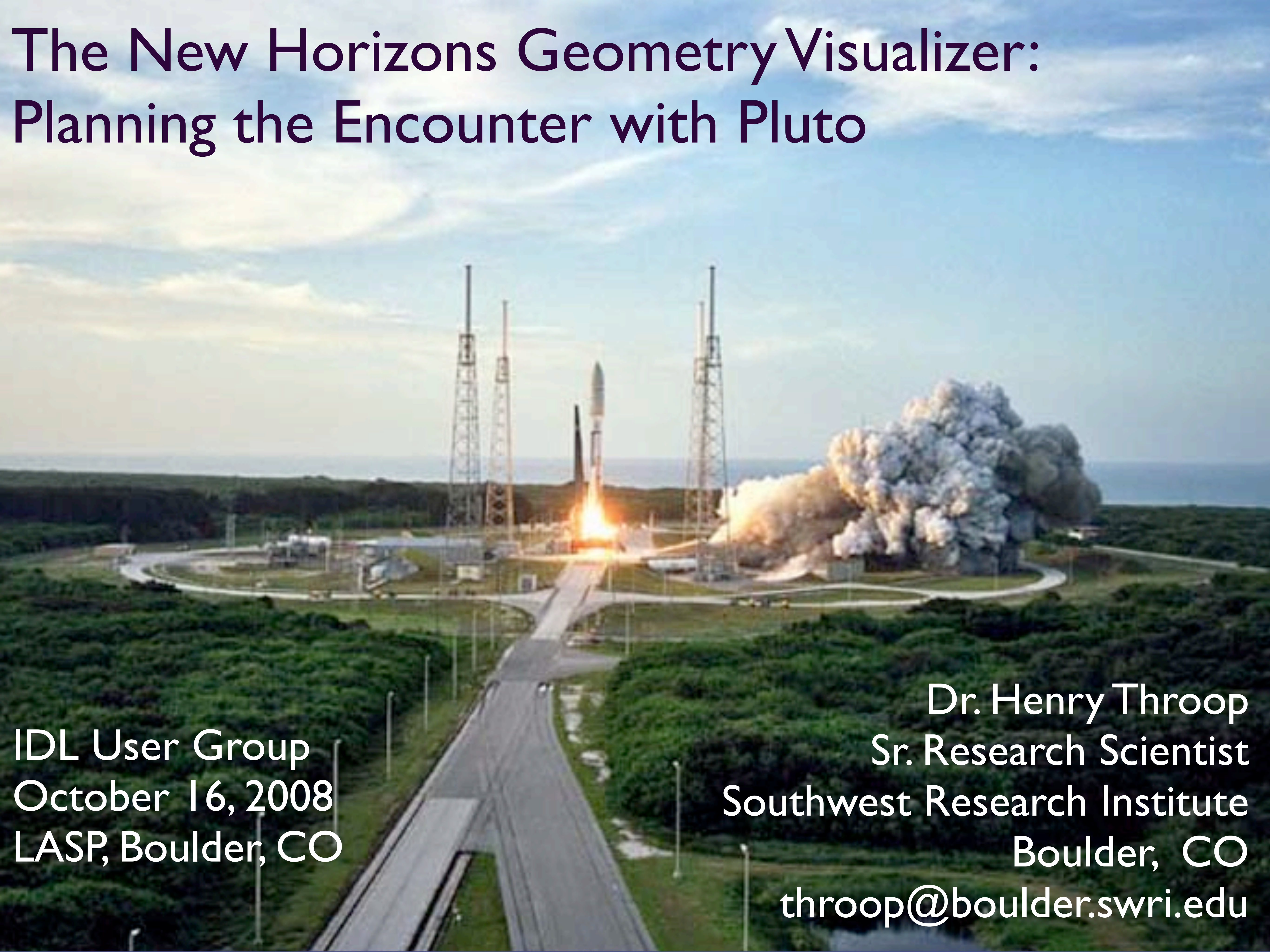
The New Horizons Geometry Visualizer: Planning the Encounter with Pluto



IDL User Group
October 16, 2008
LASP, Boulder, CO

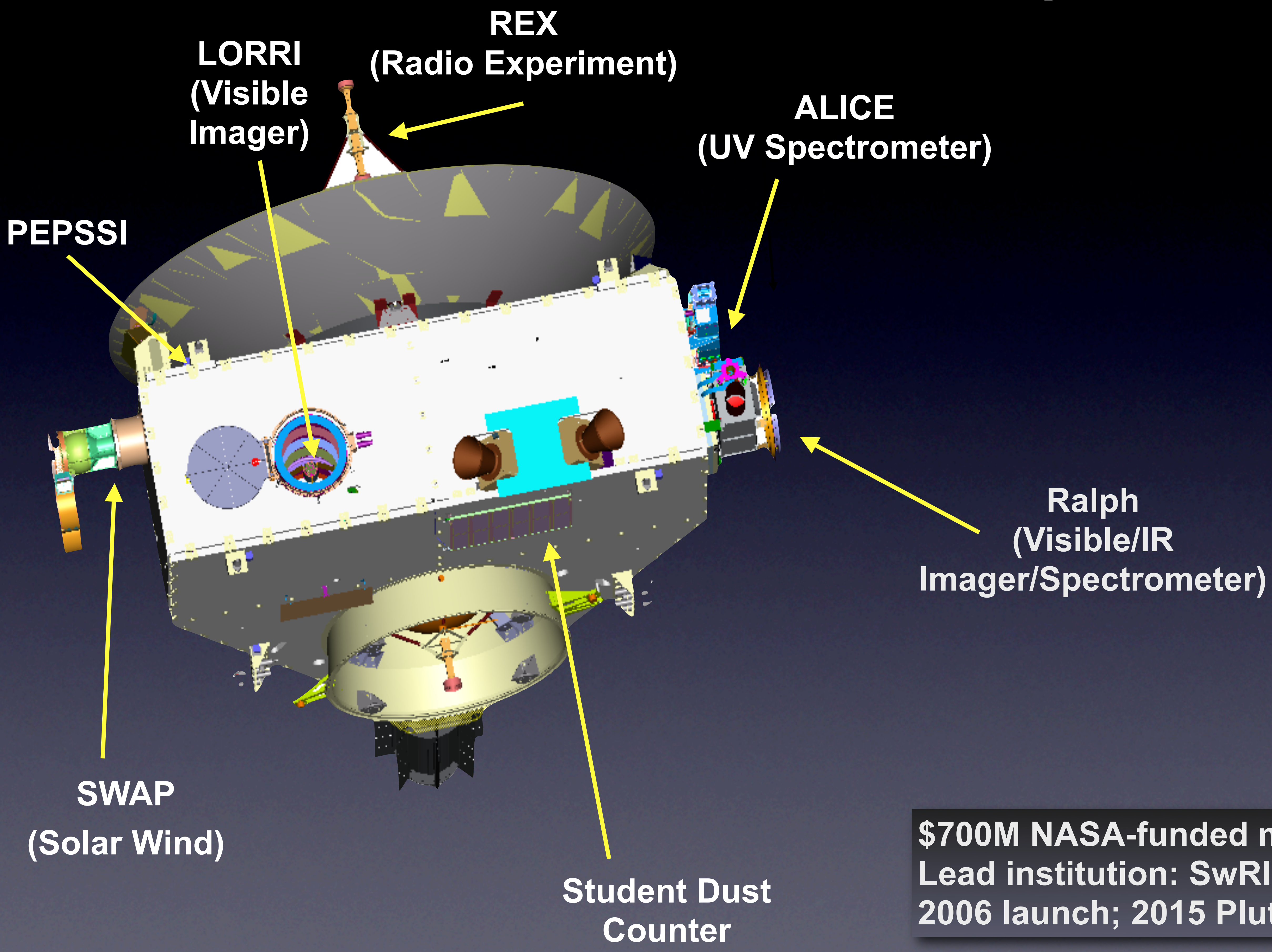
Dr. Henry Throop
Sr. Research Scientist
Southwest Research Institute
Boulder, CO
throop@boulder.swri.edu

NHGV: New Horizons Geometry Visualizer

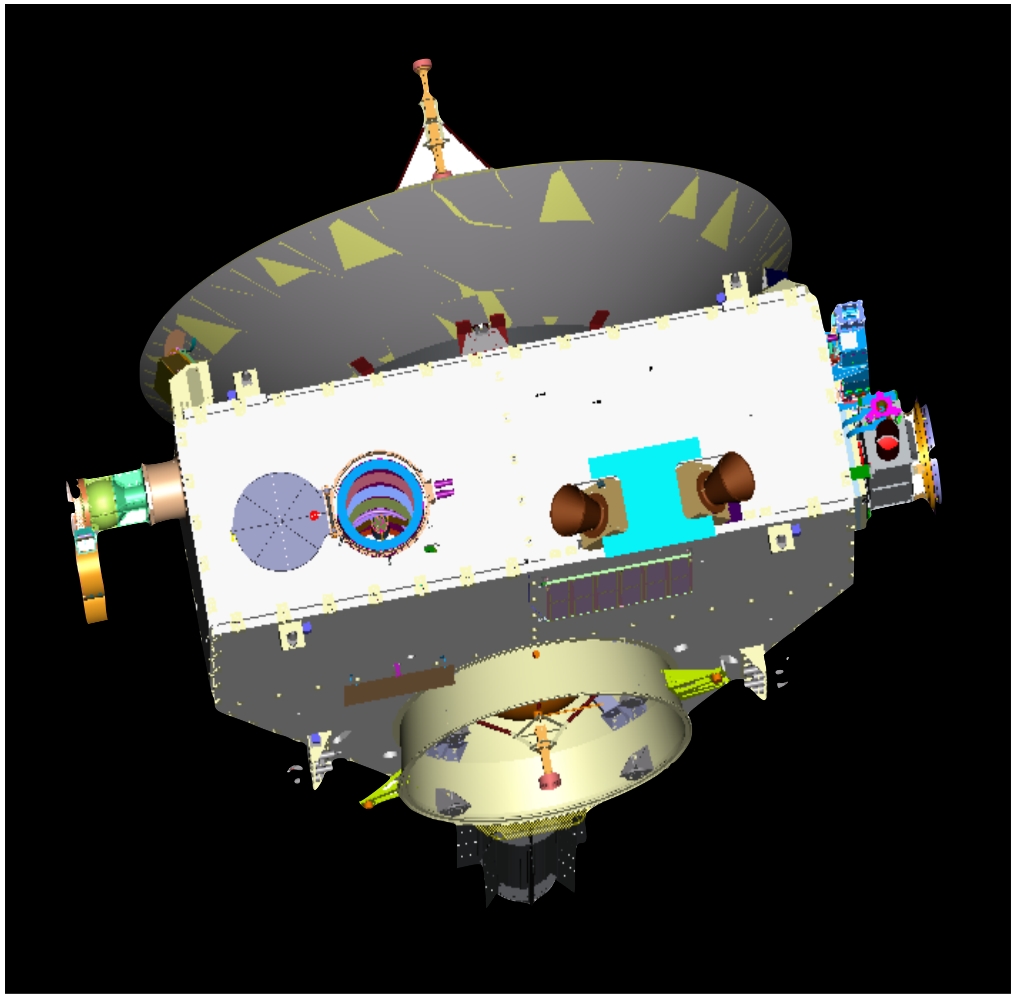
- The New Horizons Mission to Pluto
- NHGV: Brief demo and overview
- Developing NHGV in IDL

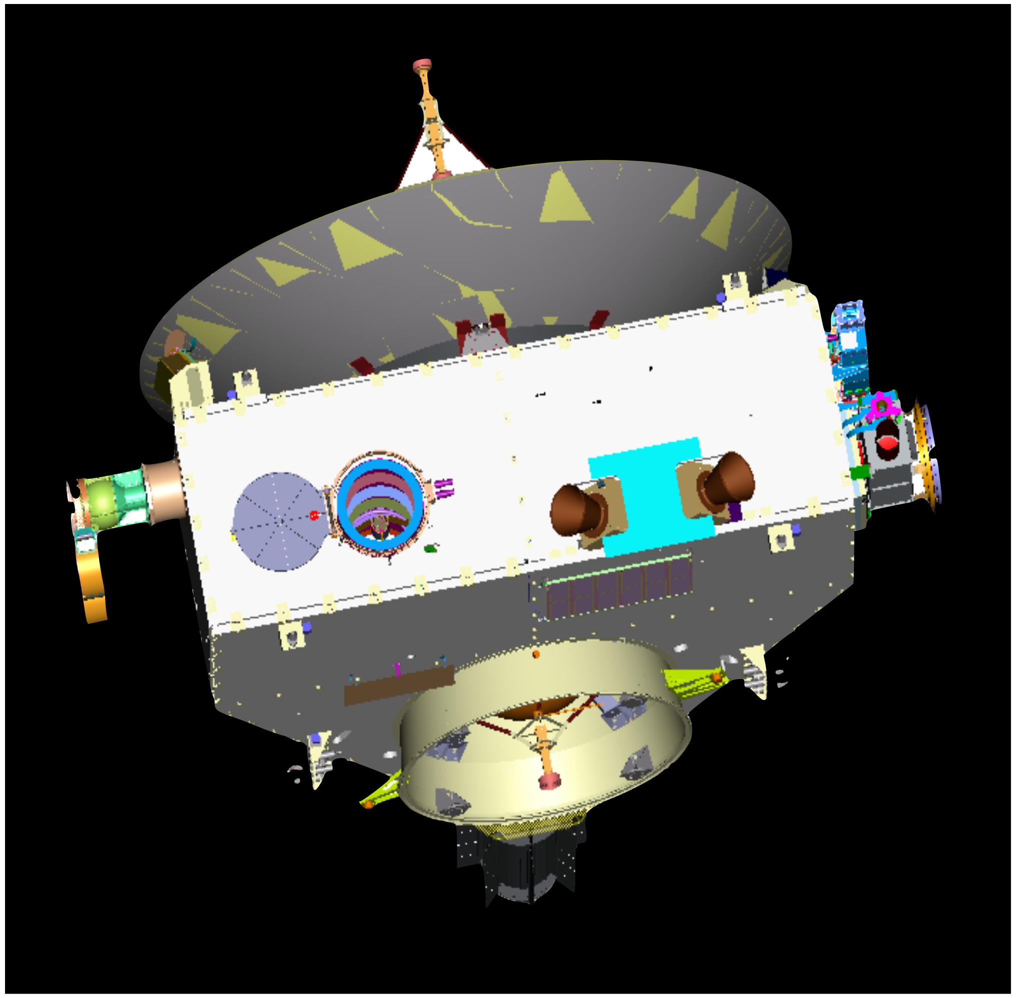
http://soc.boulder.swri.edu/gv

New Horizons Spacecraft

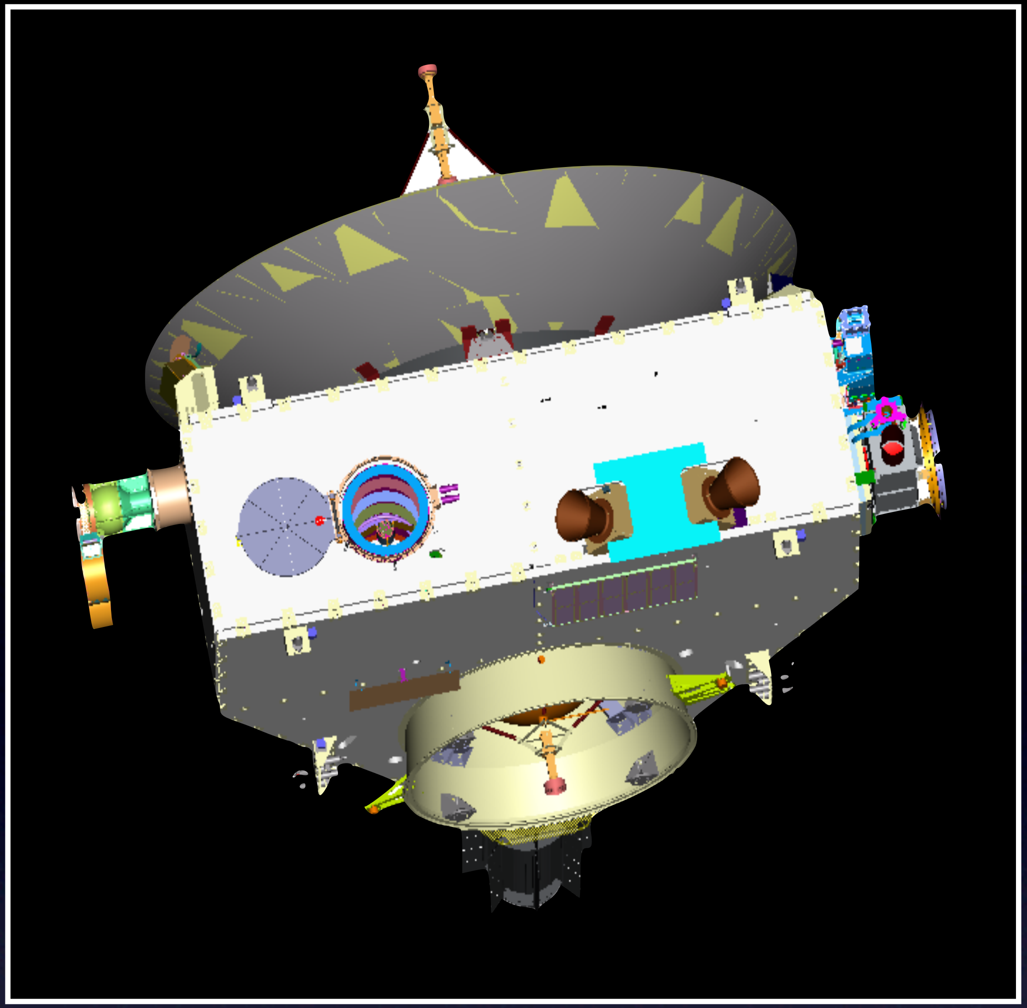


$700M NASA-funded mission
Lead institution: SwRI
2006 launch; 2015 Pluto





cingular
12:34 PM
SMS
Text
Tuesday
9
Calendar
Photos
Camera
Calculator
Stocks
280
Maps
73°
Weather
Notes
Clock
Settings
Phone
Mail
Web
iPod



