

# Using phosim for star/galaxy studies

**Introducing the users recipe book!**

Phosim Power Users Session  
Debbie Bard  
SLAC National Accelerator Lab



# Plan:

---

- Outline idea of recipe book
- Outline problem
- Documentation web page
- Code and dat files
- Lessons learned

# Recipe book

---

- Starting with phosim examples, will expand to catsim/opsim and perhaps DM stack .
- Aim: make easy-to-follow recipes for running specific tasks.
  - Show users how to do specific things
  - Show potential users the range/power of what we can do with phosim
- Eventual aim: produce libraries of functions ('utensils')
  - Users can stitch these together to create their analyses

# Recipe book

<https://github.com/DarkEnergyScienceCollaboration/ImageSimulationRecipes>

## Image Simulation Recipes

A collection of working example scripts for generating simulated LSST images given basic inputs.



Browse Recipes



View on GitHub



Get in Touch

### Scripting with Image Simulation Recipes

The links above will let you browse the recipe book scripts - but you can also build from them by forking the GitHub repository, importing the utensils and adapting the recipes to your taste.

```
$ git clone ImageSimulationRecipes
$ cd ImageSimulationRecipes
$ python
...
> import utensils
```

Image Simulation Recipes is maintained by various members of the DarkEnergyScienceCollaboration at SLAC National Accelerator Laboratory, including Debbie Bard (@dibard), Dominique Boutigny,

# Star/galaxy separation

---

## recipes.stargal module

---

### Aim:

To enable simple tests of star/galaxy separation algorithms.

### Summary:

To test algorithms for separating stars and galaxies, we will simulate separate images of fields of stars and galaxies using exactly the same atmosphere/optics for each set of images. We are effectively simulating the sky with only the stars, and then the same sky with only the galaxies.

We simulate one chip over 100 different realisations of the atmosphere. No dithering or rotation of the camera is applied - the only thing changing between the 100 realisations is the atmosphere and seeing.

`stargal()`

[\[source\]](#)

# Star/galaxy separation

---

- Basic question: what objects are stars, and which are galaxies?
- Simulate 100 visits twice – once with stars only, once with galaxies only.
  - Same atmosphere etc each time.
- Also simulate same 100 visits using a grid of bright (but not too bright!) stars.
- This is an example of mass-production of images.
- (I also test using galaxies with and without shear)

# Star/galaxy separation

---

- Run SouceExtractor on simulated images.
- Select 'stars' using some test selection algorithm on the resulting catalogues of star/galaxies.
- Interpolate the PSF to the grid points.
- Compare the model PSF to the 'true' psf.
- Average performance over 100 visits to crudely mimic stacking.

# star/gal recipe

---

- <https://github.com/DarkEnergyScienceCollaboration/ImageSimulationRecipes>
- git clone  
<https://github.com/DarkEnergyScienceCollaboration/ImageSimulationRecipes.git>
- cd python/recipes
- Look at stargal.py
- Edit stargal.py to point to your installation of phosim and data files.
- python stargal.py



