An Introduction to Ladybug Tools and Pollination for Radiance users

Radiance Workshop 2023
Innsbruck, August 28

Pollination

Agenda

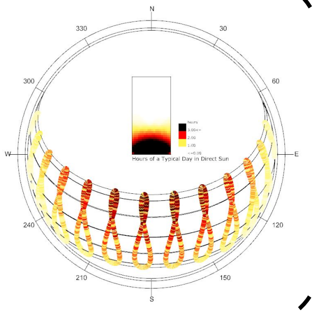
- Introduction
- Building a Honeybee Model
- Model folder
- Recipes
- break -
- honeybee-radiance-postprocess
- Pollination
- honeybee-radiance CLI
- Other useful utilities

Introduction

2013 – Ladybug

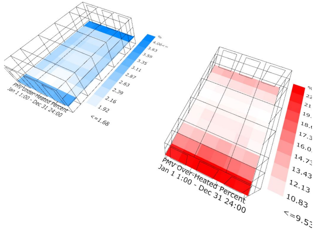
Ladybug

Climate Visualization + Analysis



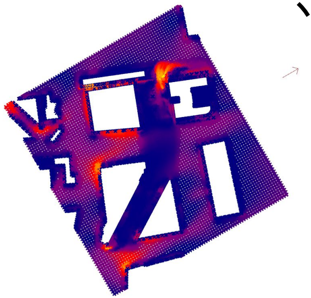
Honeybee

Building Energy, Daylight + Comfort Modelling



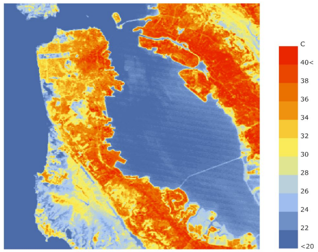
Butterfly

Airflow Modelling (CFD)



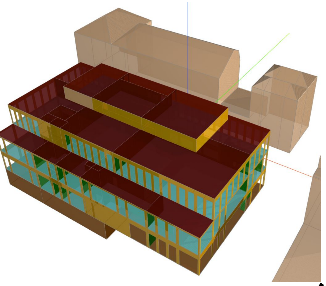
Dragonfly

Urban Modelling (urban energy, heat island, custom epw)



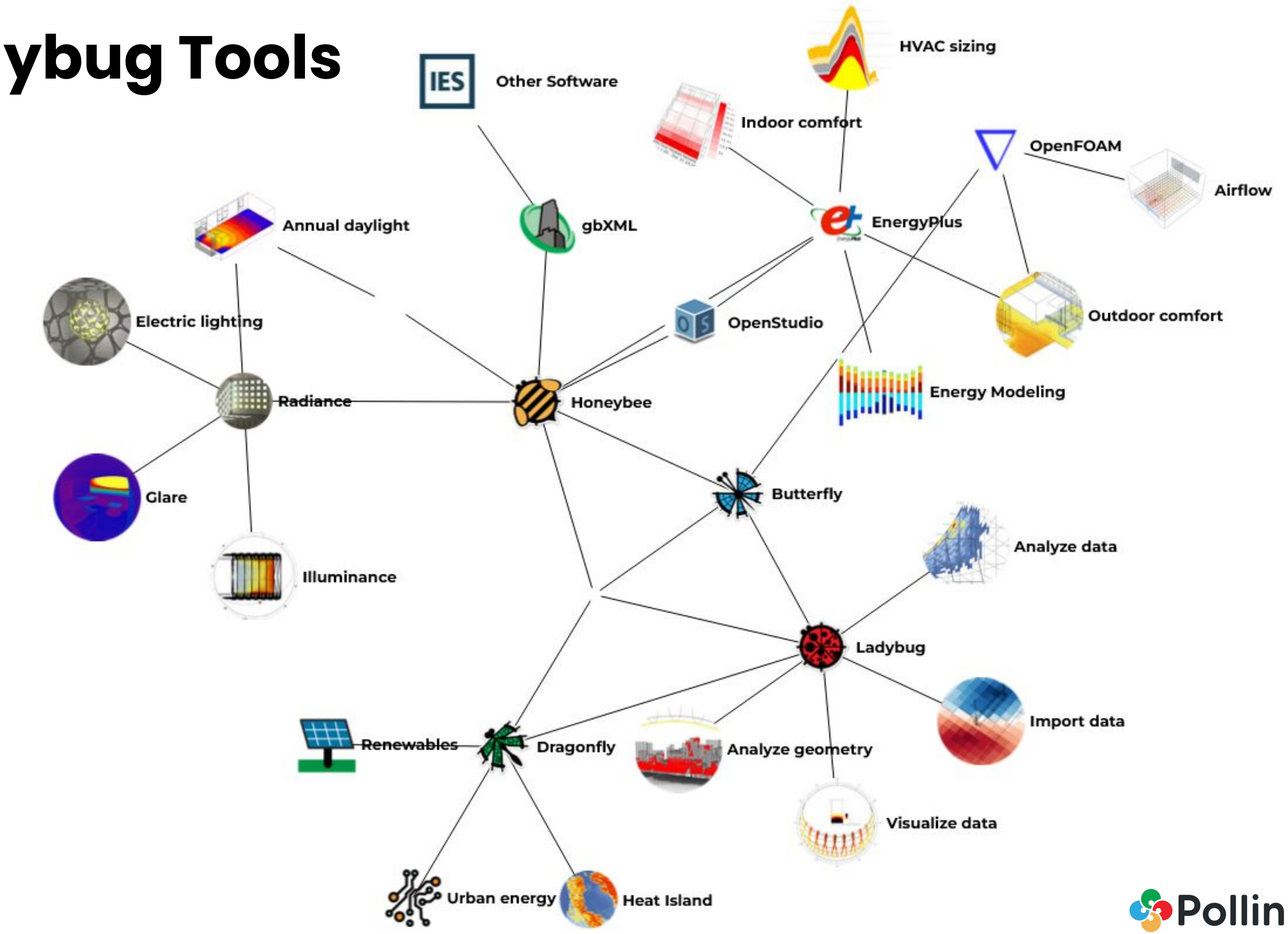
Spider

Web Visualization (sunpath, shadows, gbxml viewing/editing)