	New Horizons	iPhone
Camera	1 MP	2 MP
Storage	16 GB	16 GB
Network	NASA Deep Space Network	3G
Battery	88 years, Plutonium-238	4 hour

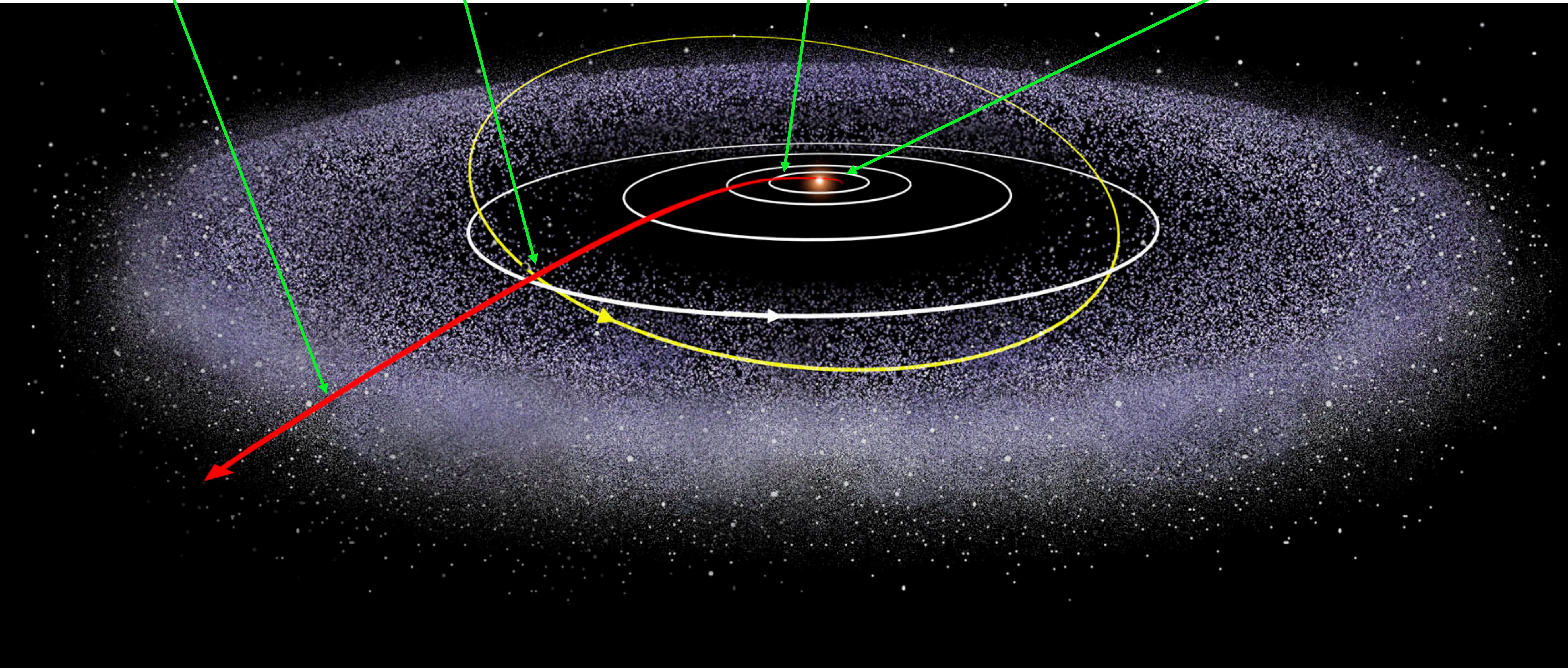
LOCKHEED MARTIN
ILS
AV-010

Pluto-Charon
July 2015

Jupiter System
March 2007

KBOs
2016-2020

Launch
January 2006



What does GV do?

- GV is a virtual planetarium of the Solar System, as seen from a moving spacecraft.
- Answers questions like:
  - When can we fit Pluto and its satellites into a single frame?
  - What stars are visible when the Sun passes behind Charon?
  - What is the sub-Solar longitude on Pluto at closest approach?
  - On what day should we approach Pluto so as to see the whole planet?
- Designed as a ‘sandbox’ for planning observations.
- Not used to program commands into spacecraft.

GV Design Goals

Priority 1:

- Accuracy

Priority 2:

- Features
- Ease of use
- Speed
- Ease of development
- Ease of distribution

Existing Programs

STK	Proprietary;
	Expensive ($70K/seat);
	Overkill for most users

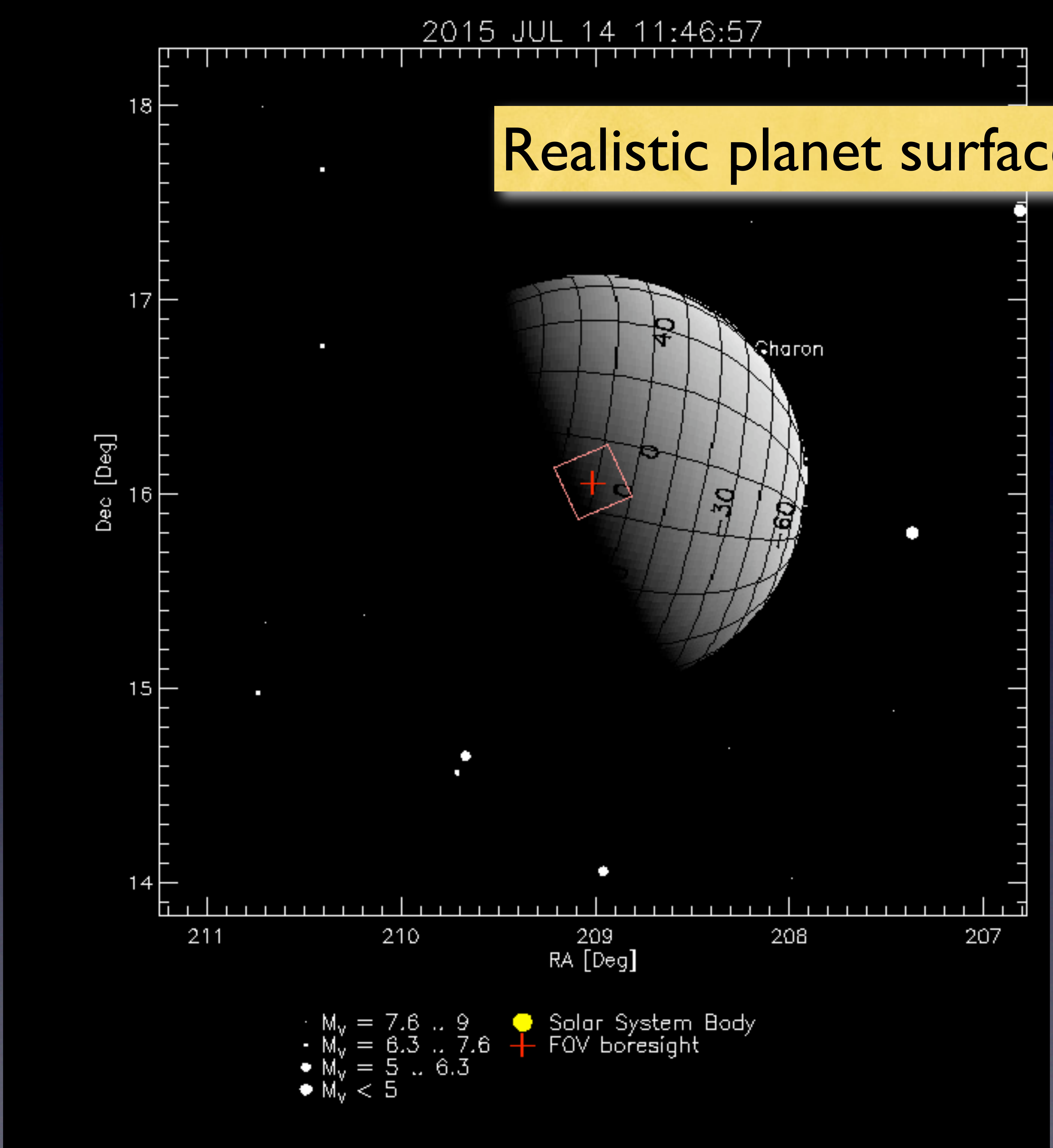
JPL tools	Minimal support; Some Solaris only
Other tools	Not SPICE-based
	Not customizable

GV is Web-Based

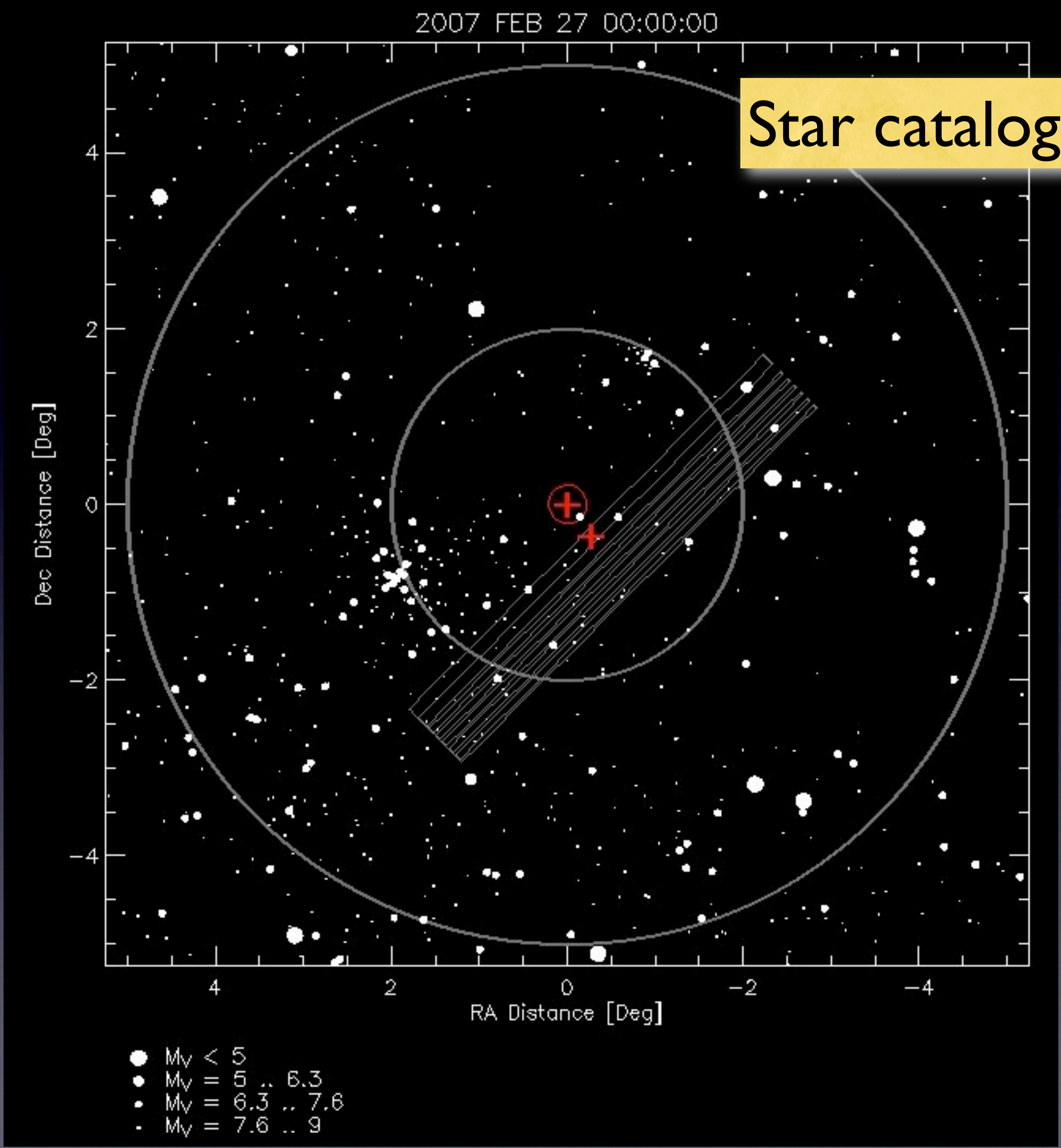
- Works on all hardware platforms
- Works from anywhere in the world
- All users are guaranteed to be using up-to-date version.
- All support files (kernel files, documentation, etc) are kept in sync.
  - There are hundreds of frequently updated kernel files, and user-maintained local repositories are prone to be out-of-sync!

