
- What is SVG (Scalable Vector Graphics)?
- How is RVG related to RMagick, ImageMagick and GraphicsMagick?
- What can you do with RVG?
- What are galactic coordinates?
- What is the closest star to our sun?
- What the heck is an orthographic projection?

Why Do This in Rails?

- To show that Rails is not just for CRUD
- To illustrate other capabilities for Rails applications that are less widely known
- To demonstrate that Rails, while still evolving, is mature enough to do graphics tasks that can be done in other languages…such as Java, etc.
- To explore the road less traveled
- For the sheer challenge of it

Where Is RVG?

- RVG is included in RMagick
- RMagick is a Ruby binding for the ImageMagick and GraphicsMagick image manipulation libraries
- GraphicsMagick is a fork of ImageMagick created when the dev team split up
- RMagick & RVG are compatible with both libs



ImageMagick

ImageMagick 6.3.3 is an open source software suite for image manipulation and display

- Includes an image manipulation app (Windows version is buggy, though)
- Command-line utilities for image processing
- API libraries for image manipulation
- Supports numerous image formats
- Needs open source delegate libraries to support some image formats
- For more info: http://www.imagemagick.org/script/index.php

GraphicsMagick

GraphicsMagick 1.1.7 is an open source software suite billed as the “Swiss army knife of image manipulation”

- Fork of ImageMagick 5.5.2, due to differences within the development team
- Claims to be a better, more stable version of the ImageMagick codebase
- No new releases since version 1.1.7 in October 2005
- Supports same basic features and API as ImageMagick
- For more info: http://www.graphicsmagick.org

RMagick

RMagick 1.15.0 is a Ruby binding for the ImageMagick and GraphicsMagick APIs.

- Version 1.0.0 released October 2003
- Mature and widely used package
- Contains 3 major classes, 30 minor classes – over 600 methods for image manipulation
- Contains Ruby Vector Graphics (RVG) library
- RMagick requires Ruby 1.6 or 1.8.x
- RVG requires Ruby 1.8.x
- Both require ImageMagick 6.0.0 or higher; or GraphicsMagick 1.0.0 or higher

RMagick Basics

- Three major classes are:
  - **Image** : image manipulation methods
  - **ImageList** : methods acting on arrays of images
  - **Draw** : primitives for drawing / writing
- RMagick drawing features currently used more commonly than RVG in Ruby and Rails apps

Ruby Vector Graphics (RVG)

- New library bundled with Rmagick
- RVG implements a substantial portion of the Scalable Vector Graphics (SVG) standard from the W3C
- RVG is an SVG-oriented wrapper around the image manipulation features provided by the RMagick Draw class
- RVG positions RMagick to be useful to a growing audience of users familiar with the SVG standard

What is SVG?

- It’s an XML format for representing 2D scalable graphics
- It’s similar to Flash, but non-proprietary
- It’s a W3C standard
- It defines a standard way of representing and manipulating images
- Metadata can be stored with graphics, enhancing their search potential
- Graphics can be manipulated with XSL and CSS
- Includes support for animation

Key Components of an SVG Graphic

- A canvas to be drawn upon
- A viewport, which is a rectangular window through which a portion of the canvas can be drawn on
- Vector Shapes: rect, circle, ellipse, line, polyline, polygon
- Images – Bitmap images
- Text – Text to be displayed on a canvas
- Symbols – Reusable elements, such as flowchart symbols
- Containers

RVG and SVG

- RVG implements key SVG objects such as a canvas, viewports, vector shapes, containers, etc.
- RVG allows images to be created server-side and used client-side
- RVG provides an SVG-oriented façade to the robust and mature Rmagick image manipulation features

Key RMagick / RVG Links

- RMagick Documentation:
  http://www.simplesystems.org/RMagick/doc/index.html
- RMagick Downloads:
  http://rmagick.rubyforge.org
- Installation Instructions:
  http://rmagick.rubyforge.org/install-faq.html
- Excellent RVG Tutorial:
  http://redux.imagemagick.org/RMagick/doc/rvgtut.html

RMagick Pre-Installation Notes

- RMagick is not a trivial install
- RMagick installs for UNIX, Linux and MacOS are more complicated than Windows, but more production-ready
- Windows installation is fairly simple, using a pre-made gem…but the version in the gem may not be compatible with the absolute latest versions of Ruby and Rails

