Is My Code Good Enough?
Improving software through code review.

Mike Jarvis
February 5, 2018
LSST DESC DE School

How do you know if your code is good?

How do you know if your code is good?

1. **Try it out**
   - Run it on different kinds of data.
   - Validate it on runs with simulated data.
2. **Tests**
   - Unit tests check each component for accuracy, reliability, etc.
   - Integration tests to check that parts work together well.
3. **Code Review**

What is a code review?

- A developer submits code changes for review.
- One or more reviewers read the code and comment, typically asking for modifications.
- The developer responds either by changing the code appropriately or replying why those suggestions are not appropriate.
- Once the reviewers are satisfied, the code is merged into the main line (e.g. master branch).
- GitHub makes all of these steps very easy.

GitHub, Inc. [US] https://github.com/GalSim-developers/GalSim/pulls

This repository Search Pull requests Issues Marketplace Explore

GalSim-developers / GalSim

Unwatch 45 Unstar 76 Fork 48

Code Issues 58 Pull requests 3 Projects 0 Wiki Insights Settings

Label issues and pull requests for new contributors Dismiss

Now, GitHub will help potential first-time contributors discover issues labeled with help wanted or good first issue

Go to Labels

Filters is:pr is:open Labels Milestones New pull request

3 Open 316 Closed Author Labels Projects Milestones Reviews Assignee Sort

VonKarman profile
#940 opened 11 days ago by jmeyers314 • Review required

#809f Define GSObject classes primarily in python with C++ layer code as an implementation detail. 3
#931 opened on Dec 29, 2017 by rmjarvis • Review required

#770 Script to create AEGIS catalog 37
#806 opened on Sep 26, 2016 by sowmyakth • Changes requested

ProTip! Type g p on any issue or pull request to go back to the pull request listing page.

LiveSlides.dmg VonKarman profile by ....htm Show All

GalSim-developers / GalSim

Unwatch 45 | Unstar 76 | Fork 48

Code | Issues 58 | Pull requests 3 | Projects 0 | Wiki | Insights | Settings

#822 Adding SED wavelength sampling #826

Edit

Merged **rmjarvis** merged 12 commits into `master` from `#822` on Nov 11, 2016

Conversation 13 | Commits 12 | Files changed 9 | +290 −0

kadrlica commented on Nov 9, 2016 — Member

Added `SED.SampleWavelength` to randomly sample wavelengths from an SED. Additional changes from Mike to add the `BaseDeviate.generate` function to create arrays of random variates. Some tests for each new functionality.

rmjarvis and others added some commits on Nov 8, 2016

- Add generate function to fill numpy arrays with random values ... 1765954
- Added 'sampleWavelength' to randomly draw wavelengths from an SED (#822) 77e15d1
- Added tests for 'sampleWavelength' (takes a while to run due to slow … ... 30f9f72
- Added notes about sampleWavelength to CHANGELOG ... ba6247a

Reviewers

rmjarvis ✓

rmandelb ✓

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Why do code reviews?

Why do code reviews?

- Helps find errors in the code.
- Encourages developers to be more thorough in the first place.
- Ensures that code base has a fairly consistent style.
- Helps developers learn coding techniques from others on the team.
- Spreads familiarity with the code base. At least 2 people have looked in detail at any given piece of the code.
- **Saves time in the long run**.

What kinds of errors?

Discuss in groups...

What kinds of errors?

Missing tests

- Unit tests of all the new parts of the code
- Tests of typical use case
- Tests of corner cases or unusual situations
- Tests of invalid user input
- Regression tests where appropriate
- Integration tests with other parts of the code
- Validation code to check accuracy of calculations

rmjarvis reviewed on Aug 23, 2016

View changes

tests/test_inclined_exponential.py

Hide outdated

```
+
+        np.testing.assert_array_almost_equal(ratio_core, np.mean(ratio_cor
+                                                err_msg = "Error in compariso
+                                                verbose=True)
```

rmjarvis on Aug 23, 2016 **Owner**

Need to add some more testing aside from just the regression tests though.

