The Development and Future of Python at STScI

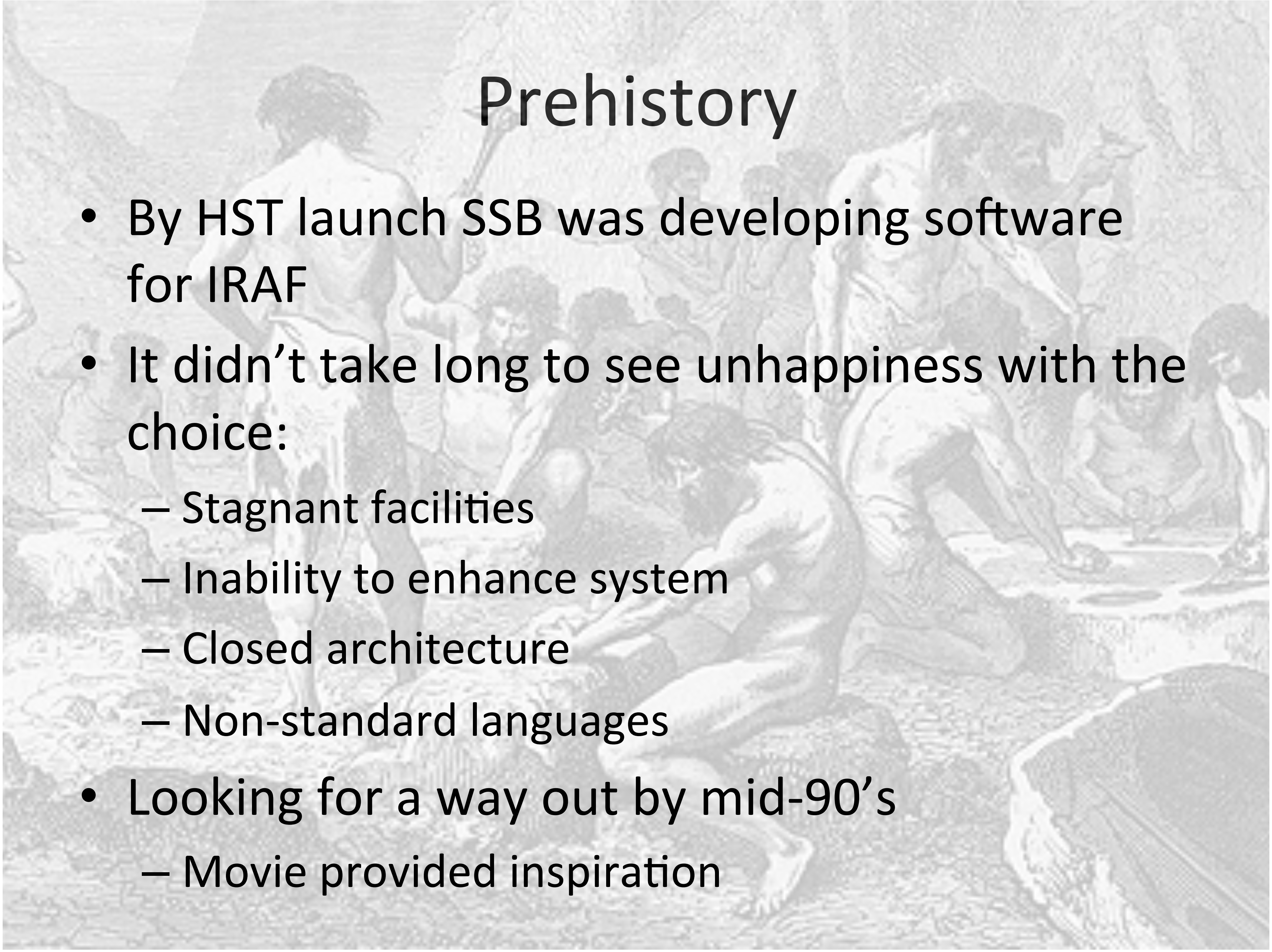
Perry Greenfield

Science Software Branch

Lorentz Workshop 2015

Outline

- Pre-Python History
- Python History
- Role for JWST
- Current work
- Challenges



Prehistory

- By HST launch SSB was developing software for IRAF
- It didn’t take long to see unhappiness with the choice:
  - Stagnant facilities
  - Inability to enhance system
  - Closed architecture
  - Non-standard languages
- Looking for a way out by mid-90’s
  - Movie provided inspiration

Foot in the door

- PyRAF: Sold as a better IRAF CL
- Python at the time was the only reasonable choice for a scripting language
- The big surprise was that it was a fantastic development language too. We wanted to write everything in it.
- But that meant lots of basic tools were needed to allow that

But Really, Why Python?