Tour of GV

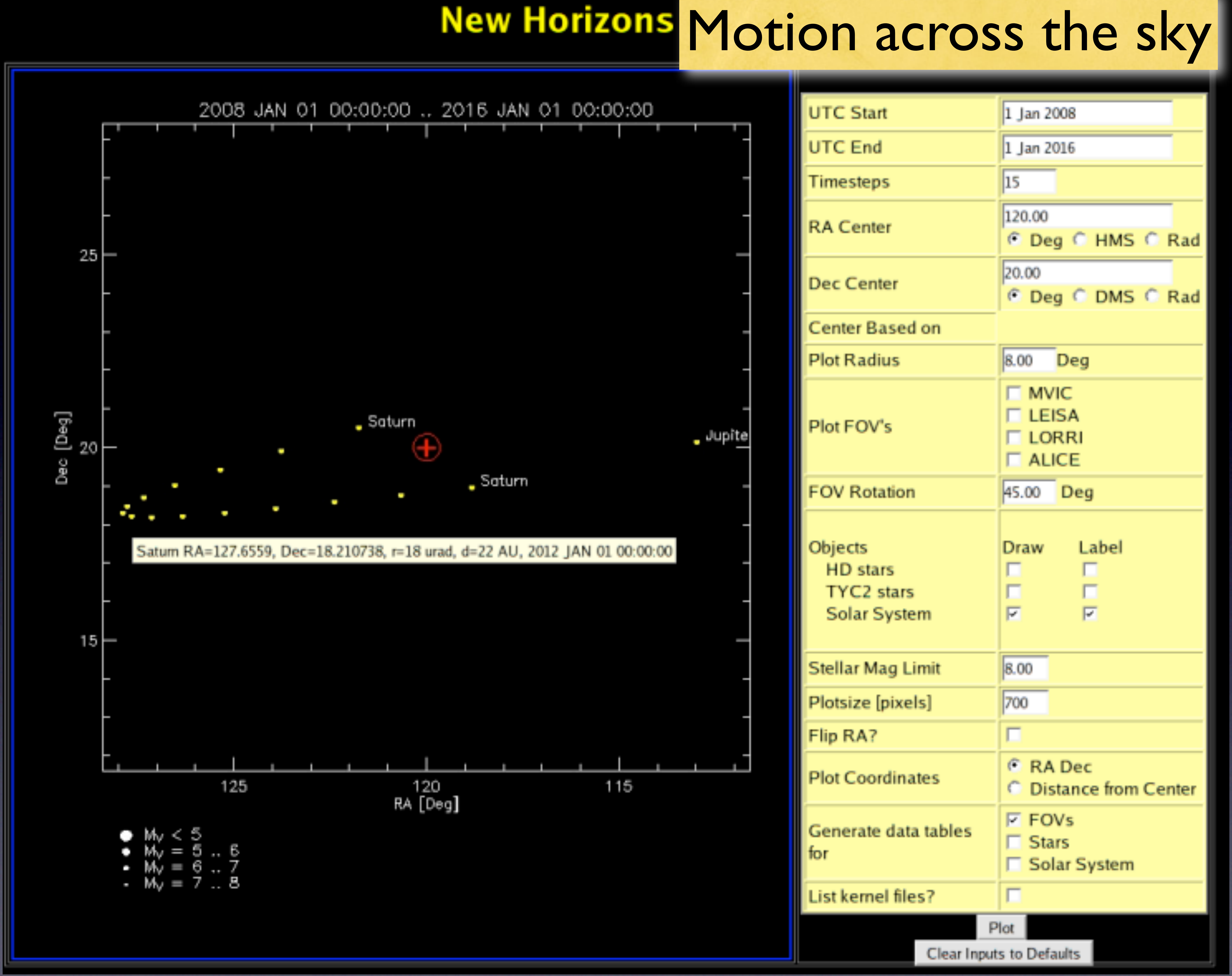
Realistic planet surface models



Star catalogs



Motion across the sky

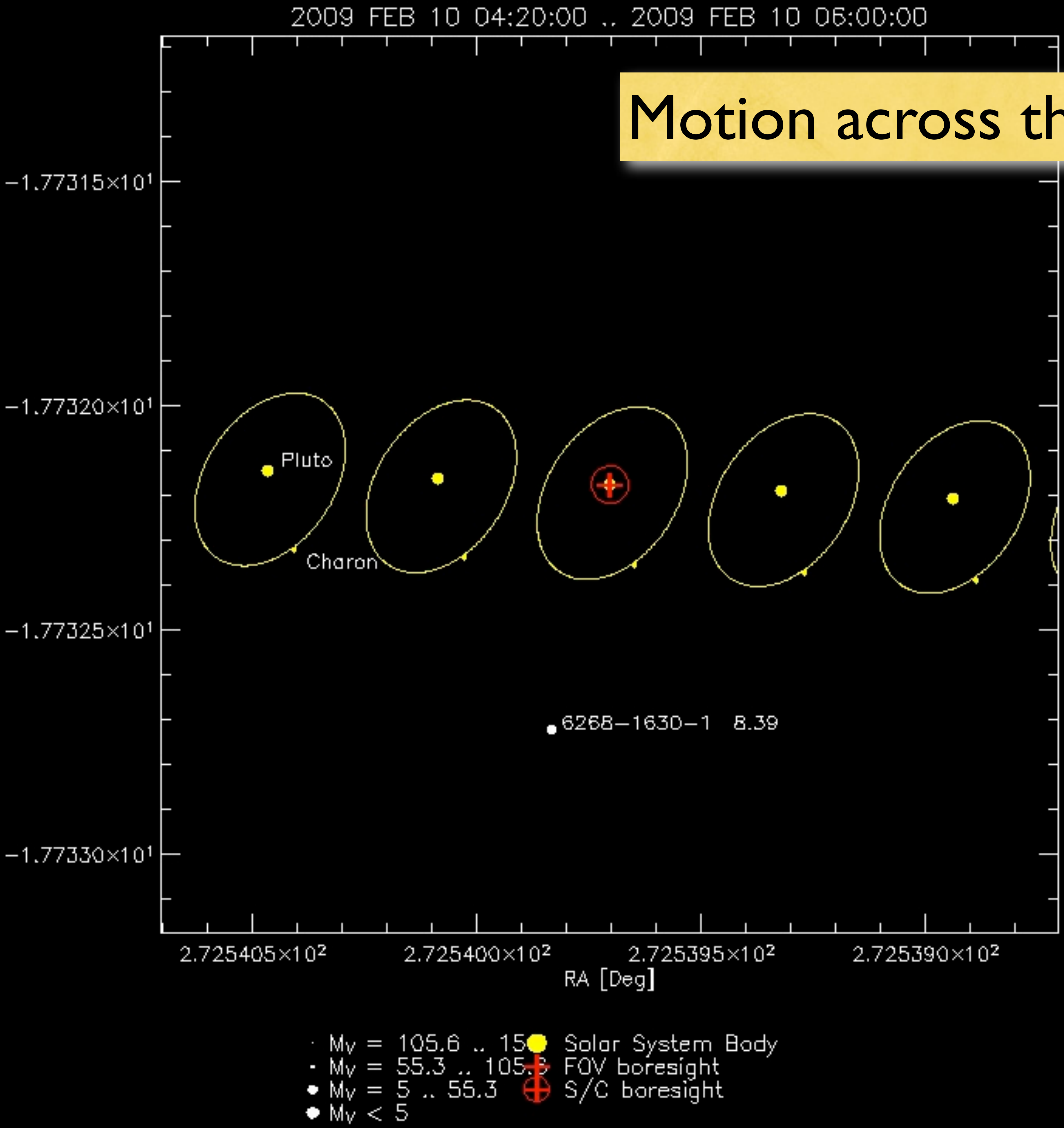


New Horizons

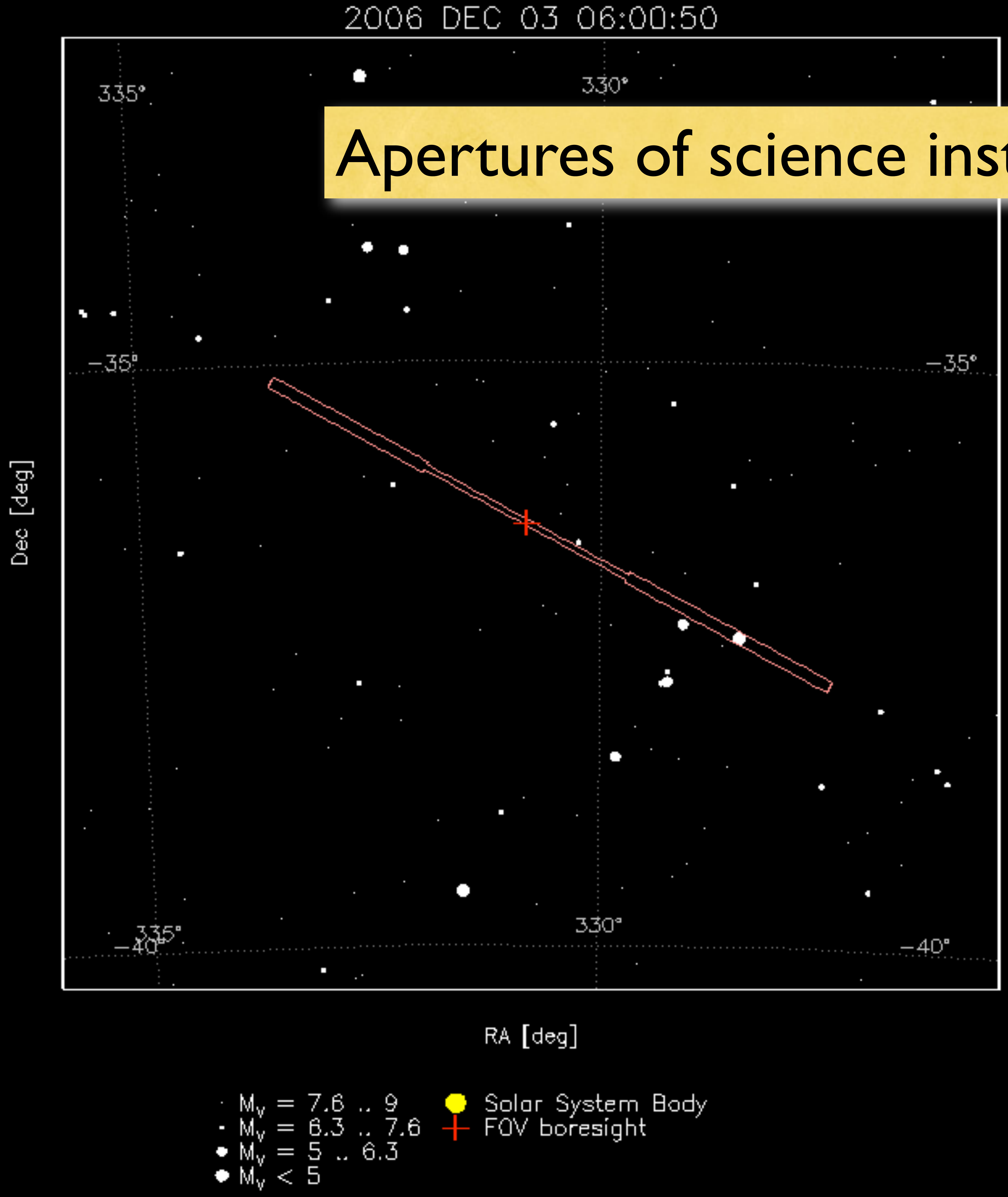
Saturn RA=127.6559, Dec=18.210738, r=18 urad, d=22 AU, 2012 JAN 01 00:00:00

UTC Start	1 Jan 2008
UTC End	1 Jan 2016
Timesteps	15
RA Center	120.00 ● Deg ○ HMS ○ Rad
Dec Center	20.00 ● Deg ○ DMS ○ Rad
Center Based on	
Plot Radius	8.00 Deg
Plot FOV's	□ MVIC □ LEISA □ LORRI □ ALICE
FOV Rotation	45.00 Deg
Objects HD stars TYC2 stars Solar System	Draw Label □ □ □ □ ☑ ☑
Stellar Mag Limit	8.00
Plotsize [pixels]	700
Flip RA?	□
Plot Coordinates	● RA Dec ○ Distance from Center
Generate data tables for	☑ FOVs □ Stars □ Solar System
List kernel files?	□