RMagick Installation - Windows

- Download **rmagick-win32** gem from: http://rubyforge.org/projects/rmagick
- Listed As: RMagick 1.14.1 gem for Ruby 1.8.5 (dated November 26, 2006)
- Extract zip file into a temp directory
- Run ImageMagick Installer for 6.3.0-7, which was bundled with the gem
- Run command from temp dir:
  gem install rmagick --local
- Note: Compatible with InstantRails 1.4, not later versions (Ruby 1.8.5, Rails 1.1.6)

RMagick Installation – Unix, Linux

- Step 1: Install delegate libraries
- Step 2: Install either ImageMagick or GraphicsMagick
- Step 3: Install RMagick
  - Can use pre-compiled binaries (easy)
  - Or full compile from source (harder)

RMagick Installation – Mac OS X

- Step 1: Install X11
- Step 2: Install Xcode tools and X11 SDK
- Step 3: Install MacPorts (formerly known as DarwinPorts) – automates open source software installations
- Step 4: Install delegate libraries
- Step 5: Install either ImageMagick or GraphicsMagick
- Step 6: Install RMagick

Onwards to the Demo Application!

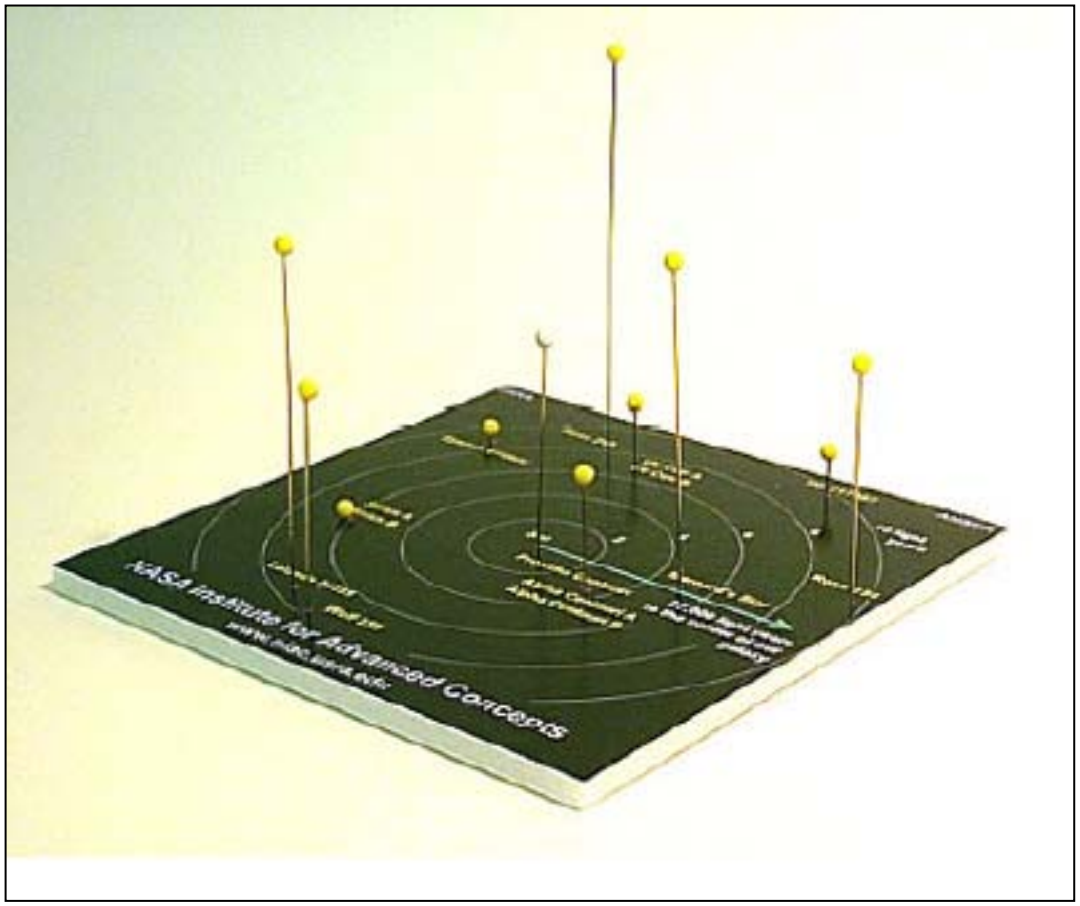
The **Star Mapper** web application is targeted as an educational service to provide users with a 3D view of our local stellar geography.