- Gives us an excuse to watch Monty Python and purchase related paraphernalia
- It's fun to use
- Excuses use of silly names
- Escape from IRAF

Slightly more seriously

- Escape from IRAF
- Out-IDL IDL
- Unify developer and astronomer languages
- Leverage much broader efforts available
- More satisfaction developing
  - increase “moxie factor”
- Avoids Møøse bites

Basic Foundations Needed

- Better array tools
  - Numeric at that time, but not good enough
  - → numarray → numpy
- Efficient and powerful way of reading and writing FITS files.
  - No complete tool available
  - Adapted and expanded Paul Barrett's pyfits
- Visualization
  - the situation was a mess
  - → chaco → matplotlib

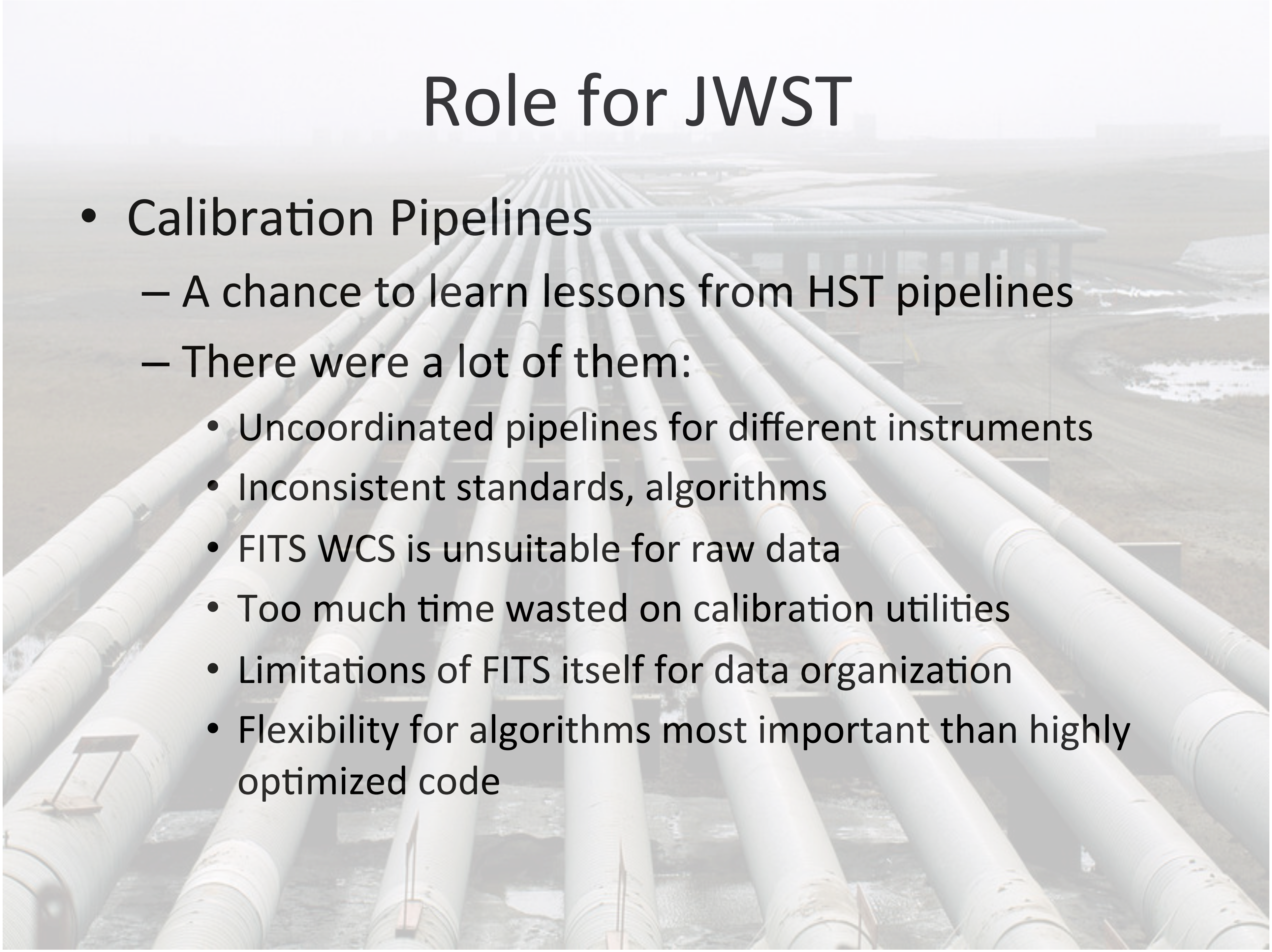
Initial use of Python

- HST pipelines and associated software
  - CALCOS
  - Multidrizzle/astrodrizzle
  - Pysynphot
- Rewrote Java-based ETCs in Python
  - Though not just for the hell of it

Data Analysis demands more

- Operations uses are comparatively narrow
- General data analysis requires much broader toolset that is easier for astronomers to use.
- Large numbers of existing tools for IRAF without corresponding tools in Python.
- Tools need to be broader than just what HST and JWST need.
- HST never really had resources to fund this
- JWST now comes into the picture.

Role for JWST



- Calibration Pipelines
  - A chance to learn lessons from HST pipelines
  - There were a lot of them:
    - Uncoordinated pipelines for different instruments
    - Inconsistent standards, algorithms
    - FITS WCS is unsuitable for raw data
    - Too much time wasted on calibration utilities
    - Limitations of FITS itself for data organization
    - Flexibility for algorithms most important than highly optimized code

Role for JWST (continued)

- Data Analysis:
  - – Needed tools to support new modes HST didn’t have:
    - MOS and IFU spectroscopy
  - – Launch delay provided an opportunity for more systematic attempt to provide a more complete suite of data analysis tools
    - Did we really want to depend on IRAF to the end of the mission?
    - i.e., we need to replace most useful IRAF capabilities

The Good

- Request to fund “IRAF replacement” well funded.
- Significant resources added (~ 5 people over 5 years or so)
- Very unusual to see this kind of support for Data Analysis work.

But…

The Bad

- Such resources are vulnerable to:
  - Priority demands
  - Political needs
  - Budget problems
  - Decreasing HST support
- Potentially worse:
  - Process entanglement

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still have to delete the image and then insert it again.

Digression: what works and what doesn’t

- Successful (IMHO):
  - IRAF/STSDAS/TABLES
  - IDL/astrolib
  - AIPS
  - Sherpa
  - MIDAS
  - PyRAF, pyfits, numarray/numpy, matplotlib, astropy
  - None of these had a heavy formal “Program Management” development process (at least to start). Some had virtually none at all.

Digression continued

- Unsuccessful (circumspection requires that I be somewhat vague.)
  - Early HST DA efforts (pre-launch); all tossed out.
  - [redacted] ++
  - [redacted] Data Analysis project
  - Most US [redacted] -related DA projects
- All these had heavy Program Management processes
  - No doubt there are other factors, and
  - It may just be a coincidence, but nevertheless…

Astropy born in 2011

JWST Data Analysis plans

General issues:

- Significant overlap with Calibration Pipeline needs
- Make it generic as possible
  - – If not able to fill in all needs, make it as easy as possible for others to do so.
    - E.g., supply templates and examples of how to do so
- Use Astropy as the focus (either in core or as affiliated packages)

JWST Plans: Core capabilities

- More useful WCS library
  - FITS standard is a very inflexible model
  - Essentially unusable for distorted data
  - Aids increasing desire to avoid resampling
- A more useful data format than FITS (ASDF)
  - We need this to store the new WCS info anyway
  - Easy to generalize to cover most FITS use cases.
- Modeling/Fitting
  - Useful in many contexts for JWST

JWST Plans: Core tools

Supporting (among other things):

- Spectral cube visualization and analysis
- Multi-Object Spectroscopy visualization and analysis
- Spectral model fitting
- PSF matching and related PSF tools
- Coronagraph reduction tools
- Image Utilities
- Source identification and characterization tools

Philosophy

- Replace important IRAF functionality, but not necessarily the same approach.
- Rather than large black-box tasks, a layered approach of useful low, medium, and high-level libraries and tasks.
- Leverage existing functionality in scipy and other existing packages
  - But may need to wrap to present a consistent and convenient interface for astronomers.