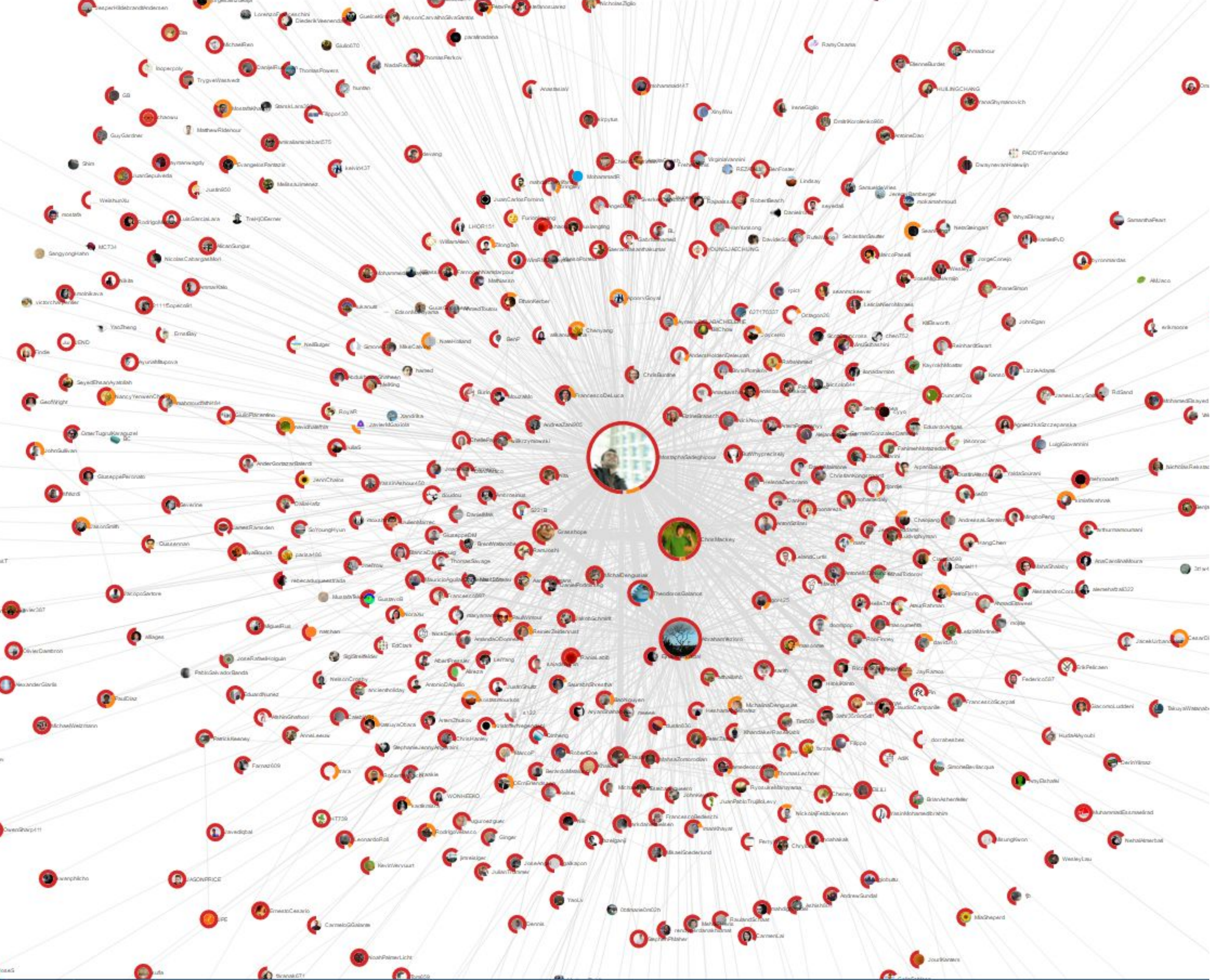


Pollination

2023 - Ladybug Tools



Pollination



Our Community

+ 650,000

Downloads - Food4Rhino

+3,000

Monthly Active Forum Members

~ 25

New Forum Topics per Day

~ 6,000

Forum Page Views per Day

Pollination

Since last time

2018

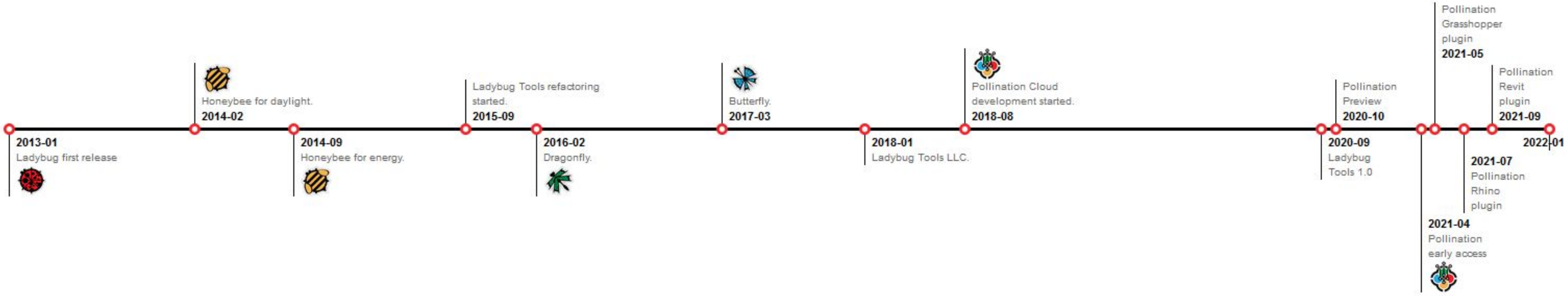
- Ladybug, Honeybee (Legacy plugins)
- Honeybee[+]

2023

- Ladybug Tools plugin

3-5 September, Loughborough University, UK
2018 International Radiance Workshop

From Ladybug Tools to Pollination Ecosystem



2013-01
Ladybug first release

Honeybee for daylight.
2014-02

2014-09
Honeybee for energy.

Ladybug Tools refactoring started.
2015-09

2016-02
Dragonfly.

Butterfly.
2017-03

2018-01
Ladybug Tools LLC.

Pollination Cloud development started.
2018-08

2020-09
Ladybug Tools 1.0

Pollination Preview
2020-10

2021-04
Pollination early access

Pollination Grasshopper plugin
2021-05

2021-07
Pollination Rhino plugin

Pollination Revit plugin
2021-09

2022-01

Pollination

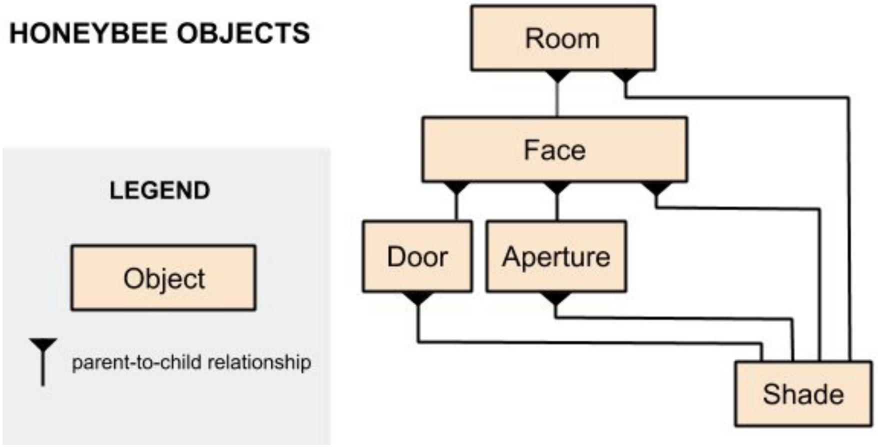
Core libraries

- honeybee
  - Core honeybee library containing building geometry objects
- honeybee-radiance
  - honeybee extension for simulation with Radiance
- honeybee-radiance-command
  - Wrapper around Radiance commands, used by honeybee-radiance
- honeybee-radiance-postprocess
  - Postprocess Radiance results
- honeybee-radiance-folder
  - Read, write and validate honeybee-radiance folders

