targeted spatio-temporal sampling using *raytraverse*

Stephen Wasilewski

Radiance Workshop 2022: Toronto, Canada

HOCHSCHULE LUZERN
EPFL

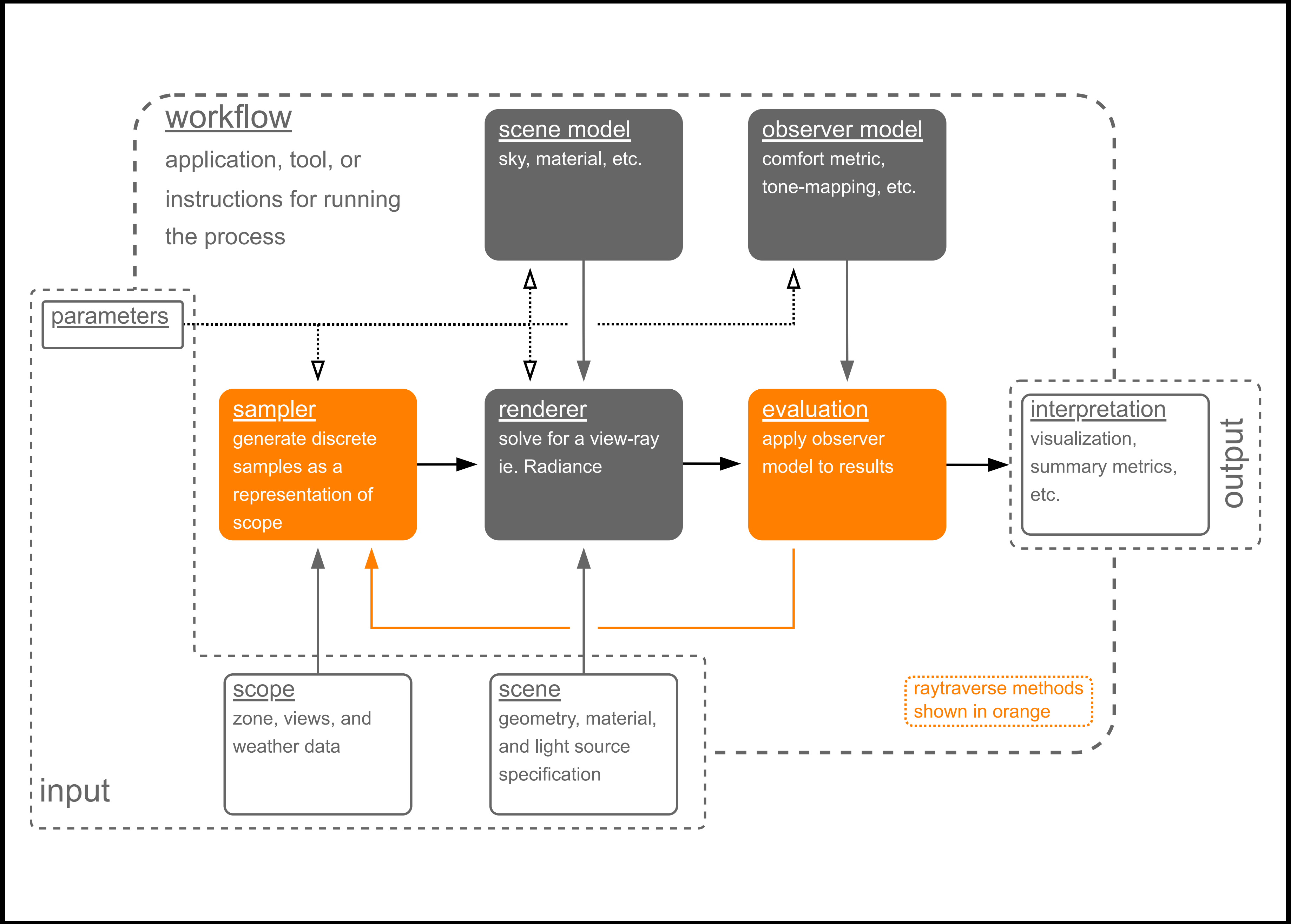
Schedule / Goals

- Introduction (20 min):
  + what is raytraverse?
  + How does it work?
- Exercise #1 (30 min): getting started
- Setting up a Simulation (20 min)
  + input data
  + CBDM methods
- Exercise #2 (30 min): hourly metrics from a point
- ----- Break -----
- Simulation Parameters (10 min)
- basic scripting (20 min)
- Exercise #3 (30 min): sensor area sampling

introduction

What is raytraverse?

method



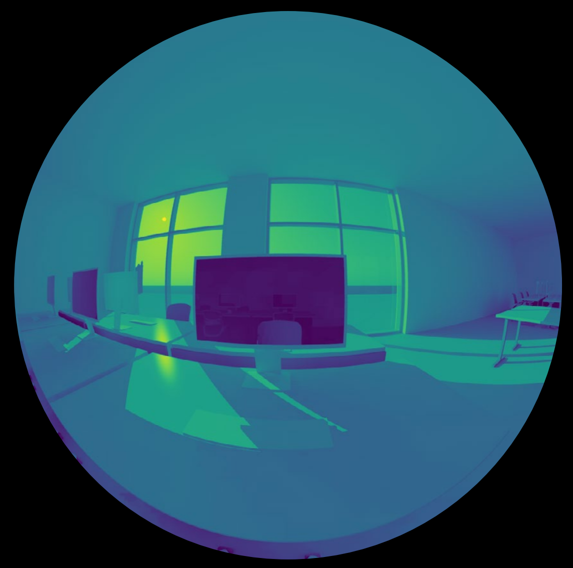
python library

```python
from raytraverse.scene import Scene
from raytraverse.renderer import Rtrace, Rcontrib
from raytraverse.sampler import SamplerPt, SamplerArea
from raytraverse.mapper import ViewMapper, SkyMapper, PlanMapper
from raytraverse.lightpoint import LightPointKD
from raytraverse.lightfield import LightPlaneKD, LightResult
from raytraverse.integrator import Integrator
from raytraverse.evaluate import MetricSet
```

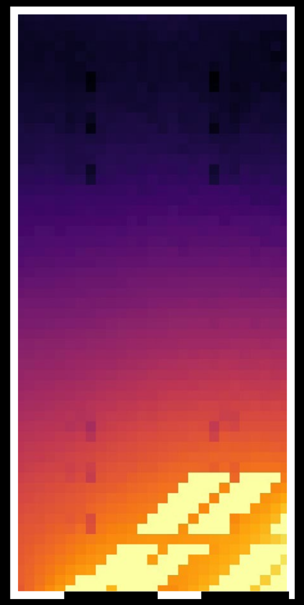
command line tool

```
Usage: raytraverse [OPTIONS] COMMAND1 [ARGS]... [COMMAND2
Commands:
  area          define sampling area
  directskyrun
  evaluate      evaluate metrics
  images        render images
  pull
  scene         define scene files for renderer and output
  skydata       define sky conditions for evaluation
  skyengine     initialize engine for skyrun
  skyrun        run scene under sky for a set of points
  sourceengine  initialize engine for sunrun
  sourcerun     run scene for a single source
  sunengine     initialize engine for sunrun
  sunrun        run scene for a set of suns
  suns          define solar sampling space
```

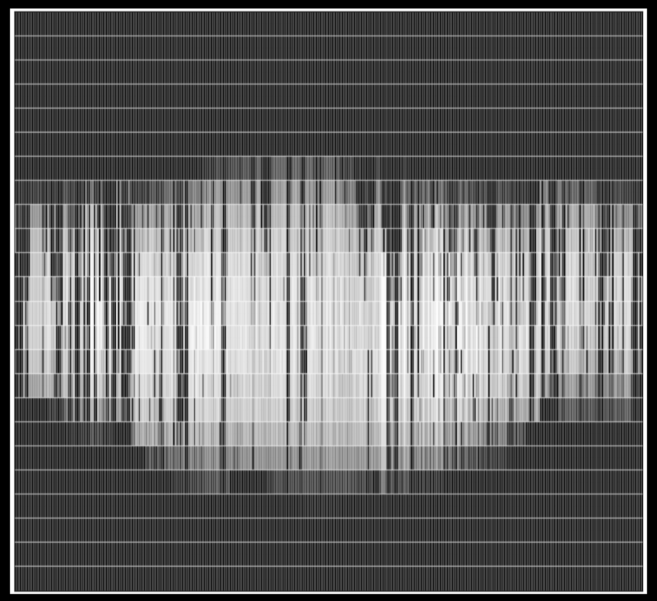
Why is sampling so important?
If one pixel is the volume of a studio apartment....



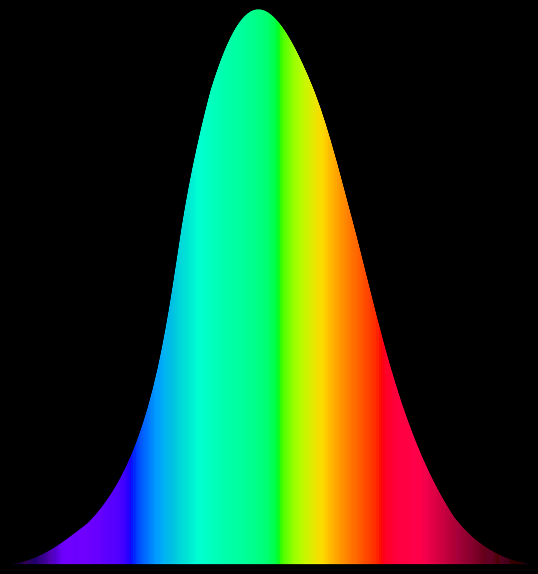
Direction
x 10^6



Position
x 10^2



Time / Source
x 10^3



Spectrum
x 10^1

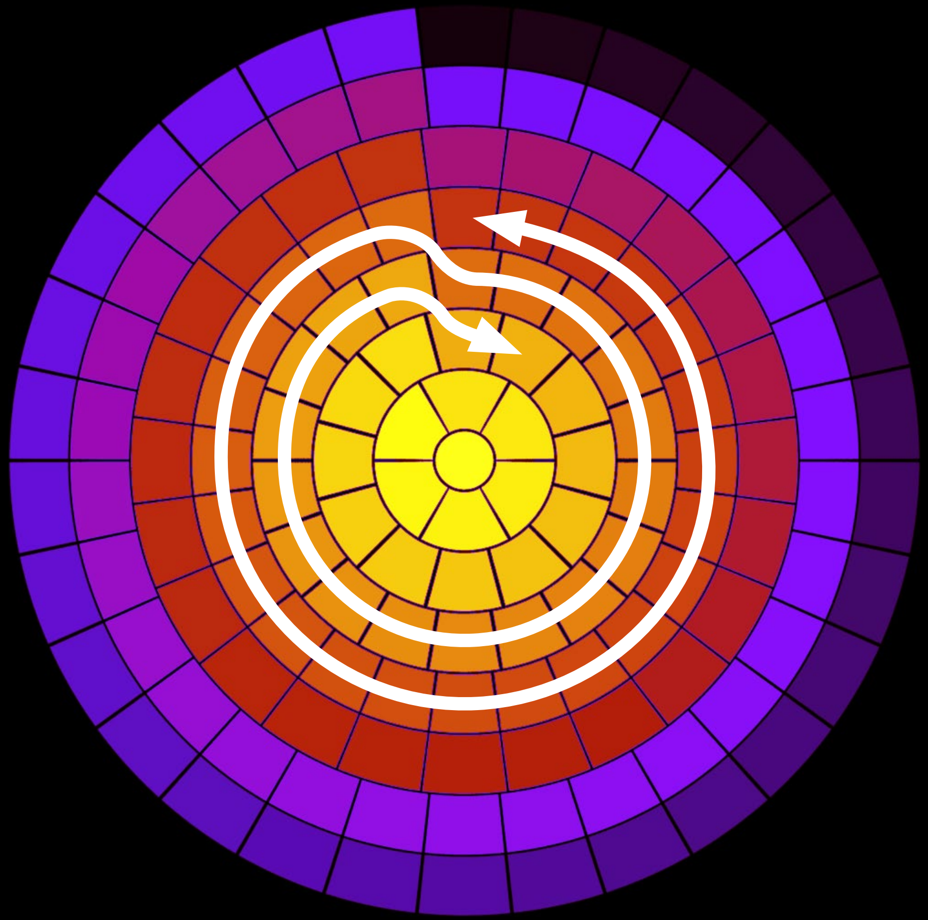
Pluto - Ganymede

Mars - Neptune

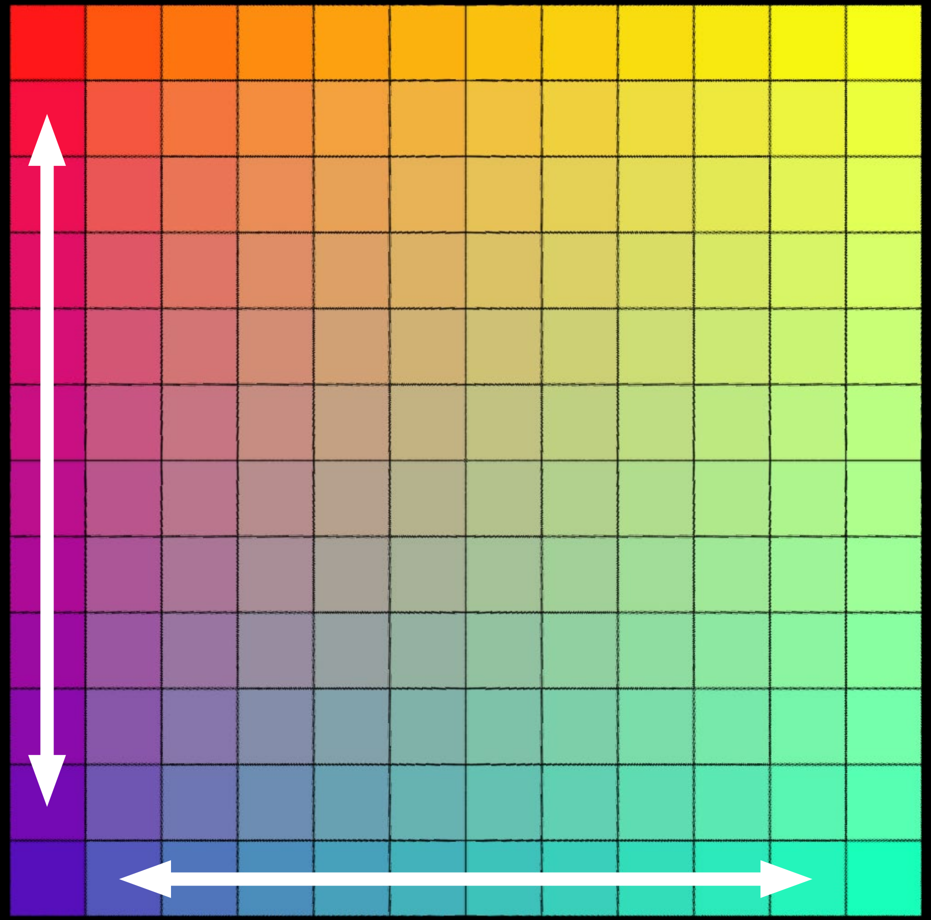
Saturn - Sun

Jupiter - Sun+

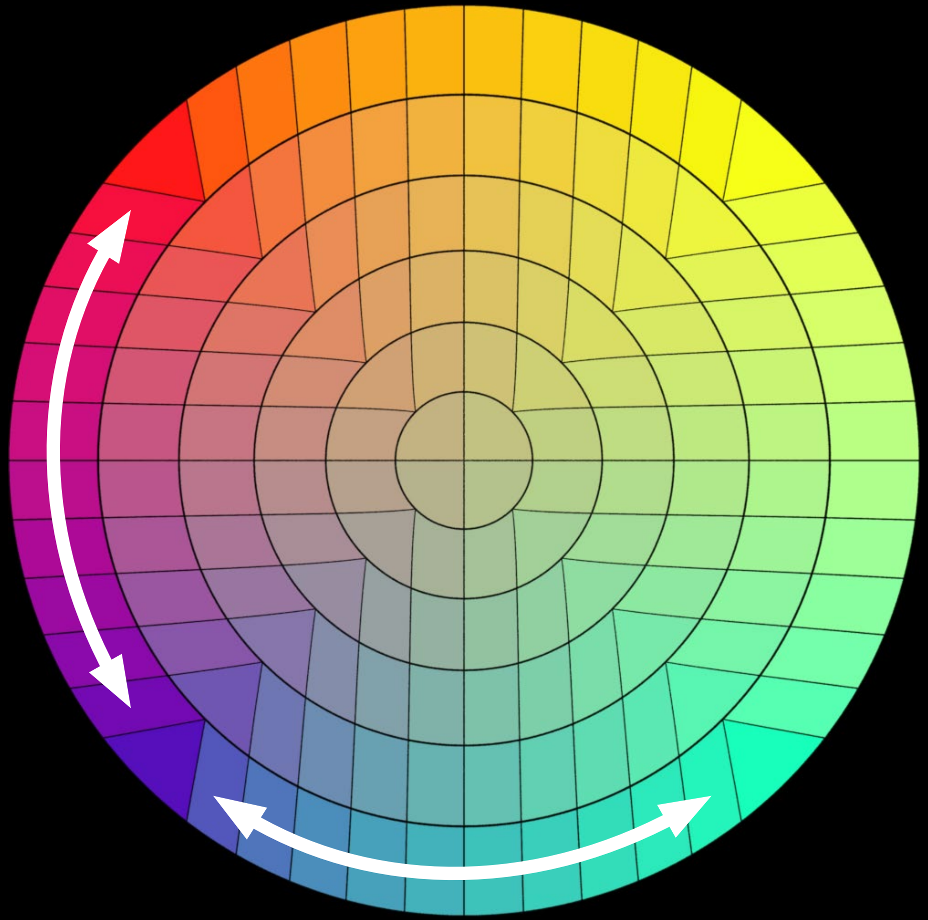
Organizing the Sampling Dimensions to efficiently encode useful variance



Tregenza
(Reinhart and Klems are similar)