Dealing with Interactive Use

- Important use case, but
- GUIs are very, very expensive to develop
  - Complex ones can be 10X more work than command line equivalent functionality
  - Very easy to waste a lot of resources with mistakes in design choices
  - Early and fast progress gives a false impression of the total amount of work required
    - Effort grows very nonlinearly
  - Basic design and technical decisions here have long-standing consequences.
  - One of my biggest worries.
  - Need to be very tough about allowing feature growth
    - Rely on plug-ins for extending functionality
- While comparatively primitive, IRAF dealt with this issue in a very smart and economical way.
  - Possibly the fallback approach.

Basic GUI questions

- What underlying technology to use?
- Web-based vs desktop GUI?
- Web pros:
  - Avoids installation issues regarding GUI framework
  - Much of GUI development moving that way
  - Makes remote use very easy (e.g., server side computations)
- Web cons:
  - Implies lots of development in Javascript
  - Can one really wall off what needs to be in JS?
    - Is there a slippery slope of re-implementing many tools already in Python?
  - Raises potential performance issues.

Basic GUI questions (cont.)

- Desktop GUI pros:
  - No need to use Javascript
  - Avoids potential Python/Javascript interface and performance issues
  - Existing tools mostly available as desktop GUIs
- Desktop GUI cons:
  - Frameworks can be difficult installs
  - Dwindling interest in general?
- Decision: Desktop GUI for now
  - Easiest way to show lots of initial progress
  - Binary packagers solve most of the installation issues (e.g., Anaconda)
  - Users need to install Python anyway, so in this case web interface doesn’t avoid users having to install software.