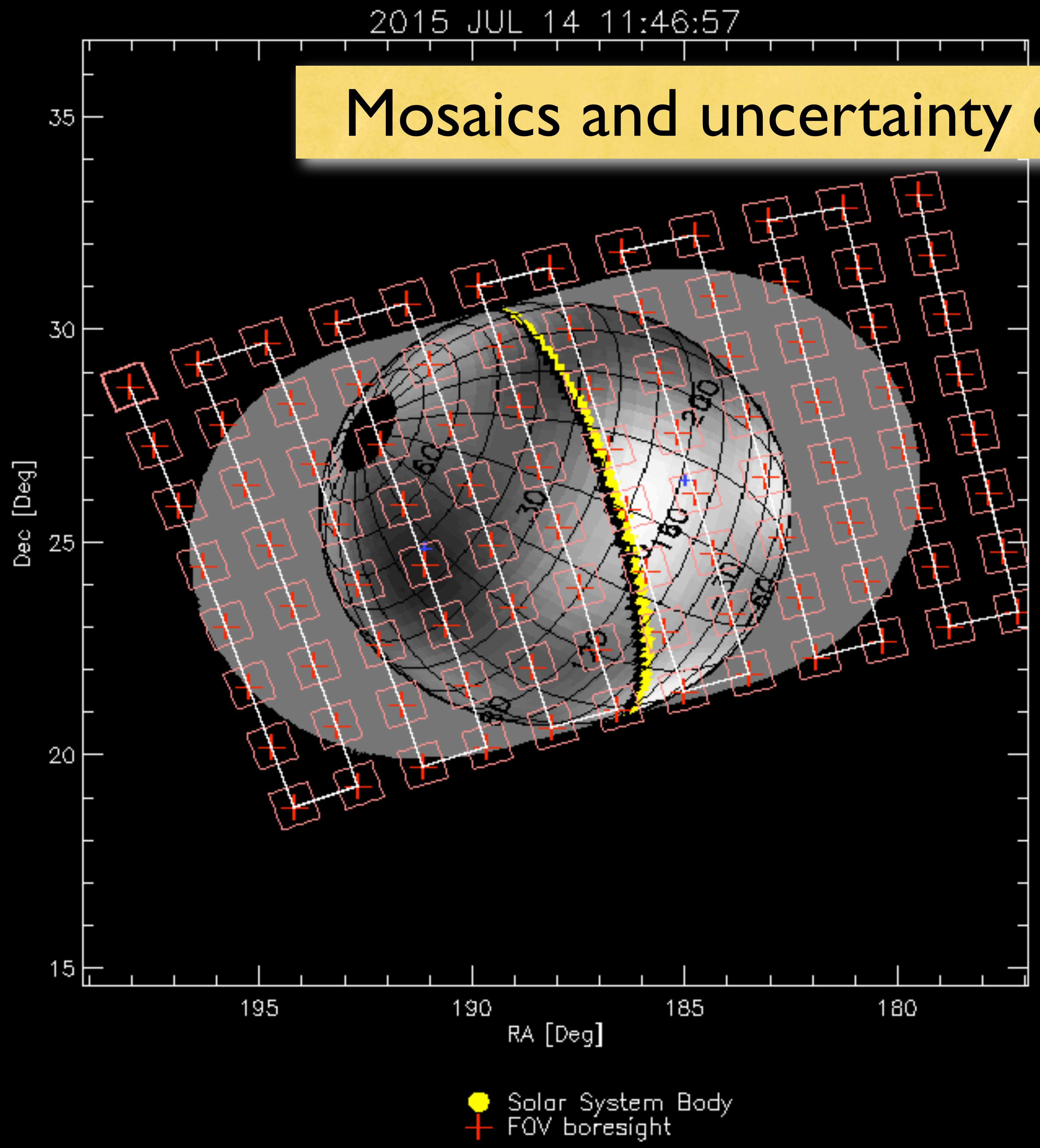
Motion across the sky



Apertures of science instruments

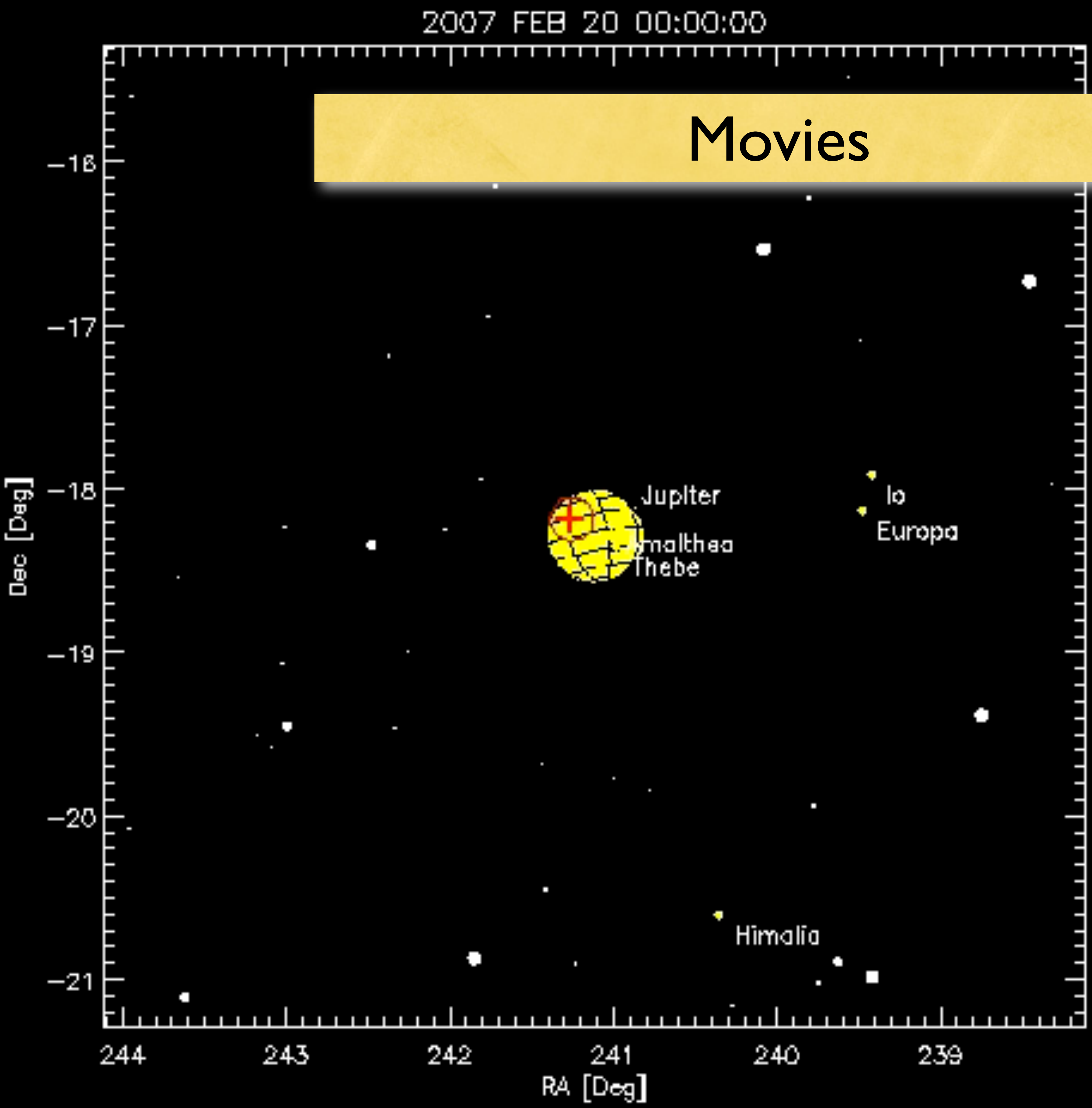


Mosaics and uncertainty ellipses

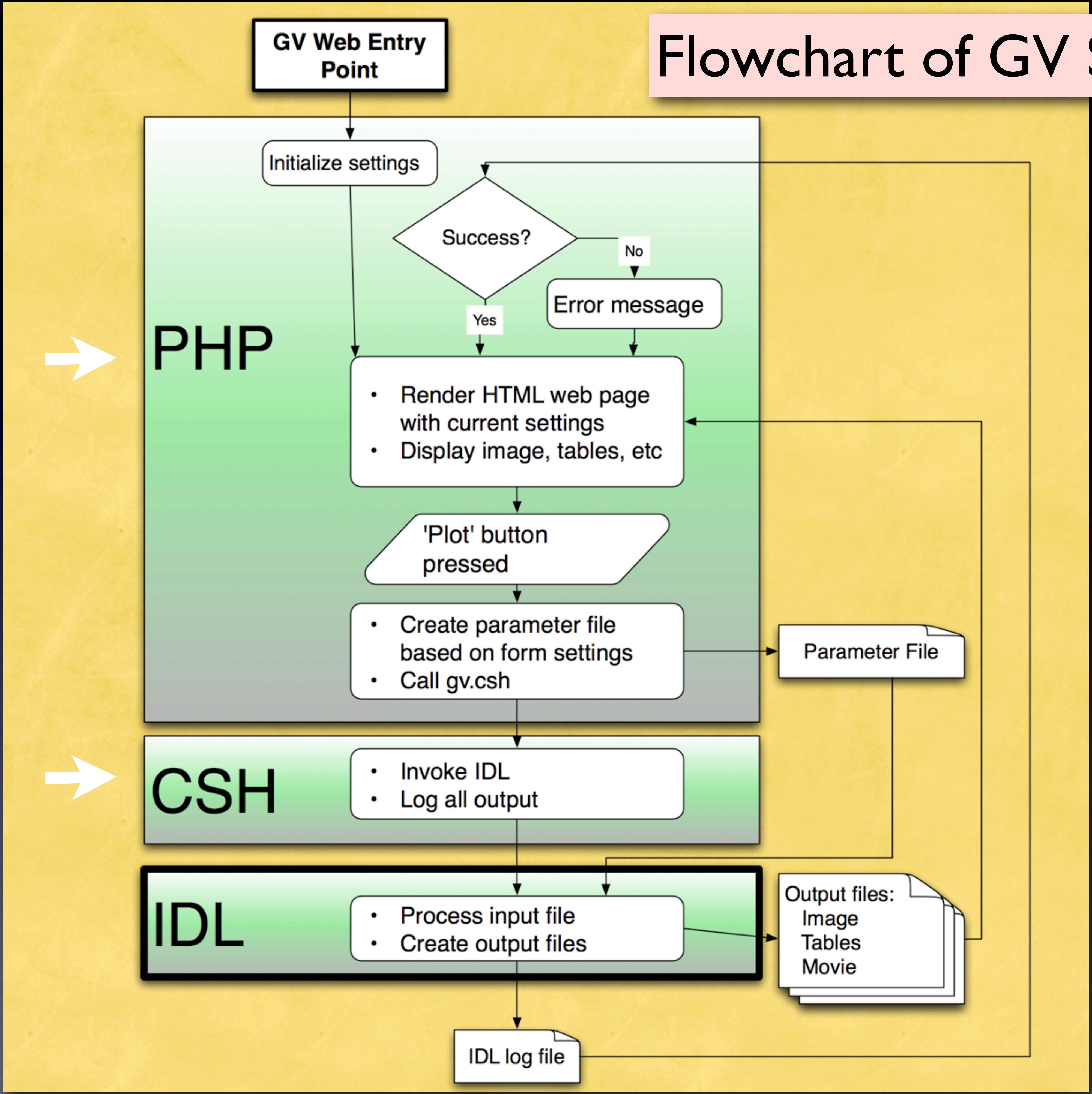


Movies

Movies


2007 FEB 20 00:00:00
Jupiter
Amalthea
Thebe
Io
Europa
Himalia
−16
−17
−18
−19
−20
−21
Dec [Deg]
244
243
242
241
240
239
RA [Deg]

Flowchart of GV System



2000 lines

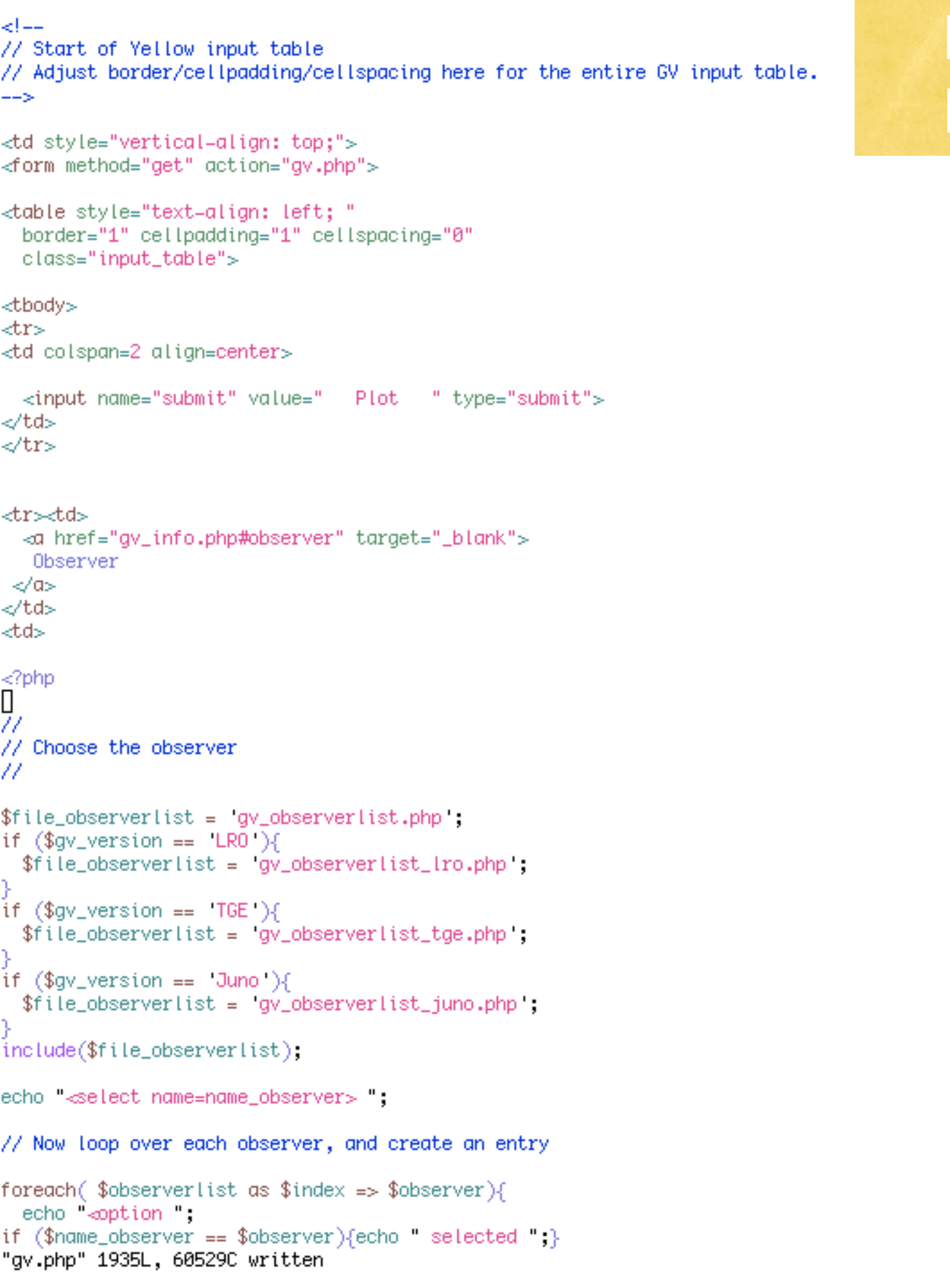
100 lines

15,000 lines

Web Interface

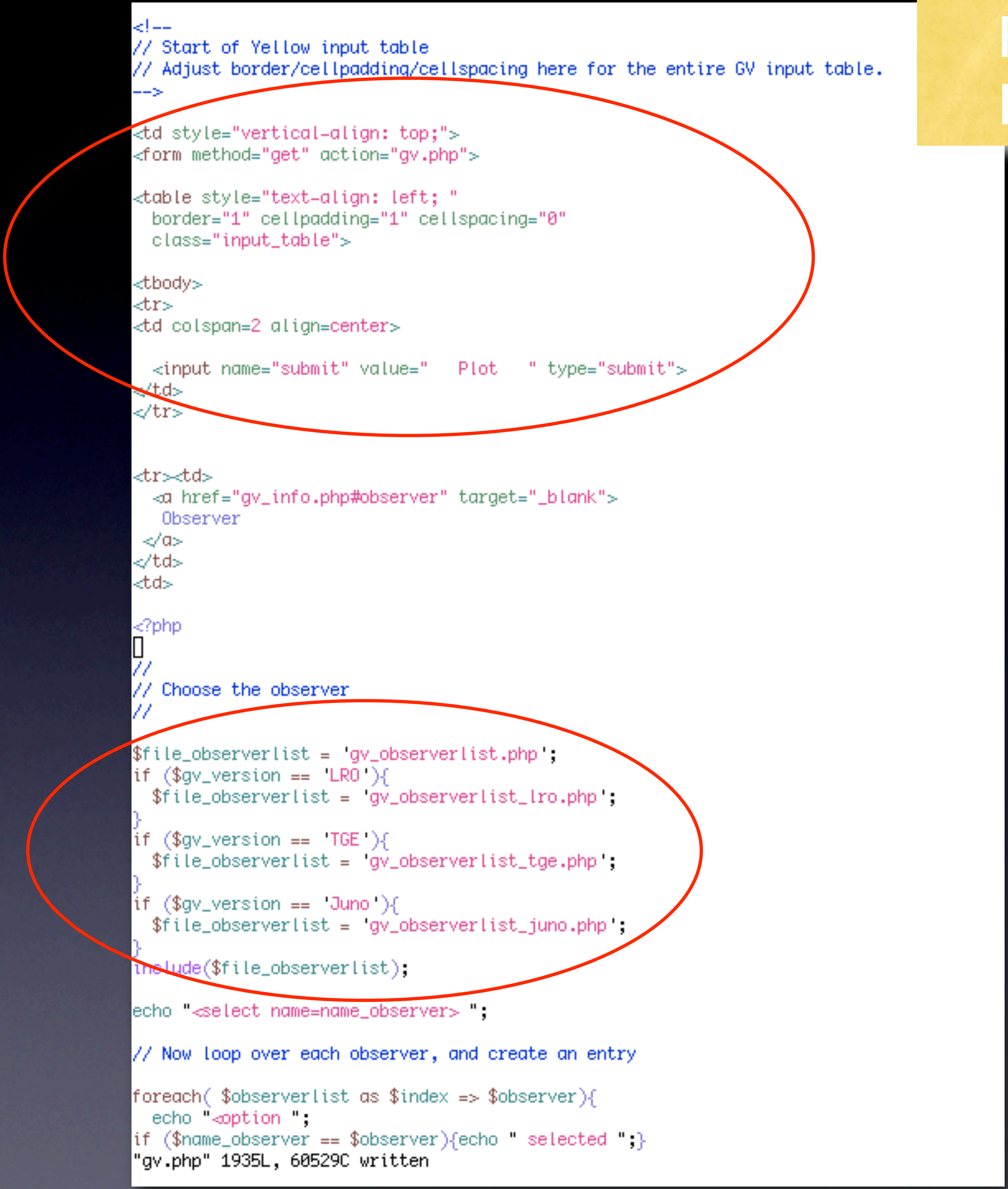
- ION (IDL-on-Net) is a discontinued product, so we can't use it!
- Instead, we use regular IDL, on a Linux Apache server.
  - PHP code draws web page, waits for user input. HTML, not widgets!
  - IDL starts up, runs, creates output files, and exits.
  - PHP reads these files and displays to screen.
- Requires starting up IDL at every web call, but more robust than other options.

PHP code generates HTML front end



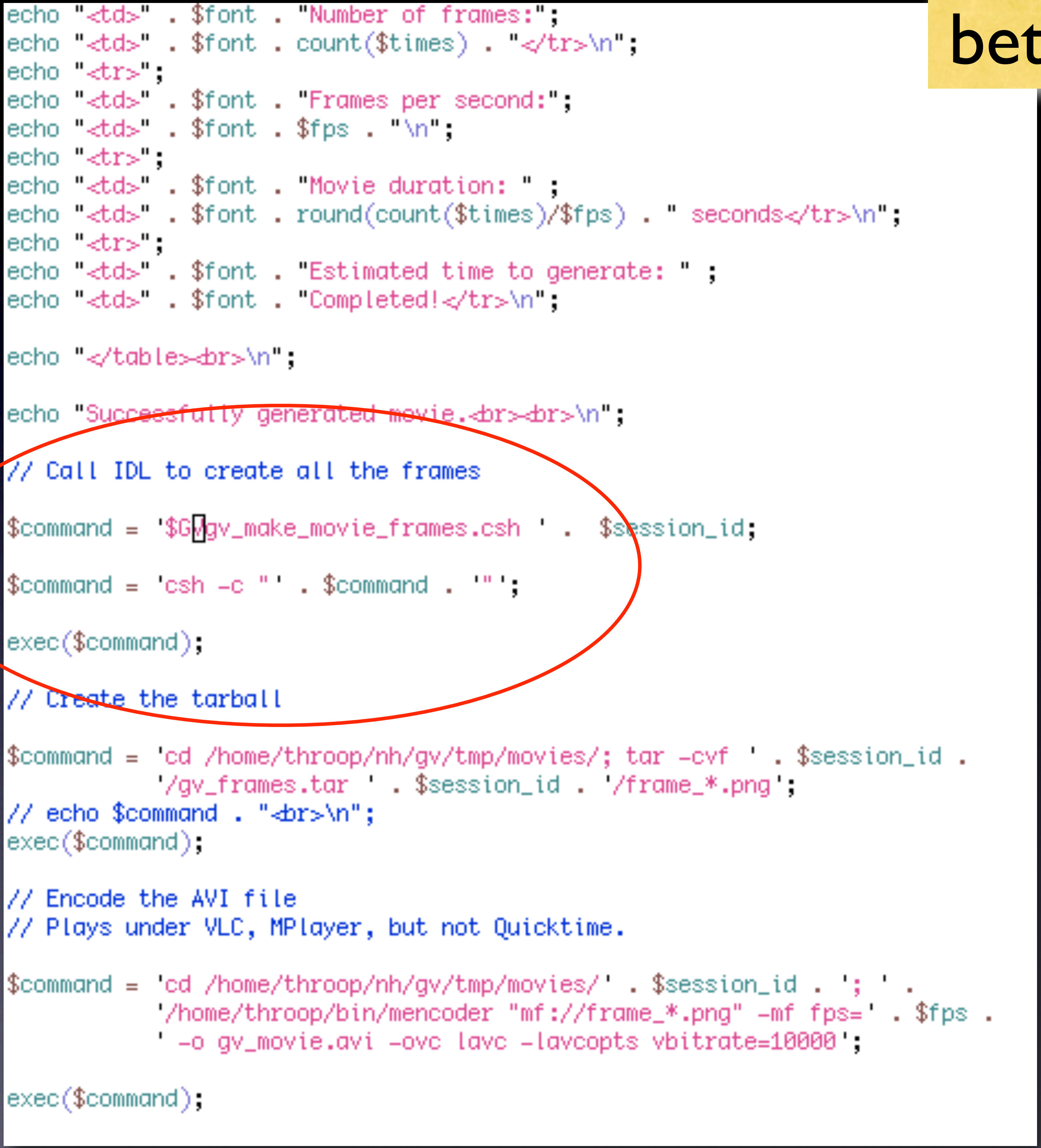
```
<!--
// Start of Yellow input table
// Adjust border/cellpadding/cellspacing here for the entire GV input table.
-->

<td style="vertical-align: top;">
<form method="get" action="gv.php">

<table style="text-align: left; "
  border="1" cellpadding="1" cellspacing="0"
  class="input_table">

<tbody>
<tr>
<td colspan=2 align=center>

  <input name="submit" value="   Plot    " type="submit">
</td>
</tr>


<tr><td>
  <a href="gv_info.php#observer" target="_blank">
   Observer
 </a>
</td>
<td>

<?php

//
// Choose the observer
//

$file_observerlist = 'gv_observerlist.php';
if ($gv_version == 'LRO'){
  $file_observerlist = 'gv_observerlist_lro.php';
}
if ($gv_version == 'TGE'){
  $file_observerlist = 'gv_observerlist_tge.php';
}
if ($gv_version == 'Juno'){
  $file_observerlist = 'gv_observerlist_juno.php';
}
include($file_observerlist);

echo "<select name=name_observer> ";

// Now loop over each observer, and create an entry

foreach( $observerlist as $index => $observer){
  echo "<option ";
if ($name_observer == $observer){echo " selected ";}
"gv.php" 1935L, 60529C written
```