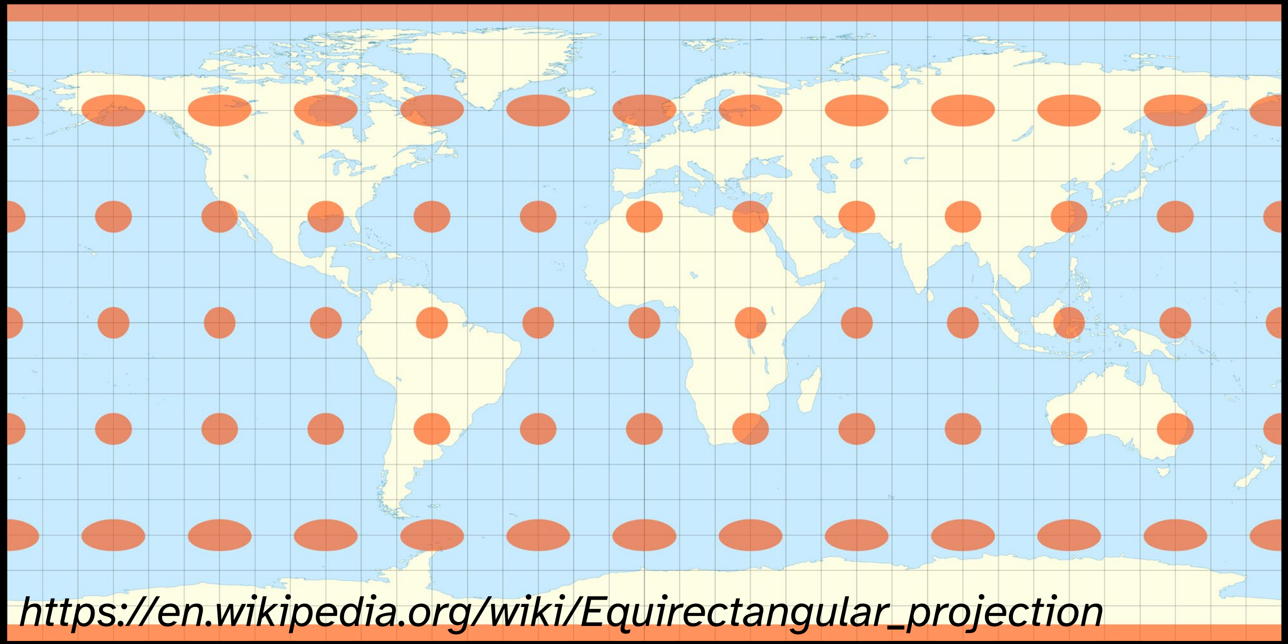


A simple 2D grid

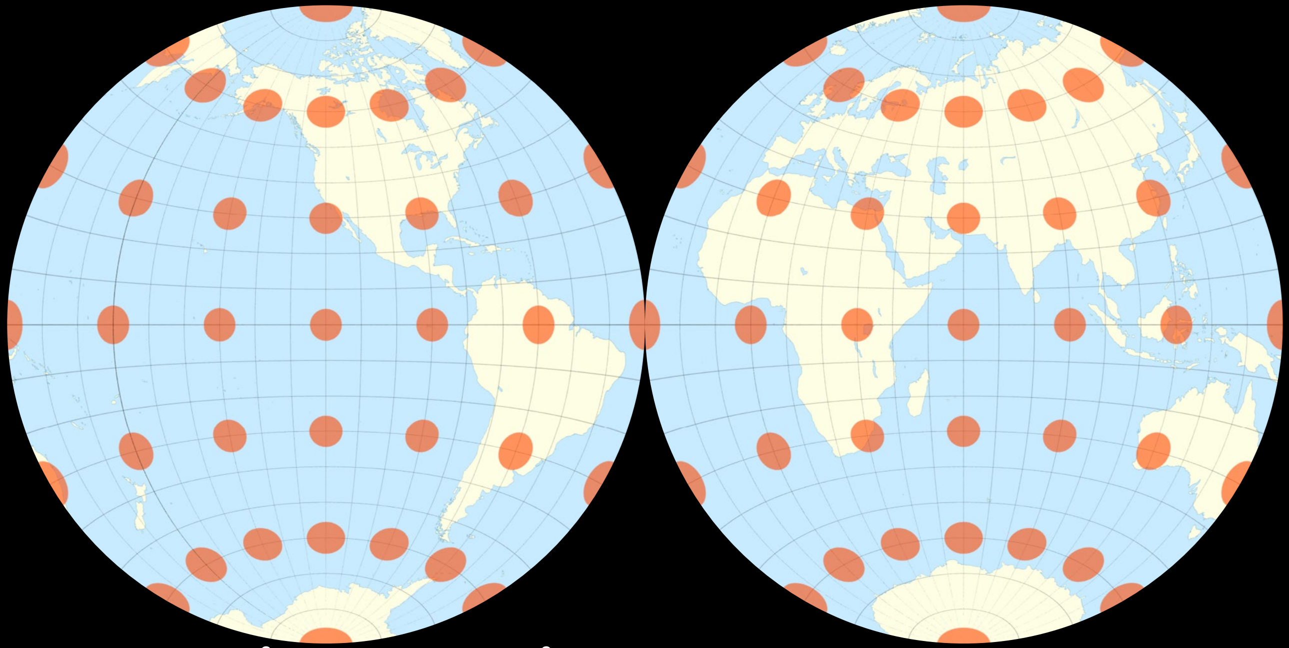


Shirley-Chiu
Low distortion map
between disk and square

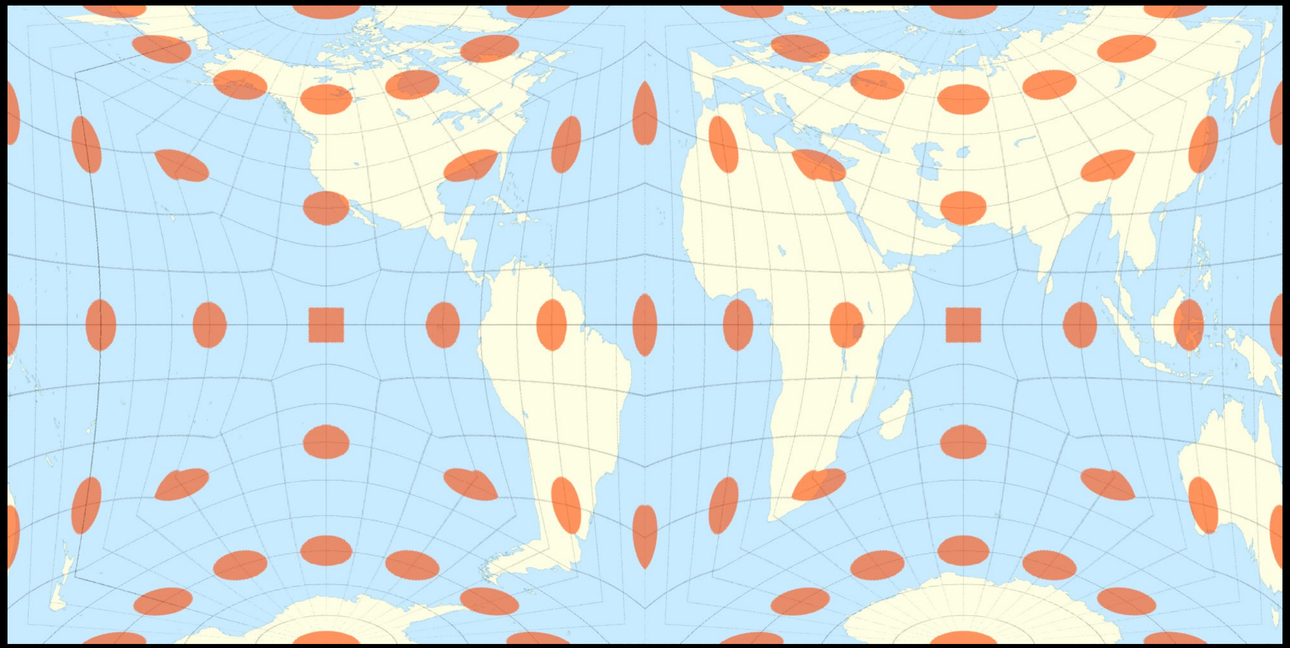
Shirley-Chiu - Useful for Sampling and DWT



~~Preserves fractional area~~
Bi-continuous
Low distortion

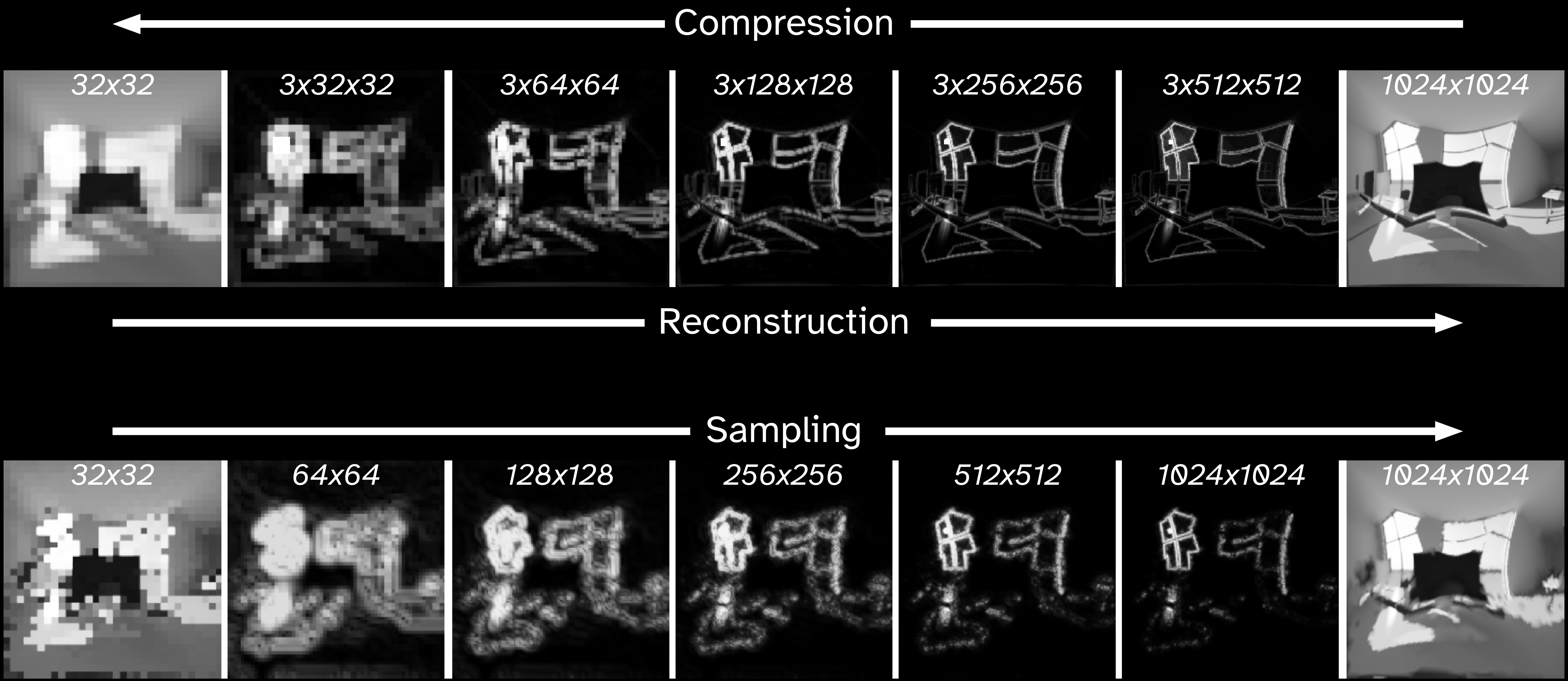


Dual Equiangular Fisheyes
~~Preserves fractional area~~
~~Bi-continuous~~
Low distortion

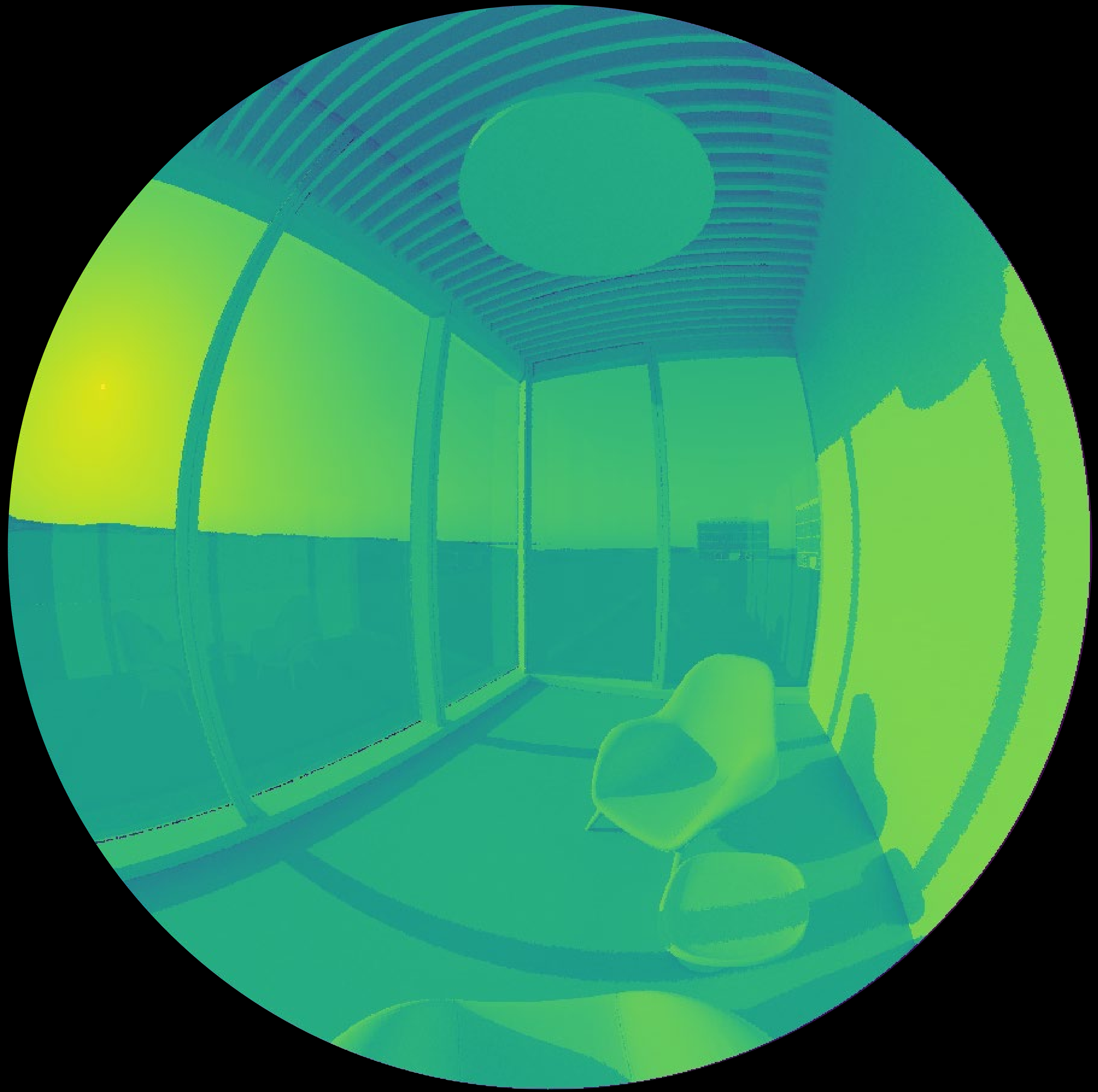


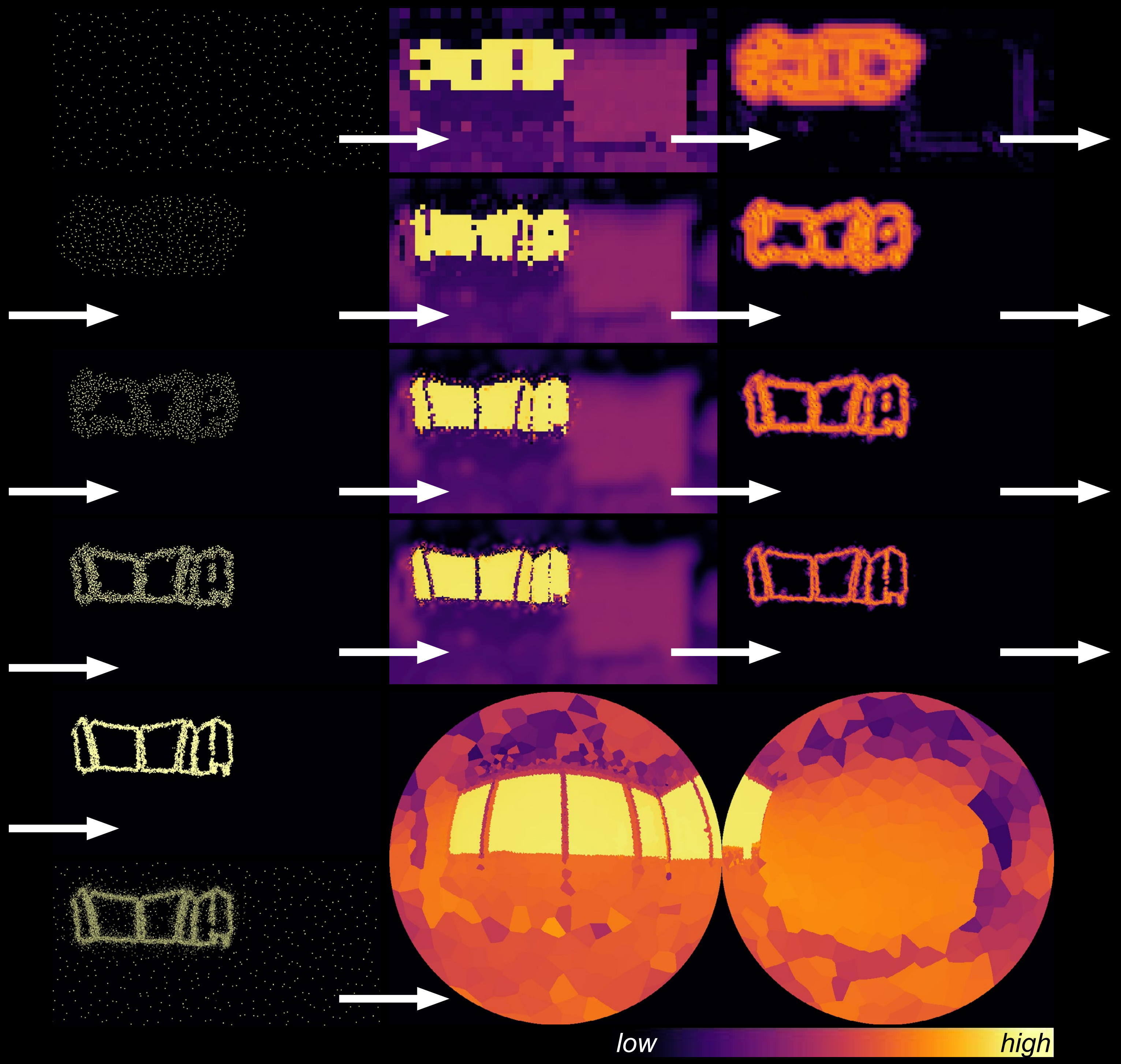
Dual Shirley-Chiu
Preserves fractional area
Bi-continuous
Low distortion