- Inspired by NASA
- Provided a good platform for learning RVG
- Serves as a good illustration of how RVG can be used to create custom charts

NASA Starmap Outreach Project

- Free kit from NASA Institute for Advanced Concepts
- Designed to illustrate astronomy concepts
- Lets kids (and adults) create a 3-D starmap of the stars within 10 light years of Earth
- Inspiration for the Starmap Rails app

NASA Starmap Outreach Project



Basic Stellar Data

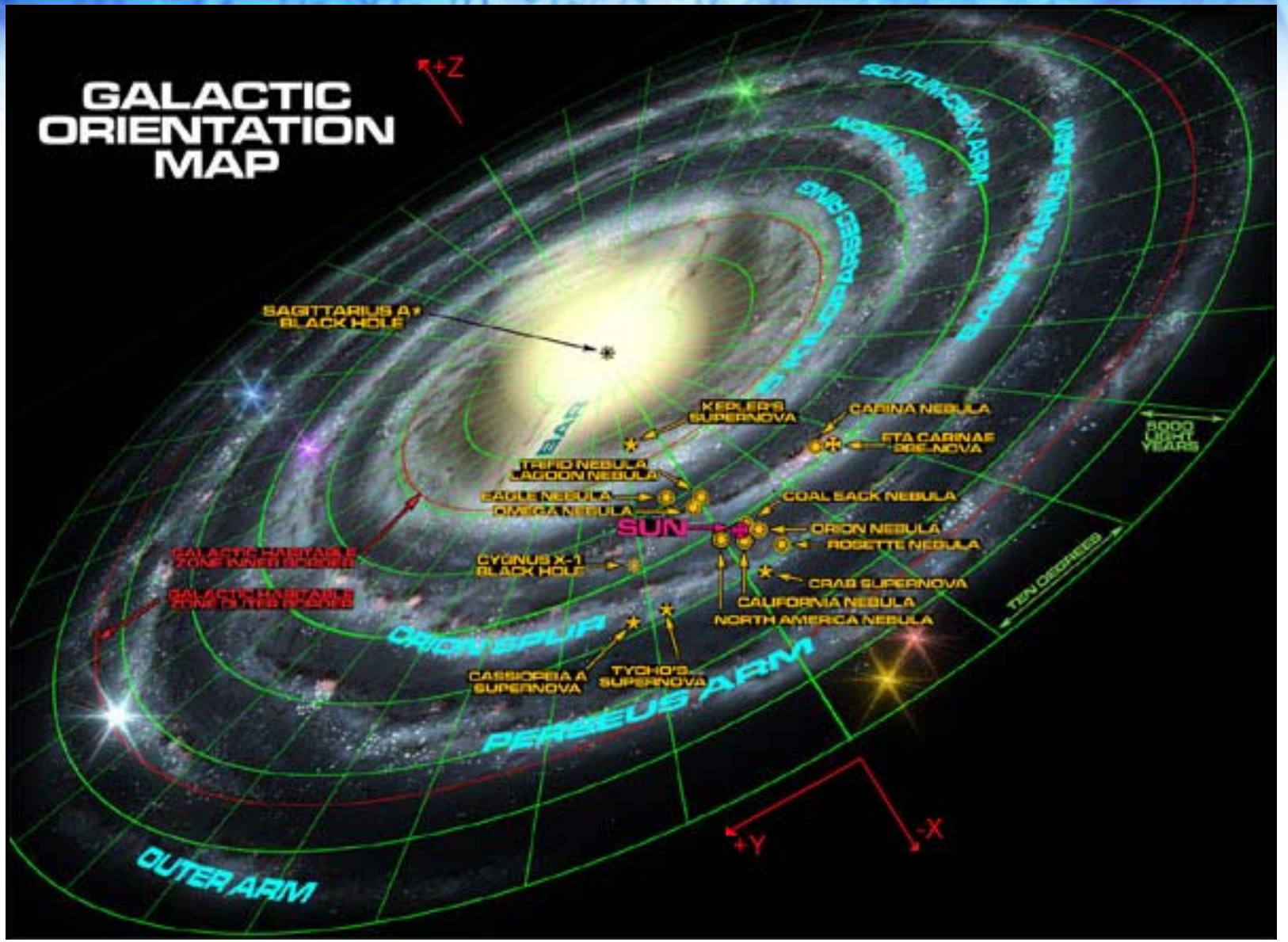
If you're going to create a starmap for the local stars, you need X-Y-Z coordinate data. Luckily, there are a number of free sources for stellar data:

- Gliese 3 – (1991) Data on stars within 25 parsecs (81.5 light years) – Easy to work with
- Hipparcos – (2005) French compilation of info on over 100,000 near stars
- Great site for information on starmaps, compliments of Winchell Chung:
  http://www.projectrho.com/starmap.html

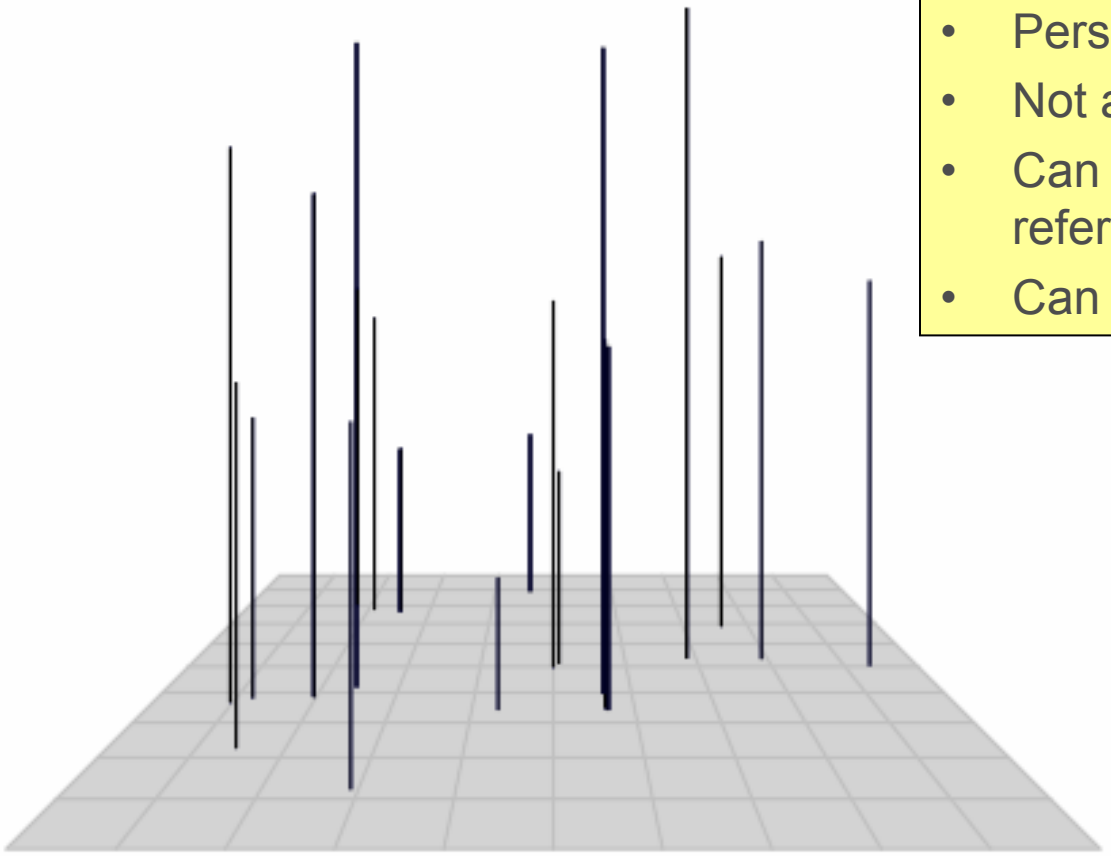
X-Y-Z Coordinate Systems

Great. You're gonna use an X-Y-Z coordinate system for your stars. Uh…which way is up?

- Epoch 1950 – Coordinates defined by Earth's position in 1950 (Gliese 3)
- Epoch 2000 – Coordinates defined by Earth's position in 2000
- Galactic Coordinates – X-Y plane is the plane of the galaxy; positive X points to galactic center; positive Y to spinward; our sun is above the plane by about 50 light years