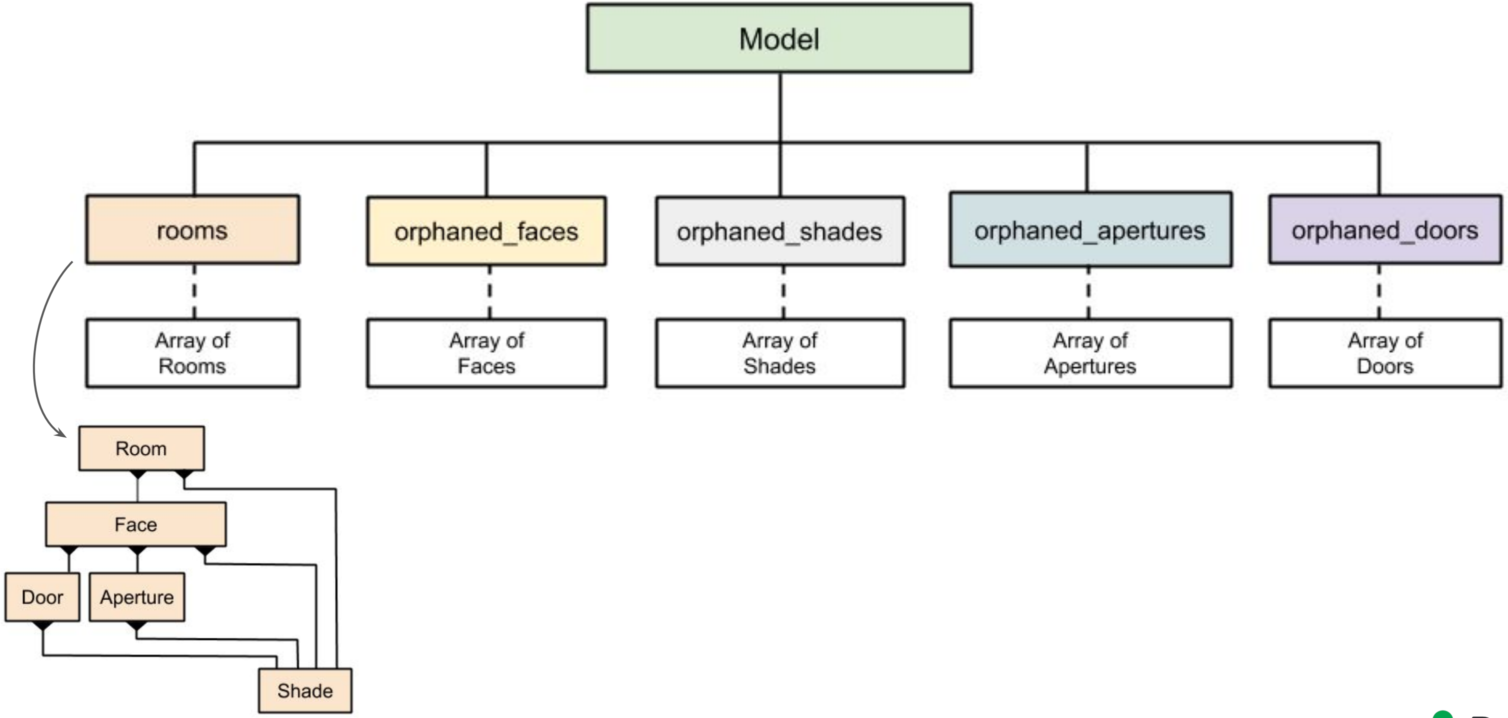
honeybee-schema

The five geometry objects

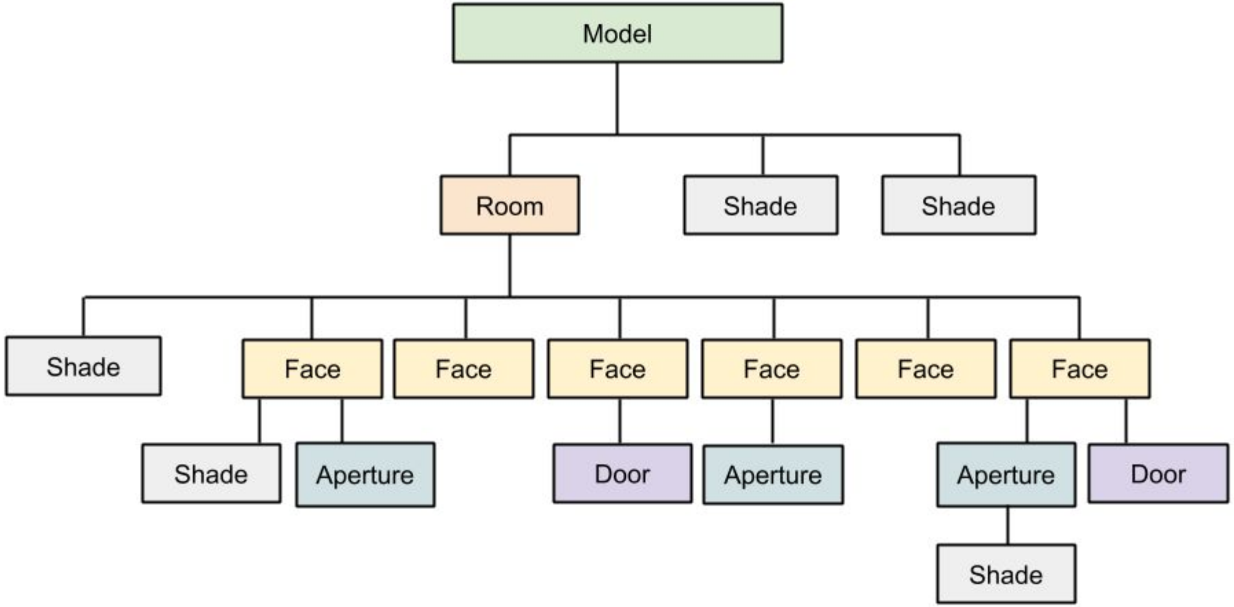
- Room
- Face
- Aperture
- Door
- Shade



honeybee-schema



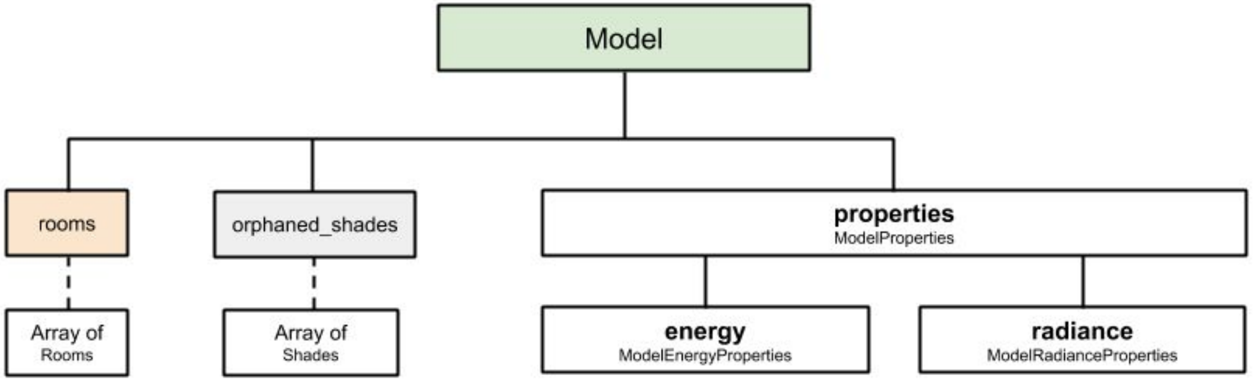
honeybee-schema



Pollination

honeybee-schema

- Extending the model-schema with properties for simulation engines.
  - EnergyPlus/OpenStudio and Radiance.
- Properties can be assigned to geometry.
  - E.g. modifiers, dynamic_group_identifier, states.



Pollination

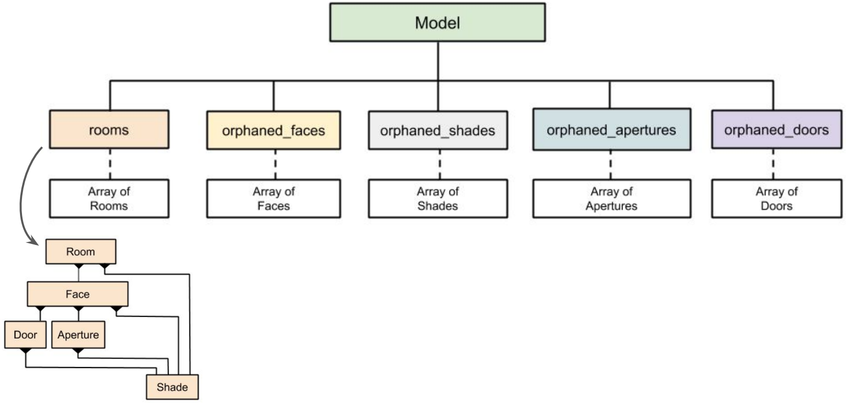
Building a Honeybee Model

(in Grasshopper)

Building a Honeybee Model (in Grasshopper)

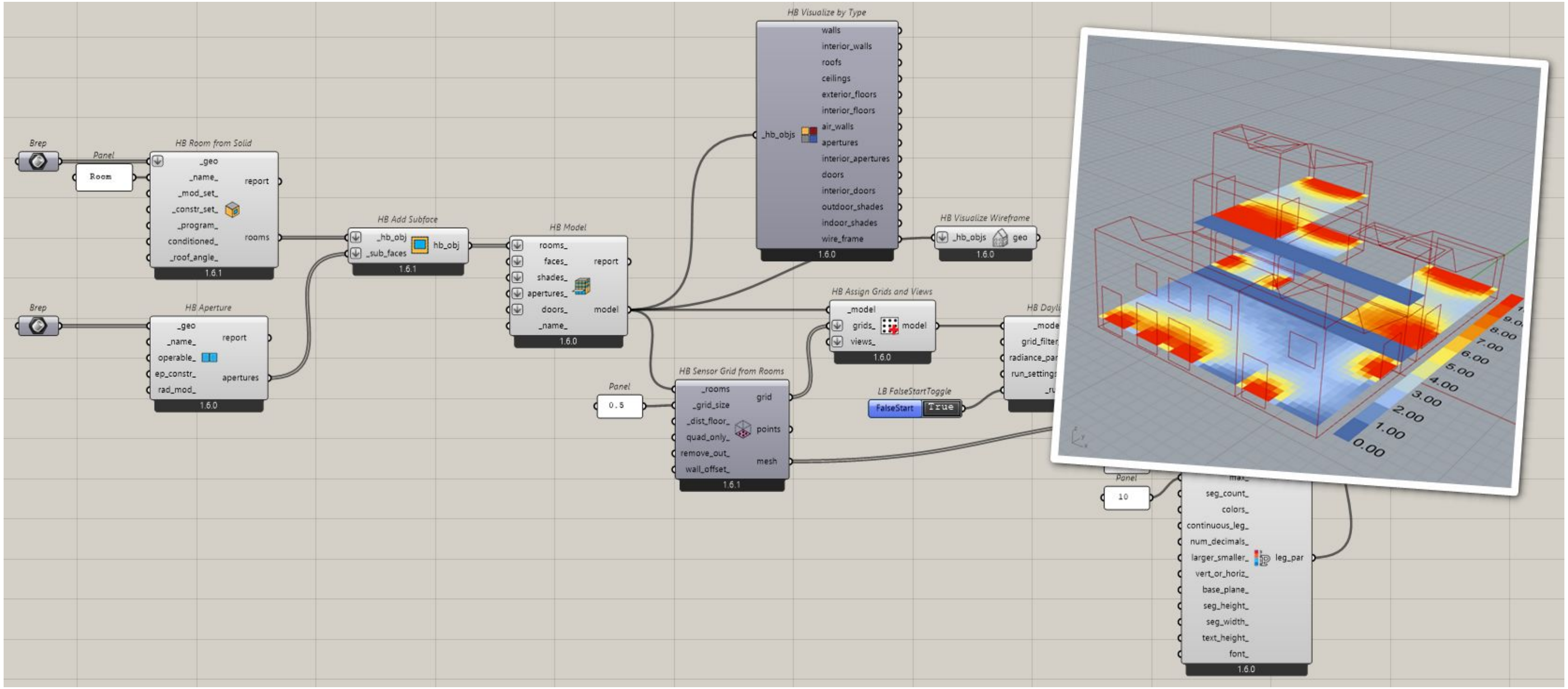
There are two approaches to building a HB Model:

- Surface by surface
- Room-based



Pollination

Run a daylight factor study

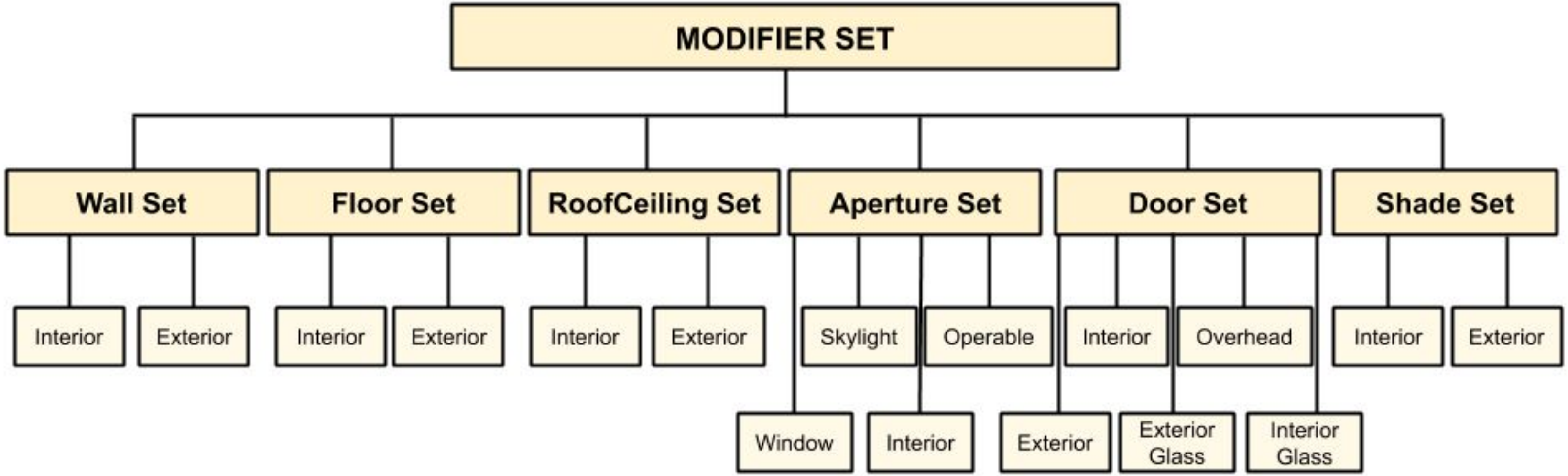


Adding modifiers

- Surface by surface
  - Adding modifiers surface by surface.
- Room-based
  - Adding modifiers by using a ModifierSet.