Wavelet Guided Sampling

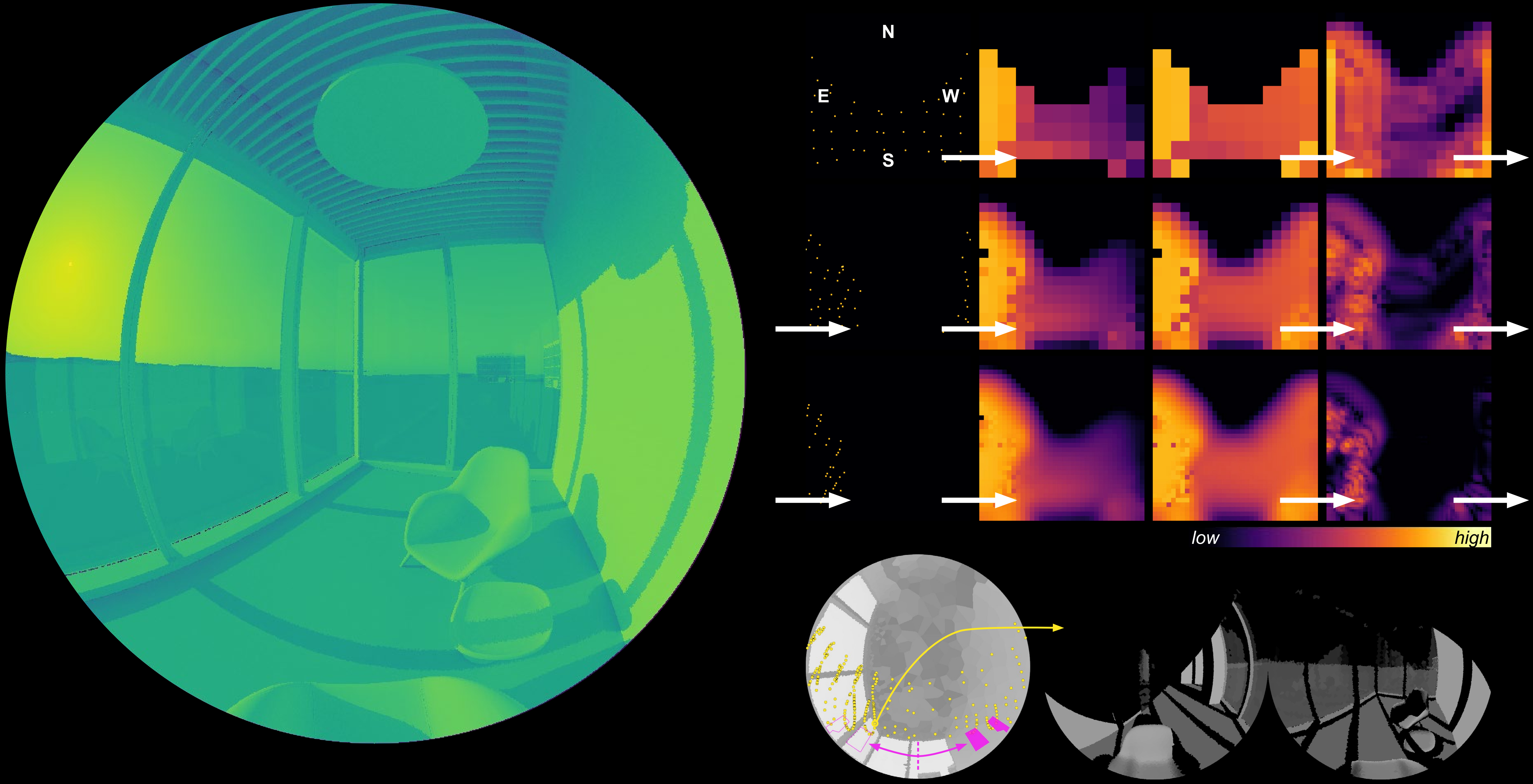


Sampling Directions from a Point

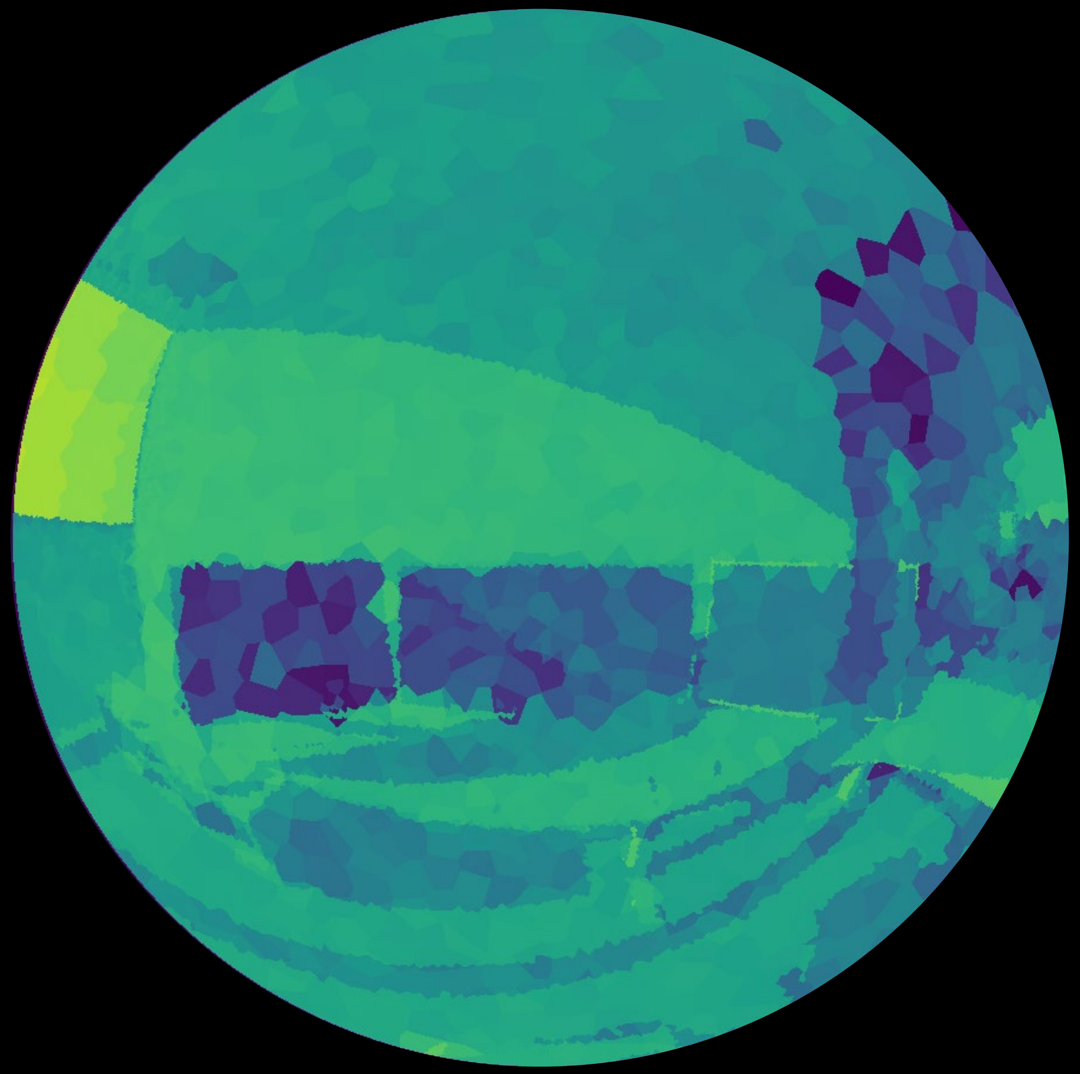




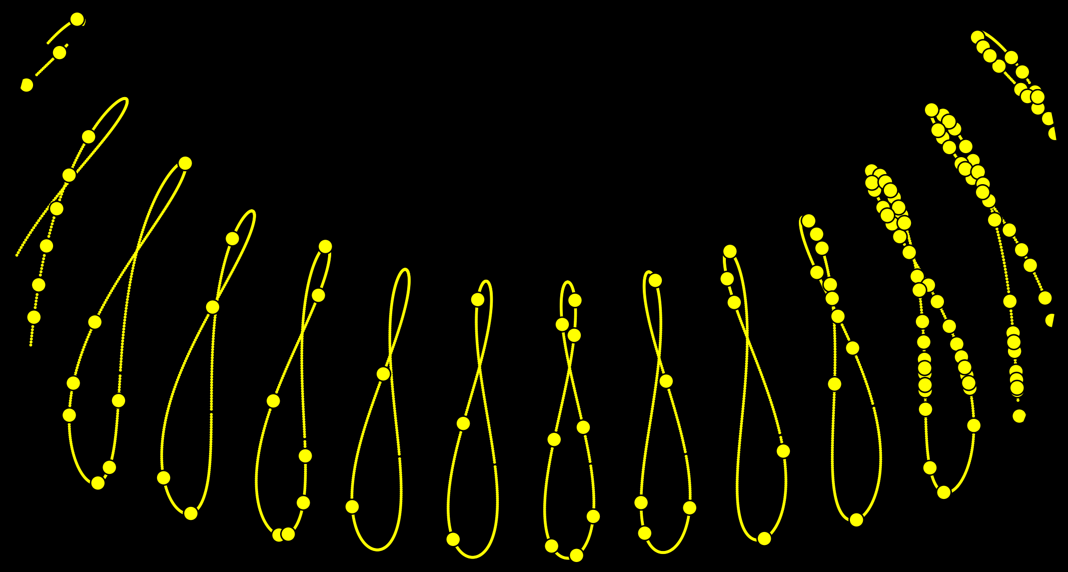
Sampling Sun Positions



What does this get us?
If one ray is the volume of a studio apartment....

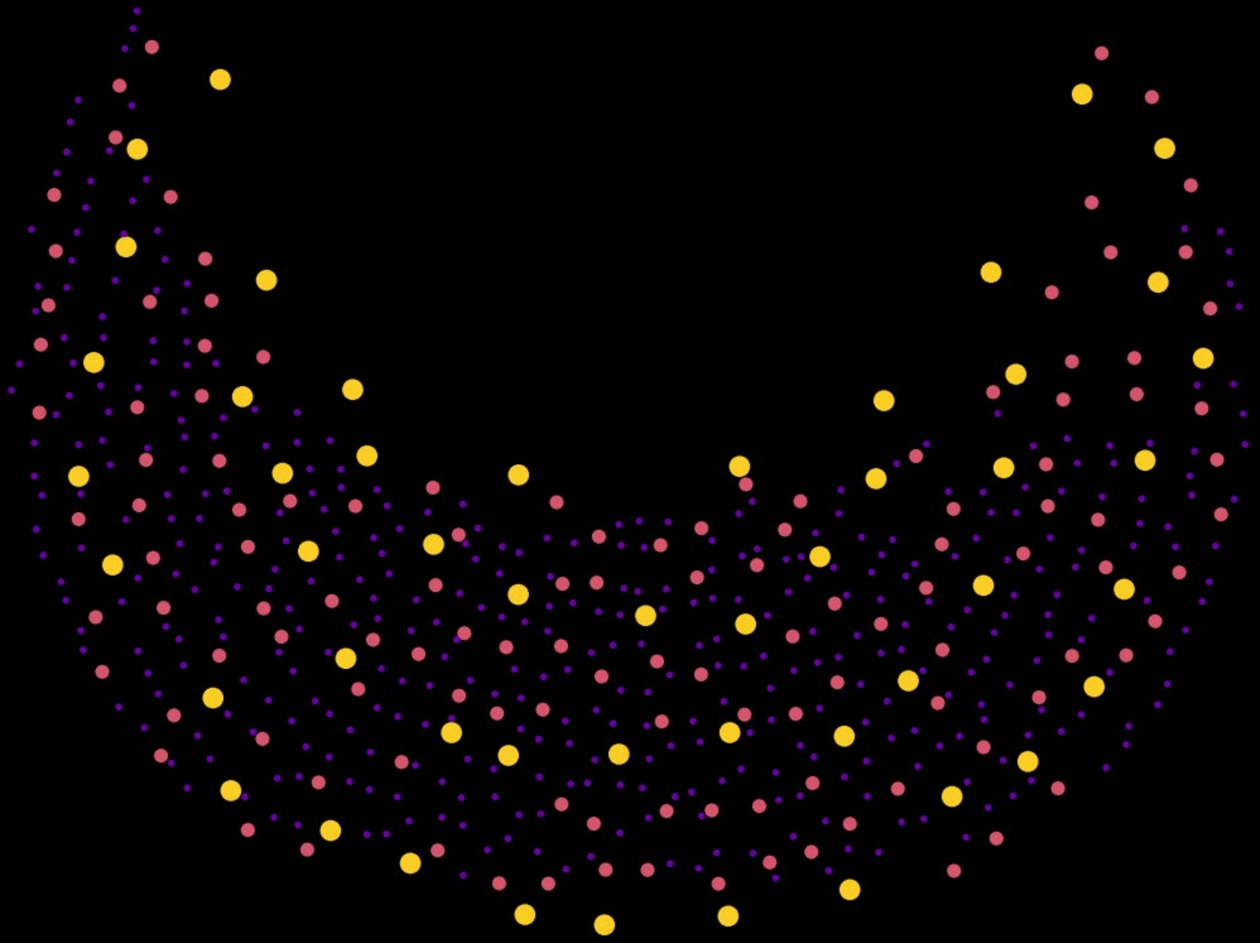


Direction

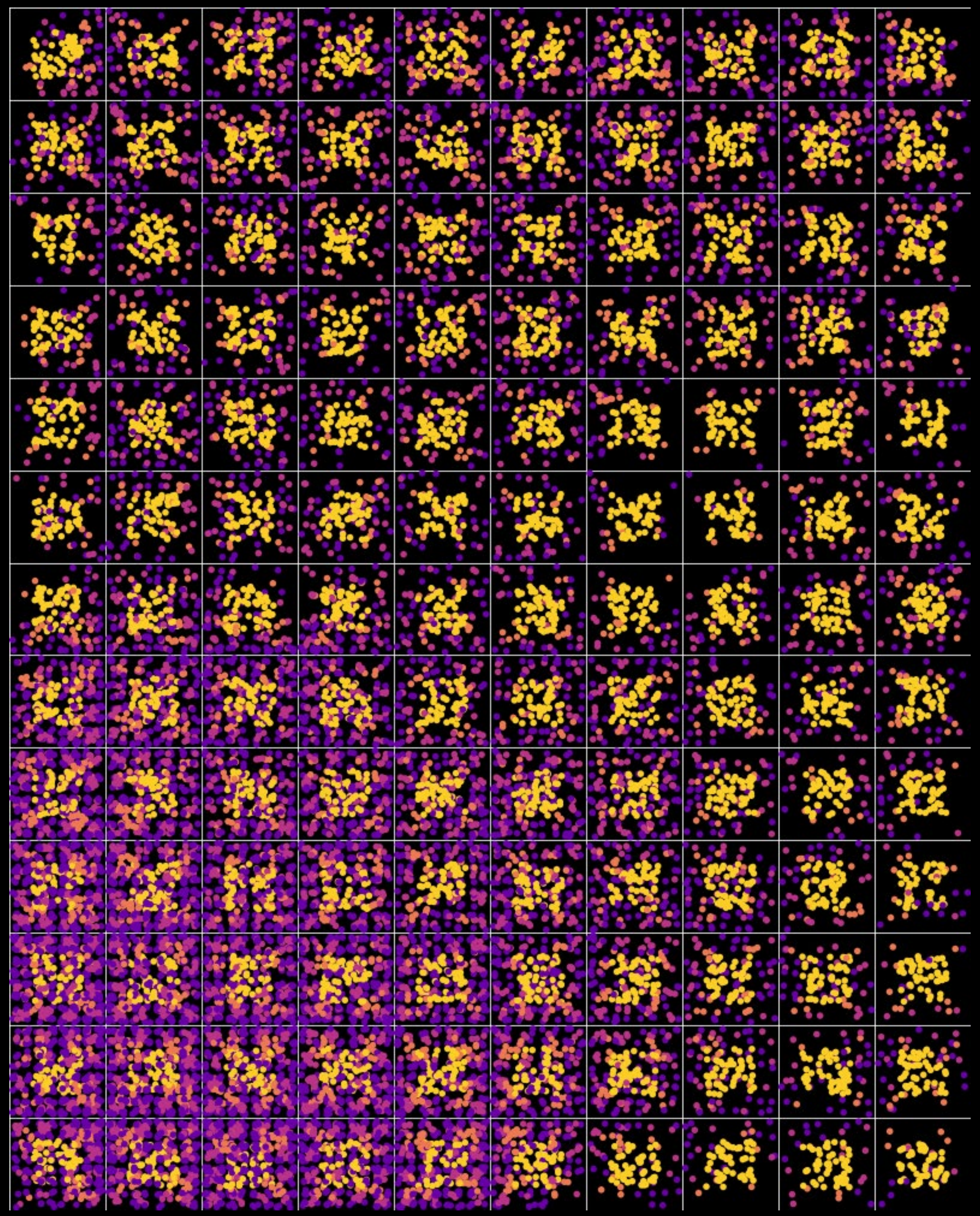


Time / Source

Sampling an Area and Sun Positions

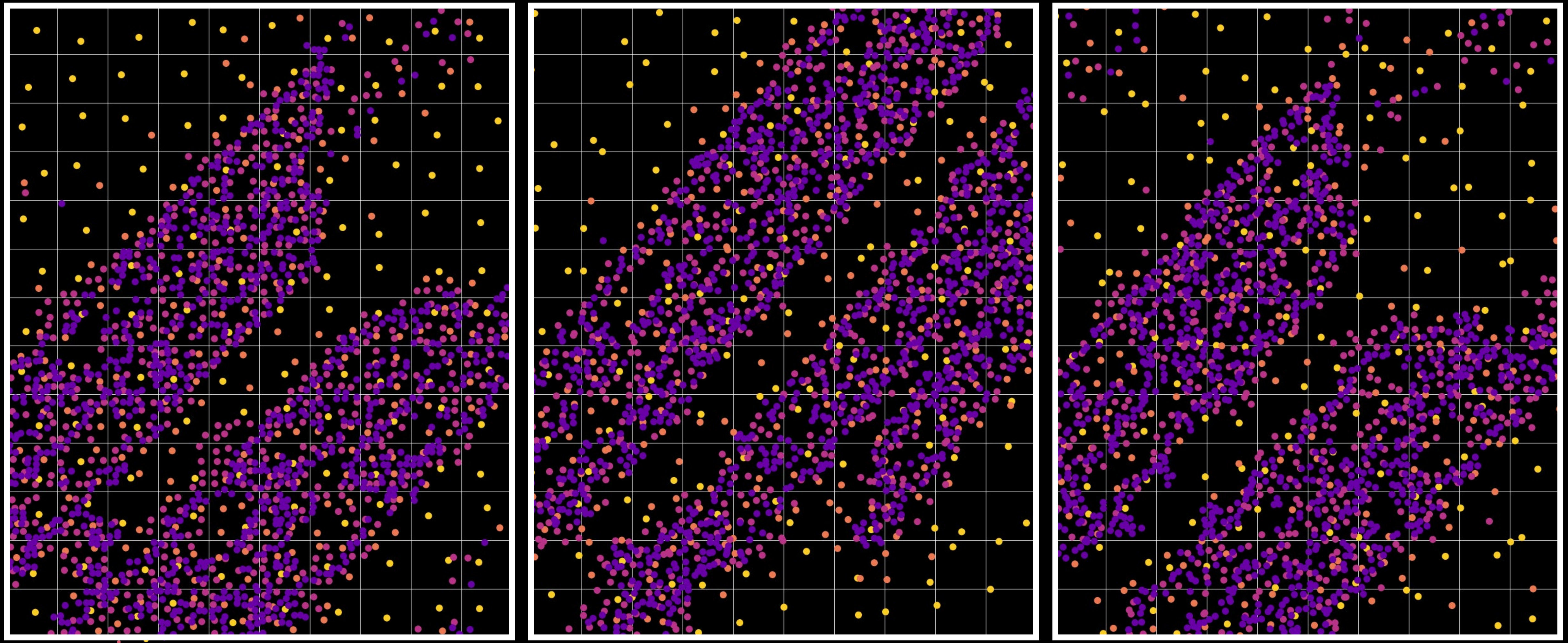


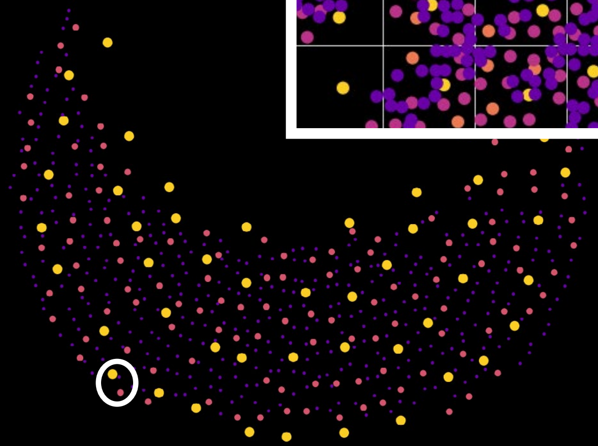
3 levels of sun sampling
for full zone



point samples for
level 0 suns

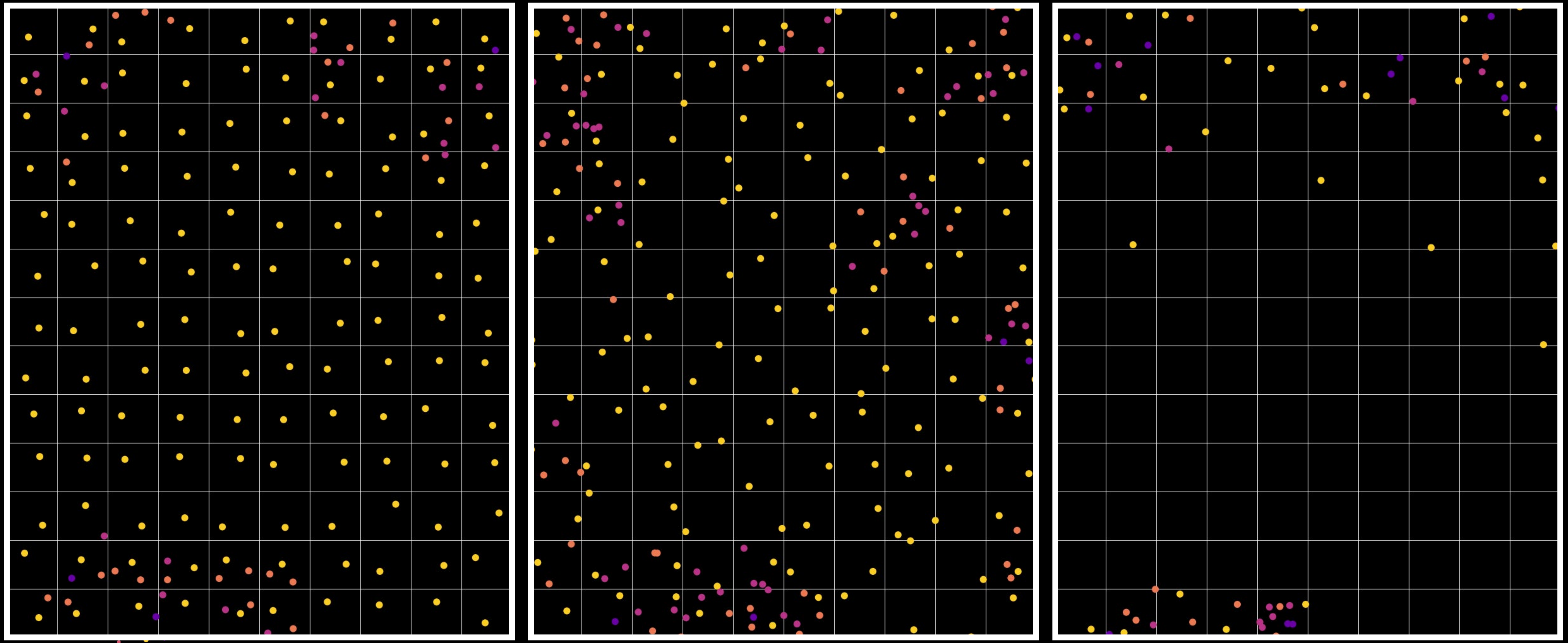
Sampling an Area and Sun Positions

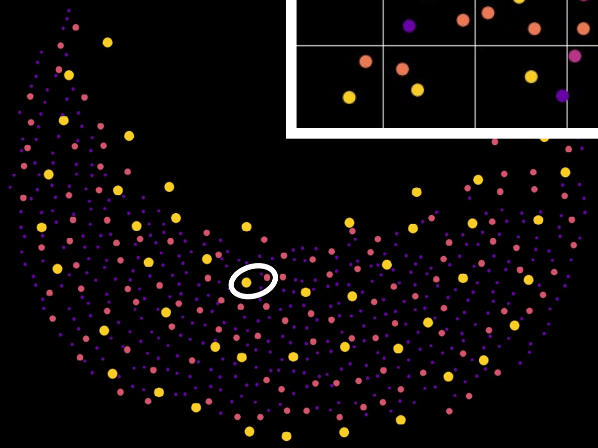




3 levels of sun sampling in high variance region

Sampling an Area and Sun Positions





3 levels of sun sampling
in low variance region

Exercise #0: prerequisites

Windows

- Install Docker from: https://www.docker.com/products/docker-desktop/
  - click on “Windows” and then follow the installation instructions.
- Open the newly installed Docker Desktop application
  - you do not need to sign in or create an account
- download and unzip <<rwt>>
- open a command prompt and navigate to rwt/
- docker build . --tag rwt:latest
  - copy text from Dockerfile