PHP code generates HTML front end



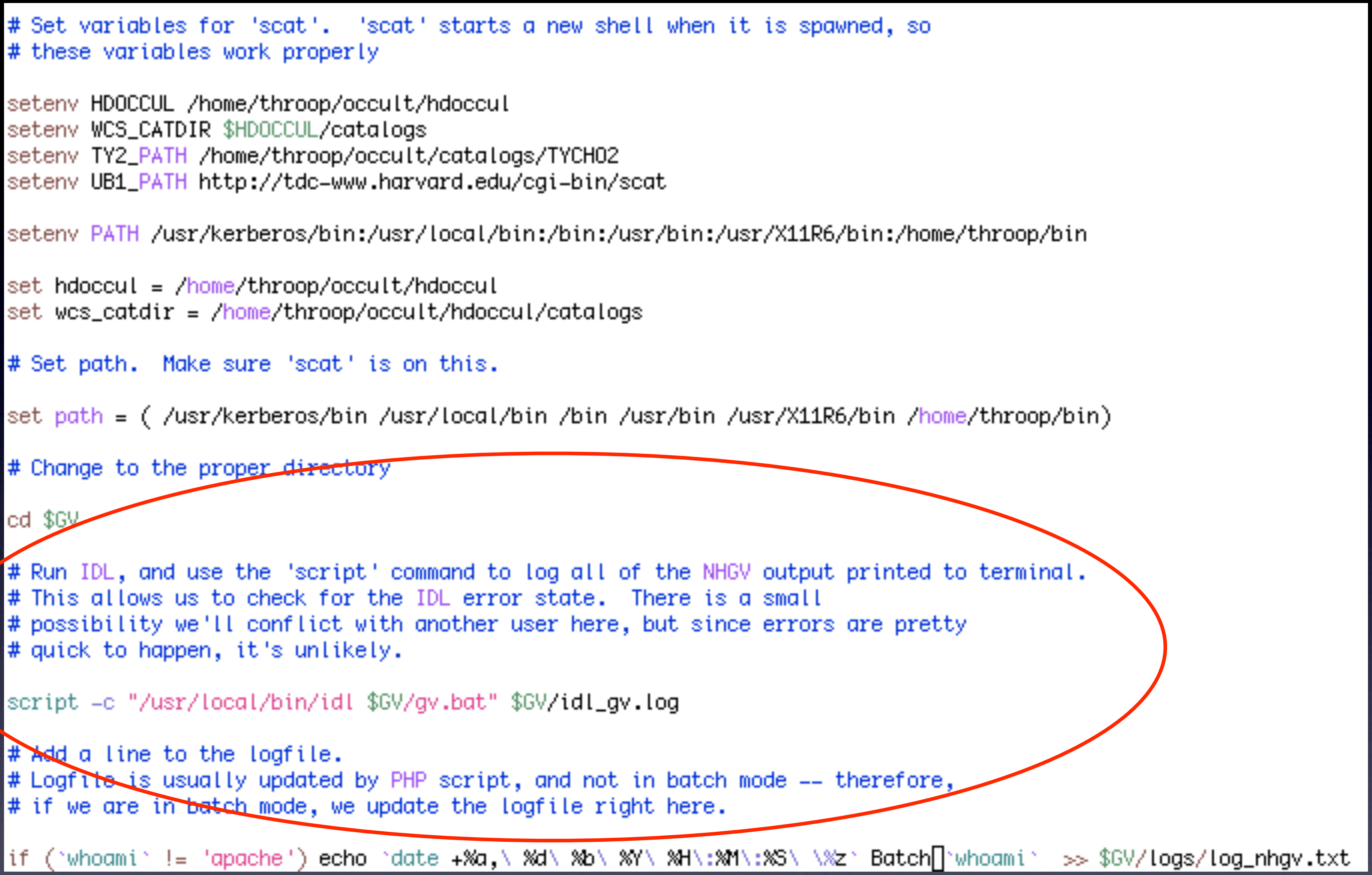
```
<!--
// Start of Yellow input table
// Adjust border/cellpadding/cellspacing here for the entire GV input table.
-->

<td style="vertical-align: top;">
<form method="get" action="gv.php">

<table style="text-align: left; "
  border="1" cellpadding="1" cellspacing="0"
  class="input_table">

<tbody>
<tr>
<td colspan=2 align=center>

  <input name="submit" value="   Plot    " type="submit">
</td>
</tr>


<tr><td>
  <a href="gv_info.php#observer" target="_blank">
   Observer
  </a>
</td>
<td>

<?php

//
// Choose the observer
//

$file_observerlist = 'gv_observerlist.php';
if ($gv_version == 'LRO'){
  $file_observerlist = 'gv_observerlist_lro.php';
}
if ($gv_version == 'TGE'){
  $file_observerlist = 'gv_observerlist_tge.php';
}
if ($gv_version == 'Juno'){
  $file_observerlist = 'gv_observerlist_juno.php';
}
include($file_observerlist);

echo "<select name=name_observer> ";

// Now loop over each observer, and create an entry

foreach( $observerlist as $index => $observer){
  echo "<option ";
if ($name_observer == $observer){echo " selected ";}
"gv.php" 1935L, 60529C written
```

PHP is interface between user and IDL



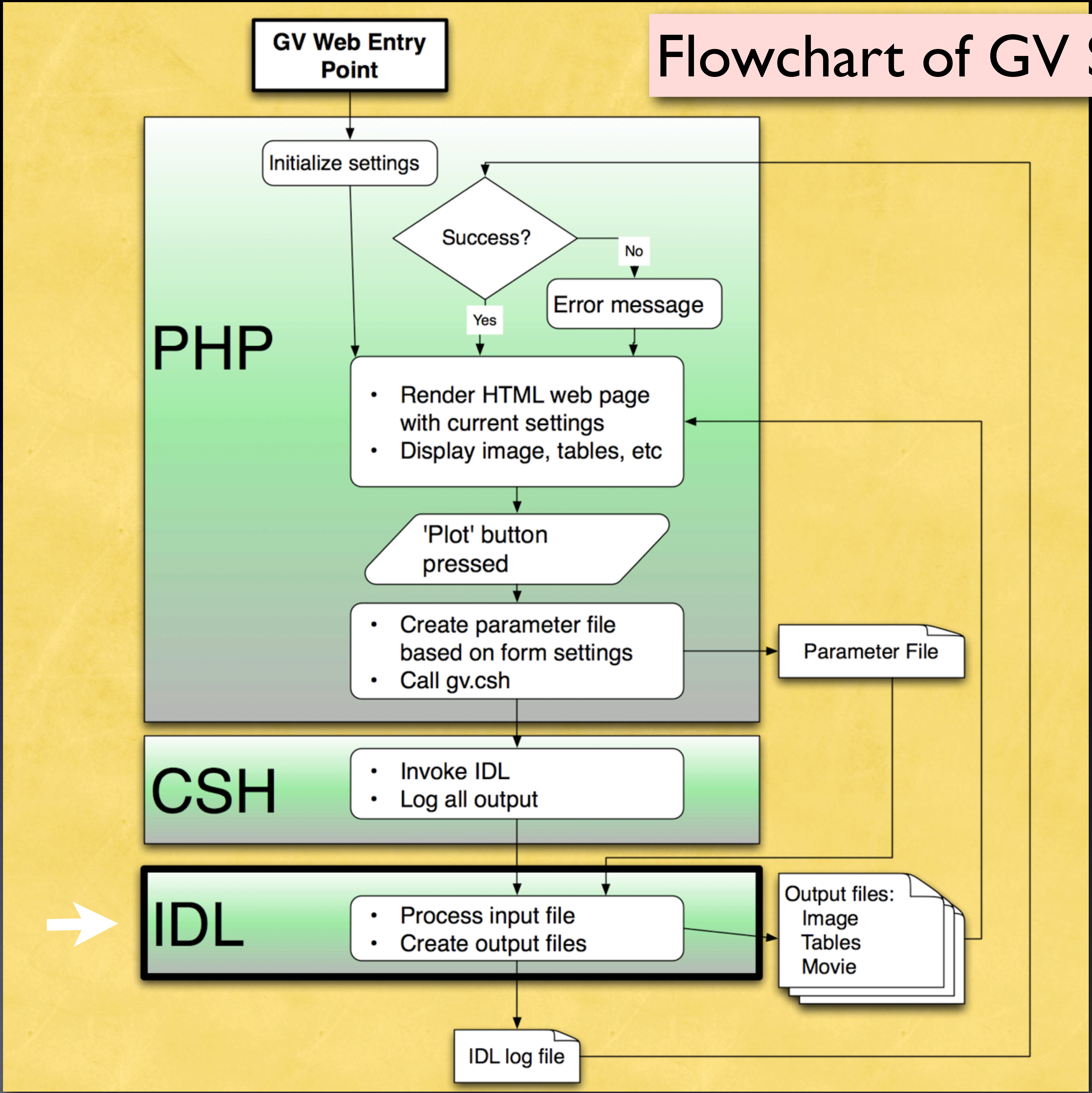
```
echo "<td>" . $font . "Number of frames:";
echo "<td>" . $font . count($times) . "</tr>\n";
echo "<tr>";
echo "<td>" . $font . "Frames per second:";
echo "<td>" . $font . $fps . "\n";
echo "<tr>";
echo "<td>" . $font . "Movie duration: " ;
echo "<td>" . $font . round(count($times)/$fps) . " seconds</tr>\n";
echo "<tr>";
echo "<td>" . $font . "Estimated time to generate: " ;
echo "<td>" . $font . "Completed!</tr>\n";

echo "</table><br>\n";

echo "Successfully generated movie.<br><br>\n";

// Call IDL to create all the frames

$command = '$G/gv_make_movie_frames.csh ' .  $session_id;

$command = 'csh -c "' . $command . '"';

exec($command);

// Create the tarball

$command = 'cd /home/throop/nh/gv/tmp/movies/; tar -cvf ' . $session_id .
           '/gv_frames.tar ' . $session_id . '/frame_*.png';
// echo $command . "<br>\n";
exec($command);

// Encode the AVI file
// Plays under VLC, MPlayer, but not Quicktime.

$command = 'cd /home/throop/nh/gv/tmp/movies/' . $session_id . '; ' .
           '/home/throop/bin/mencoder "mf://frame_*.png" -mf fps=' . $fps .
           ' -o gv_movie.avi -ovc lavc -lavcopts vbitrate=10000';

exec($command);
```

csh scripts: Invoke IDL and log output



```
# Set variables for 'scat'.  'scat' starts a new shell when it is spawned, so
# these variables work properly

setenv HDOCCUL /home/throop/occult/hdoccul
setenv WCS_CATDIR $HDOCCUL/catalogs
setenv TY2_PATH /home/throop/occult/catalogs/TYCHO2
setenv UB1_PATH http://tdc-www.harvard.edu/cgi-bin/scat

setenv PATH /usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/throop/bin

set hdoccul = /home/throop/occult/hdoccul
set wcs_catdir = /home/throop/occult/hdoccul/catalogs

# Set path.  Make sure 'scat' is on this.

set path = ( /usr/kerberos/bin /usr/local/bin /bin /usr/bin /usr/X11R6/bin /home/throop/bin)

# Change to the proper directory

cd $GV

# Run IDL, and use the 'script' command to log all of the NHGV output printed to terminal.
# This allows us to check for the IDL error state.  There is a small
# possibility we'll conflict with another user here, but since errors are pretty
# quick to happen, it's unlikely.

script -c "/usr/local/bin/idl $GV/gv.bat" $GV/idl_gv.log

# Add a line to the logfile.
# Logfile is usually updated by PHP script, and not in batch mode -- therefore,
# if we are in batch mode, we update the logfile right here.

if (`whoami` != 'apache') echo `date +%a,\ %d\ %b\ %Y\ %H\:%M\:%S\ \%z` Batch□`whoami`  >> $GV/logs/log_nhgv.txt
```

Flowchart of GV System



GV Web Entry Point

PHP

Initialize settings

Success?