ModifierSet

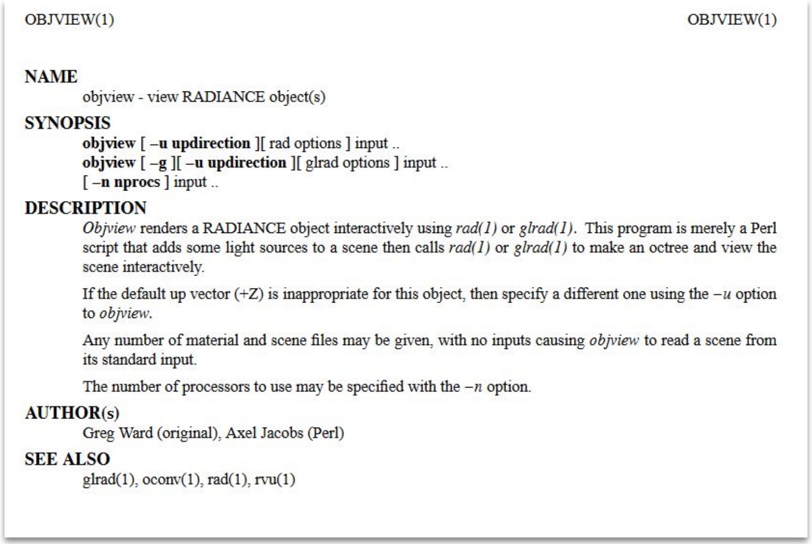
- A ModifierSet is a collection of modifiers that can be applied to Rooms.



Pollination

Check Scene

- Use the Check Scene component to visualize the model with Radiance.
  - Can be used to check if the HB Model seems correct when translated to a Radiance model.
- Somewhat similar to objview.



OBJVIEW(1) OBJVIEW(1)

NAME

objview - view RADIANCE object(s)

SYNOPSIS

objview [**–u updirection**][rad options] input ..
objview [**–g**][**–u updirection**][glrad options] input ..
[**–n nprocs**] input ..

DESCRIPTION

Objview renders a RADIANCE object interactively using *rad(1)* or *glrad(1)*. This program is merely a Perl script that adds some light sources to a scene then calls *rad(1)* or *glrad(1)* to make an octree and view the scene interactively.

If the default up vector (+Z) is inappropriate for this object, then specify a different one using the *–u* option to *objview*.

Any number of material and scene files may be given, with no inputs causing *objview* to read a scene from its standard input.

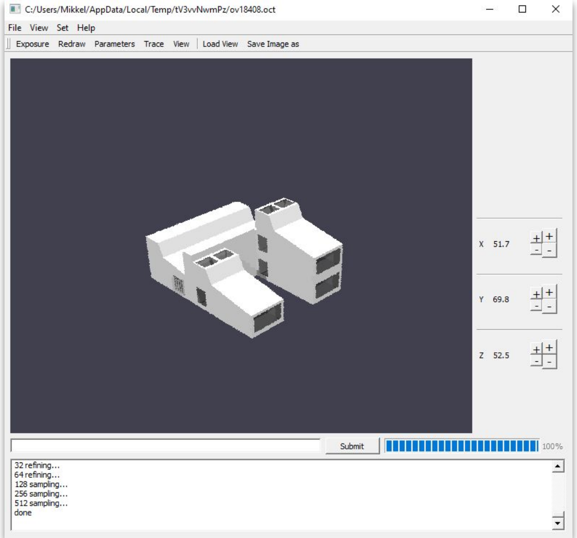
The number of processors to use may be specified with the *–n* option.

AUTHOR(s)

Greg Ward (original), Axel Jacobs (Perl)

SEE ALSO

glrad(1), oconv(1), rad(1), rvu(1)



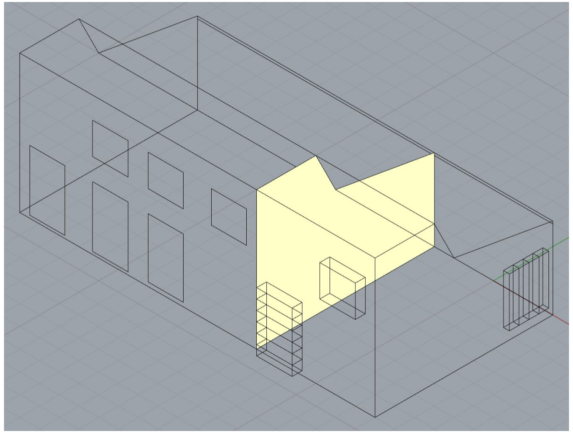
C:/Users/Mikkel/AppData/Local/Temp/tV3vvNwmPz/ov18408.oct
File View Set Help
Exposure Redraw Parameters Trace View Load View Save Image as
X 51.7
Y 69.8
Z 52.5
Submit
100%

```
32 refining…
64 refining…
128 sampling…
256 sampling…
512 sampling…
done
```

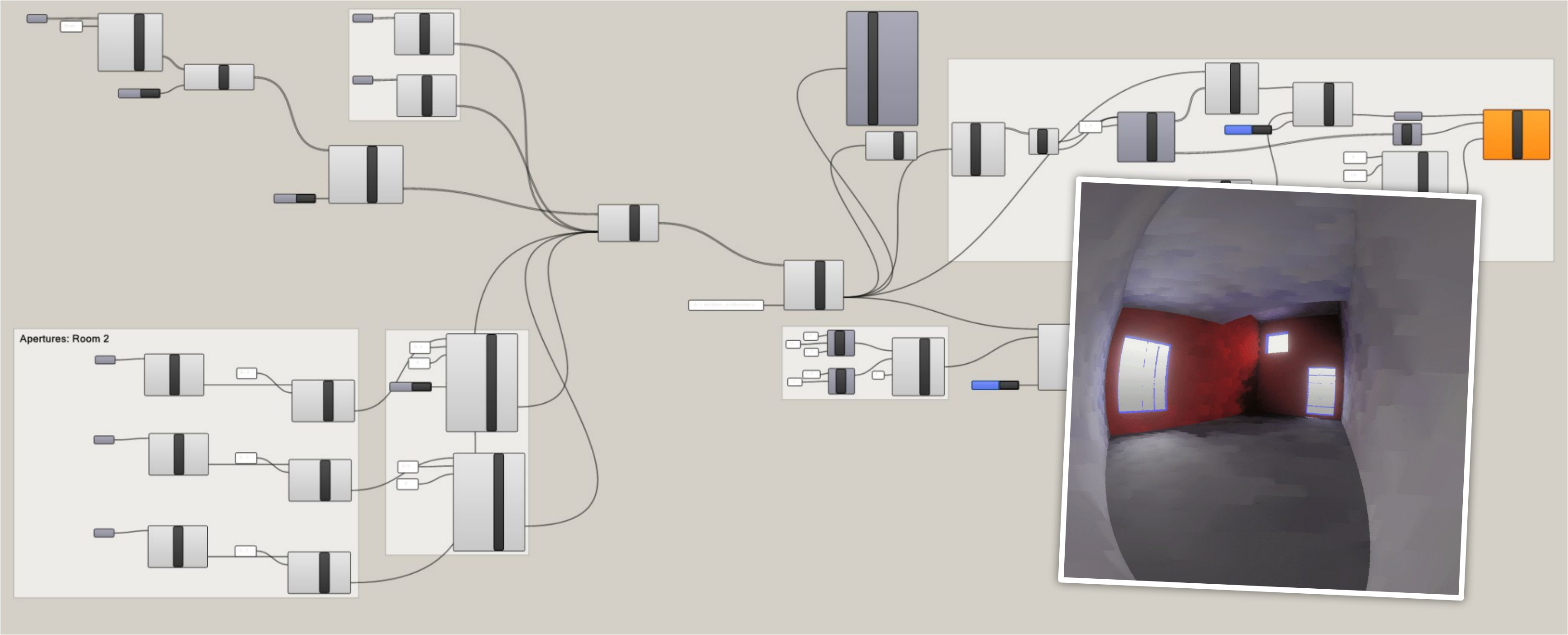
AirBoundary

- AirBoundaries can be used to create an invisible wall between two Rooms.
- The AirBoundary is using a trans modifier, however, …
- … when writing the HB Model to a Radiance model folder, the AirBoundaries are excluded.

```
void trans air_boundary
0
0
7 1.0 1.0 1.0 0.0 0.0 1.0 1.0
```



Adding and modifying modifiers

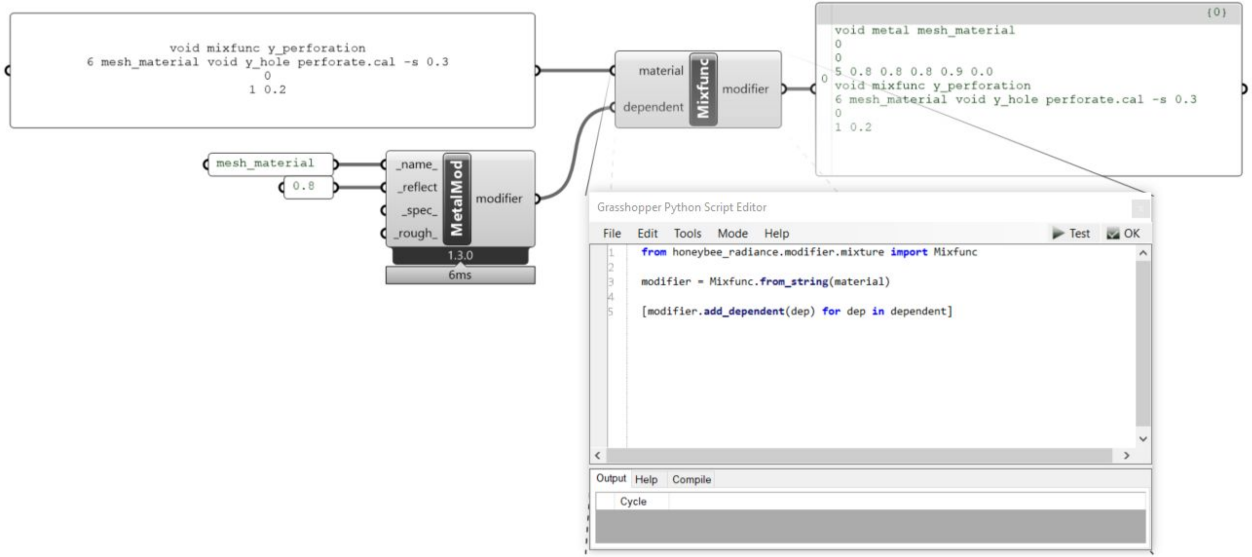


Creating custom modifiers

- Not all Radiance modifiers have a Python class in honeybee-radiance.
  - Some are just implemented as generic Radiance primitives.
  - Modifiers used for less generic Radiance studies, e.g., BRTDfunc, Dielectric, Illum, Mixfunc, and Transfunc are generic Radiance primitives.