Macos/Linux

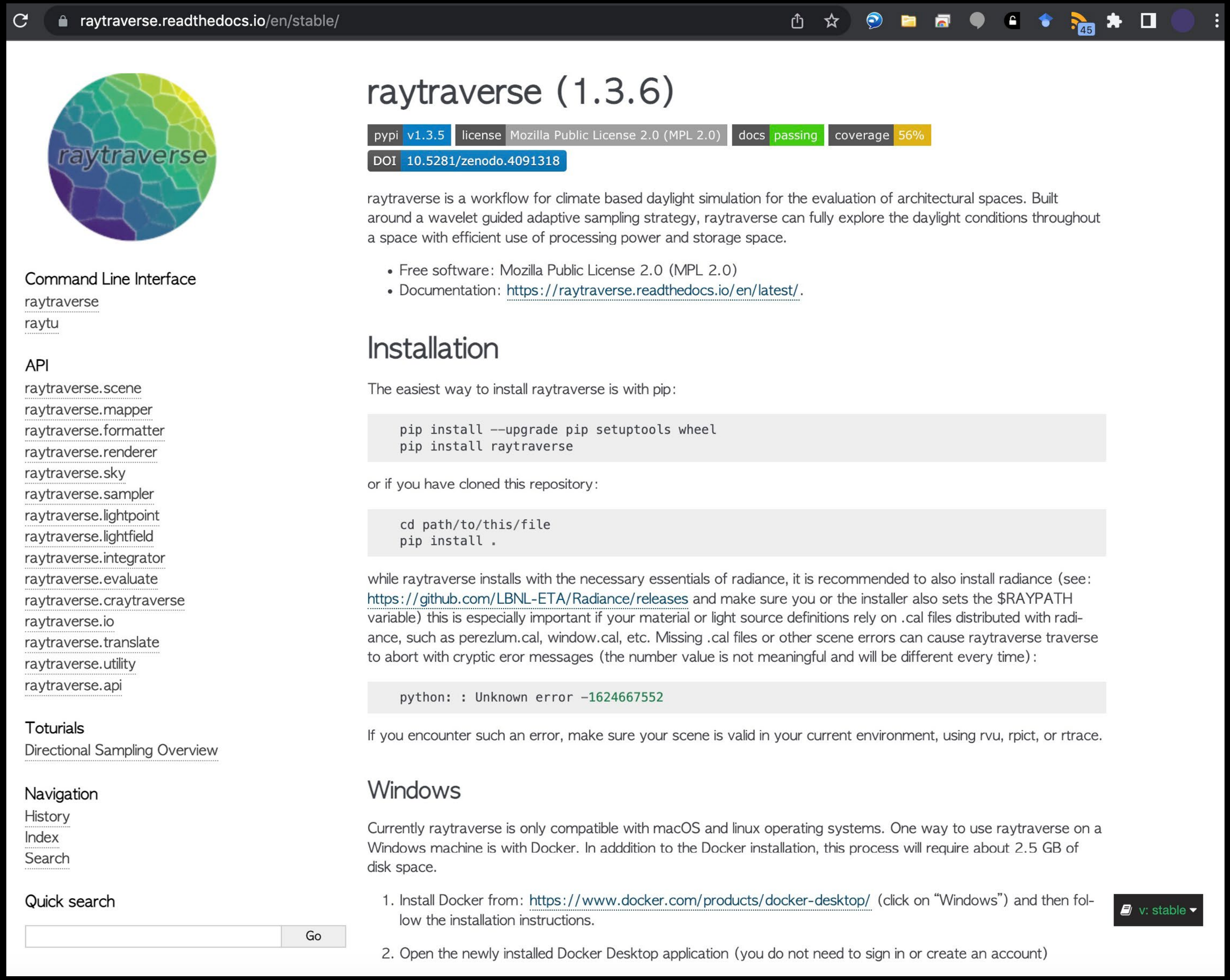
- (as an alternative, on a mac, you can follow the windows instructions)
- Ensure python3.7 - python 3.10 is installed on your machine
  - check in a terminal with: python3 --version
  - for mac, I suggest using homebrew: (find tutorial link)
  - For ubuntu, make sure python has pip (find commands for this)
- install radiance: (github url)
- set path/raypath/manpath in .bash_profile
- open a terminal/command prompt and navigate to rwt/
- python3 -m venv rwtenv
- source rwtenv/bin/activate
- pip install -r Requirements.txt

see README.rst in rwt

exercise #1

Help

raytraverse.readthedocs.io



raytraverse.readthedocs.io/en/stable/

Command Line Interface
raytraverse
raytu

API
raytraverse.scene
raytraverse.mapper
raytraverse.formatter
raytraverse.renderer
raytraverse.sky
raytraverse.sampler
raytraverse.lightpoint
raytraverse.lightfield
raytraverse.integrator
raytraverse.evaluate
raytraverse.craytraverse
raytraverse.io
raytraverse.translate
raytraverse.utility
raytraverse.api

Toturials
Directional Sampling Overview

Navigation
History
Index
Search

Quick search

Go

raytraverse (1.3.6)

pypi v1.3.5 | license Mozilla Public License 2.0 (MPL 2.0) | docs passing | coverage 56%

DOI 10.5281/zenodo.4091318

raytraverse is a workflow for climate based daylight simulation for the evaluation of architectural spaces. Built around a wavelet guided adaptive sampling strategy, raytraverse can fully explore the daylight conditions throughout a space with efficient use of processing power and storage space.

- Free software: Mozilla Public License 2.0 (MPL 2.0)
- Documentation: https://raytraverse.readthedocs.io/en/latest/.

Installation

The easiest way to install raytraverse is with pip:

```
pip install --upgrade pip setuptools wheel
pip install raytraverse
```

or if you have cloned this repository:

```
cd path/to/this/file
pip install .
```

while raytraverse installs with the necessary essentials of radiance, it is recommended to also install radiance (see: https://github.com/LBNL-ETA/Radiance/releases and make sure you or the installer also sets the $RAYPATH variable) this is especially important if your material or light source definitions rely on .cal files distributed with radiance, such as perezlum.cal, window.cal, etc. Missing .cal files or other scene errors can cause raytraverse traverse to abort with cryptic eror messages (the number value is not meaningful and will be different every time):

```
python: : Unknown error -1624667552
```

If you encounter such an error, make sure your scene is valid in your current environment, using rvu, rpict, or rtrace.

Windows

Currently raytraverse is only compatible with macOS and linux operating systems. One way to use raytraverse on a Windows machine is with Docker. In adddition to the Docker installation, this process will require about 2.5 GB of disk space.

1. Install Docker from: https://www.docker.com/products/docker-desktop/ (click on "Windows") and then follow the installation instructions.
2. Open the newly installed Docker Desktop application (you do not need to sign in or create an account)

v: stable

raytraverse --help

```
Usage: raytraverse [OPTIONS] COMMAND1 [ARGS]... [COMMAND2 [ARGS]...]...

  the raytraverse executable is a command line interface to the
  raytraverse python package for running and evaluating climate based
  daylight models. sub commands of raytraverse can be chained but should
  be invoked in the order given.
  the easiest way to manage options is to use a configuration file, to
  make a template:
     raytraverse --template > run.cfg
  after adjusting the settings, then each command can be invoked in turn
  and any dependencies will be loaded with the correct options, a
  complete run and evaluation can then be called by:
     raytraverse -c run.cfg skyrun sunrun evaluate
  as all required precursor commands will be invoked automatically as
  needed.


Options:

VALUE OPTIONS:

  -config, -c PATH
      path of config file to load

  -n INTEGER
      sets the environment variable RAYTRAVERSE_PROC_CAP set to 0 to clear (parallel processes will use cpu_limit)

  -out DIRECTORY


FLAGS (DEFAULT FALSE):

  --template / --no-template
      write default options to std out as config  [default: no-template]


HELP:

  -opts, --opts
      check parsed options  [default: False]

  --debug
      show traceback on exceptions  [default: False]

  --help
      Show this message and exit.  [default: False]

  --version
      Show the version and exit.  [default: False]


Commands:
  area          define sampling area
  directskyrun
  evaluate      evaluate metrics
  images        render images
  pull
  scene         define scene files for renderer and output directory
  skydata       define sky conditions for evaluation
  skyengine     initialize engine for skyrun
  skyrun        run scene under sky for a set of points (defined by area)
  sourceengine  initialize engine for sunrun
  sourcerun     run scene for a single source (or multiple defined in a single scene file)
  sunengine     initialize engine for sunrun
  sunrun        run scene for a set of suns (defined by suns) for a set of points (defined by area)
  suns          define solar sampling space
```

Exercise #1.a: Up and Running

- open a terminal/command prompt and navigate to rwt/
- for windows users (copy from ./Dockerfile):
  + make sure docker is running
  + docker run -it --name rayt --mount type=bind,source="$(pwd)",target=/working rwt /bin/bash
- for mac/linux:
  + source rwtenv/bin/activate
- cd ex1
- raytraverse --version

see README.rst in rwt

Exercise #1.b: Configuration File

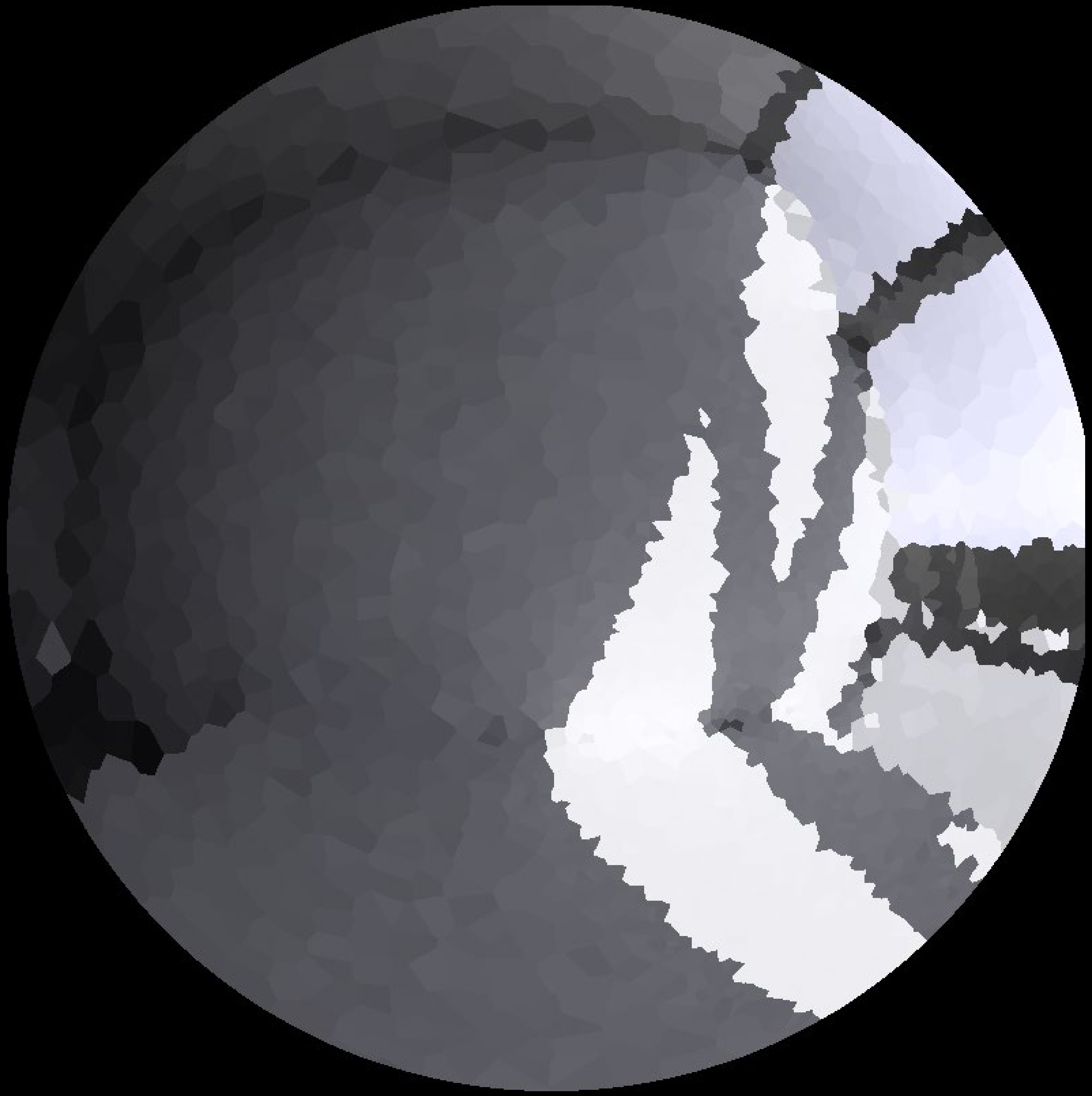
```
raytraverse scene --opts

Options:

debug..........False
log............True
opts...........False
out............None
overwrite......False
reload.........True
scene..........None
```

```
rayt.cfg:

[raytraverse_scene]
out = office
scene = office_glz.rad

[raytraverse_area]
name = vp1
static_points = 0.6,1,1.2

[raytraverse_sourceengine]
source = may3pm
rayargs = -ab 4
srcfile = MAY03-1530.rad

[raytraverse_images]
sensors = 0.6,1,1.2,1,0,0
basename = default
simtype = may3pm
```

```
raytraverse -c rayt.cfg scene --opts

Options:

debug..........False
log............True
opts...........False
out............office
overwrite......False
reload.........True
scene..........office_glz.rad
```

Exercise #1.c: Sampling a Single Sky from a Point

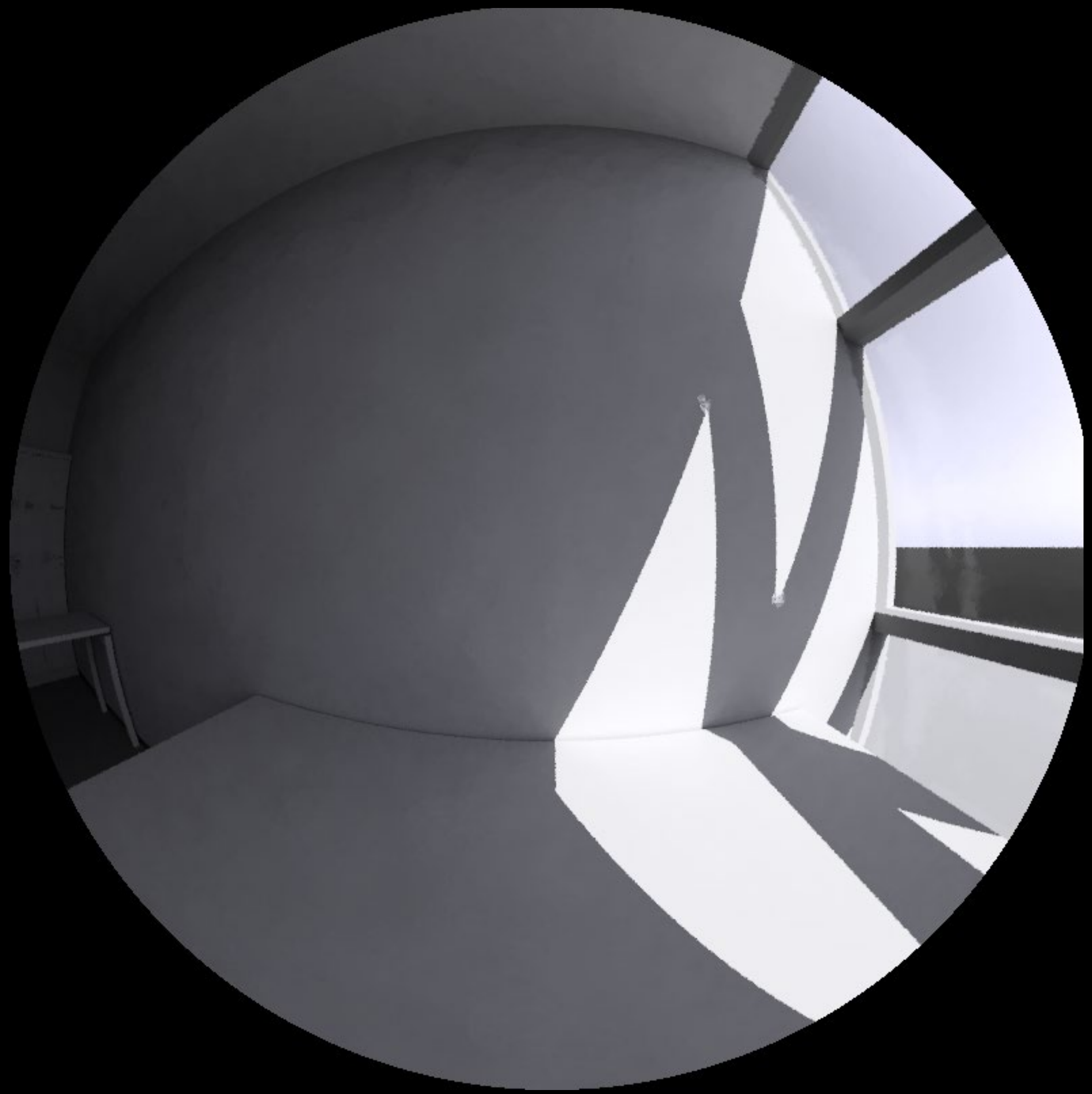
```
$ raytraverse -c rayt.cfg  sourcerun images
27-Jun-2022 12:58:14  SamplerArea  Started sampling office at vp1 with may3pm
27-Jun-2022 12:58:14  SamplerArea  level         shape  samples     rate
27-Jun-2022 12:58:14  SamplerArea  1 of 1        [1 1]        1  100.00%
27-Jun-2022 12:58:14  SrcSamplerPt  Started sampling office at view_000000 with may3pm
27-Jun-2022 12:58:14  SrcSamplerPt  level         shape  samples     rate
27-Jun-2022 12:58:14  SrcSamplerPt  1 of 6      [64 32]     2048  100.00%
27-Jun-2022 12:58:15  SrcSamplerPt  2 of 6    [128  64]     1617   19.74%
27-Jun-2022 12:58:16  SrcSamplerPt  3 of 6    [256 128]     1252    3.82%
27-Jun-2022 12:58:17  SrcSamplerPt  4 of 6    [512 256]     1030    0.79%
27-Jun-2022 12:58:17  SrcSamplerPt  5 of 6  [1024  512]      941    0.18%
27-Jun-2022 12:58:18  SrcSamplerPt  6 of 6  [2048 1024]      768    0.04%
27-Jun-2022 12:58:19  SrcSamplerPt  total sampling:  -      7656    0.37%
27-Jun-2022 12:58:19  SamplerArea  total sampling:  -         1  100.00%
27-Jun-2022 12:58:19  Integrator  Making Images for 1 view directions at 1 points
under 1 skies
27-Jun-2022 12:58:19  Integrator  Calculating 1 sun/sky/pt combinations
  Making Images (01 of 01): 100%|████████████████████████████████| 1/1
[00:00<00:00,  1.30it/s]
default_may3pm-0000_pt-0.6_1.0_1.2_vd-1.0_0.0_0.0.hdr
```