1. Need to check the accuracy of the repr, pickling etc. cf. uses of the function `do_pickle` in other test files.
2. Some tests of edge-on inclination == Pi/2.
3. Should test close, but not quite edge-on. Probably on both sides of the magic cut-off you impose on i.
4. Should test that `i==0` is functionally equivalent to an `Exponential`.
5. Check some basic properties like that kValue at k=maxK() is below `maxk_threshold` and that less than `folding_threshold` fraction of the light falls outside of r=Pi/stepK() for a few inclination angles.
6. Sanity checks like kValue(0,0) == flux, centroid() == (0,0), total flux draw with large enough image == flux.

What kinds of errors?

API redesign

- Feels clunky in some typical uses.
- Seems prone to user error from misuse.
- Doesn't match corresponding API of other parts of the code.
- Won't be extensible enough given future development plans.
- Doesn't have the appropriate options for some use cases.

galsim/base.py

Hide outdated

```
+                warnings.warn("Cannot propagate noise through chromatic tr
+        else:
+            jac = np.diag([scale, scale])
+        new_obj = galsim.Transform(self, jac=jac)
```

rmjarvis on Sep 30, 2016 **Owner**

I'm trying to decide whether all these changes to the GSObject transformations are actually a net improvement. It's a lot of duplicated code that is already in ChromaticObject. And AFAICT the thing that it enables is for people to write

```
gal = gal.expand(lambda wave: wave**1.2)
```

rather than

```
gal = ChromaticObject(gal).expand(lambda wave: wave**1.2)
```

So I guess I have a few questions/comments about this.

1. How often do we expect people to want to do this? I'm sure there is some use case, but I would have thought the wavelength-dependent transformation would pretty much just be applied to things that are already chromatic.
2. Is the first syntax any clearer than the second? The second one isn't much more typing, and it seems useful to me to be explicit that you want to treat the galaxy as having a uniform (unit, dimensionless) SED and then expand it as a function of wavelength. So the extra `ChromaticObject` bit seems useful to keep.
3. I'm a little concerned that the former would more often be a bug in the code rather than what the user really wanted, e.g. They might have some function

What kinds of errors?

Failures of tests

- Sometimes errors are very system-specific.
- Travis should catch errors on one of several standard systems.
- Helpful for others on development team to try new code on their own machines to get a variety of OS's, Python versions, gcc versions, numpy versions, etc.
- Even if you don't have time to review the code, this is a useful service to perform for the team.

rmandelb commented on Apr 7, 2017 Owner

Tests pass on my Mac. I'm getting some failures on my Linux cluster (and based on previous issues I went so far as to do a completely clean install of this branch starting from `rm -rf .scon*`, but they persisted).

```
FAIL: test_sensor.test_sensor_wavelengths_and_angles
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/opt/python27/lib/python2.7/site-packages/nose-1.3.0-py2.7.egg/nose/case.py", line
    self.test(*self.arg)
  File "/home/rmandelb/git/GalSim/tests/galsim_test_helpers.py", line 542, in f2
    result = f(*args, **kwargs)
  File "/home/rmandelb/git/GalSim/tests/test_sensor.py", line 395, in test_sensor_wavelengt
    assert r4 > r1
AssertionError:
-------------------- >> begin captured stdout << ---------------------
Starting test_wavelengths_and_angles
Testing Wavelength and Angle sampling - i band
Flux = 3539:                sum        peak          radius
No lamb, no angles:         3503.0     435.00       0.309004
W/ lamb, no angles:         3502.0     435.00       0.308952
No lamb, w/ angles:         3503.0     429.00       0.308881
W/ lamb, w/ angles:         3503.0     408.00       0.320105
check r4 > r1 due to added wavelengths and angles
r1 = 0.309004, r4 = 0.320105, 2*sigma_r = 0.010086
```

What kinds of errors?