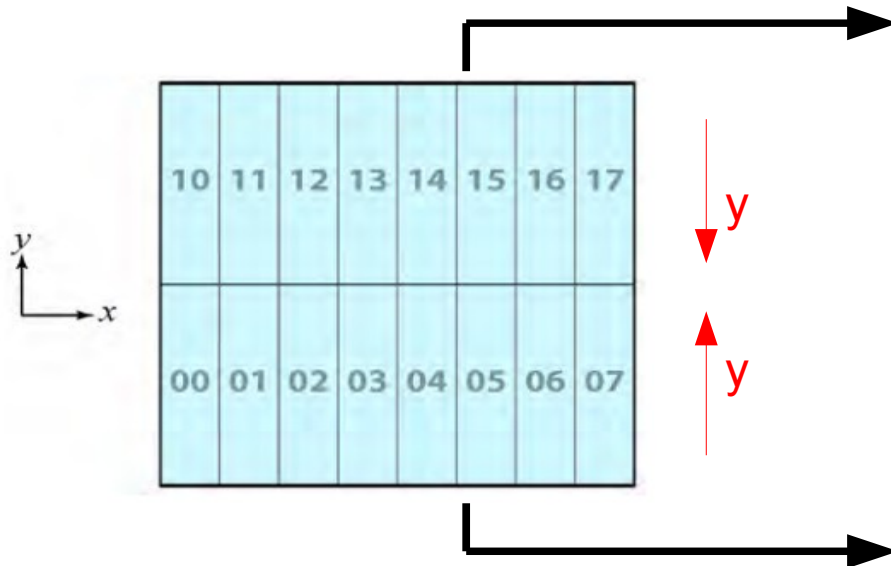
# Lessons learned

---

- This is mass production – 100 atmospheres x 4 (or more) types of objects x however many chips I want to simulate.
- Need to name work/output directories carefully.
- I run on nfs disk space – sometimes overload the disk I/O.
  - Easy solution: don't submit jobs all at once!
- Each chip requires ~350MB working files, so I have to be careful of disk space.
  - I often have >100GB of working files.
- Batch system time limits sometimes exceeded.
  - I'm careful to select which chips I use to avoid too many bright stars (eg no more than 5 stars mag 13-15; none with mag<13) otherwise it takes **forever** to run.
- SLAC pipeline may be useful for me?

# Lessons learned

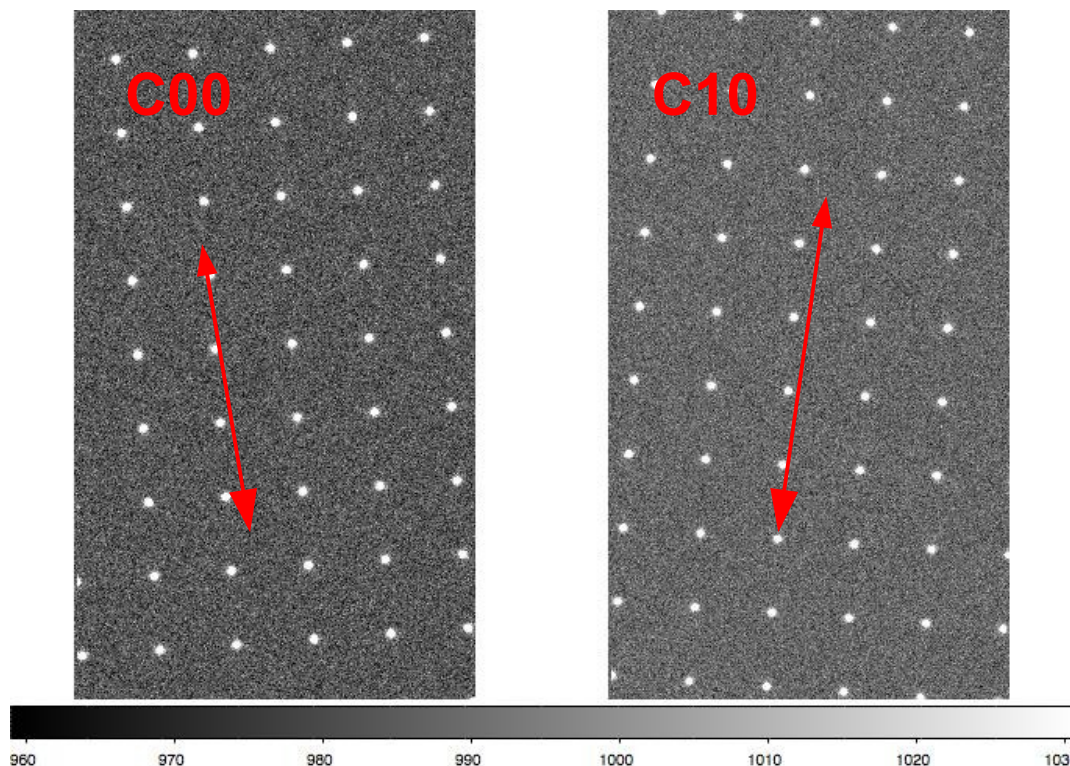
- In processing the resulting images, I ran into problems because I didn't realise the chip coordinate systems are rotated according to which direction the segments are read out



- If you stitch together the chip segments as though pixel-y is always in the same direction, segments 10-17 will be upside down!
- Is this a bug or a feature?

# Lessons learned

- In processing the resulting images, I ran into problems because I didn't realise the chip coordinate systems are rotated according to which direction the segments are read out



- If you stitch together the chip segments as though pixel-y is always in the same direction, segments 10-17 will be upside down!
- Is this a bug or a feature?