Exercise #1.d: Command Line Options: Interpolation

Naive Interpolation
~9 sec (sampling + interpolation)

Context Filtering
~14 sec (sampling + interpolation)



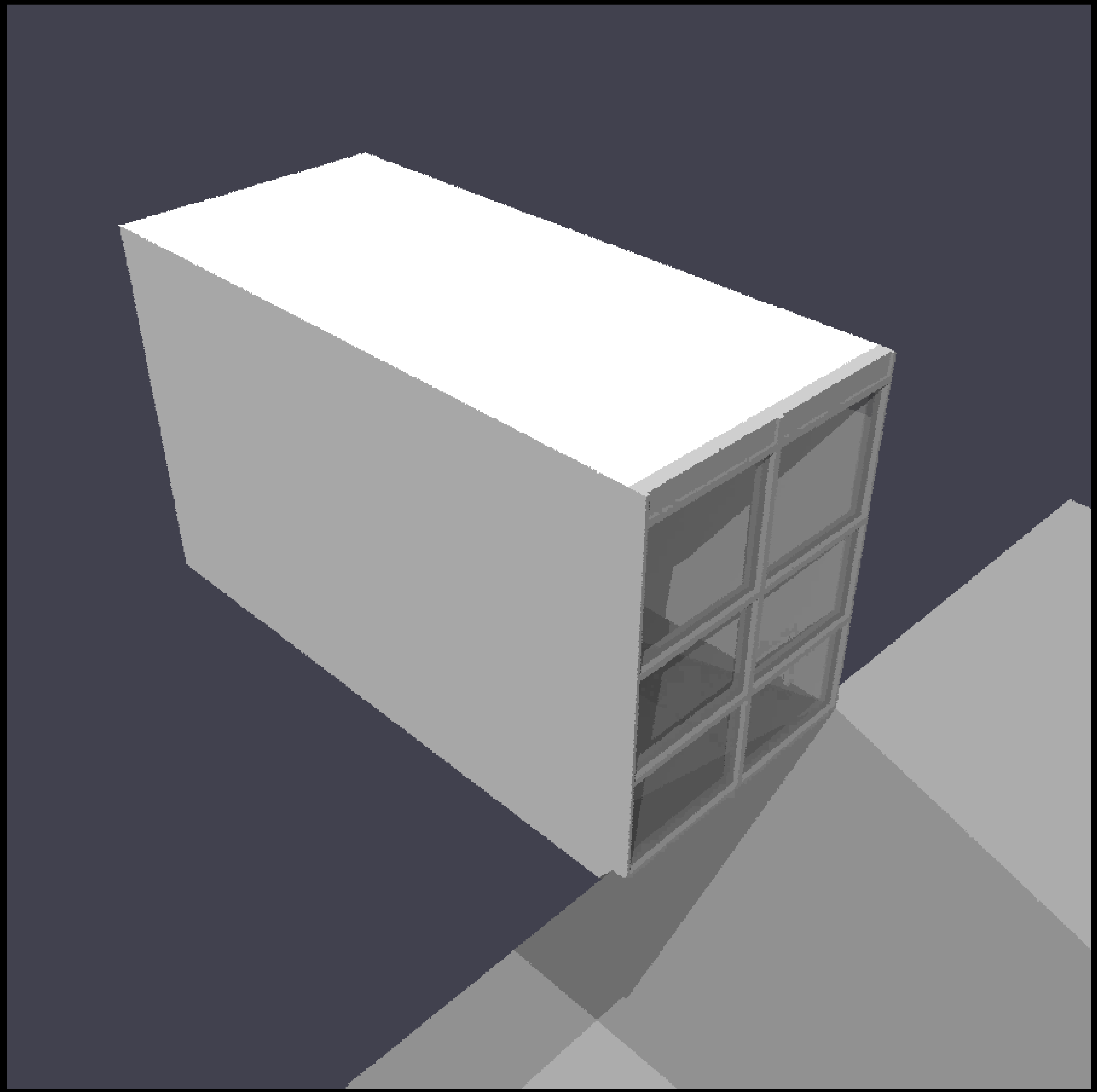
Context Sampling + higher accuracy
~2.5 min (sampling + interpolation)

```
raytraverse -c rayt.cfg images -basename inter -interpolate high
raytraverse -c rayt.cfg images -basename context -interpolate highc
raytraverse -c rayt.cfg sourceengine -accuracy .125 sourcerun --overwrite --scenedetail images\
            -basename detail -interpolate high
```

setting up a simulation

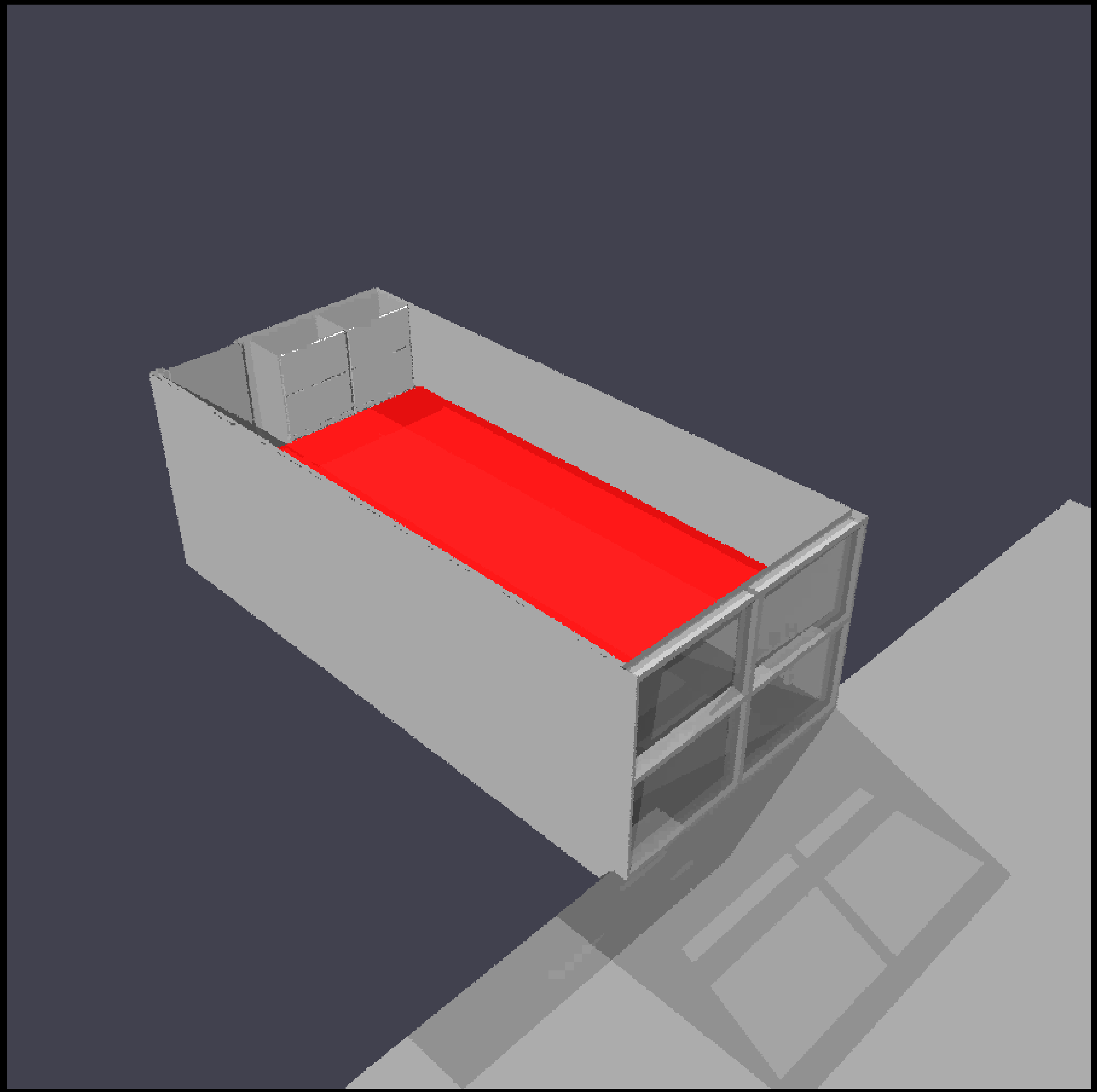
Scene

```
[raytraverse_scene]
log = True
out = None
overwrite = False
reload = True
scene = None
```



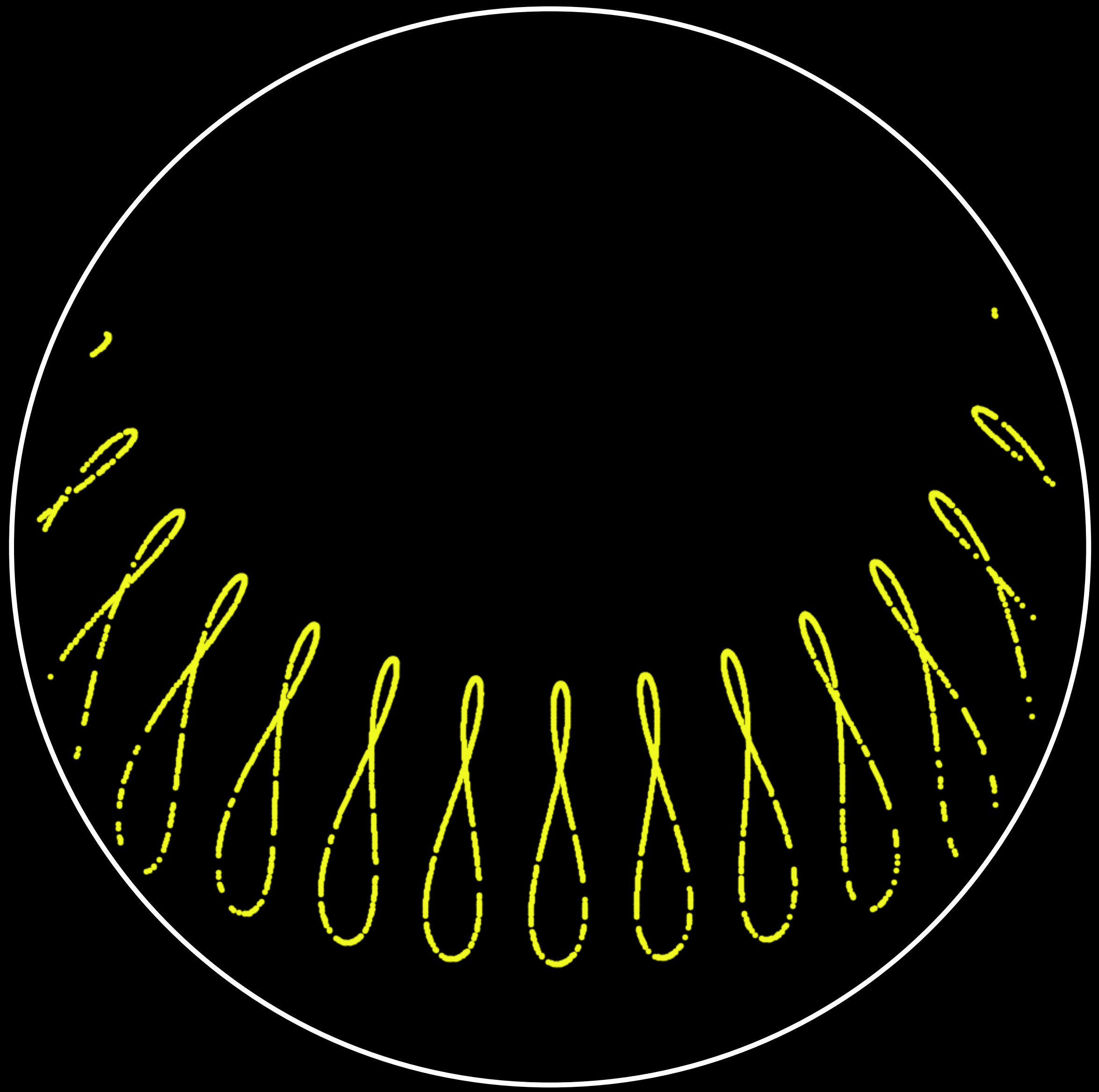
Area

```
[raytraverse_area]
jitterrate = 0.5
name = plan
printdata = False
printlevel = None
ptres = 1.0
rotation = 0.0
static_points = None
zheight = None
zone = None
```

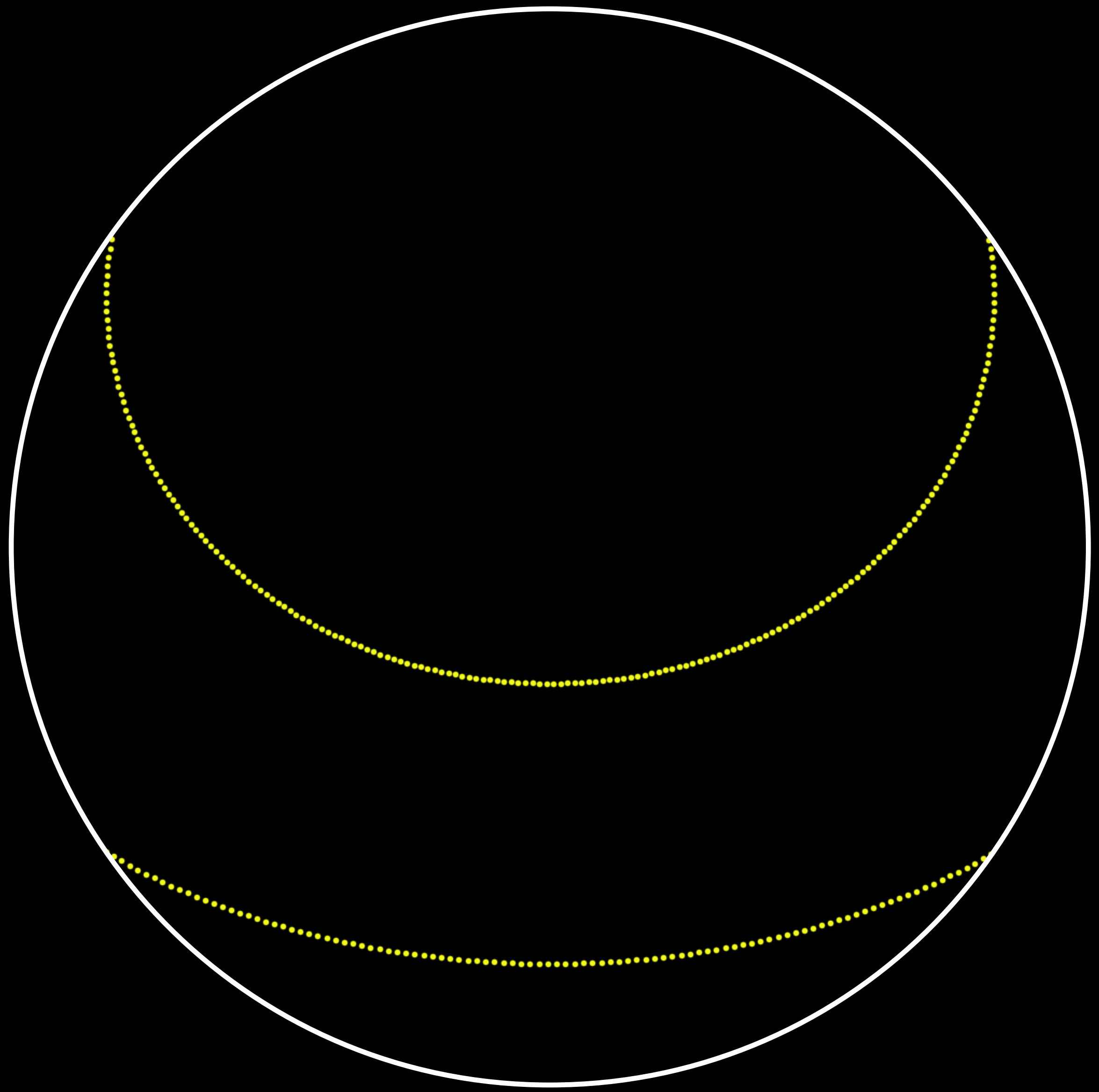


Suns

```
[raytraverse_suns]
epwloc = False
jitterrate = 0.5
loc = None
name = suns
printdata = False
printlevel = None
skyro = 0.0
sunres = 9
```