Style

- Make sure code is clear. **Readability is paramount.**
- Make sure classes, function names indicate what they do.
- Make sure variable names match style elsewhere in code.
- If style if very bad throughout, suggest using a linter.
- If some section is very hard to read because of style, suggest specific changes.

galsim/compound.py

Hide outdated

```
+    Initialization
+    --------------
+    @param  npoints  Number of point sources to generate.
+    @param  hlr      Half light radius of the distribution of points.  Thi
```

rmjarvis on Oct 26, 2016 **Owner**

In other classes, we spell this out as half_light_radius, so probably should keep to that convention here.

What kinds of errors?

Documentation

- Improve (or add) user documentation of new features.
- Add new features list to the CHANGELOG.
- Give overview of algorithm in in-line comments.
- Reference relevant papers where formulae or algorithms come from.
- If using Sphinx or similar, make sure new docs process correctly.
- Include new features in demos if appropriate.

galsim/phase_psf.py

Hide outdated

```diff
@@ -23,6 +23,9 @@

 where x, y are focal plane coordinates and u, v are pupil plane coordinate

+Alternatively, drawing using photon-shooting can use a fast geometric opti
+Fourier optics.
```

rmandelb on Dec 21, 2016 **Owner**

Be more explicit: this is now the default when photon-shooting such objects.

jmeyers314 on Dec 26, 2016 **Owner**

I rewrote the beginning of this file docstring:

```
Utilities for creating PSFs from phase screens.

For PSFs drawn using real-space or Fourier methods, these utilities essentiall
optics diffraction equation:

PSF(x, y) = int( |FT(aperture(u, v) * exp(i * phase(u, v, x, y, t)))|^2, dt)

where x, y are focal plane coordinates and u, v are pupil plane coordinates.

For PSFs drawn with method='phot', an often significantly faster geometric app
instead.  To use photon-shooting without this approximation, set `geometric_sh
creating the PSF.
```

What kinds of errors?

Inefficient code

- Avoid gratuitously inefficient code, such as loops in Python that are easy to convert to a list comprehension or calculations that can be pulled out of a loop.
- But don't prematurely optimize. If it's not in a "tall pole" section of the code, readability is more important than speed.
- Try to keep the inefficient algorithm in unit tests (since it might be "true by inspection") to compare against a faster algorithm.
- Consider pulling repeated code out into a helper function.

galsim/phase_screens.py

Hide outdated

```
+    coefs = np.array(coefs)
+    coefs[1:] *= np.cumprod(np.full((n,), b, dtype=float))  # powers of b
+    coefs[:-1] *= np.cumprod(np.full((n,), a, dtype=float))[::-1]  # power
+    return coefs
```

rmjarvis on Oct 21, 2016 **Owner**

I'm sure this isn't at all important, but a slightly more efficient algorithm for this (factor of 3 according to some timeit tests I ran) is

```
def generate():
    c = a**n
    yield c
    for i in range(n):
        c *= (n-i)/(i+1) * b / a
        yield c
return np.fromiter(generate(), float, n+1)
```

jmeyers314 on Oct 21, 2016 **Owner**

Always happy to see a clever improvement. Switched to your algorithm.

Further optimize binomial coefficients. ···

✓ f1f0adc

Who should do the review?

Who should do the review?

- Ideally at least two other developers on the same project.
- Anyone who already worked on portions of the code being changed.
- Encourage junior developers to participate as reviewers.
- Users of the code are excellent as reviewers of API design and docs.
- Occasionally bring in outside expertise to review portions of the code. (E.g. DESC architects)

How can you make your code easier to review?

Discuss in groups...

How can you make your code easier to review?

Keep it short.

- The PR should cover a single topic.
- Ideally, a reviewer should be able to fully understand the code changes in a few hours at most.
- For large refactorings, perhaps break up into a few PRs covering one set of related changes at a time.
- In some cases, trivial changes can be pushed directly to master before the PR. E.g. splitting up the contents of a file into multiple files or whitespace edits.