Creating custom modifiers

- Even fewer have their own honeybee-radiance Grasshopper component.
    - However, modifiers can be created by using the honeybee-radiance core library.

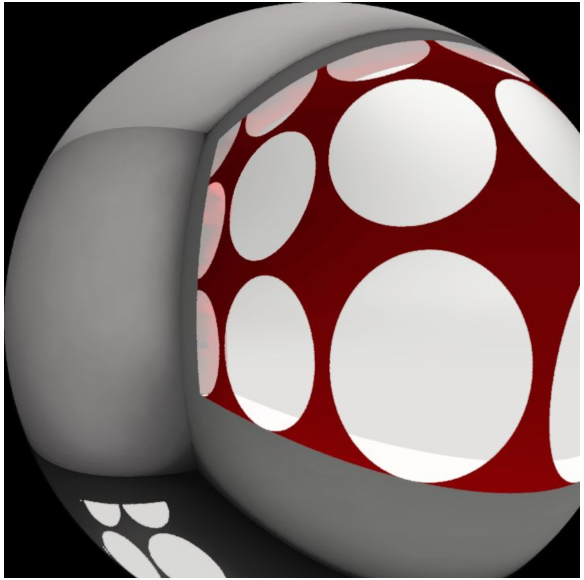


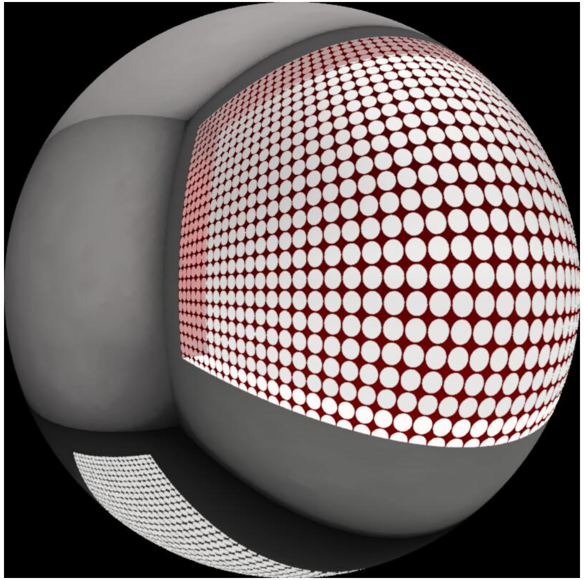
```
void mixfunc y_perforation
6 mesh_material void y_hole perforate.cal -s 0.3
0
1 0.2
```

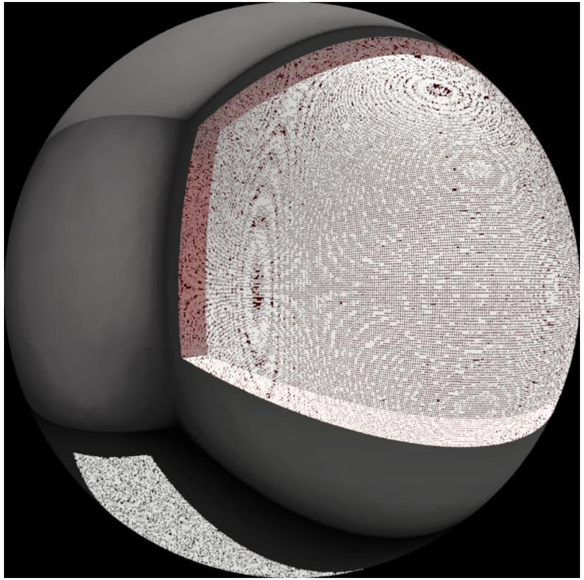
```
void metal mesh_material
0
0
5 0.8 0.8 0.8 0.9 0.0
void mixfunc y_perforation
6 mesh_material void y_hole perforate.cal -s 0.3
0
1 0.2
```

```python
from honeybee_radiance.modifier.mixture import Mixfunc

modifier = Mixfunc.from_string(material)

[modifier.add_dependent(dep) for dep in dependent]
```

Creating custom modifiers





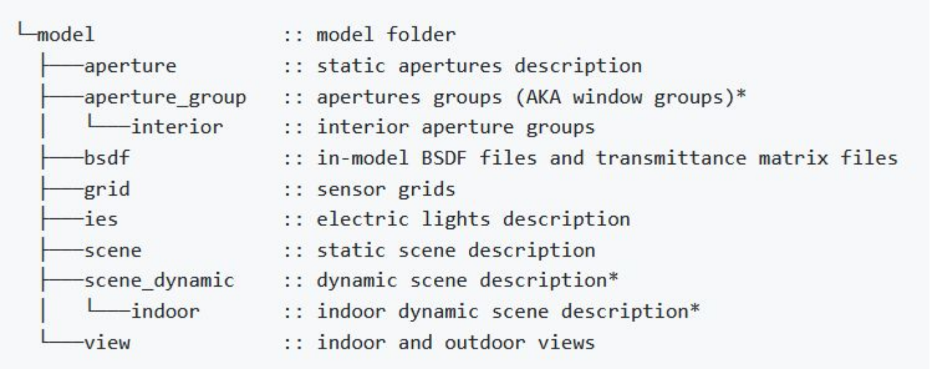


Model folder

honeybee-radiance-folder

The model folder

- The model folder is a standardized folder that describes the geometry, modifiers, and dynamic parts of the model.
- Reusable between studies.
    - For this reason there is no information about, e.g., skies.

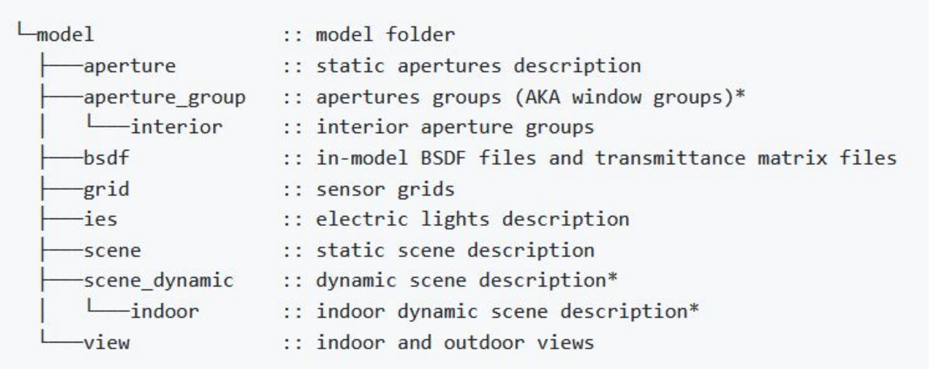


```
└─model                   :: model folder
  ├──aperture             :: static apertures description
  ├──aperture_group       :: apertures groups (AKA window groups)*
  │   └──interior         :: interior aperture groups
  ├──bsdf                 :: in-model BSDF files and transmittance matrix files
  ├──grid                 :: sensor grids
  ├──ies                  :: electric lights description
  ├──scene                :: static scene description
  ├──scene_dynamic        :: dynamic scene description*
  │   └──indoor           :: indoor dynamic scene description*
  └──view                 :: indoor and outdoor views
```

Pollination

The model folder

- *.rad files includes only the geometry.
- *.mat files includes only the modifiers.
- *.blk files includes black modifiers used for direct sunlight calculation (transparent modifiers are not black).
- For aperture groups there are also files used when calculating daylight and view matrices.