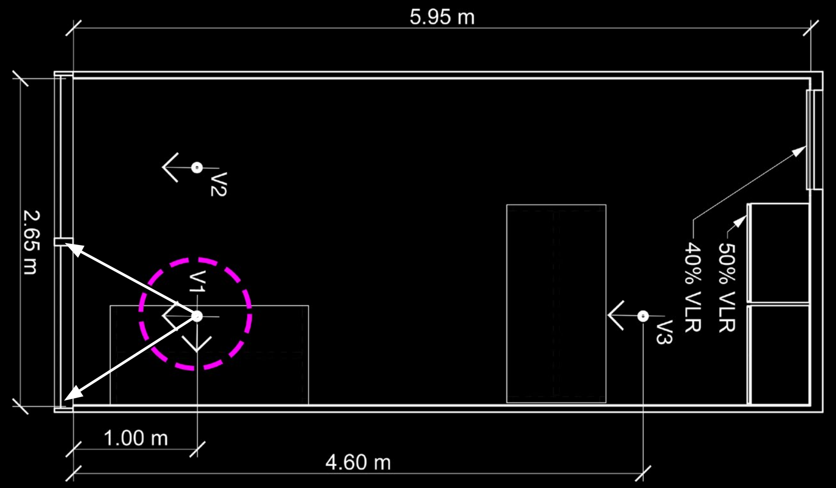


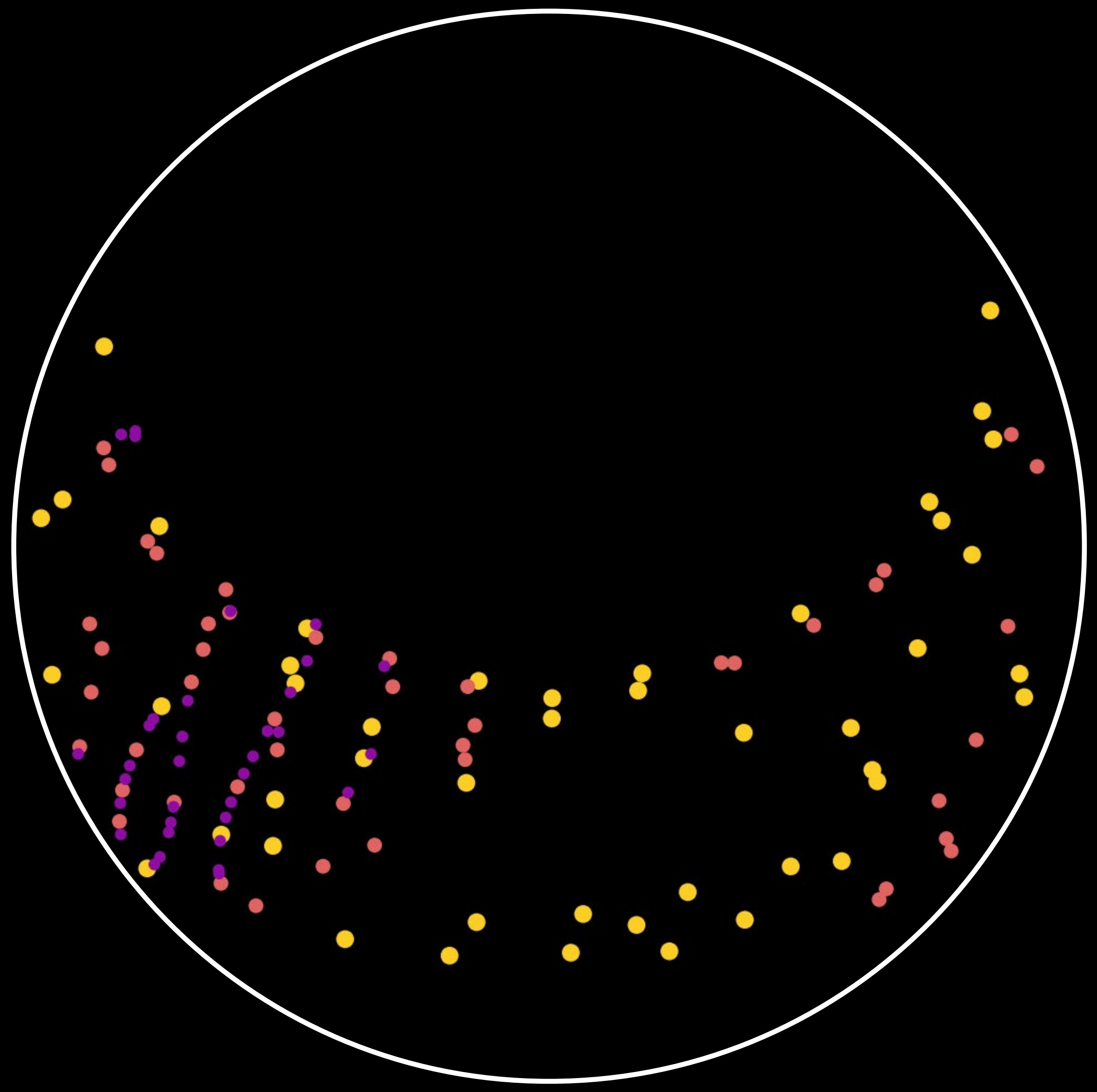
--no-epwloc // epwloc = False



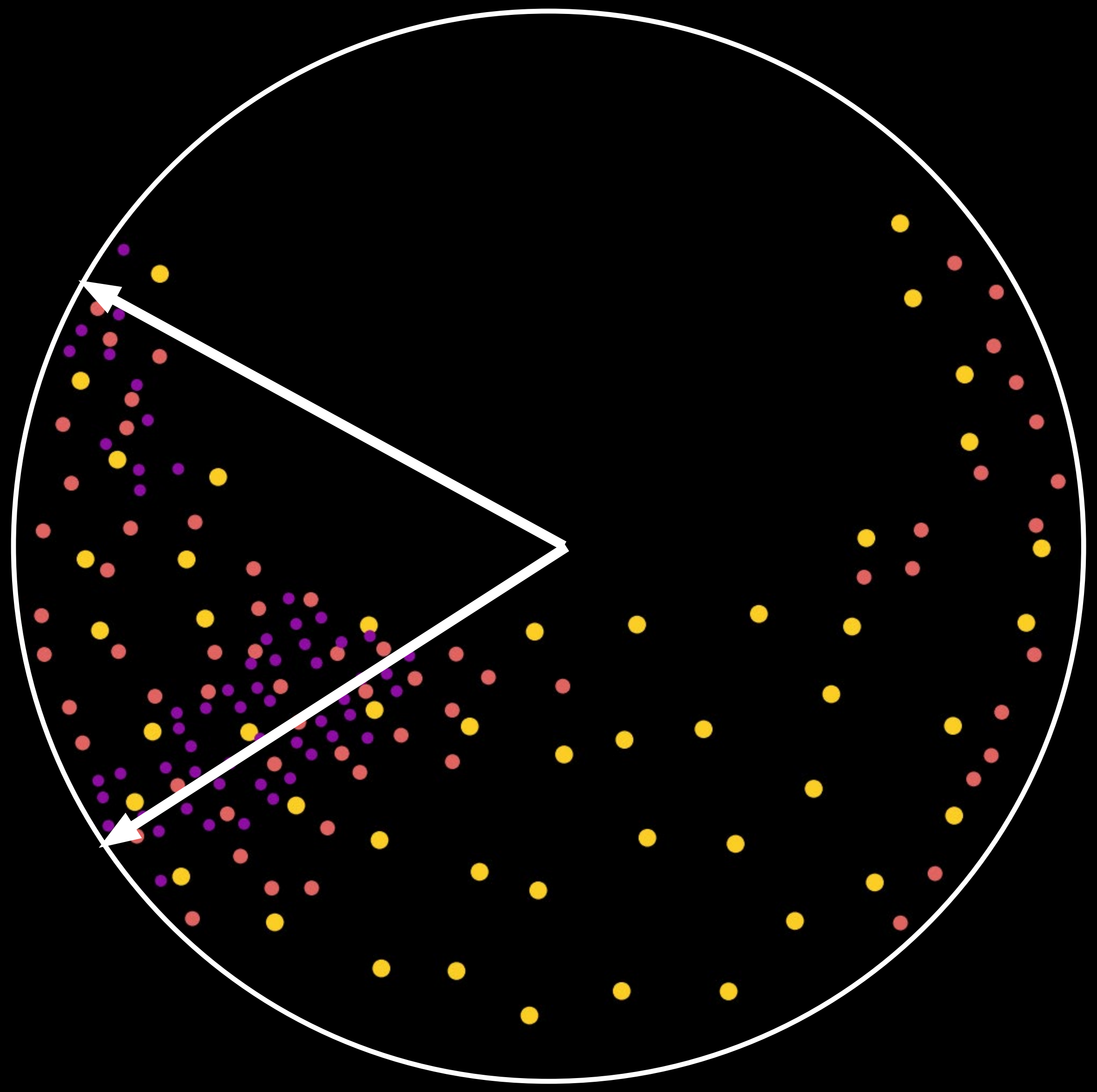
--epwloc // epwloc = True

Suns





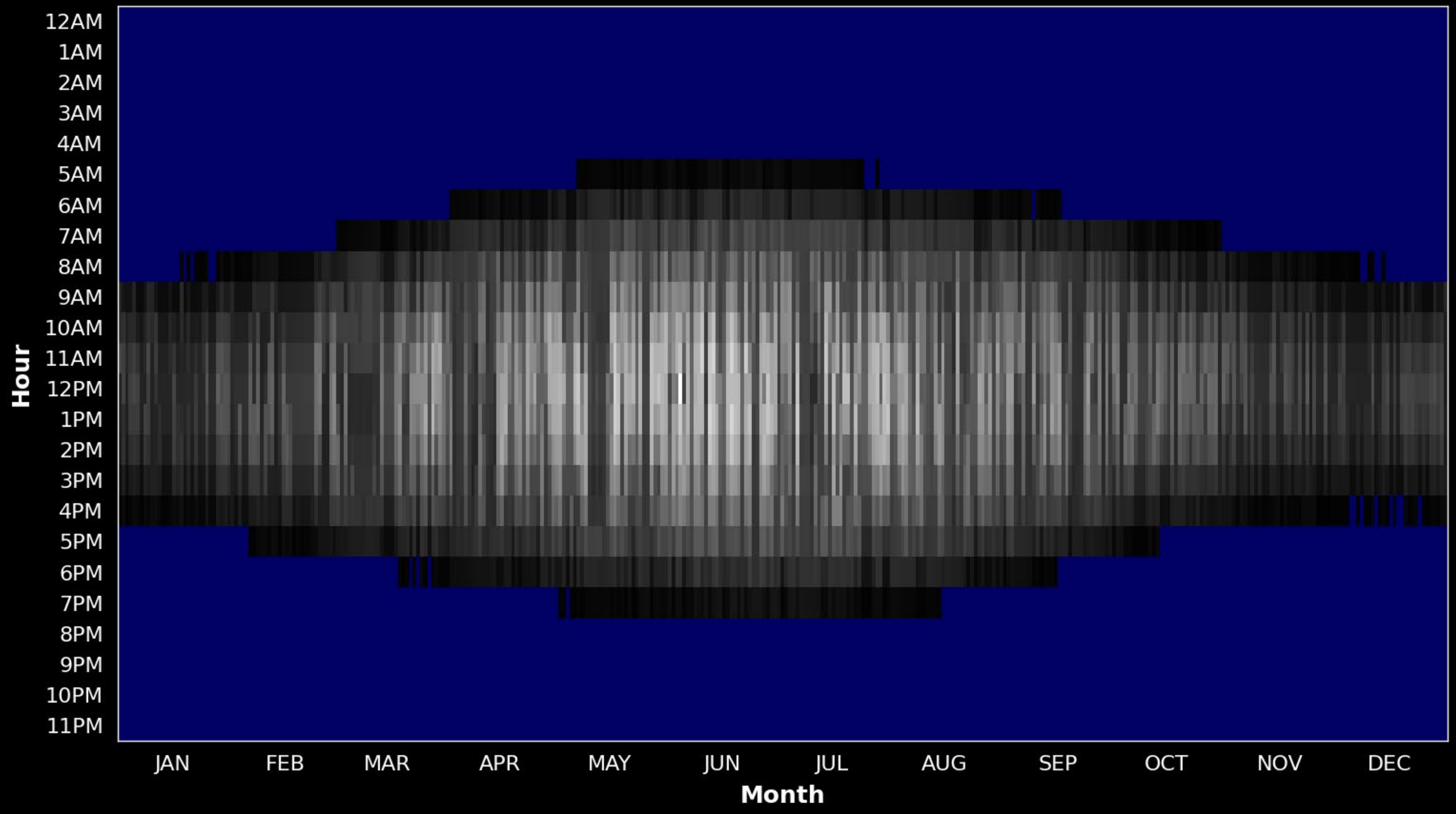
--no-epwloc // epwloc = False



--epwloc // epwloc = True

Skydata

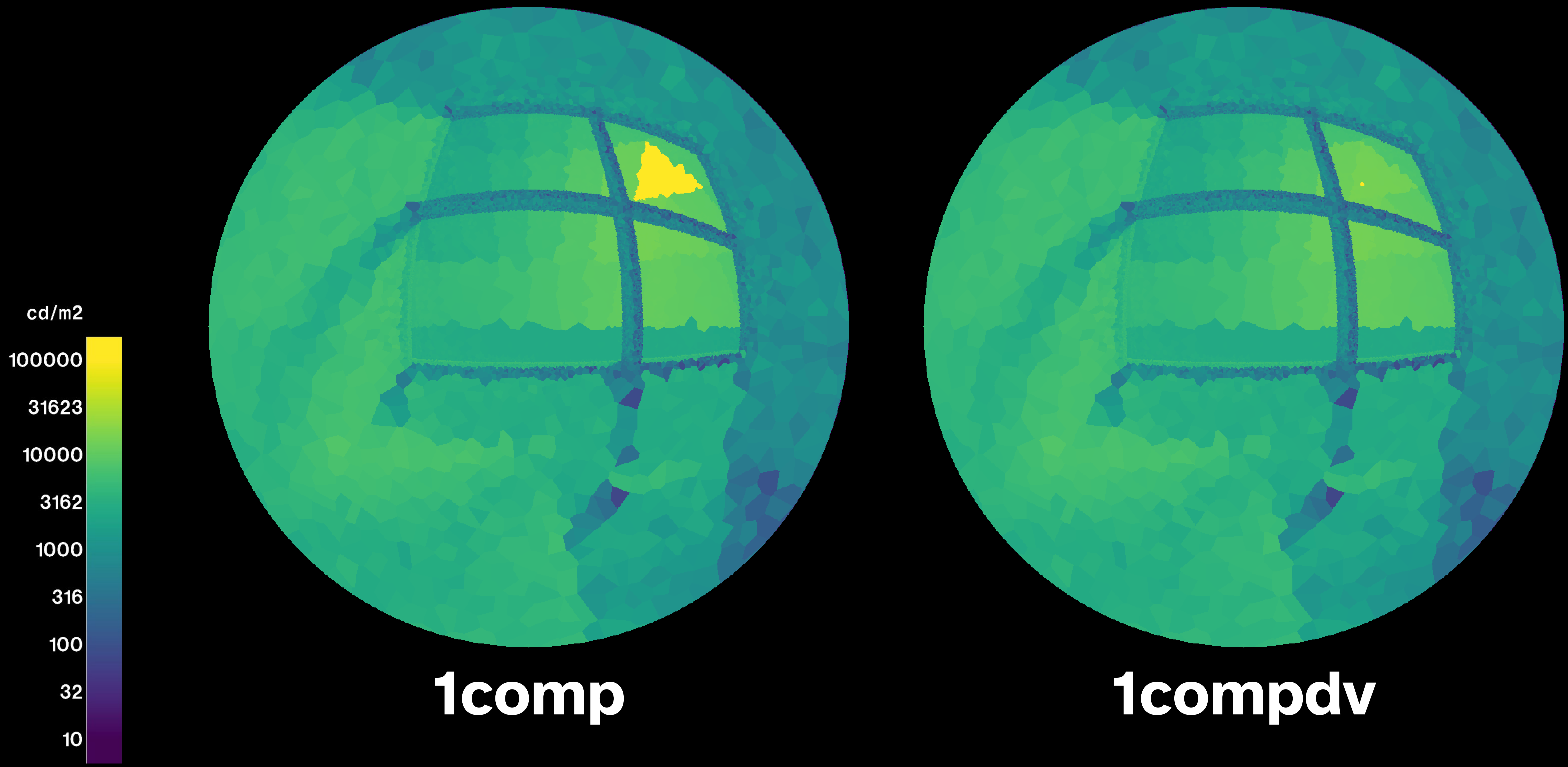
```
[raytraverse_skydata]
ground_fac = 0.2
loc = None
minalt = 2.0
mindiff = 5.0
mindir = 0.0
name = skydata
printdata = False
printfull = False
reload = True
skyres = 15
skyro = 0.0
wea = None
```



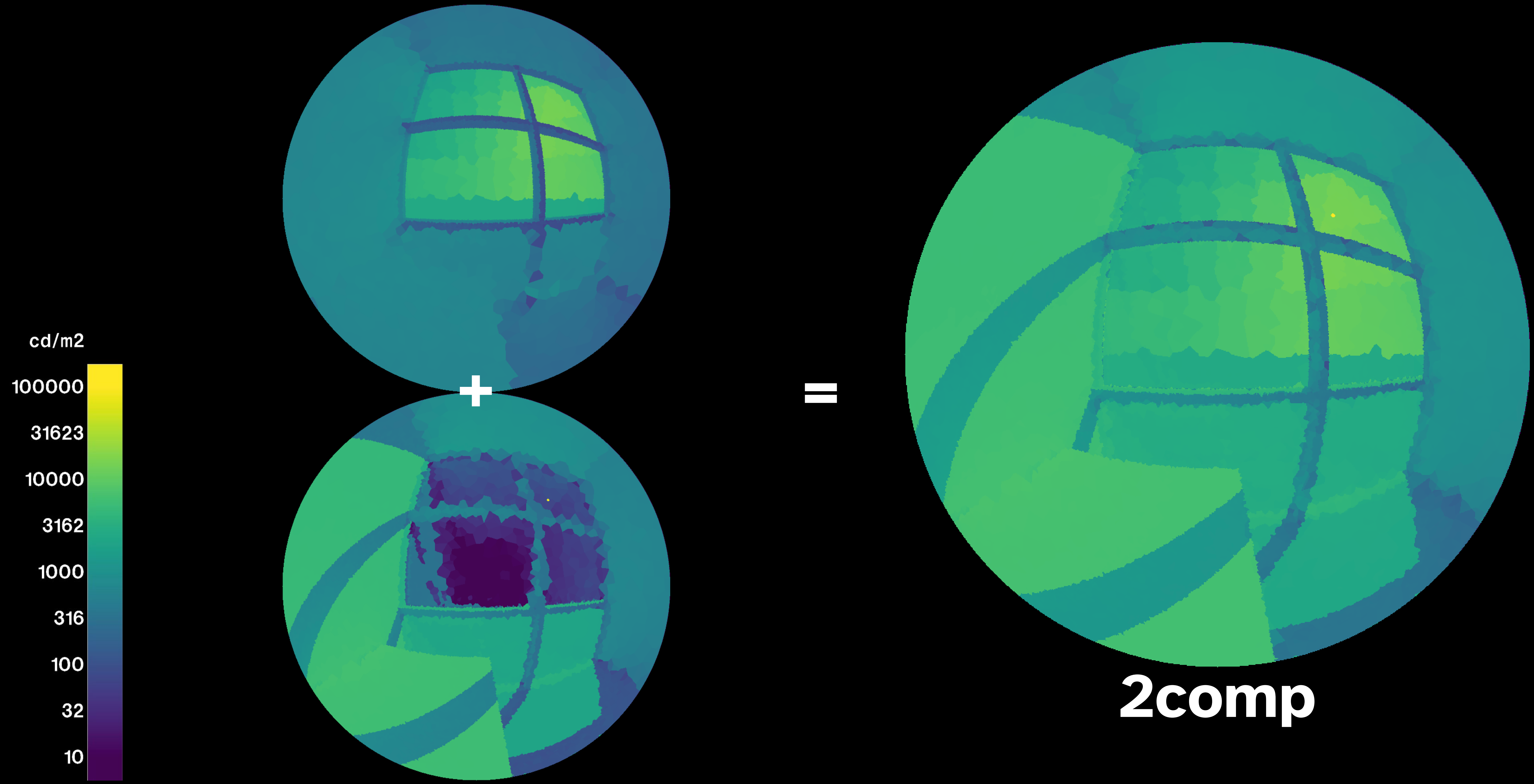
Available Methods

- **1comp**: daylight coefficient method, solar energy in sky patch
- **1compdv**: 1comp, but with direct view replacement of sun and specular reflections
- **2comp**: sky patch for sky contribution, sun run for sun contribution, depth of contributions depends on skyengine and sunengine settings, no approximation for sun from sky patch
- **3comp**: 2-phase DDS, sky handles sky+indirect sun, sun handles direct sun requires directskyrun -ab 1 and sunrun -ab 0
- **<sourcename>**: use a single source file (electric lights, single sky + sun, etc…)