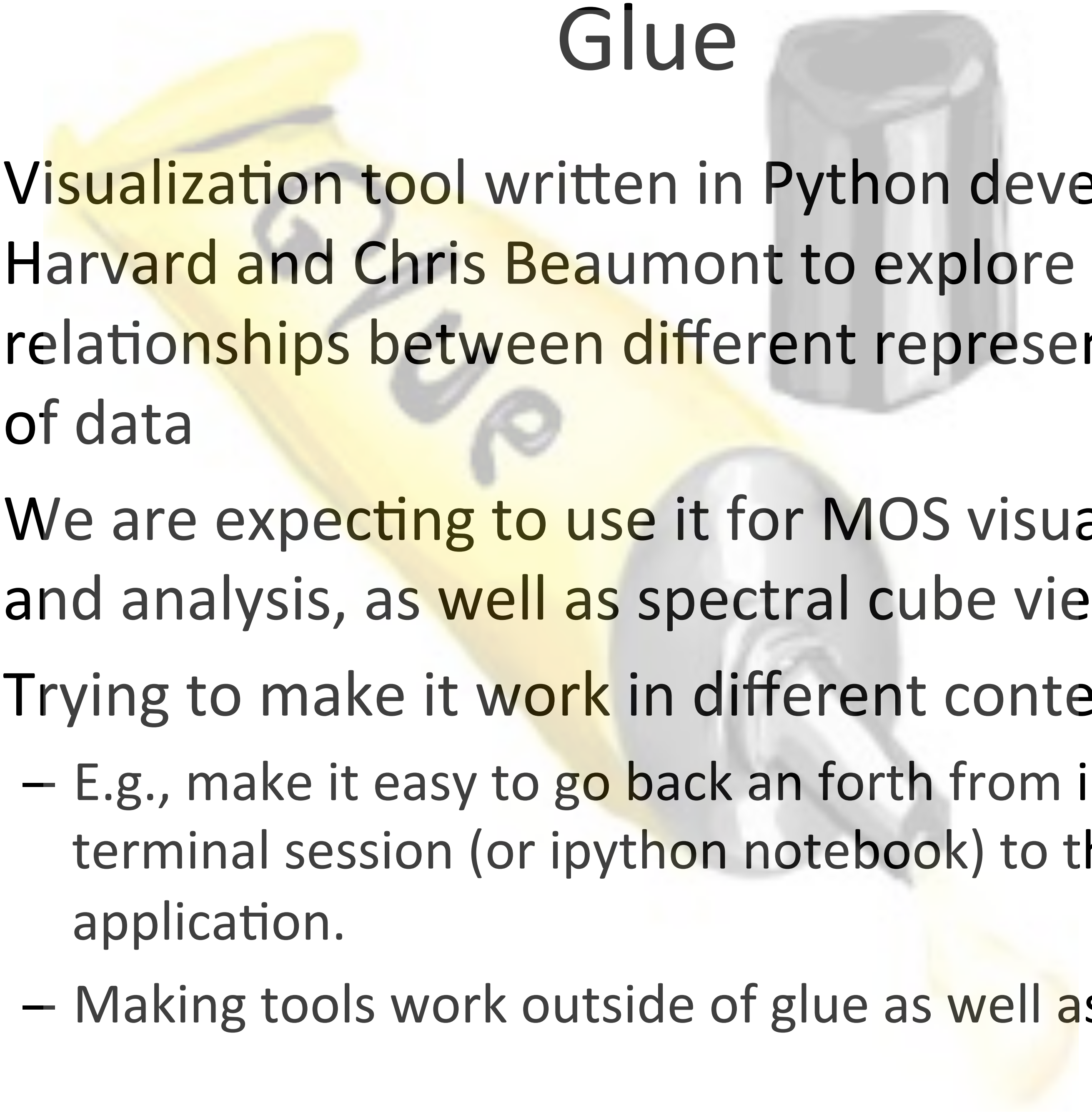
Desktop GUI issues

- Qt is the obvious choice (for now)
- Modern interface with many features and tools
- Cross platform (supports all common OS's)
- Used by the tools we are most interested in adopting, namely:
  - Glue
  - Ginga
- Reusing these should save a lot of effort. Both are written in Python

Ginga

- Basic, responsive Image viewer developed by Erik Jescke at Suburu.
- Well set up to handle plug-ins for customized applications
- Can be used within Glue!