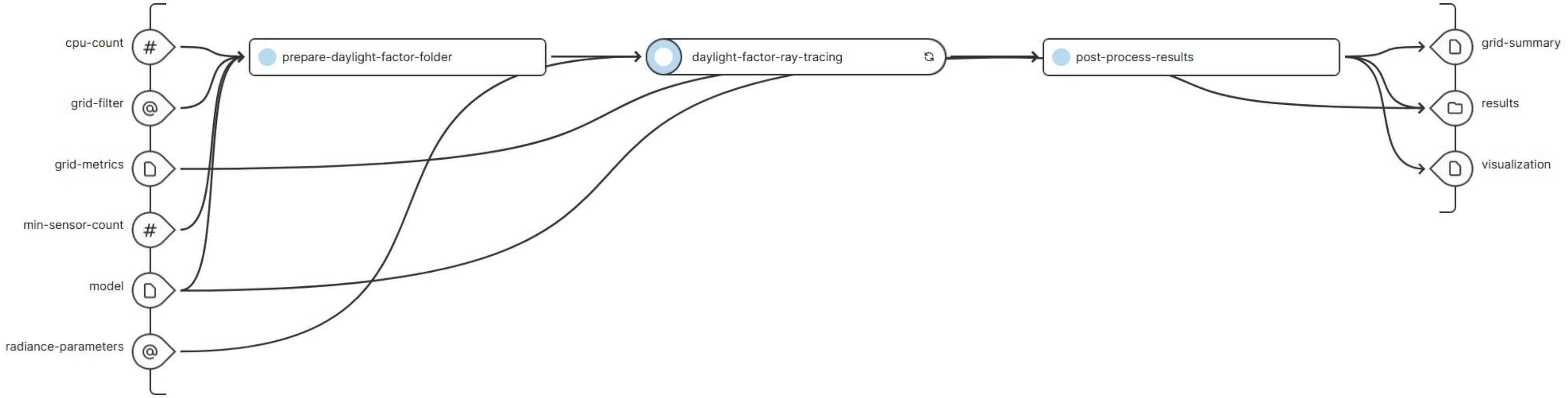
```
└─model                  :: model folder
  ├──aperture            :: static apertures description
  ├──aperture_group      :: apertures groups (AKA window groups)*
  │   └──interior        :: interior aperture groups
  ├──bsdf                :: in-model BSDF files and transmittance matrix files
  ├──grid                :: sensor grids
  ├──ies                 :: electric lights description
  ├──scene               :: static scene description
  ├──scene_dynamic       :: dynamic scene description*
  │   └──indoor          :: indoor dynamic scene description*
  └──view                :: indoor and outdoor views
```

Pollination

Recipes

What is a recipe?

- A set of tasks to create and translate files to run a specific study.
- Some recipes seem similar but have different post-processing.
- The tasks can be visualized on Pollination.
    - daylight-factor



Pollination

Recipes

rcontrib/rfluxmtx-based

- annual-daylight
- annual-daylight-enhanced
- annual-daylight-en17037
- two-phase-daylight-coefficient
- three-phase
- imageless-annual-glare
- leed-daylight-option-one
- direct-sun-hours
- annual-irradiance
- sky-irradiance
- cumulative-radiation

rtrace-based

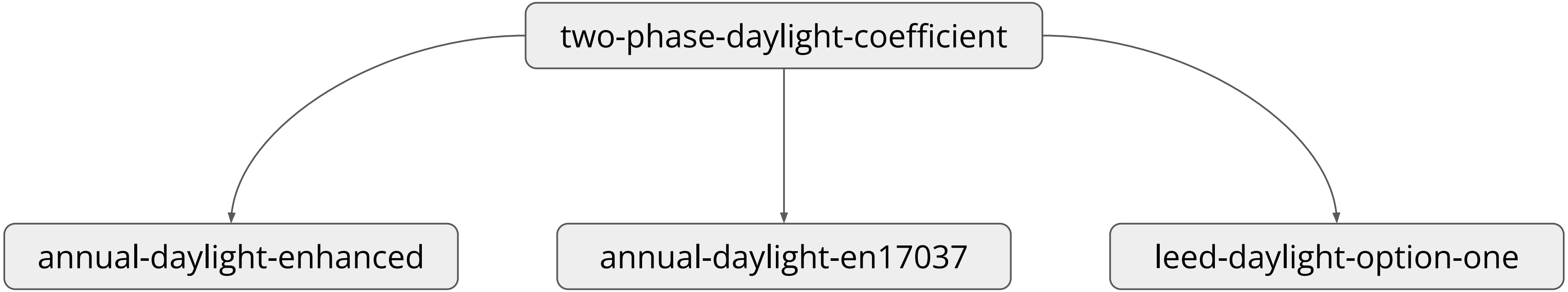
- daylight-factor
- point-in-time-grid
- leed-daylight-option-two
- sky-view

rpict-based

- point-in-time-view

Recipe dependency

- A recipe can be a dependency of another recipe.



two-phase-daylight-coefficient

annual-daylight-enhanced

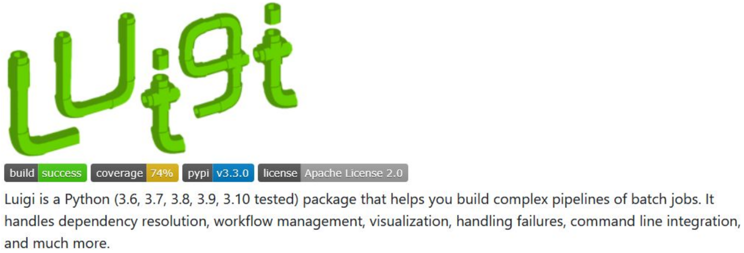
annual-daylight-en17037

leed-daylight-option-one

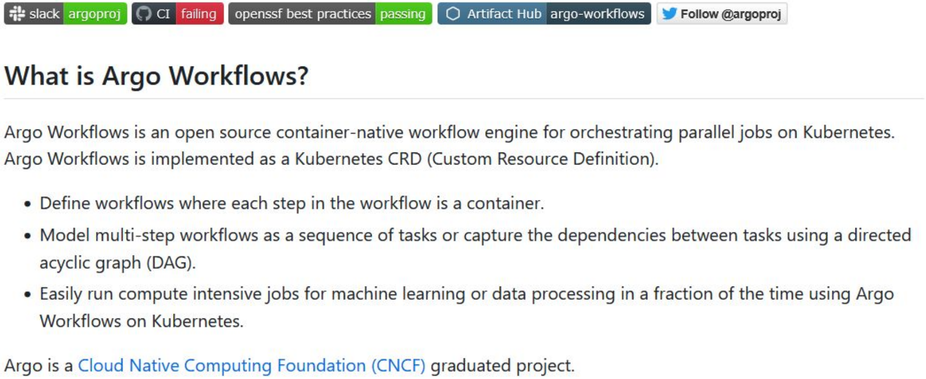
Luigi and Argo

- The recipes are using Luigi (locally) or Argo (cloud*), but for the end user this is not important.

  *Pollination Cloud: https://app.pollination.cloud/



Luigi is a Python (3.6, 3.7, 3.8, 3.9, 3.10 tested) package that helps you build complex pipelines of batch jobs. It handles dependency resolution, workflow management, visualization, handling failures, command line integration, and much more.



What is Argo Workflows?

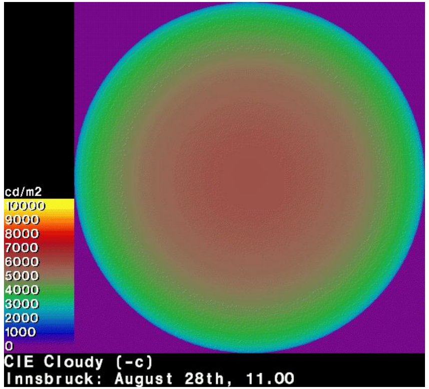
Argo Workflows is an open source container-native workflow engine for orchestrating parallel jobs on Kubernetes. Argo Workflows is implemented as a Kubernetes CRD (Custom Resource Definition).

- Define workflows where each step in the workflow is a container.
- Model multi-step workflows as a sequence of tasks or capture the dependencies between tasks using a directed acyclic graph (DAG).
- Easily run compute intensive jobs for machine learning or data processing in a fraction of the time using Argo Workflows on Kubernetes.

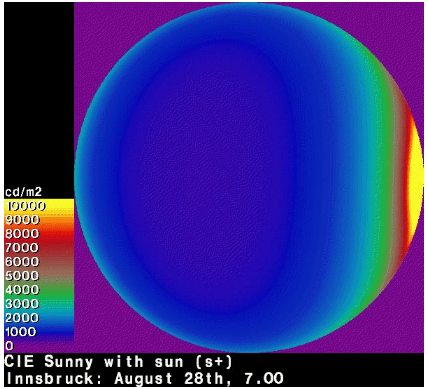
Argo is a Cloud Native Computing Foundation (CNCF) graduated project.

Skies

- Several ways to create skies.
  - Certain Illuminance.
  - CIE Standard Sky.
  - Climate based Sky.
  - Custom Sky.