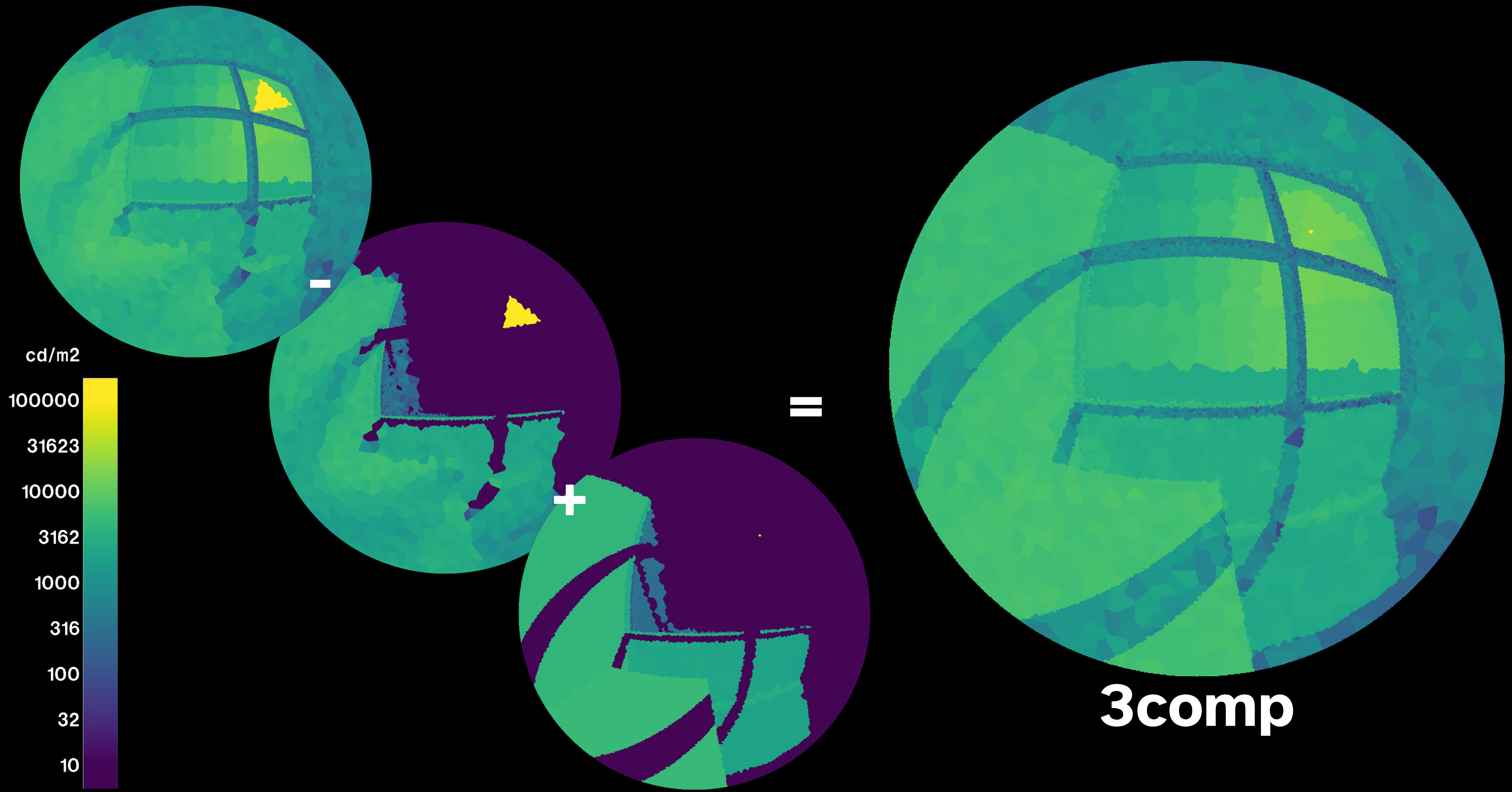
Sky Patch Coefficients



Sky Patch + Sun



Sky Patch + Direct Sun



Single Source

may3pm

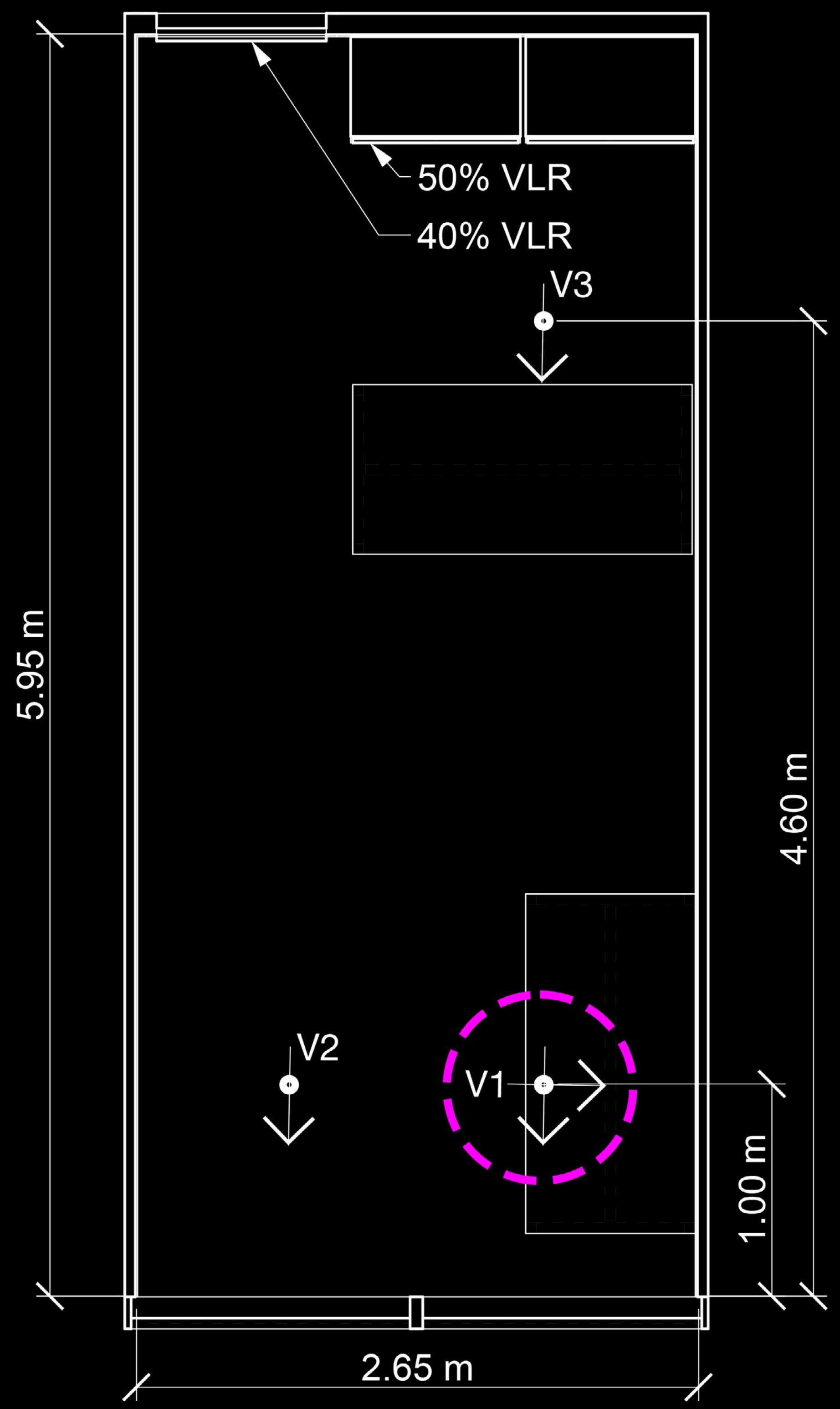
sep2pm

Simulation Types

1comp	1compdv	2comp	3comp	<srcname>
Run:				
skyrun	*skyrun* *directskyrun*	*skyrun* *sunrun*	*skyrun* *directskyrun* *sunrun*	*sourcerun*
Config:				
[raytraverse_images] simtype = 1comp [raytraverse_evaluate] simtype = 1comp	[raytraverse_skyengine] dcompargs = -ab 0 [raytraverse_images] simtype = 1compdv [raytraverse_evaluate] simtype = 1compdv	[raytraverse_sunengine] rayargs = -ab ACTUAL [raytraverse_images] simtype = 2comp [raytraverse_evaluate] simtype = 2comp	[raytraverse_skyengine] dcompargs = -ab 1 [raytraverse_images] simtype = 3comp [raytraverse_evaluate] simtype = 3comp	[raytraverse_images] simtype = srcname [raytraverse_evaluate] simtype = srcname

exercise #2

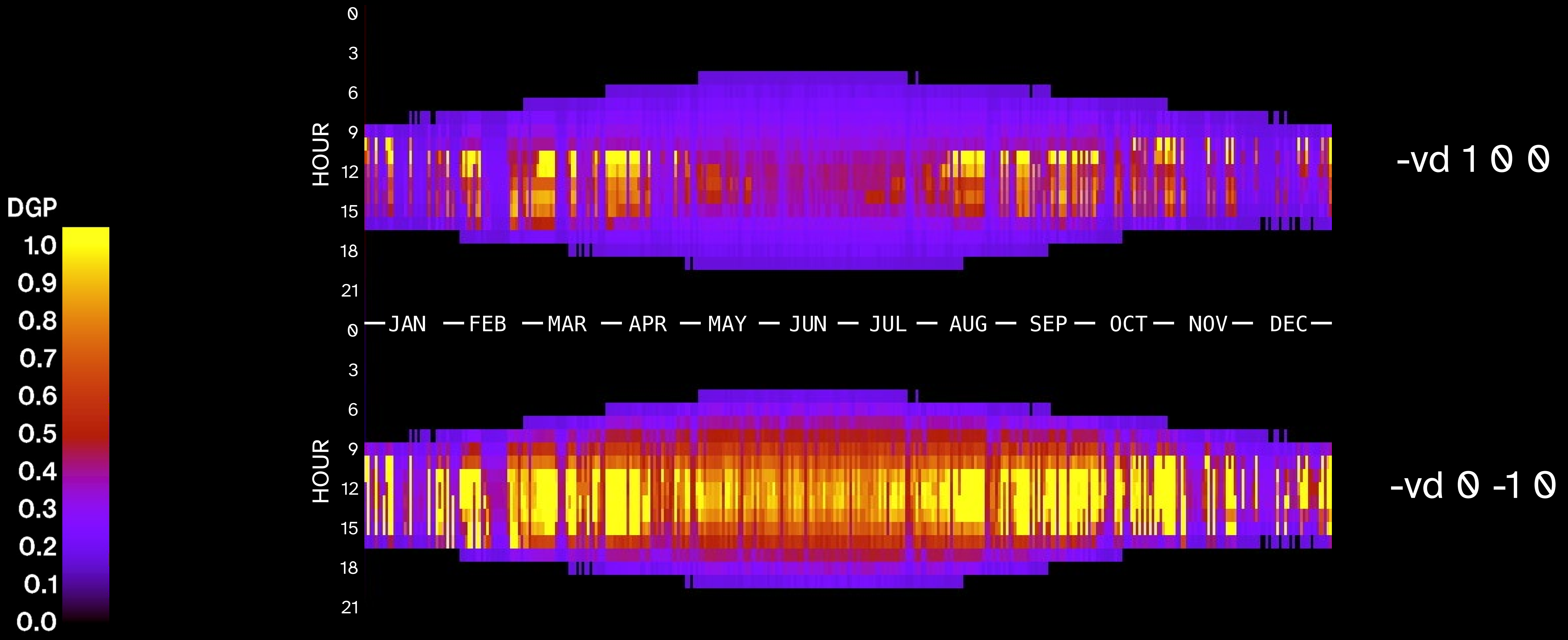
exercise #2: Annual Metrics From a Point




```
[shared]
wea = ../scene/geneva.epw

[raytraverse_scene]
out = office
scene = ../scene/office_glz.rad

[raytraverse_area]
name = vp1
static_points = 0.6,1,1.2

[raytraverse_suns]
epwloc = True
loc = ${shared:wea}

[raytraverse_skydata]
wea = ${shared:wea}

[raytraverse_skyengine]
dcompargs = -ab 0

[raytraverse_evaluate]
basename = results
blursun = True
metrics = illum dgp
resampleview = True
sdirs = 1,0,0 0,-1,0
sensors = 0.6,1,1.2
simtype = 1compdv
threshold = 2000.0

[raytraverse_pull]
col = view metric
gridhdr = True
lr = results.npz
metricfilter = dgp
ofiles = dgp
skyfill = office/skydata.npz
```

Exercise #2.a: Annual Metrics From a Point

```
# run a 1compdv simulation from a single point,
# evaluate DGP and Illum for 2 view directions,
# create hdr heatmaps
raytraverse -c rayt.cfg skyrun directskyrun evaluate pull
```



Exercise #2.b: Using Pull

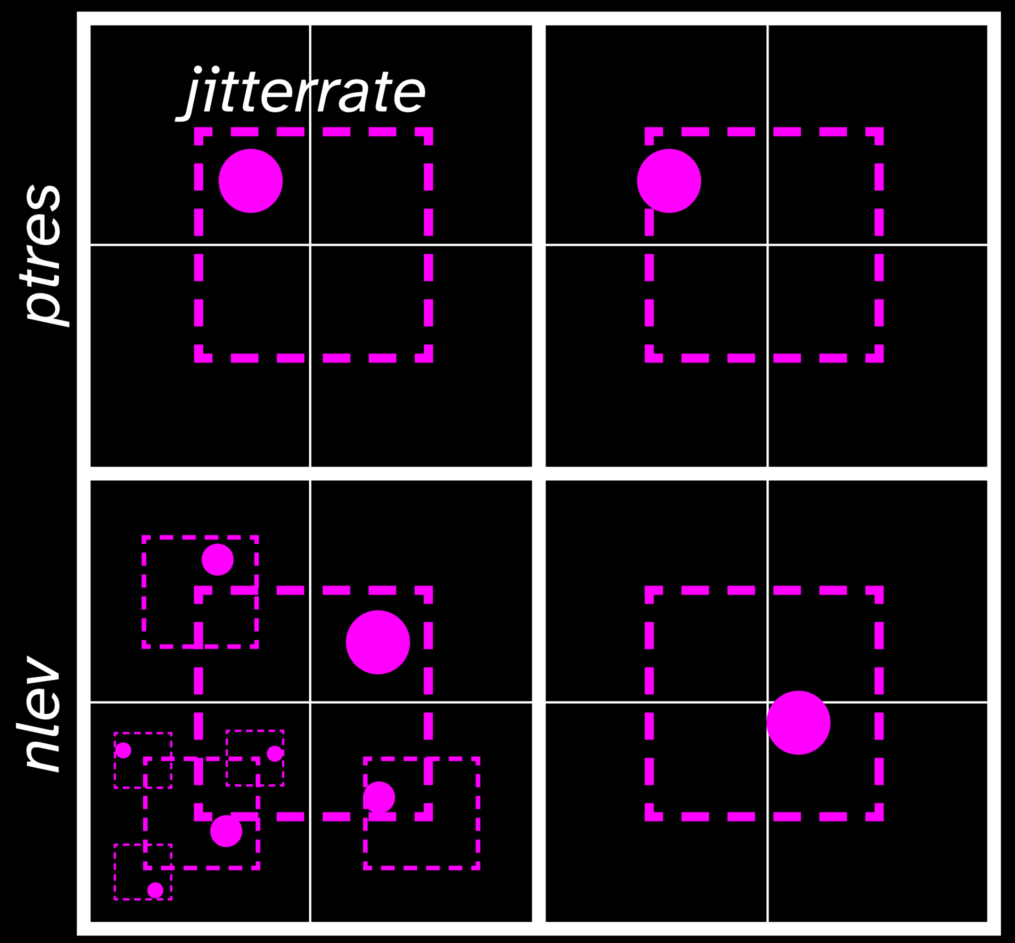
```
# inspect result object:
raytraverse pull -lr results.npz --info

# create tsv txt files for each view:
raytraverse pull -lr results.npz -col ‘metric view’ -ofiles test

# pad to 8760 data:
raytraverse pull -lr results.npz -col ‘metric view’ -ofiles test8760 -skyfill office/skydata.npz

# look at 12pm each day for 1 view:
raytraverse pull -lr results.npz -viewfilter 0 -skyfilter 12:8760:12 -skyfill office/skydata.npz

# calculate percent of daylight hours with perceptible glare for both views:
raytraverse pull -lr results.npz -metricfilter dgp --no-header --no-rowlabel -col view \
   | rcalc -e ‘$1=if($1-.35,1,0);$2=if($2-.35,1,0)’ | total -m
```

simulation parameters

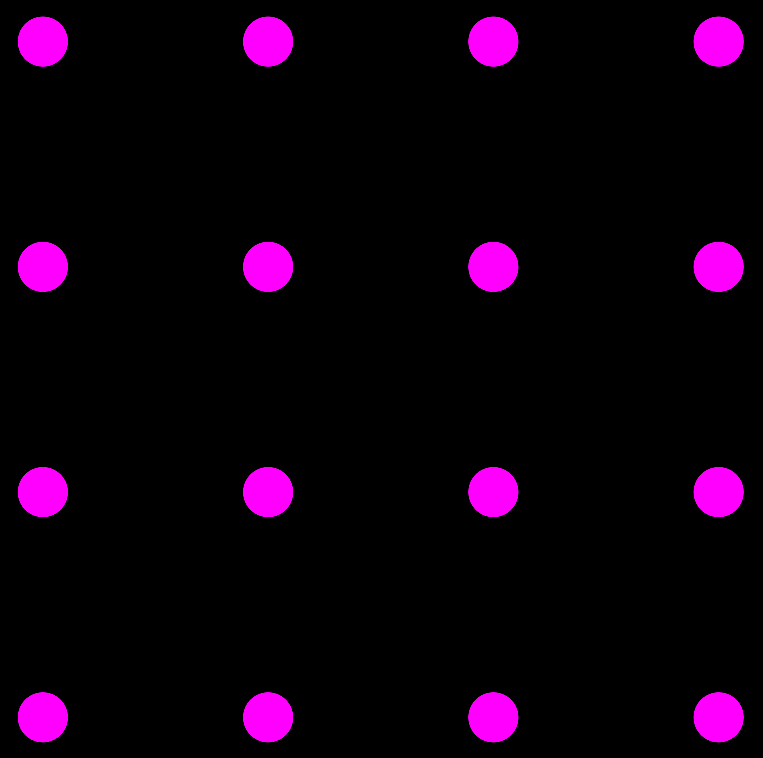
Daylight Coefficients / Sky Simulation

```
[raytraverse_area]
jitterrate = 0.5
ptres = 1.0
static_points = None
zone = None

[raytraverse_skyengine]
accuracy = 1.0
dcompargs = -ab 1
default_args = True
idres = 32
nlev = 5
rayargs = None
skyres = 15
vlt = 0.64

[raytraverse_skyrun]
accuracy = 1.0
edgemode = reflect
jitter = True
nlev = 3
```