GALACTIC ORIENTATION MAP
+Z
SAGITTARIUS A*
BLACK HOLE
SCUTUM-CROSS ARM
NORMA ARM
PERSEUS ARM
SAGITTARIUS ARM
BAR
KEPLER'S SUPERNOVA
CARINA NEBULA
ETA CARINAE PRE-NOVA
TRIFID NEBULA
LAGOON NEBULA
EAGLE NEBULA
OMEGA NEBULA
COAL SACK NEBULA
SUN
ORION NEBULA
ROSETTE NEBULA
CYGNUS X-1
BLACK HOLE
CRAB SUPERNOVA
CALIFORNIA NEBULA
NORTH AMERICA NEBULA
CASSIOPEIA A
SUPERNOVA
TYCHO'S
SUPERNOVA
ORION SPUR
PERSEUS ARM
OUTER ARM
+Y
-X
5000 LIGHT YEARS
TEN DEGREES

Nearby Stars - Starmap



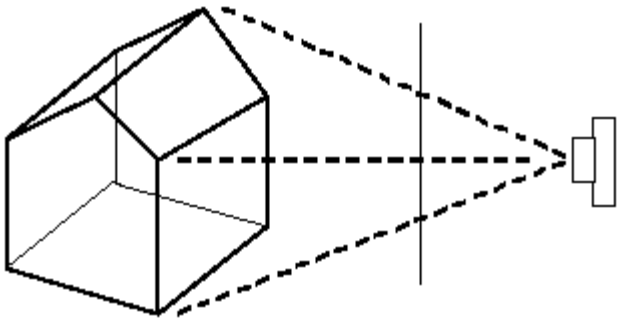
- Perspective projection
- Not as fancy as I would like (yet)
- Can show stars from different reference points
- Can show different sets of stars

Basic Drawing With RVG

```
require 'rvg/rvg'
include Magick

class Demo

  def self.create_demo       # Demonstrates how to draw the floor in perspective
    RVG::dpi = 72

    ptc = { :x => 50, :y => 50, :z => -100 }  # Reference Point
    pt1 = { :x => 0, :y => 0, :z => 0 }       # Corner 1
    pt2 = { :x => 100, :y => 0, :z => 0 }     # Corner 2
    pt3 = { :x => 100, :y => 0, :z => 100 }   # Corner 3
    pt4 = { :x => 0, :y => 0, :z => 100 }     # Corner 4

    rvg = RVG.new(7.in, 7.in).viewbox(0,0,500,500) do |canvas|
      canvas.background_fill = 'white'

      canvas.g.translate(450, 450).rotate(0) do |floor|
        floor.styles(:fill=>'lightgray', :stroke=>'silver', :stroke_width=>1)

        xs1 = self.transform(pt1, ptc)  # Perform perspective transformations here
        xs2 = self.transform(pt2, ptc)
        xs3 = self.transform(pt3, ptc)
        xs4 = self.transform(pt4, ptc)

        floor.polygon(xs1[:x], xs1[:y], xs2[:x], xs2[:y],
          xs3[:x], xs3[:y], xs4[:x], xs4[:y])
      end # Floor

      # Do more operations on the canvas…
    end  # Canvas
  end
  rvg
end
```

Perspective Projection

A perspective projection traces lines between a viewing point and elements in a 3D model in order to perform a 2D projection onto a screen.



- Trace lines from the reference point, also known as the center of the projection, and the 3D model.
- Where the lines pass through a designated plane defines the 2D image that can be displayed on a screen.

Transformation Algorithm

```
def self.transform(pt, ptc)
  npt = { :x => 0.0, :y => 0.0, :z => 0.0 }
  vdist = 4.0

  d = ptc[:z] - pt[:z]
  if d < 0.0
    d = 0.0 - d
  end

  if d < 1
    npt[:x] = (pt[:x].to_f - ptc[:x].to_f) * 1000000.0
    npt[:y] = (pt[:y].to_f - ptc[:y].to_f) * 1000000.0
    npt[:z] = 1000000.0
  else
    npt[:x] = (vdist * ((pt[:x].to_f * ptc[:z].to_f) - (ptc[:x].to_f * pt[:z].to_f))) / d
    npt[:y] = (vdist * ((pt[:y].to_f * ptc[:z].to_f) - (ptc[:y].to_f * pt[:z].to_f))) / d
    npt[:z] = pt[:z].to_f / d
  end

  npt   # Return a hash containing X-Y-Z coordinates for the transformed point
end
```

The transform method from the StarMap Model:

pt - A hash containing X-Y-Z coordinates for the point to be transformed

ptc - X-Y-Z coordinates for the reference point, sometimes referred to as the center of the projection

Summary

- Both RMagick and RVG can be used to add dynamic custom charting capabilities to Rails applications
- SVG, and implementations of it such as RVG, have the potential to radically change our experiences with Web 2.0