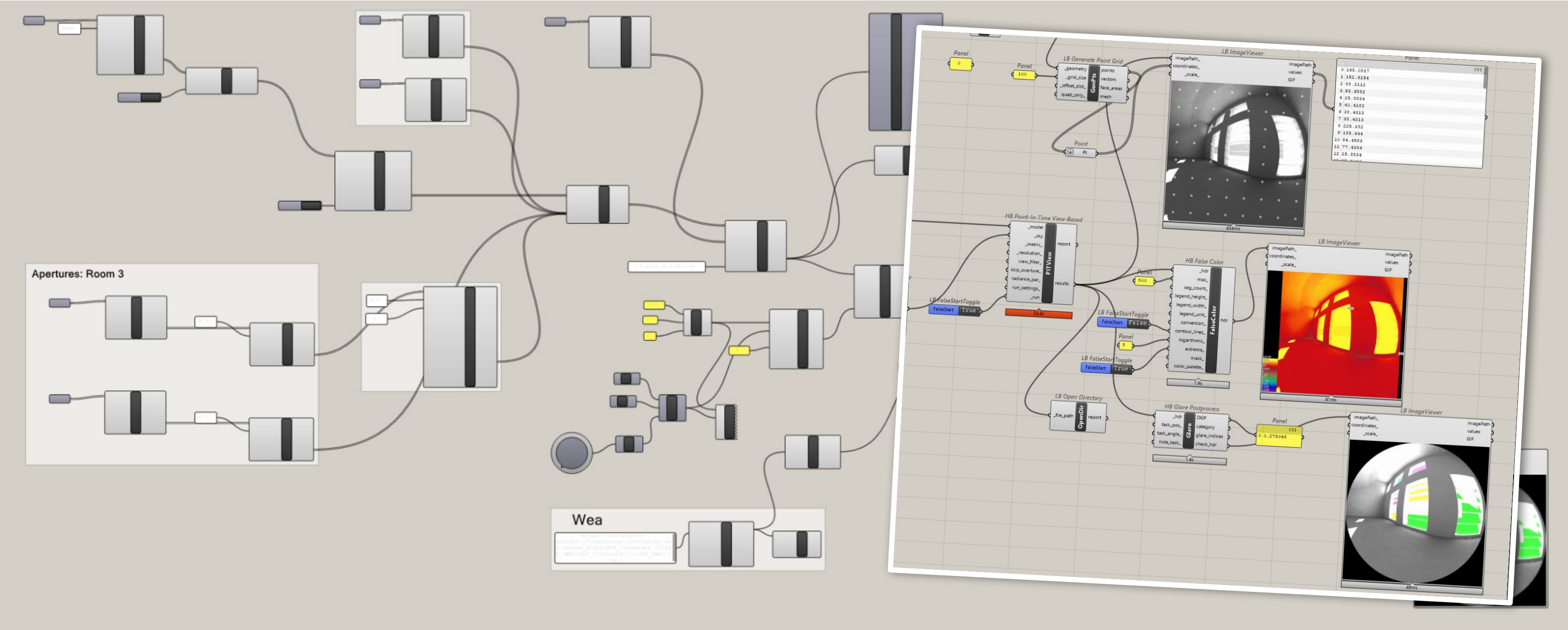


CIE Cloudy (-c)
Innsbruck: August 28th, 11.00



CIE Sunny with sun (s+)
Innsbruck: August 28th, 7.00

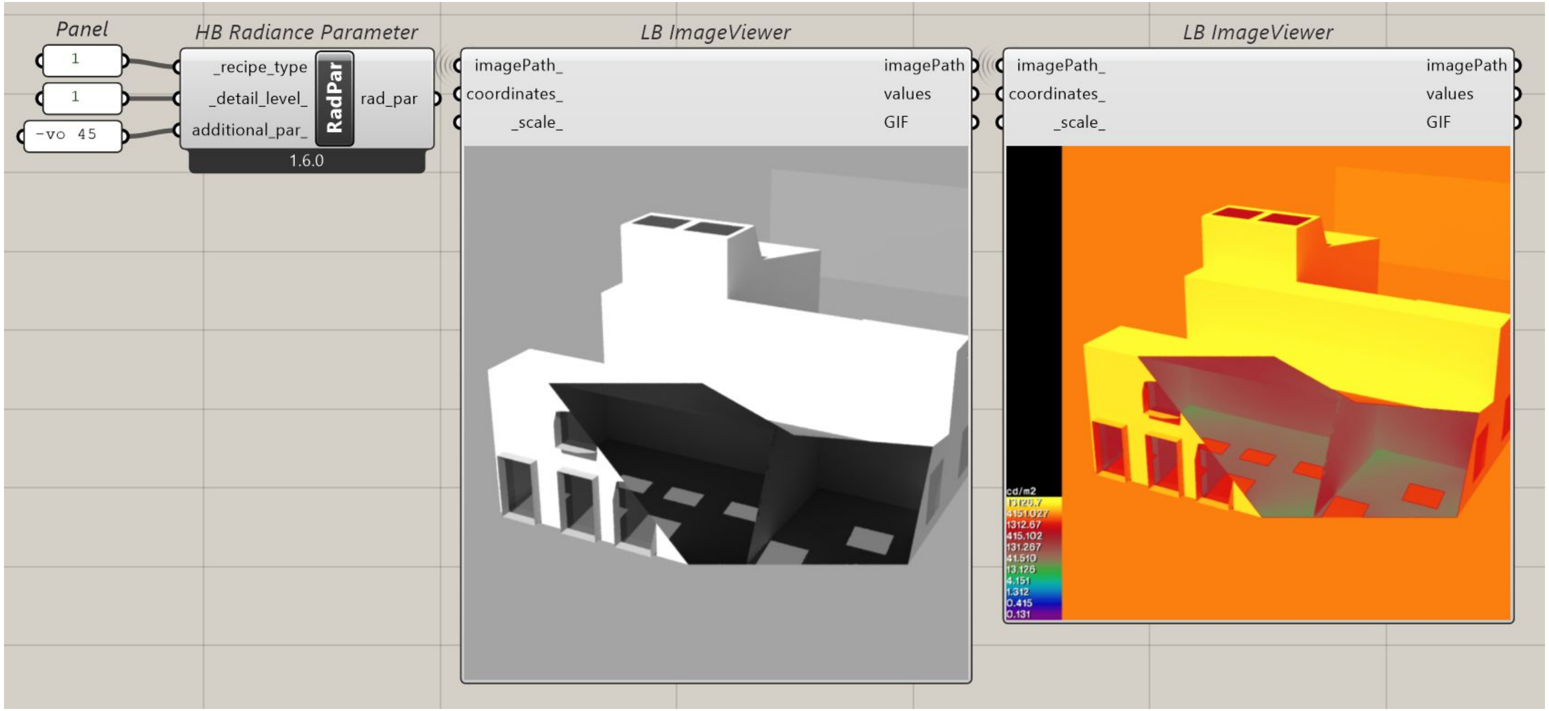
point-in-time-view



```
{0}
0 165.0917
1 152.6154
2 88.2112
3 63.6882
4 25.8834
5 40.4182
6 38.4313
7 38.4313
8 225.182
9 185.444
10 94.4583
11 77.4354
12 25.8834
```

```
{0}
0 0.275046
```

Using clipping planes (-vo and -va)



Panel

```
1
1
-vo 45
```

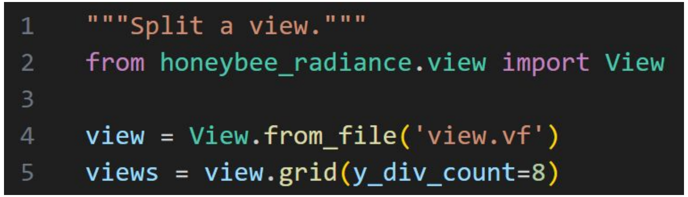
HB Radiance Parameter

- _recipe_type: 1
- _detail_level_: 1
- additional_par_: -vo 45

Splitting the view

view.vf

rvu -vtv -vp 12.0 1.0 3.0 -vd -2.0 2.0 0.0 -vu 0.0 0.0 1.0 -vh 90.0 -vv 60.0

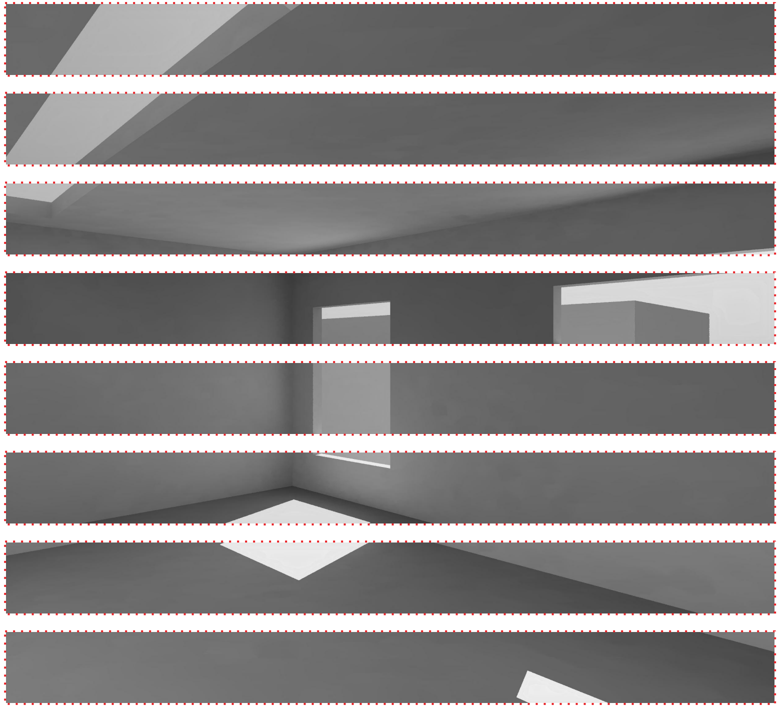


```python
"""Split a view."""
from honeybee_radiance.view import View

view = View.from_file('view.vf')
views = view.grid(y_div_count=8)
```

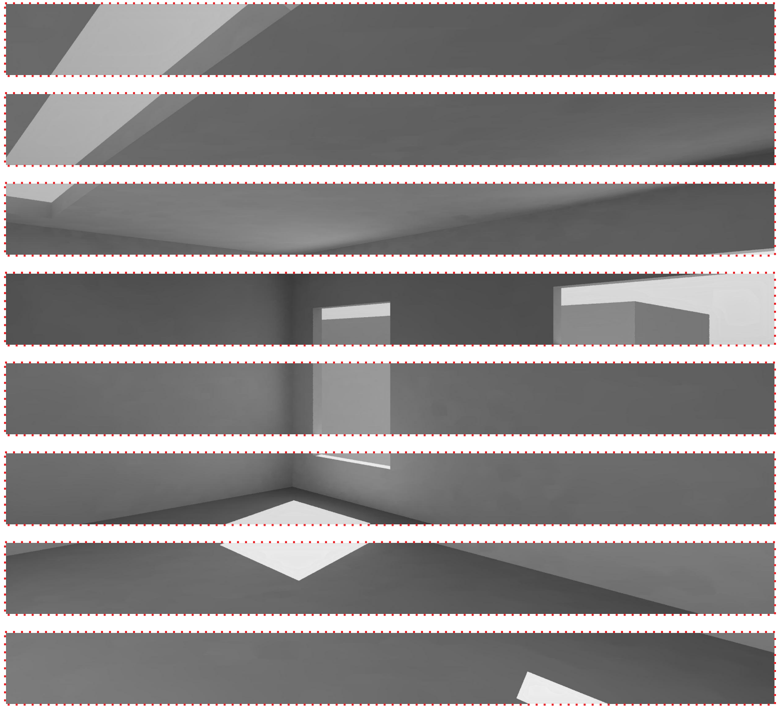
```
-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl 3.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl 2.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl 1.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl 0.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl -0.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl -1.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl -2.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl -3.5
```