No

Yes

Error message

- Render HTML web page with current settings
- Display image, tables, etc

'Plot' button pressed

- Create parameter file based on form settings
- Call gv.csh

Parameter File

CSH

- Invoke IDL
- Log all output

IDL

- Process input file
- Create output files

Output files:
Image
Tables
Movie

IDL log file

2000 lines

100 lines

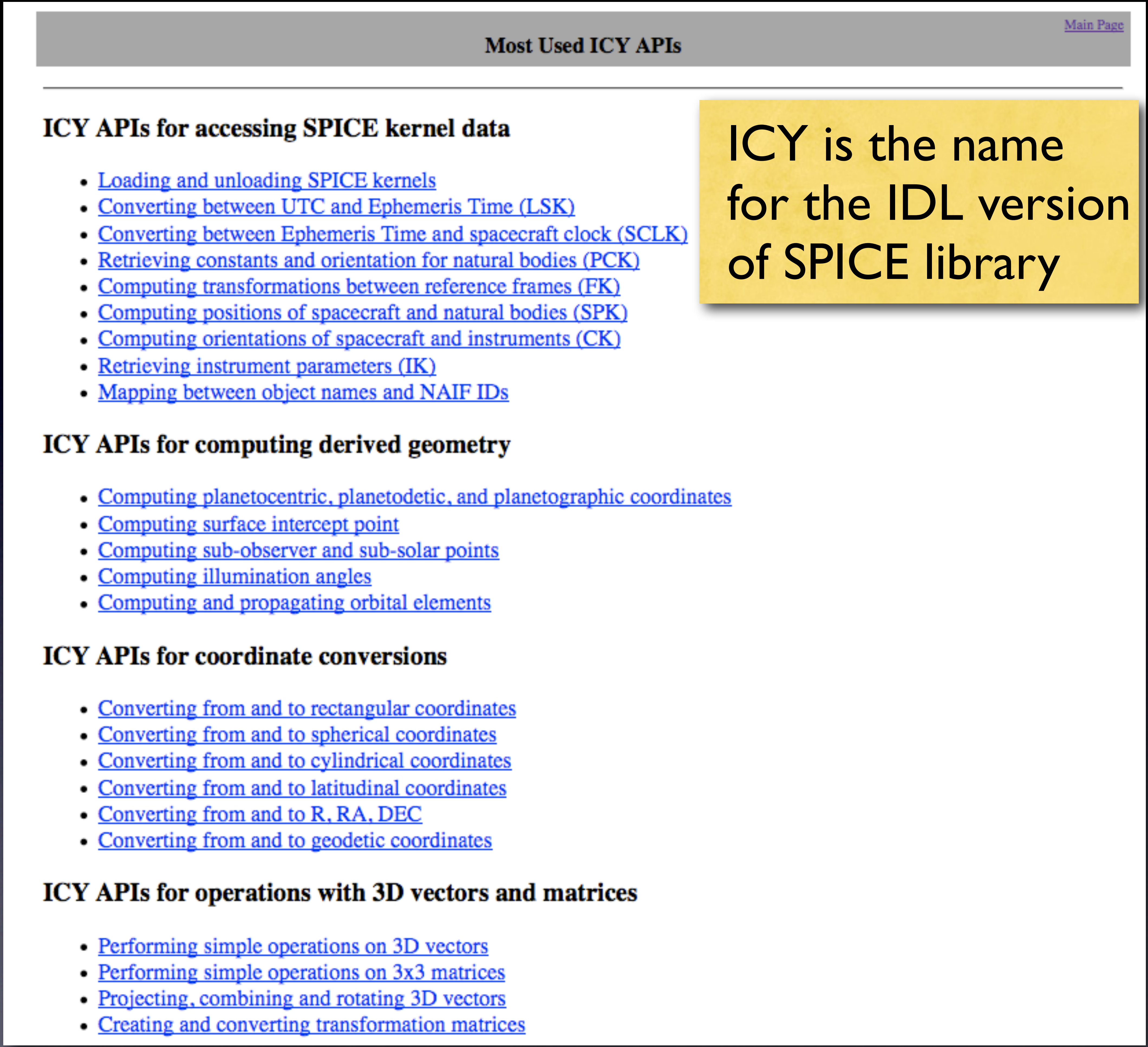
15,000 lines

SPICE *

(*) SPacecraft, Instrument, Camera kErnels

- SPICE is a spacecraft geometry toolset developed and maintained by NASA-JPL.
- Highly accurate astrometric / geometric computations. Hundreds of functions.
- Distributed for C, FORTRAN, IDL.
- SPICE is **the** standard for interplanetary spacecraft navigation.
- ‘Kernel’ files define trajectories, ephemerides, FOVs, leap-seconds, planet masses, etc.
- Using SPICE assures that everyone gets the same results!

http://naif.jpl.nasa.gov



Main Page

Most Used ICY APIs

ICY APIs for accessing SPICE kernel data

- Loading and unloading SPICE kernels
- Converting between UTC and Ephemeris Time (LSK)
- Converting between Ephemeris Time and spacecraft clock (SCLK)
- Retrieving constants and orientation for natural bodies (PCK)
- Computing transformations between reference frames (FK)
- Computing positions of spacecraft and natural bodies (SPK)
- Computing orientations of spacecraft and instruments (CK)
- Retrieving instrument parameters (IK)
- Mapping between object names and NAIF IDs

ICY APIs for computing derived geometry

- Computing planetocentric, planetodetic, and planetographic coordinates
- Computing surface intercept point
- Computing sub-observer and sub-solar points
- Computing illumination angles
- Computing and propagating orbital elements

ICY APIs for coordinate conversions

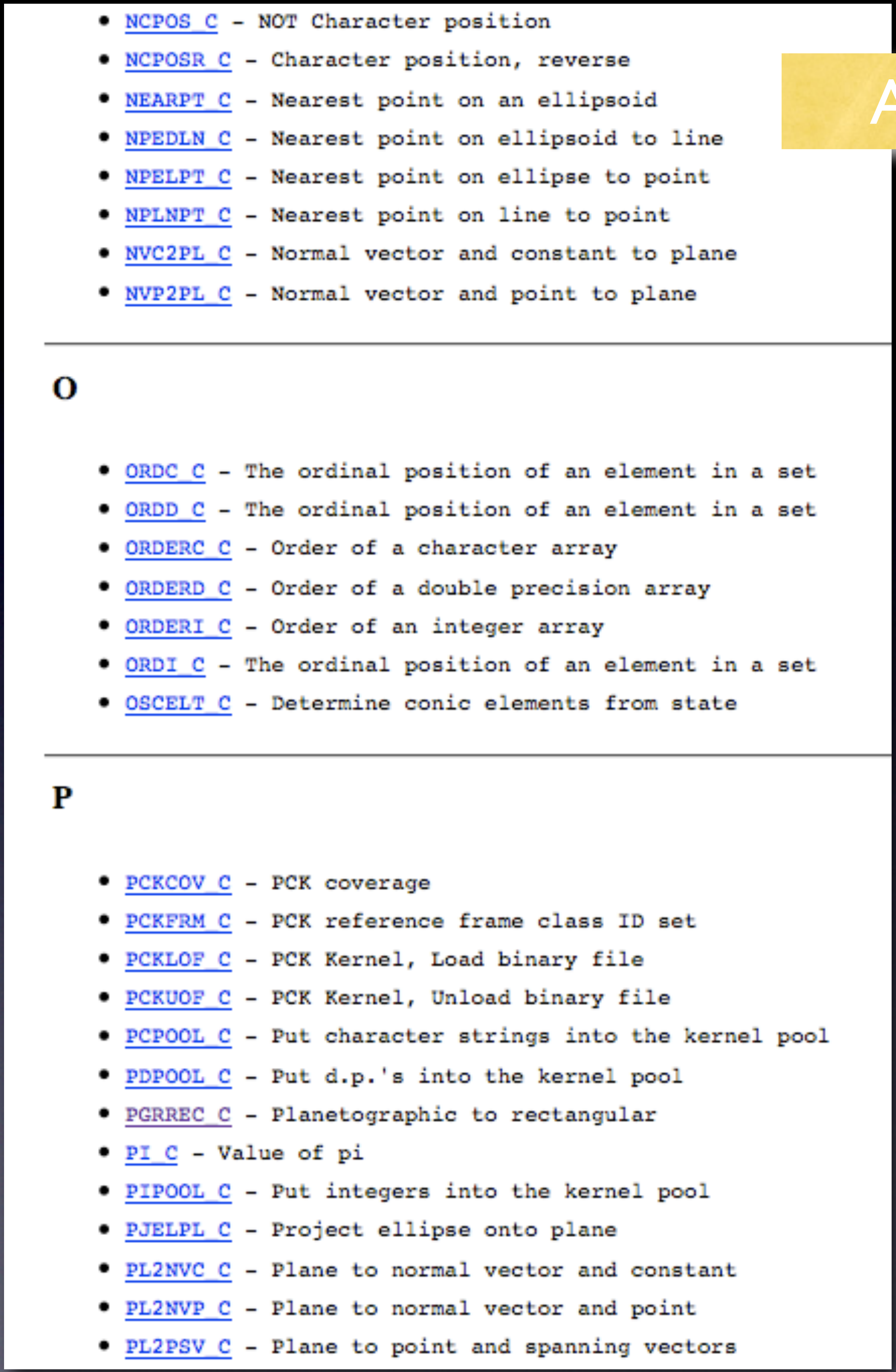
- Converting from and to rectangular coordinates
- Converting from and to spherical coordinates
- Converting from and to cylindrical coordinates
- Converting from and to latitudinal coordinates
- Converting from and to R, RA, DEC
- Converting from and to geodetic coordinates

ICY APIs for operations with 3D vectors and matrices

- Performing simple operations on 3D vectors
- Performing simple operations on 3x3 matrices
- Projecting, combining and rotating 3D vectors
- Creating and converting transformation matrices

ICY is the name for the IDL version of SPICE library

A few sample SPICE routines...



- NCPOS_C - NOT Character position
- NCPOSR_C - Character position, reverse
- NEARPT_C - Nearest point on an ellipsoid
- NPEDLN_C - Nearest point on ellipsoid to line
- NPELPT_C - Nearest point on ellipse to point
- NPLNPT_C - Nearest point on line to point
- NVC2PL_C - Normal vector and constant to plane
- NVP2PL_C - Normal vector and point to plane

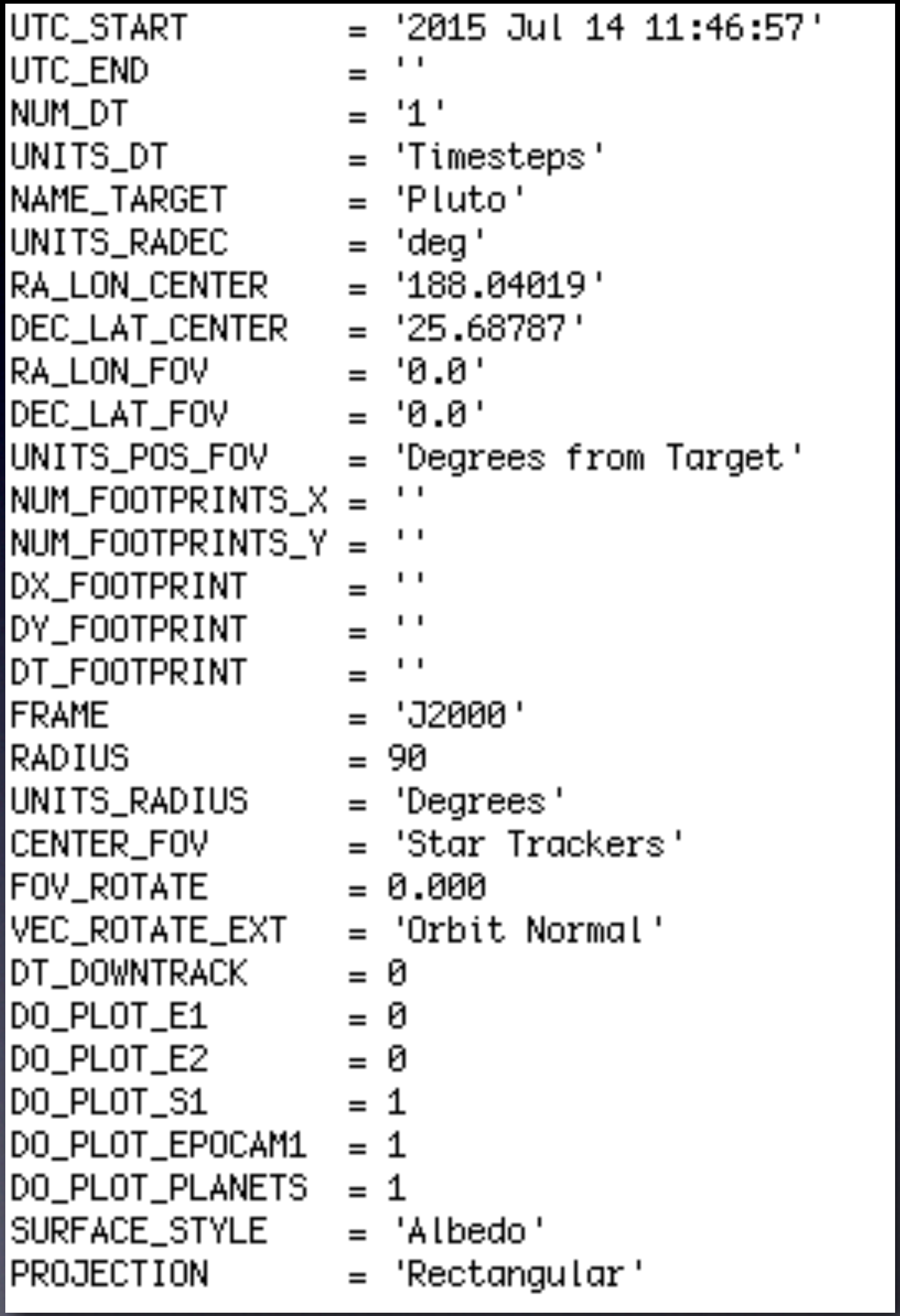
O

- ORDC_C - The ordinal position of an element in a set
- ORDD_C - The ordinal position of an element in a set
- ORDERC_C - Order of a character array
- ORDERD_C - Order of a double precision array
- ORDERI_C - Order of an integer array
- ORDI_C - The ordinal position of an element in a set
- OSCELT_C - Determine conic elements from state

P

- PCKCOV_C - PCK coverage
- PCKFRM_C - PCK reference frame class ID set
- PCKLOF_C - PCK Kernel, Load binary file
- PCKUOF_C - PCK Kernel, Unload binary file
- PCPOOL_C - Put character strings into the kernel pool
- PDPOOL_C - Put d.p.'s into the kernel pool
- PGRREC_C - Planetographic to rectangular
- PI_C - Value of pi
- PIPOOL_C - Put integers into the kernel pool
- PJELPL_C - Project ellipse onto plane
- PL2NVC_C - Plane to normal vector and constant
- PL2NVP_C - Plane to normal vector and point
- PL2PSV_C - Plane to point and spanning vectors

GV Parameter file



```
UTC_START        = '2015 Jul 14 11:46:57'
UTC_END          = ''
NUM_DT           = '1'
UNITS_DT         = 'Timesteps'
NAME_TARGET      = 'Pluto'
UNITS_RADEC      = 'deg'
RA_LON_CENTER    = '188.04019'
DEC_LAT_CENTER   = '25.68787'
RA_LON_FOV       = '0.0'
DEC_LAT_FOV      = '0.0'
UNITS_POS_FOV    = 'Degrees from Target'
NUM_FOOTPRINTS_X = ''
NUM_FOOTPRINTS_Y = ''
DX_FOOTPRINT     = ''
DY_FOOTPRINT     = ''
DT_FOOTPRINT     = ''
FRAME            = 'J2000'
RADIUS           = 90
UNITS_RADIUS     = 'Degrees'
CENTER_FOV       = 'Star Trackers'
FOV_ROTATE       = 0.000
VEC_ROTATE_EXT   = 'Orbit Normal'
DT_DOWNTRACK     = 0
DO_PLOT_E1       = 0
DO_PLOT_E2       = 0
DO_PLOT_S1       = 1
DO_PLOT_EPOCAM1  = 1
DO_PLOT_PLANETS  = 1
SURFACE_STYLE    = 'Albedo'
PROJECTION       = 'Rectangular'
```

Values entered by user.

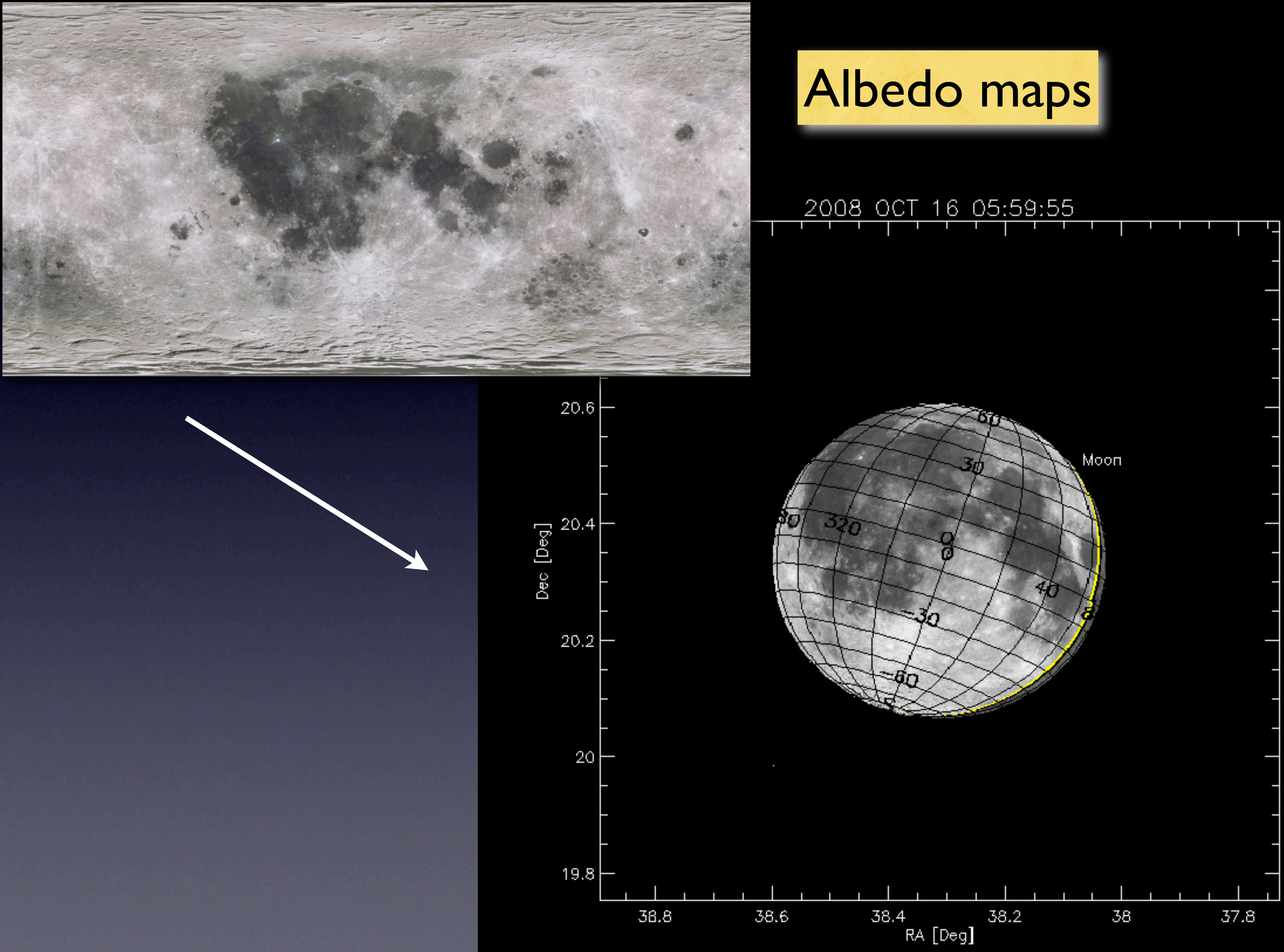
File generated by PHP.