Glue

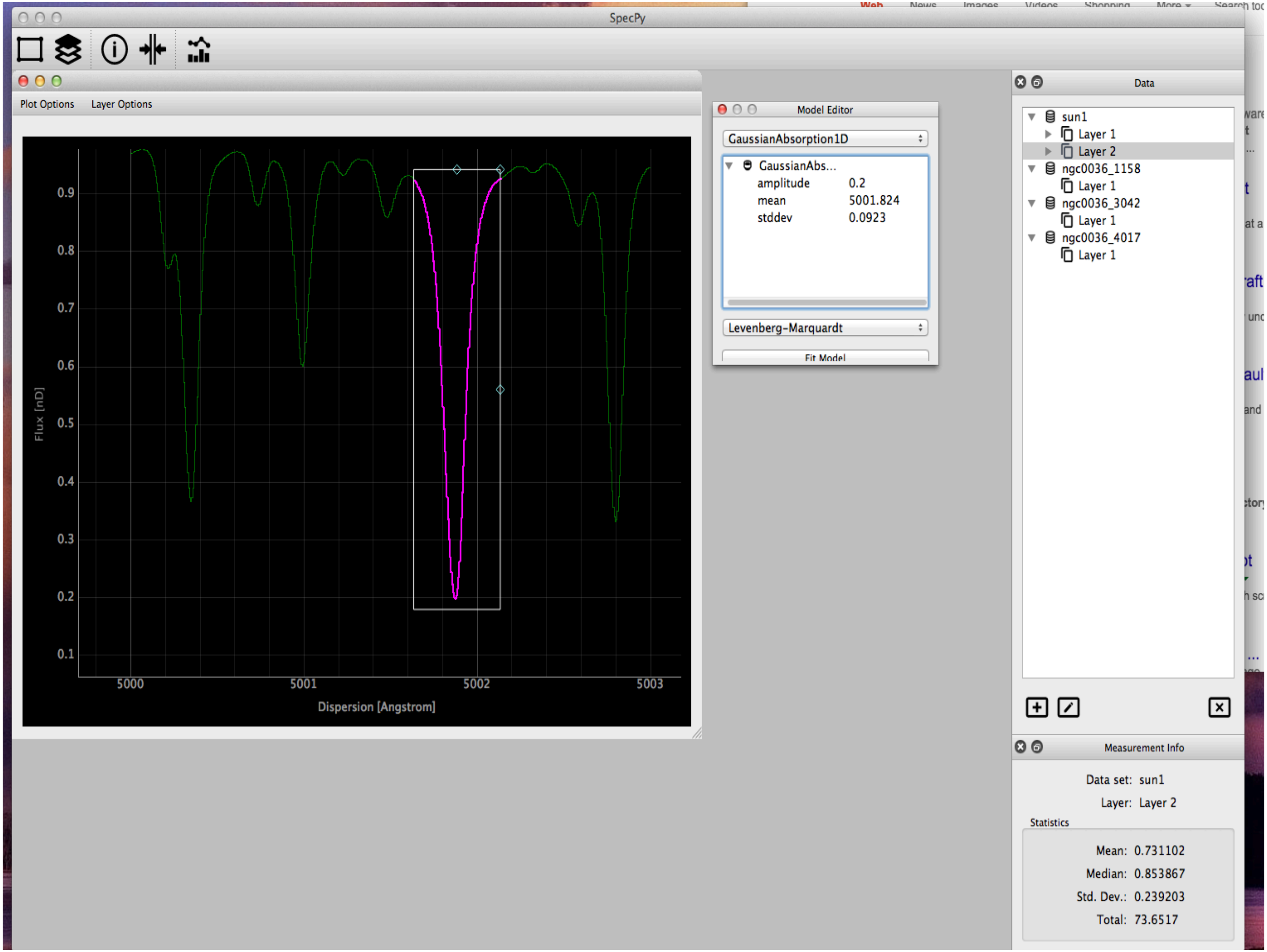


- Visualization tool written in Python developed by Harvard and Chris Beaumont to explore relationships between different representations of data
- We are expecting to use it for MOS visualization and analysis, as well as spectral cube viewing.
- Trying to make it work in different contexts
  - E.g., make it easy to go back an forth from interactive terminal session (or ipython notebook) to the application.
  - Making tools work outside of glue as well as within.

Interactive modeling

Currently working on splot kind of functionality.

- Display multiple overlaid 1-d spectra, model fits, residuals, etc
- Interactions include selecting regions, setting initial parameters (e.g., lines to be fit), etc.
- Separate but coordinating widget to interact with models
  - changing definitions of models
  - adding constraints
    - Upper, lower bounds
    - Equations relating to other parameters, e.g., fixed wavelength ratios.
  - fixing or unfixing parameters
- Keep the basics generic enough to use in other contexts
  - E.g., modeling widget for use in image fitting or WCS fitting
  - 1-d visualization for time series, etc



GaussianAbsorption1D

GaussianAbs...

amplitude	0.2
mean	5001.824
stddev	0.0923

Levenberg-Marquardt

- sun1
  - Layer 1
  - Layer 2
- ngc0036_1158
  - Layer 1
- ngc0036_3042
  - Layer 1
- ngc0036_4017
  - Layer 1

Data set: sun1

Layer: Layer 2

Statistics

Mean:	0.731102
Median:	0.853867
Std. Dev.:	0.239203
Total:	73.6517

Engaging Users in Development

- STScI using "sprints" to try to engage staff astronomers in helping define data analysis tools.
- 3 2-week sprints held so far with 2 per month or so planned.
- Using Trello as a means of floating tasks and features that should be considered as part of a sprint.
- Not sprints in the usual software sense, but still very useful to get feedback
  - Essentially commits an astronomer to be available for helping prioritize features and define user interfaces.
- Not clear if it can be sustained.

Engaging Users (cont.)

- I think it is necessary to involve individuals outside STScI to make this work.
- In other words, make user involvement astropy-wide.
- We welcome others that are interested in providing input on priorities, use cases, user testing (and even code!)
- What is the best way of interacting with the community on when we want info?
  - Mailing list?

Random Observations

- Ignorance is good (in some contexts)
- Do not over-analyze or over-design software
  - Much better to implement and redesign than try to get everything right the first time.
- Worse is better (!?) Google it
- Don't force Python on those not open to trying it!