Zone



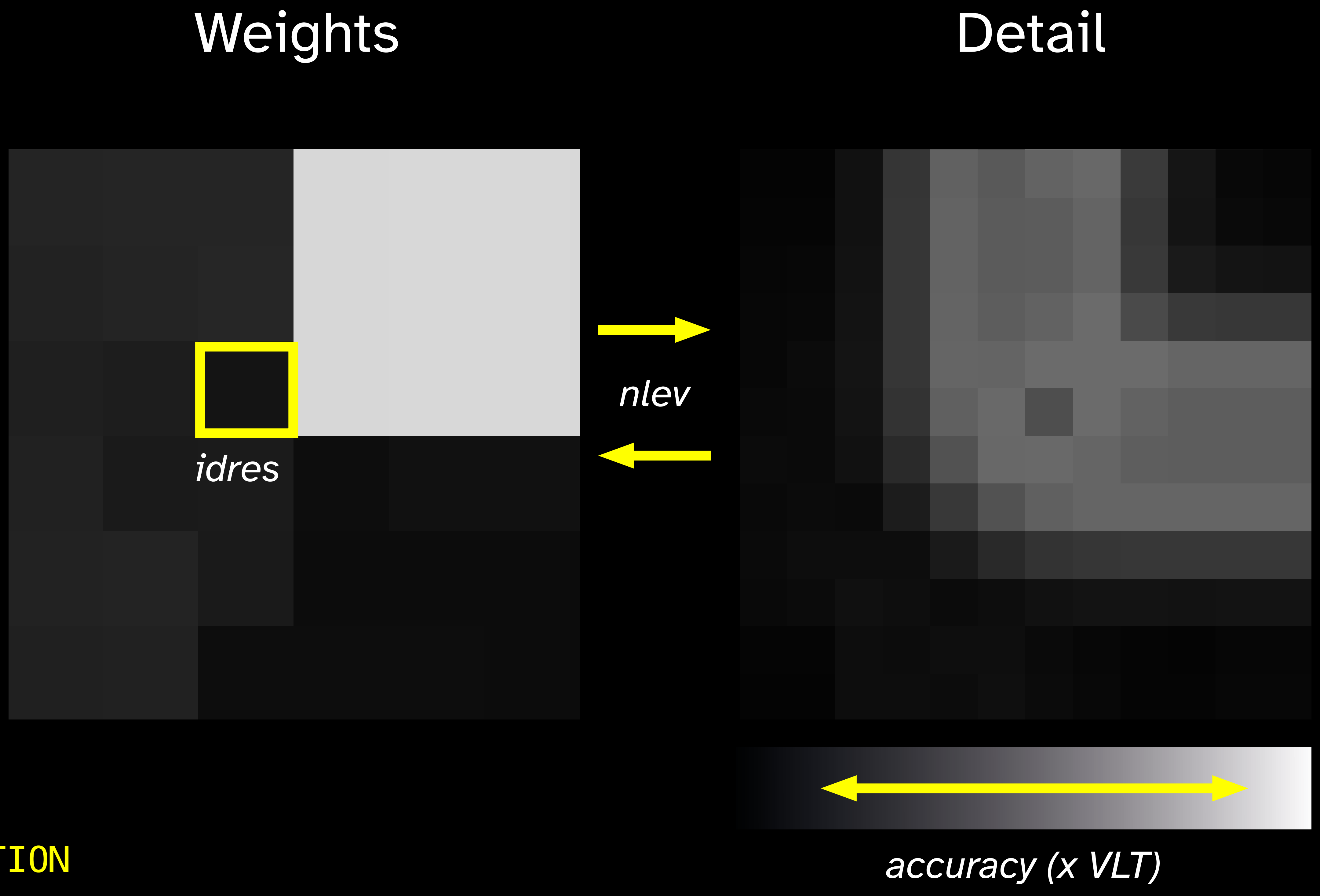
Static Points



AREA
DIRECTION
RAY

Daylight Coefficients / Sky Simulation

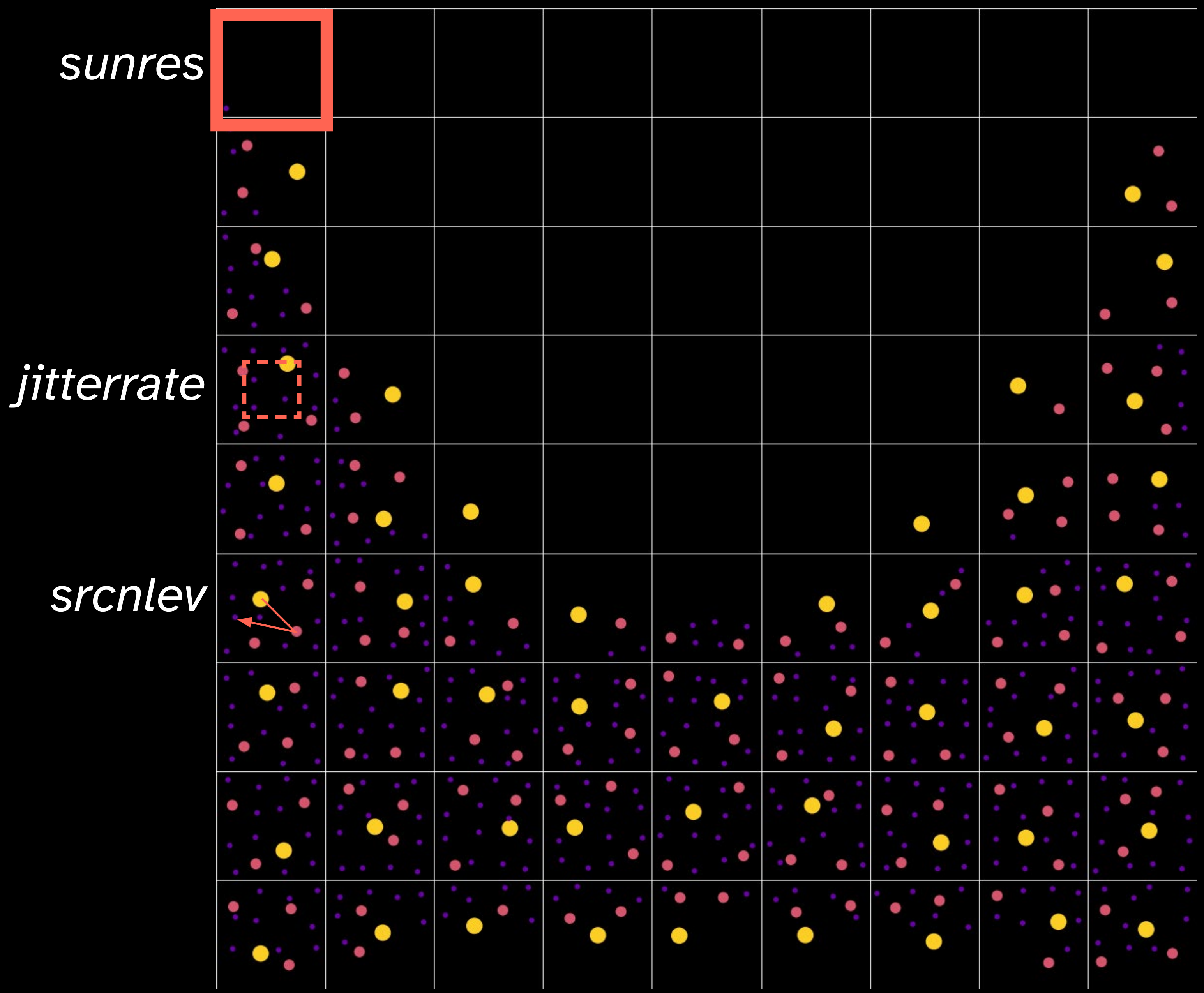
```
[raytraverse_area]
jitterrate = 0.5
ptres = 1.0
static_points = None
zone = None

[raytraverse_skyengine]
accuracy = 1.0
dcompargs = -ab 1
default_args = True
idres = 32
nlev = 5
rayargs = None
skyres = 15
vlt = 0.64

[raytraverse_skyrun]
accuracy = 1.0
edgemode = reflect
jitter = True
nlev = 3
```



AREA
DIRECTION
RAY

Sun Coefficients Simulation

```
[raytraverse_area]
jitterrate = 0.5
ptres = 1.0
static_points = None
zone = None

[raytraverse_suns]
jitterrate = 0.5
sunres = 9

[raytraverse_sunengine]
accuracy = 1.0
default_args = True
idres = 32
nlev = 6
rayargs = -ab 0
vlt = 0.64

[raytraverse_sunrun]
accuracy = 1.0
edgemode = reflect
jitter = True
nlev = 3
srcaccuracy = 1.0
srcjitter = True
srcnlev = 3
```

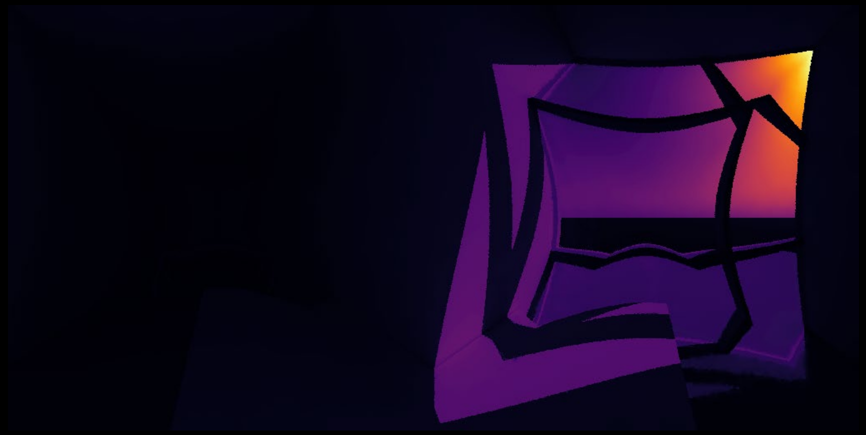


AREA
DIRECTION
RAY
SUN

Source Simulation / Scene Detail

```
[raytraverse_area]
jitterrate = 0.5
ptres = 1.0
static_points = None
zone = None

[raytraverse_sourceengine]
accuracy = 1.0
color = True
default_args = True
idres = 32
nlev = 6
rayargs = None
vlt = 1.0

[raytraverse_sourcerun]
accuracy = 1.0
distance = 0.5
edgemode = reflect
jitter = True
nlev = 3
normal = 5.0
scenedetail = False
```

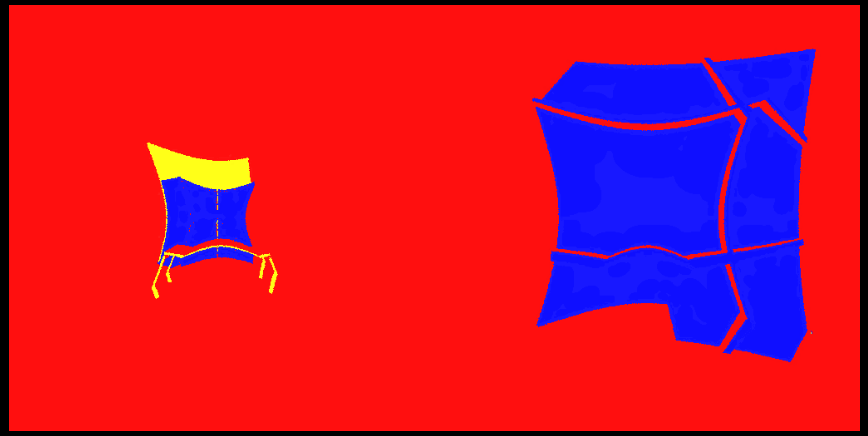
AREA
DIRECTION
RAY
SUN



Luminance

Surface Normal X

Distance



Surface Normal Y

Material ID

Surface Normal Z

basic scripting

Setup

```python
from raytraverse.scene import Scene
from raytraverse.mapper import PlanMapper, SkyMapper
from raytraverse.sky import SkyData

def setup():
    scn = Scene("room", "room.rad")
    pm = PlanMapper("plane.rad")
    skyd = SkyData("weather.epw")
    skyd.write(scene=scn)
    sm = SkyMapper(skyd.loc)
    return scn, pm, sm, skyd
```

```
raytraverse scene -out room -scene room.rad \
    area -zone plane.rad \
    skydata -wea weather.epw \
    suns -loc weather.epw --epwloc
```

raytraverse.readthedocs.io/en/stable/scene.html#scene
raytraverse.readthedocs.io/en/stable/mapper.html#planmapper
raytraverse.readthedocs.io/en/stable/sky.html#skydata

Reload:

```python
from raytraverse import api

scn, pm, skyd = api.auto_reload("room", "plane.rad")
```

raytraverse.readthedocs.io/en/stable/api.html

Sample

```python
from raytraverse.sampler import ISamplerArea, ISamplerSuns, Sensor
from raytraverse.renderer import Rcontrib, Rtrace
```

```python
def sample(scn, pm, sm):
    rcontrib = Rcontrib("-I+", scene=scn.scene)
    sensor = Sensor(rcontrib)
    sampler = ISamplerArea(scn, sensor, stype='sky')
    skylp = sampler.run(pm)

    Rcontrib.reset()
    Rcontrib("-I+ -ab 1", scene=scn.scene)
    skydlp = sampler.repeat(skylp, 'skydcomp')
    Rcontrib.reset()

    rtrace = Rtrace("-I+ -ab 0", scene=scn.scene)
    sensor = Sensor(rtrace)
    sampler = ISamplerSuns(scn, sensor)
    sunlp = sampler.run(sm, pm)
    Rtrace.reset()
```

raytraverse.readthedocs.io/en/stable/renderer.html
raytraverse.readthedocs.io/en/stable/sampler.html#sensor

Evaluate

```
from raytraverse import api

from raytraverse.integrator import SensorIntegrator
from raytraverse.lightfield import SensorPlaneKD, SunSensorPlaneKD
```

```
scn, pm, skyd = api.auto_reload("office", "../scene/plane.rad")

skylp = SensorPlaneKD(scn, None, pm, "sensor_sky")
skydlp = SensorPlaneKD(scn, None, pm, "sensor_skydcomp")
sunlp = SunSensorPlaneKD(scn, None, pm, "sensor_suns_sun")

itg = SensorIntegrator(skylp, sunlp, skydlp,
                       ptype=("skysun", "sun", "patch"), factors=(1, 1, -1))

zlr = itg.zonal_evaluate(skyd, pm)
zlr.write("results.npz")
```

Sensor based evaluation:
currently no command line

```
itg = api.get_integrator(scn, pm, simtype="1comp")

lr = itg.evaluate(skyd, [2, 2, 1.2])
lr.write("result_22.npz")
```

Full evaluation:
raytraverse evaluate

Pull

```python
import numpy as np
from raytraverse import api
from raytraverse.lightfield import ZonalLightResult, LightResult
```

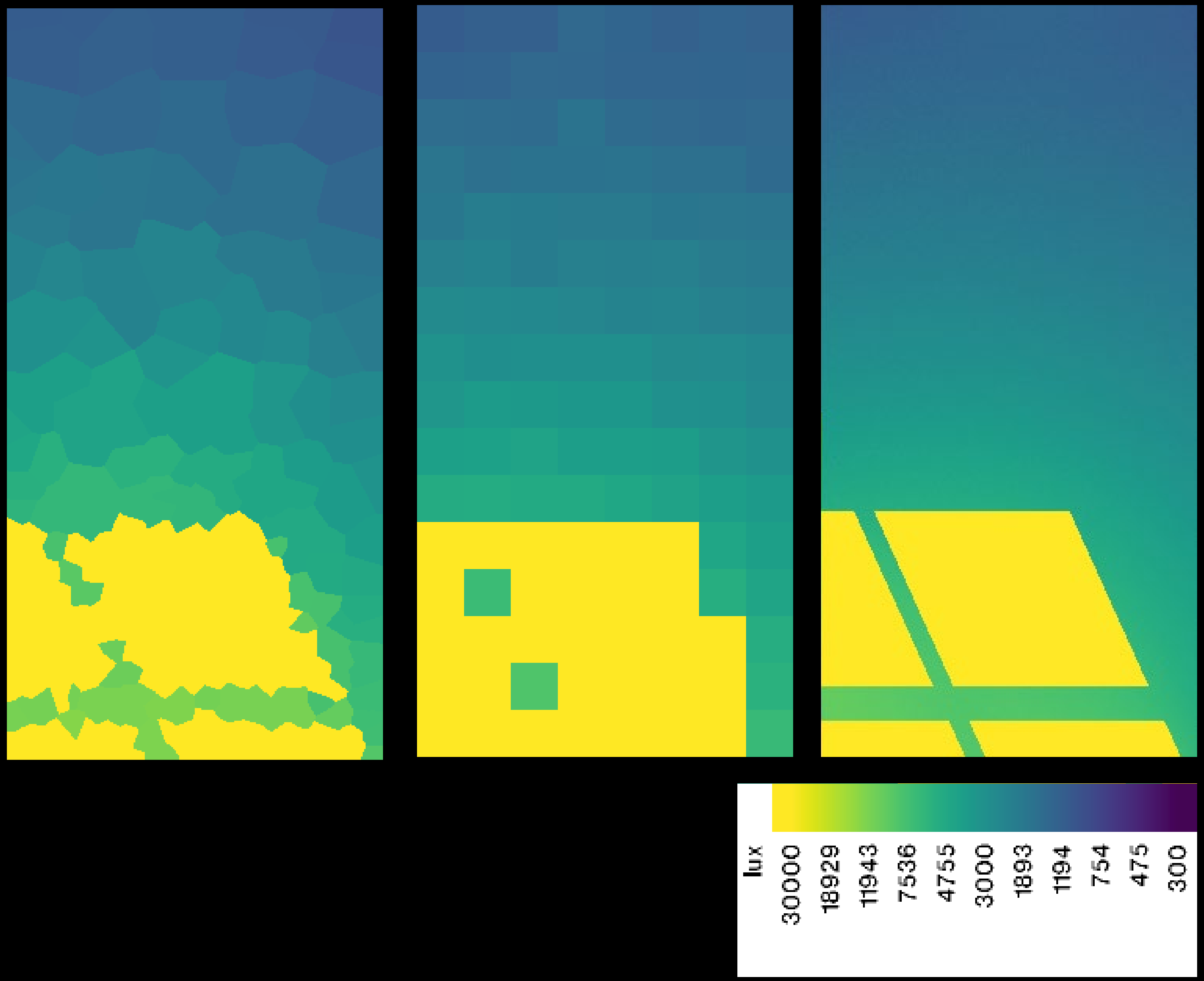
```python
scn, pm, skyd = api.auto_reload("office", "plane.rad", ptres=0.6)

lr = LightResult("sunresults_grid.npz")
skyfilter = skyd.masked_idx(np.arange(1896, 1920))
lr.pull2planhdr("plane.rad", "testg", sky=skyfilter)

zlr = ZonalLightResult("sunresults_full.npz")
d, l, n = zlr.pull("metric", metric=[zlr.axis("metric").index("illum")])
print(d.shape)  # (572262, 1)
lr2 = zlr.rebase(pm.point_grid(False, level=2))
d, l, n = lr2.pull("metric", metric=[lr2.axis("metric").index("illum")])
print(d.shape)  # (2179584, 1)
```

```
raytraverse pull \
-lr sunresults_grid.npz \
-skyfill office/skydata.npz \
-skyfilter 1896:1920 \
-ofiles pullg --gridhdr \
-imgzone plane.rad
```

exercise #3

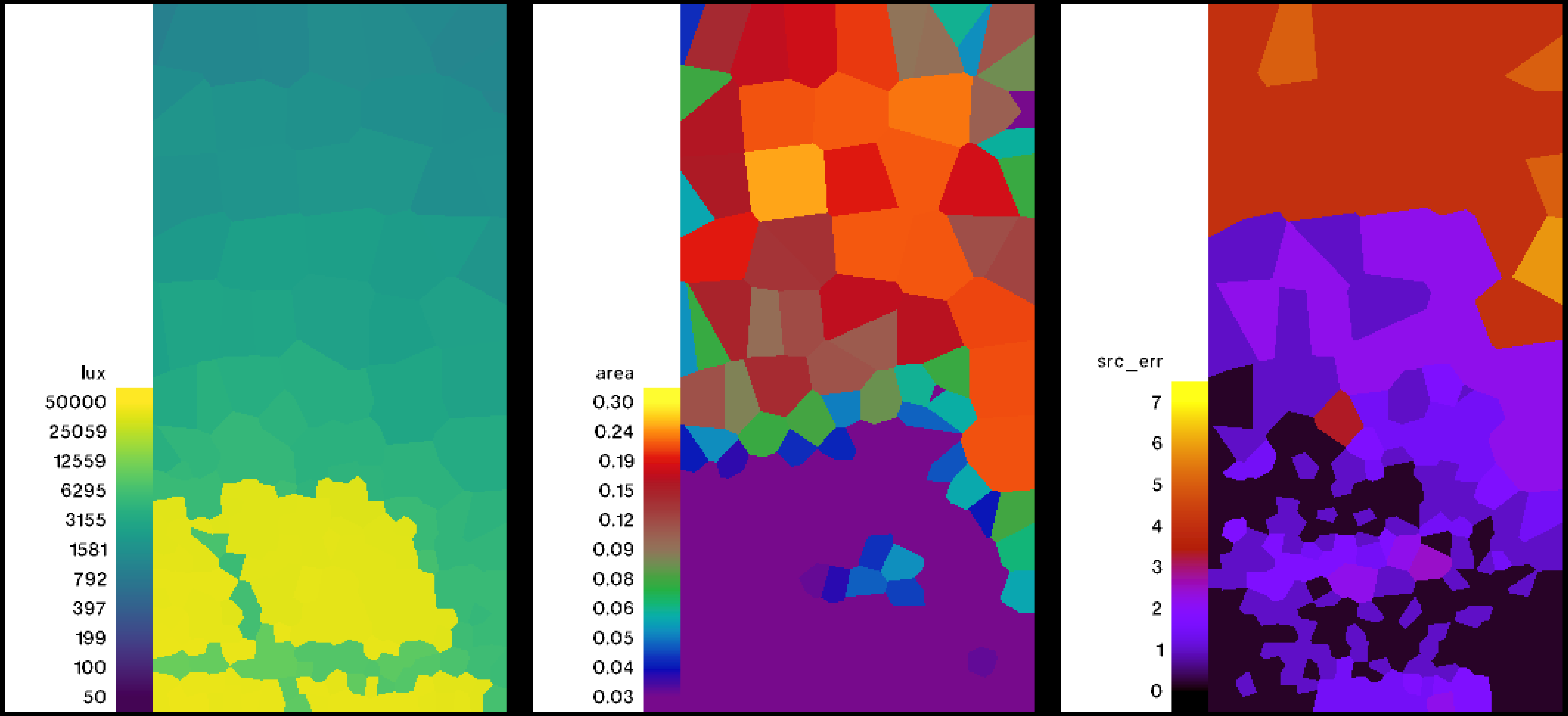
exercise #3: Sensor Point Simulation

- Use Sensor points to sample illuminance in a space using 3 component method
- Synthesize illuminance plans for a range of dates
- Evaluate direct sun illuminance annually to compute ASE
- compare different interpolation resolutions with zonal analysis



exercise #3.a: sample

```
# run sensor sampler for 3 components and evaluate for mar 21st.
./sample.py
# use command line to pull to hdr images
raytraverse pull -lr results_0321.npz -imgzone plane.rad --gridhdr -ofiles plan \
 -metricfilter "illum src_err src_idx area"
```



exercise #3.a: evaluate

```
# reload sampling results and run full annual evaluation of sun contributions
./evaluate.py
```

```
(dev39) Stephens-MacBook-Pro:ex3 stephenwasilewski$ raytraverse pull -lr sunresults_full.npz --info
ZonalLightResult sunresults_full.npz:
Has 4 axes: ['sky', 'zone', 'view', 'metric']
  Axis 'sky' has length 4257:
      0 (1., 1., 9.5)
      1 (1., 1., 10.5)
      2 (1., 1., 11.5)
      3 (1., 1., 12.5)
      4 (1., 1., 13.5)
  ...
   4252 (12., 31., 12.5)
   4253 (12., 31., 13.5)
   4254 (12., 31., 14.5)
   4255 (12., 31., 15.5)
   4256 (12., 31., 16.5)
  Axis 'zone' has length 1:
      0 plan
  Axis 'view' has length 1:
      0 (0., 0., 0., 0., 0., 1.)
  Axis 'metric' has length 5:
      0 x
      1 y
      2 z
      3 area
      4 illum
```

exercise #3.c: analyze

```
# use lightresult object to calculate ASE in different ways
./calc.py
# compare convergence of different interpolation levels on ASE
cl_plot scatter ase_interpolated.txt -x_vals 0 -y_vals 1:5 -labels “raw 2.0 1.0 0.5” \
 --legend -axes ‘hours above,0,2000,fraction of floor area,0,.75’ -ms 0 -lw ‘3 0’ -ms ‘0 3’\
 -colors viridis_r -mew 0
```