Read and processed by IDL:

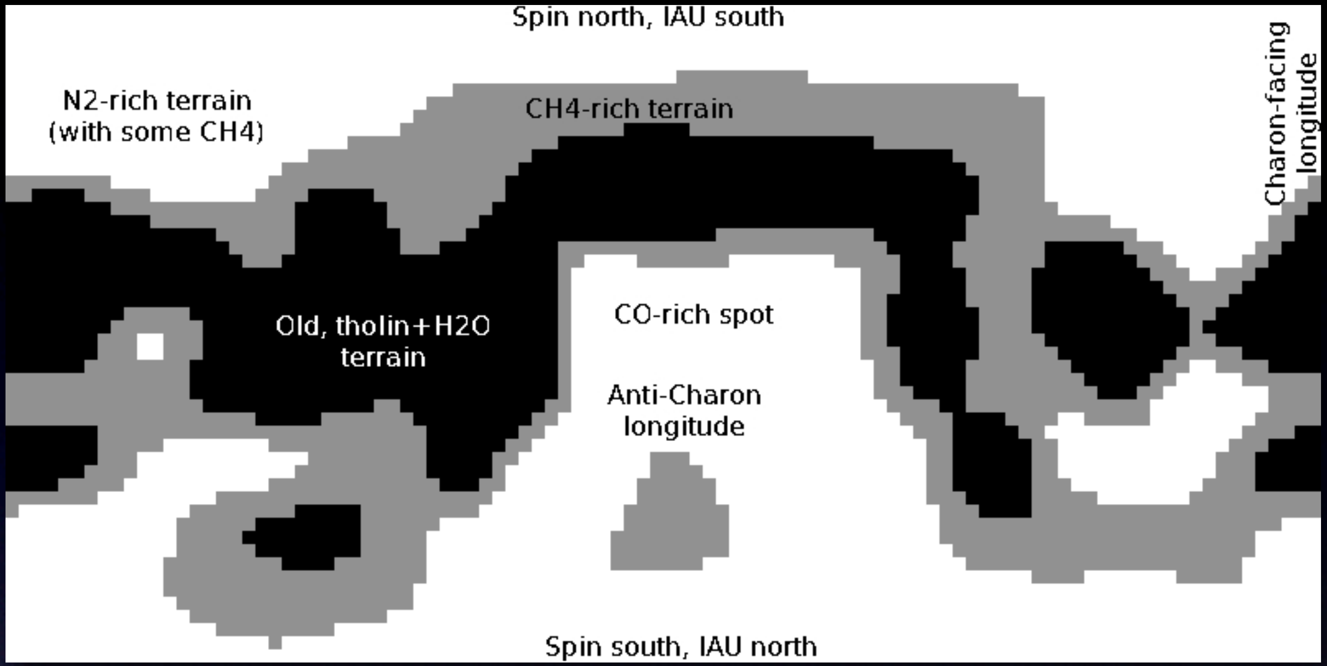
Compiled on-the-fly and executed as IDL code.

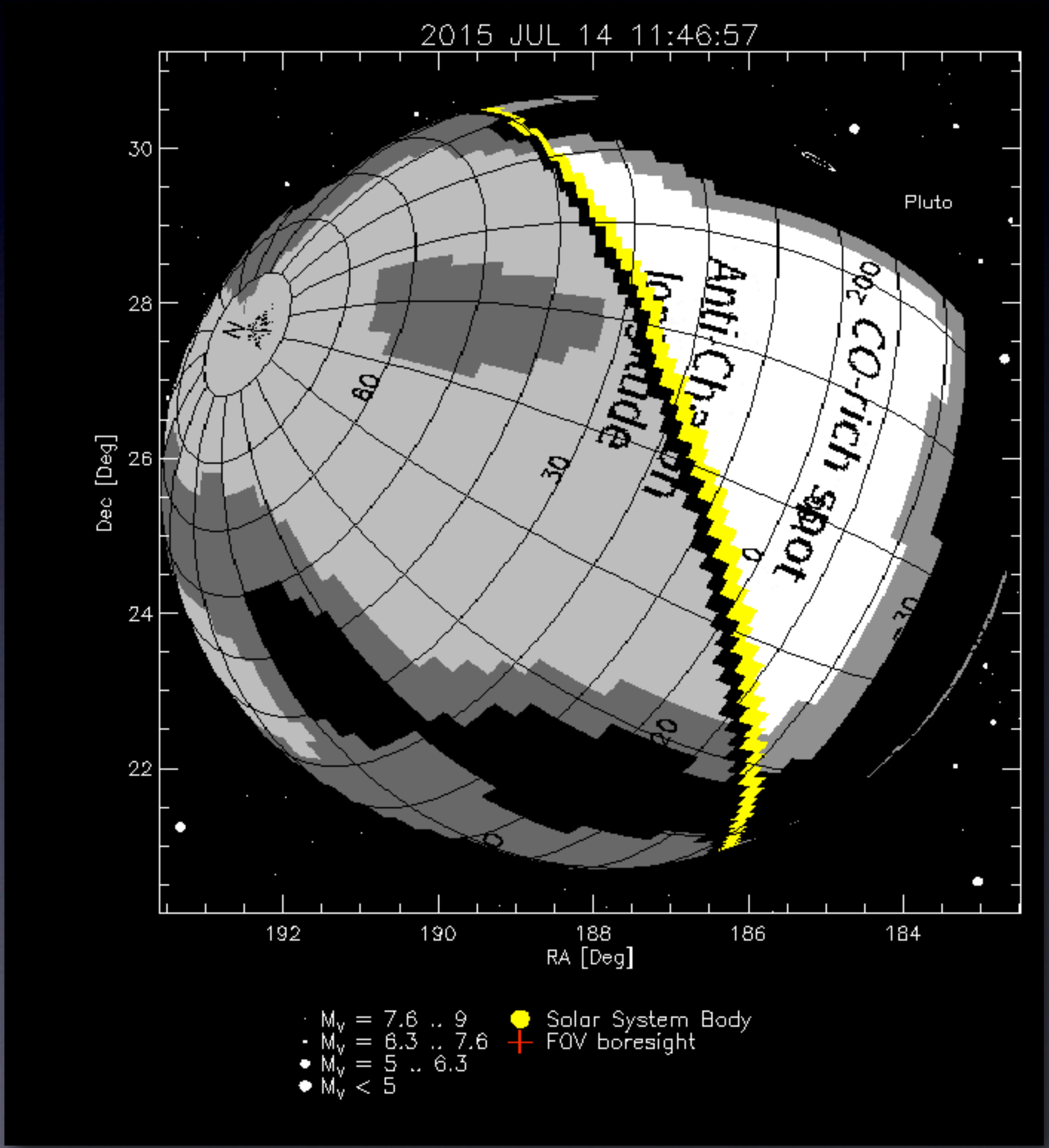
Map Projections

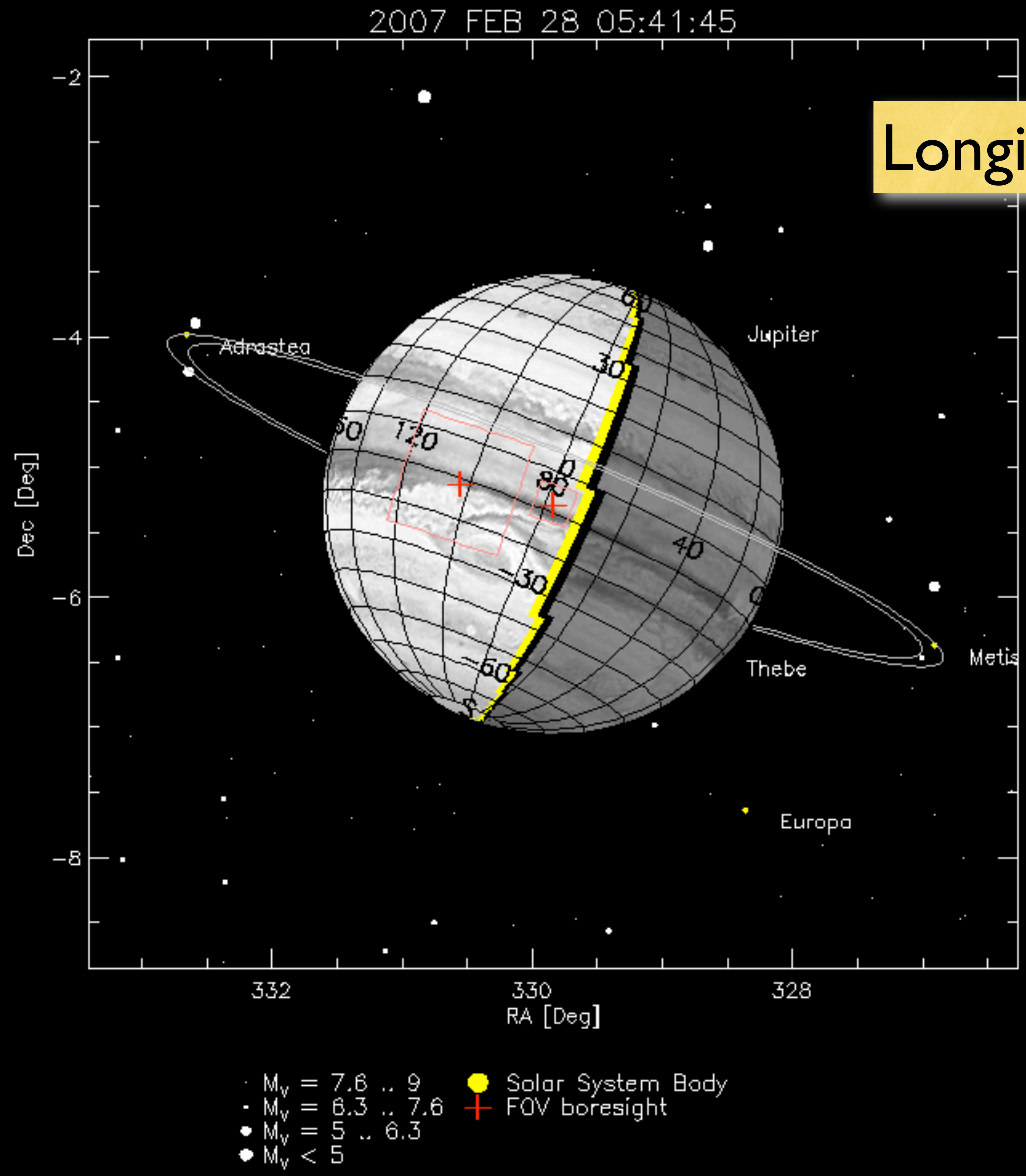
Albedo maps


2008 OCT 16 05:59:55
Moon
60
30
30
320
0
0
40
-30
-60
20.6
20.4
20.2
20
19.8
Dec [Deg]
38.8
38.6
38.4
38.2
38
37.8
RA [Deg]