#715 #817

Merged **rmjarvis** merged 26 commits into `master` from `#715` on Oct 31, 2016

Conversation 10 | Commits 26 | Files changed 43 | +212 −91

zunyibrt commented on Oct 19, 2016 — Member

Add support for reading in of unsigned int Images

zunyibrt and others added some commits on Oct 5, 2016

- Add uint16 and uint32 template instantiations to Image.cpp — 3bba8fd
- Add uint16 and uint32 support to pysrc/Image.cpp — 84aef37
- Add uint16 and uint32 support to image.py — 162dcb2
- Add uint16 and uint32 special cases to ImageArith.h — fd18349
- Add uint16/32 template instantiations to Noise.cpp — 62dde2a
- Add uint16/32 templates to SBShapelet.cpp — 3567eb8
- Add more uint32/64 template instantiations to SBShapelet.cpp — 1ba4119
- Add uint16/32 suport to Noise.h — 5aa93b8
- Add uint16/32 support to pysrc/NumpyHelper.h — 1abd069
- Fix a mistake in template instantiation for uint in pysrc/Noise.cpp — e011b78

Reviewers

rmjarvis

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Notifications

How can you make your code easier to review?

Summarize your changes

- Give the reviewers sufficient context about your code changes.
- List the major changes your PR includes
- Point out tricky corner cases that you considered.
- Discuss any algorithmic or API decisions that you struggled with.
- Make sure to reference any open issues that the PR resolves or where design discussions have happened.
- Can write inline comments yourself before reviewers take a look.

#809e Switch from boost::shared_ptr to std::shared_ptr #911

Edit

Merged **rmjarvis** merged 66 commits into `noboost` from `#809e` on Nov 8, 2017

Conversation 17 | Commits 66 | Files changed 176 | +5,899 −5,230

rmjarvis commented on Sep 20, 2017 • edited

Owner

Here is the next installment in the effort to rid GalSim of the boost dependency.

This PR nominally is about switching from `boost::shared_ptr` to either `std::shared_ptr` or `std::tr1::shared_ptr` depending on which is available. At least one of these is available back to GCC 4.1, so I don't think there is any worry about people not having this option.

The other big result from this effort is to get rid of the `NumpyHelper.h` file. This has a lot of complicated wrapping details, especially when we wanted to return a numpy array allocated in C++, that were hard to transfer over to pybind11 (never mind cffi or other wrapping modes). Now everything that used to return a numpy array instead has the numpy array allocated in python and then passed in to C++ to be filled in. This vastly simplifies the wrapping layer code with respect to numpy arrays.

Along the way, i made a few other changes, partly to make that switch easier (in the pysrc code) and partly just as general code cleanup. Some of these also ended up in PR #904 as deprecations.

- Switched from `SBProfile` to `_sbp` as the name of the SBProfile attribute. Mostly so the C++-layer SBProfile object is not part of the public API. This means we don't need to have the SBProfile classes be picklable, which simplifies a lot of the wrapping requirements. All the python-layer classes are still fully picklable, but their `_sbp` attributes are not necessarily so.
- Changed the Image class `image` attribute to `_image` to make it an implementation detail rather than something that users should ever use.
- Rewrote Bounds and Position in python with `_b` and `_p` properties to return the version that is needed for any calls to the C++ layer.
- Similarly, rewrote `GSParams`, `ShapeParams`, and `HSMParams` in python with attributes that are used for C++ calls. This means that the python layer is responsible for the memory handling, which means

Reviewers

jmeyers314 ✓

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Notifications

Unsubscribe

You're receiving notifications because you modified the open/close state.

3 participants

How can you make your code easier to review?

Discuss work while developing

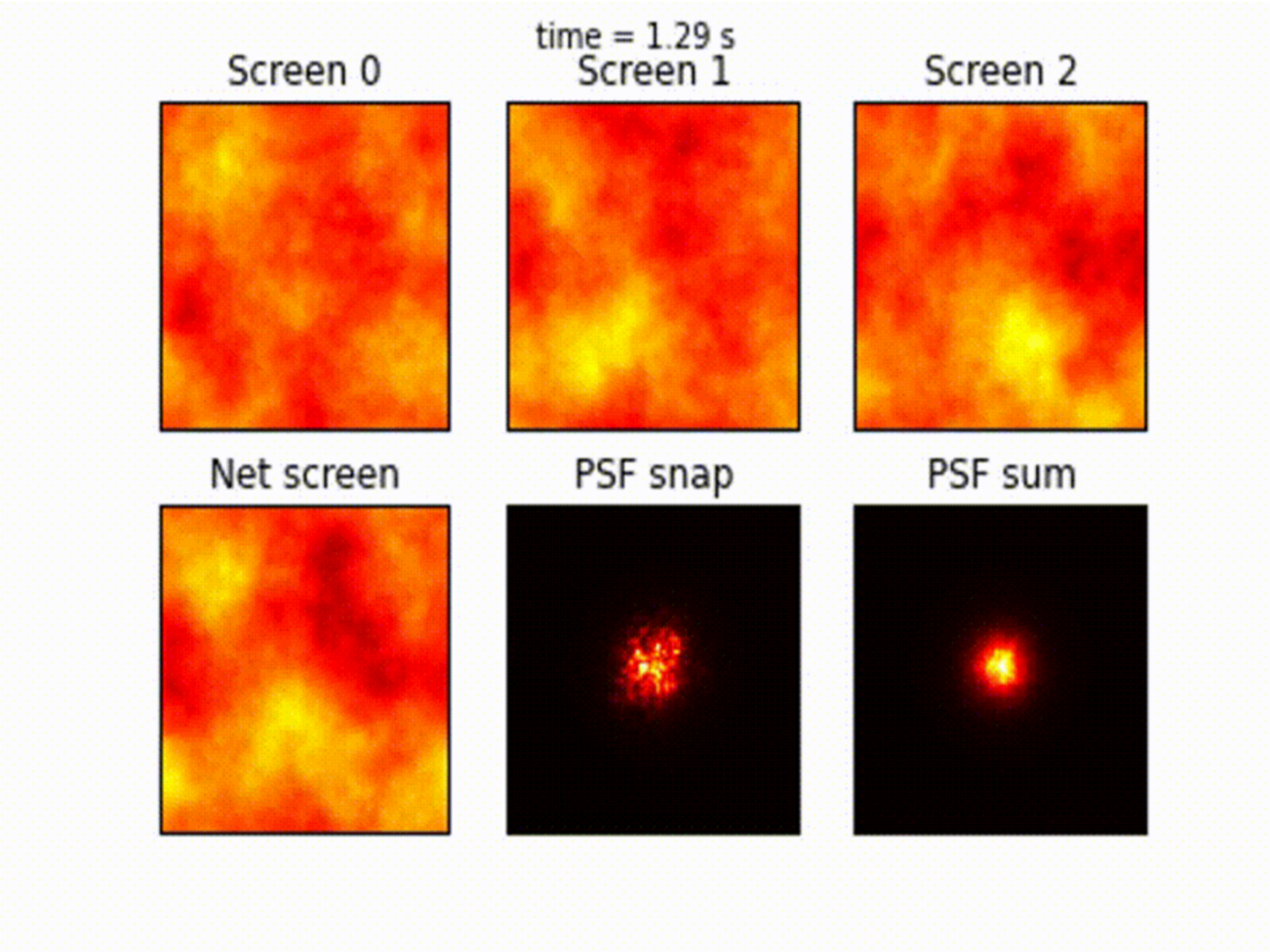
- Don't wait until code review to get feedback on design choices.
- Use the issue page to propose several possible API designs and ask about potential downsides of each.
- Advertise validation results in issue page before code is ready for review.
- Assign as code reviewers the people who participated in these discussions.

jmeyers314 added a commit that referenced this issue on Nov 2, 2015

`keep track of all screens, not just sum` ··· 171d71a

jmeyers314 commented on Nov 2, 2015 Owner

I found the bug that was preventing multiple layers. Here's the new version:



Next step: track down the magic factors of sqrt(0.00058) and such.

Conclusions

- Developing in a team needs a different workflow than when developing solo code.
- Code reviews save time in the long run and improve the quality of code.
- Both reviewing and being reviewed will improve your skillz.
- For further reading and recommendations from CI group, see the LSST DESC Coding Guidelines.