Composition maps



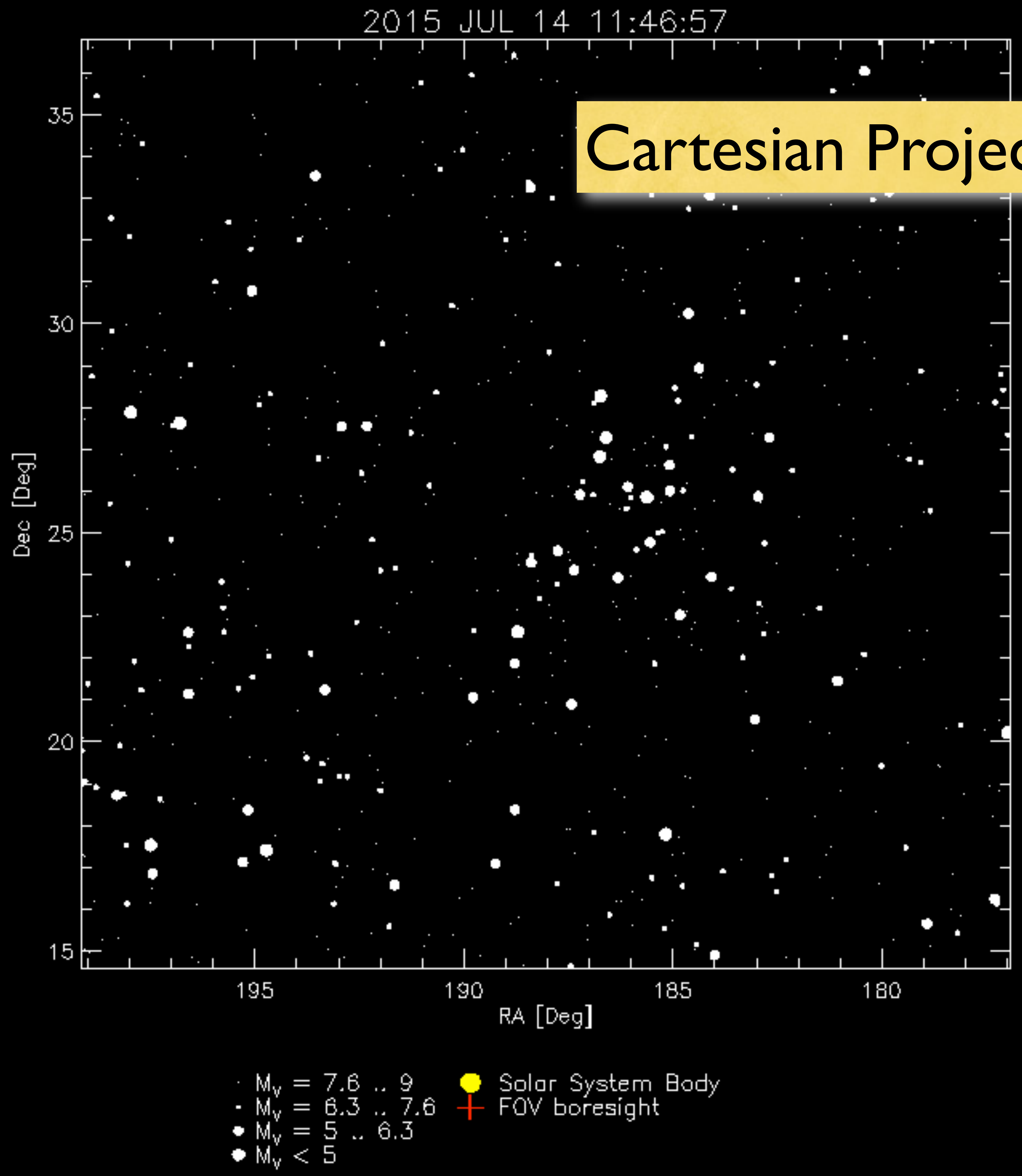




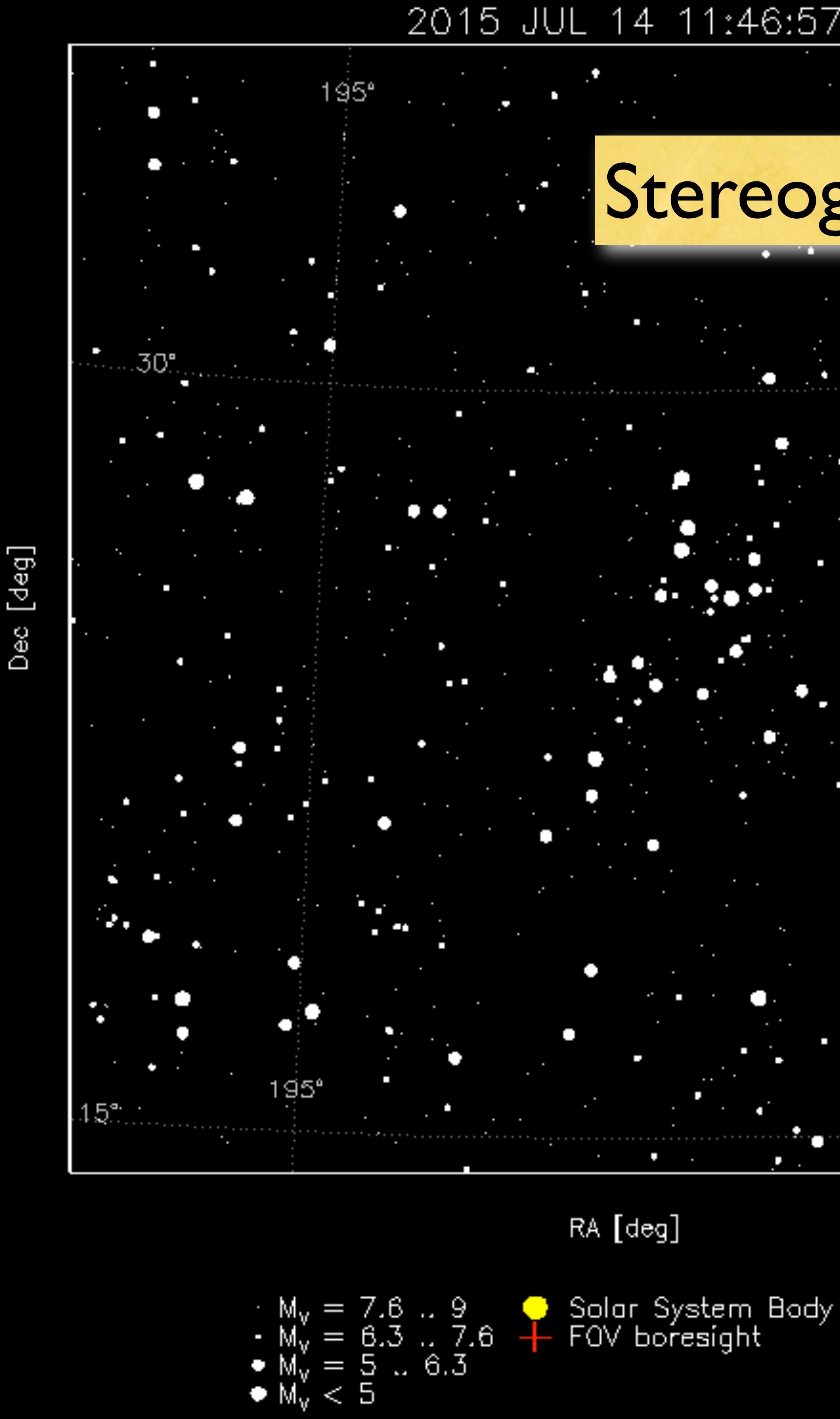
Longitude systems

- IAU vs IDL longitude systems
  - Most go 0 .. 360, not 180 W, 180 E
  - Some have N pole in different directions
  - Spherical bodies only

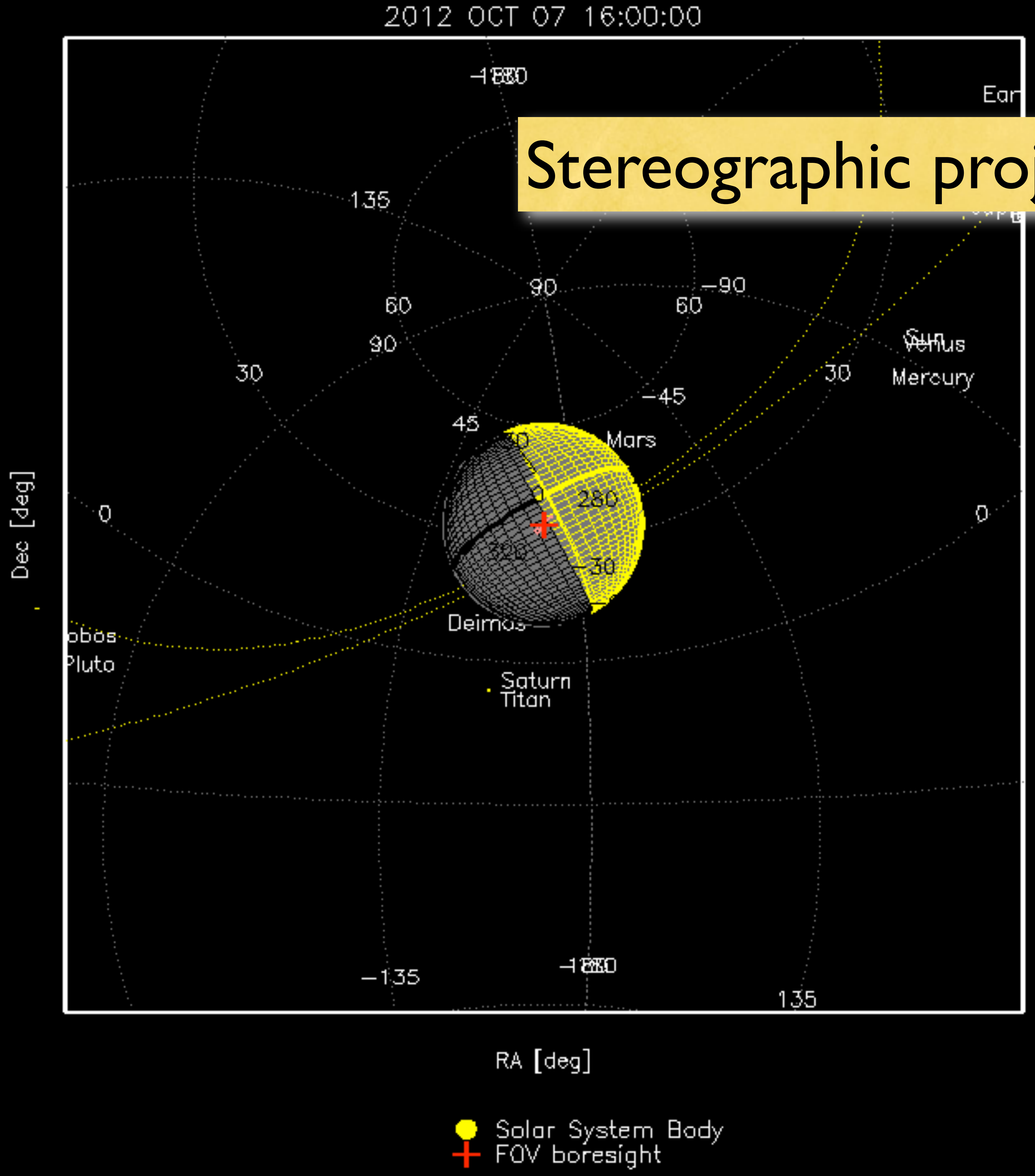
Cartesian Projection



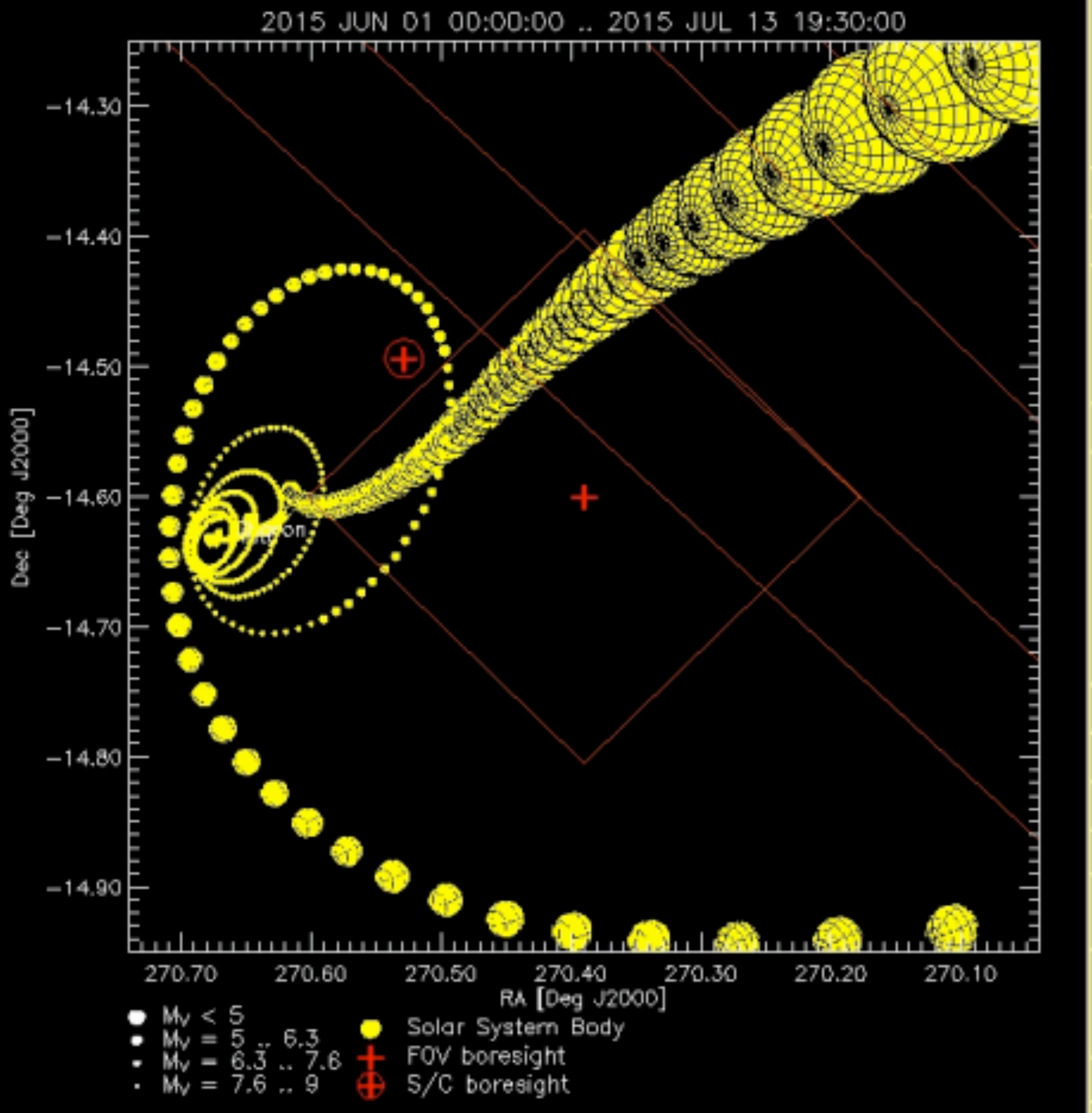
Stereographic projection



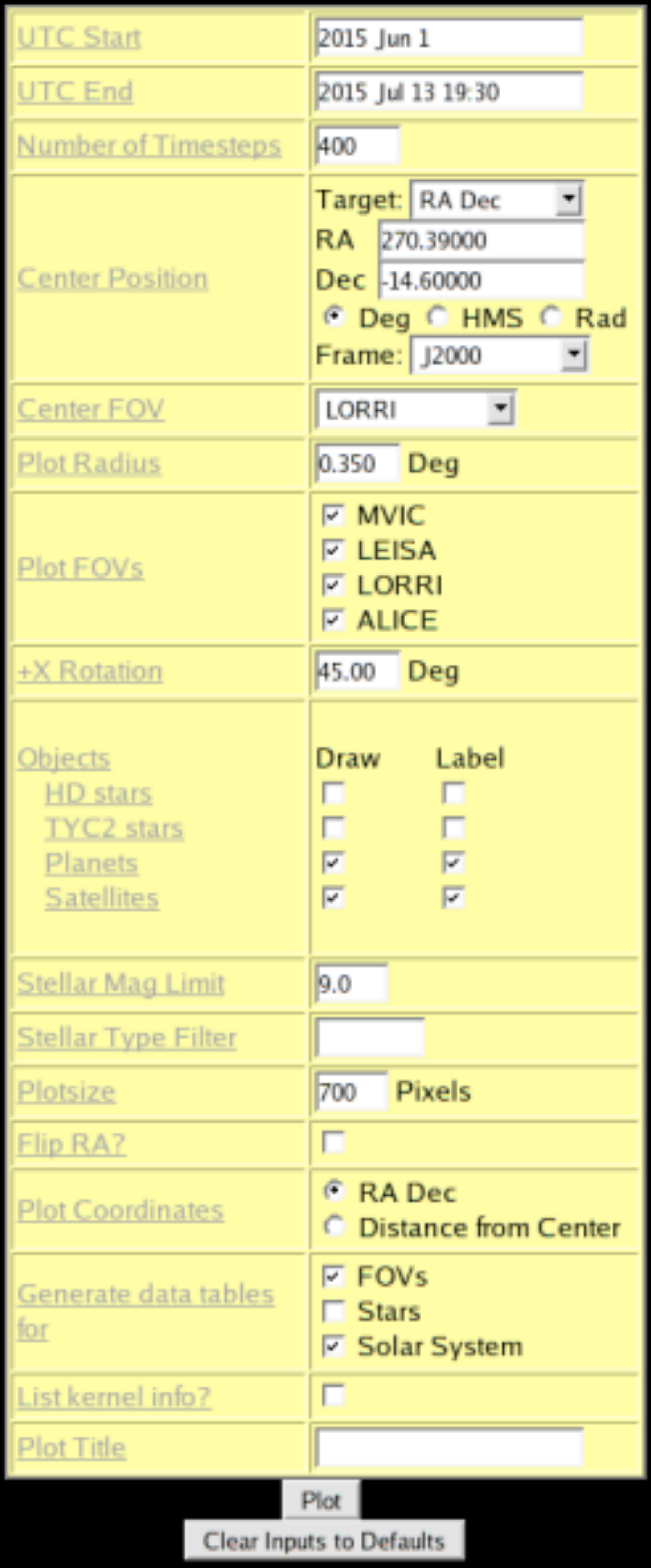
Stereographic projection



Multiple stereographic projections



Download as JPEG



UTC Start	2015 Jun 1
UTC End	2015 Jul 13 19:30
Number of Timesteps	400
Center Position	Target: RA Dec RA 270.39000 Dec -14.60000 ● Deg ○ HMS ○ Rad Frame: J2000
Center FOV	LORRI
Plot Radius	0.350 Deg
Plot FOVs	☑ MVIC ☑ LEISA ☑ LORRI ☑ ALICE
+X Rotation	45.00 Deg
Objects	Draw Label
HD stars	□ □
TYC2 stars	□ □
Planets	☑ ☑
Satellites	☑ ☑
Stellar Mag Limit	9.0
Stellar Type Filter	
Plotsize	700 Pixels
Flip RA?	□
Plot Coordinates	● RA Dec ○ Distance from Center
Generate data tables for	☑ FOVs □ Stars ☑ Solar System
List kernel info?	□
Plot Title	

Things that Worked

- SPICE
    - Robust and powerful; forms the basis of all of GV's computations.
- Calling IDL from Web, using PHP interface
- Unix shell utilities and external functions
    - File management
    - Star catalog searches
    - Movie encoding
- IDL: Rapid development, etc.

Things that didn’t work as easily...

- Direct Graphics: should use Object Graphics!
  - Higher quality
  - Much faster 3D rendering than direct graphics.
  - More flexible text labels.
- Direct graphics has bugs, quirks.
  - Polyfill works differently on Mac, Unix
  - [1,1] pixel works differently on X, Zbuffer
  - Clipping works differently on X, Zbuffer
- IDL’s map routines support Earth’s longitude system but not other planets’
- IDLDE (Workbench) is nice but no easy integration with VIM editor

Onward...

http://soc.boulder.swri.edu/gv