Splitting the view

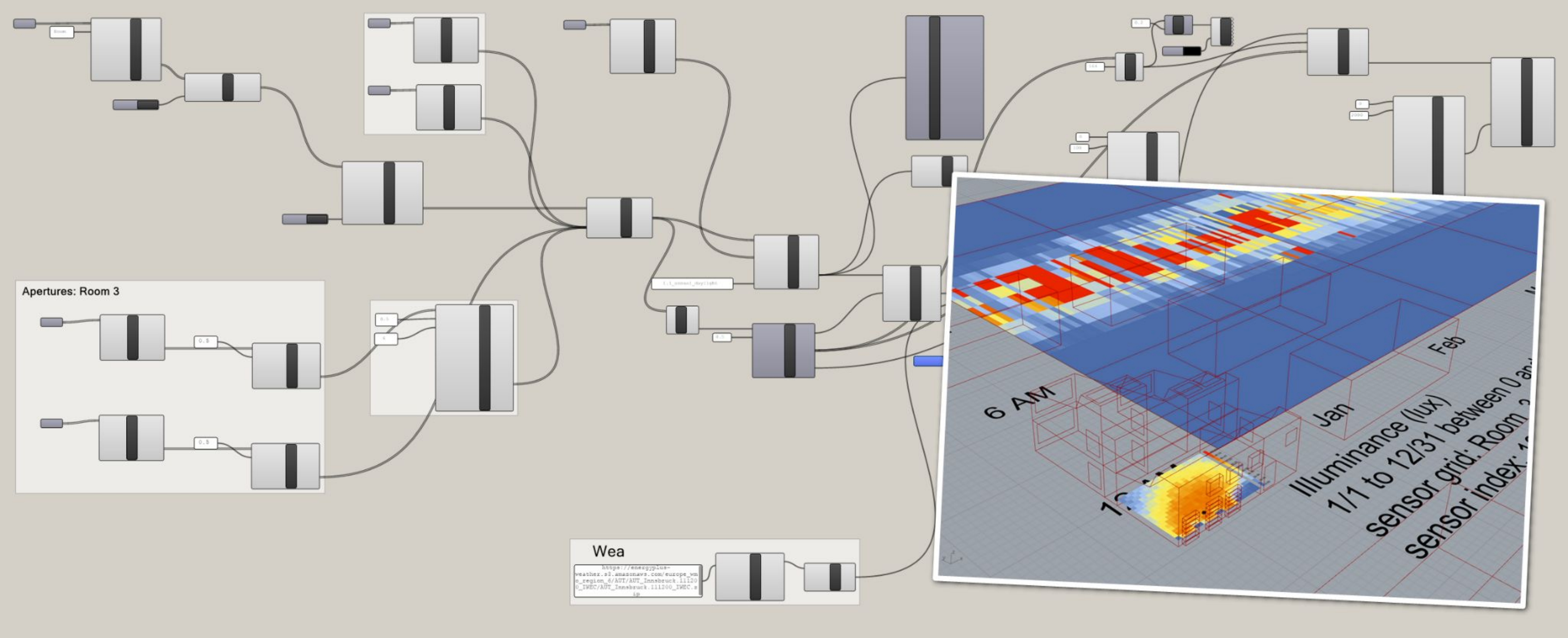
```
-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl 3.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl 2.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl 1.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl 0.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl -0.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl -1.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl -2.5

-vtv -vp 12 1 3 -vd -2 2 0 -vu 0 0 1 -vh 76.2921 -vv 8.25562 -vo 0 -va 0 -vs 0 -vl -3.5
```



Splitting the view



annual-daylight


Apertures: Room 3
0.5
0.5
1.1_annual_daylight
Wea
https://energyplus-weather.s3.amazonaws.com/europe_wmo_region_6/AUT/AUT_Innsbruck.111200_IWEC/AUT_Innsbruck.111200_IWEC.zip
6 AM
Jan
Feb
Illuminance (lux)
1/1 to 12/31 between 0 an
sensor grid: Room 3
sensor index: 1

Pollination

honeybee-radiance-postprocess

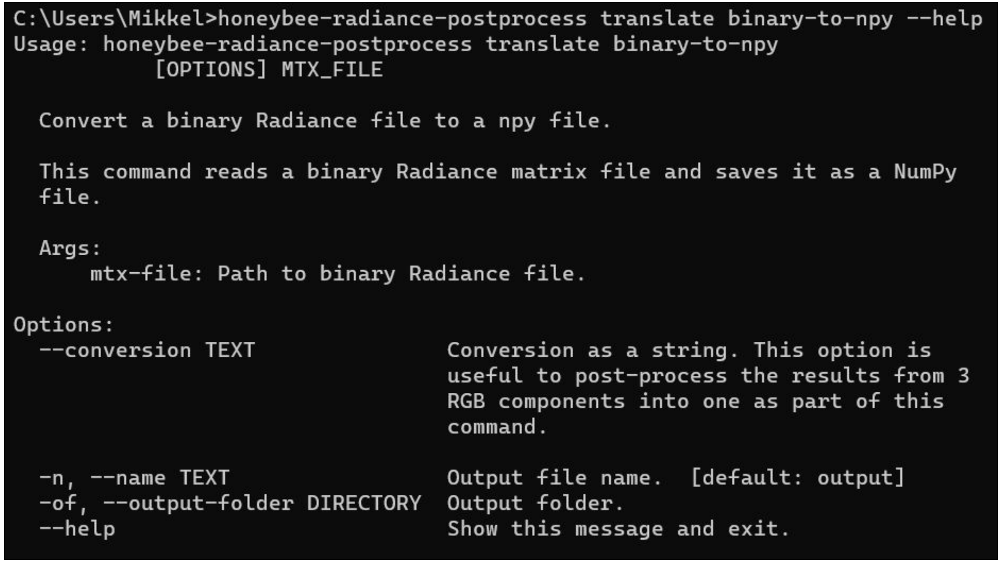
Pollination

honeybee-radiance-postprocess

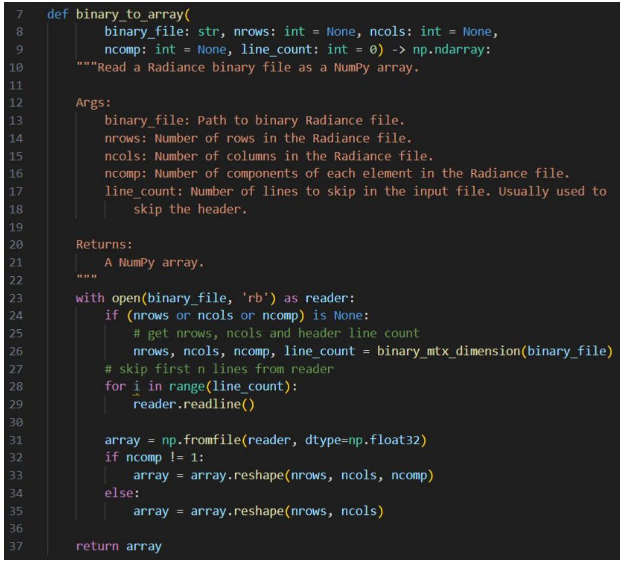
- Needed something quicker for calculation of daylight metrics.
- Uses NumPy which cannot be used in Grasshopper.
    - CLI commands are used to bypass this.
- Mainly used for annual-daylight results.

honeybee-radiance-postprocess

- The illuminance matrices from rcontrib are saved as NumPy files (.npy).



```
C:\Users\Mikkel>honeybee-radiance-postprocess translate binary-to-npy --help
Usage: honeybee-radiance-postprocess translate binary-to-npy
           [OPTIONS] MTX_FILE

  Convert a binary Radiance file to a npy file.

  This command reads a binary Radiance matrix file and saves it as a NumPy
  file.

  Args:
      mtx-file: Path to binary Radiance file.

Options:
  --conversion TEXT                Conversion as a string. This option is
                                   useful to post-process the results from 3
                                   RGB components into one as part of this
                                   command.

  -n, --name TEXT                  Output file name.  [default: output]
  -of, --output-folder DIRECTORY  Output folder.
  --help                           Show this message and exit.
```

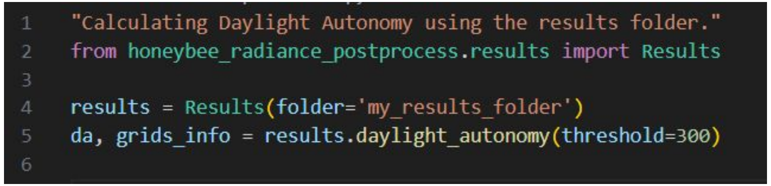


```python
def binary_to_array(
        binary_file: str, nrows: int = None, ncols: int = None,
        ncomp: int = None, line_count: int = 0) -> np.ndarray:
    """Read a Radiance binary file as a NumPy array.

    Args:
        binary_file: Path to binary Radiance file.
        nrows: Number of rows in the Radiance file.
        ncols: Number of columns in the Radiance file.
        ncomp: Number of components of each element in the Radiance file.
        line_count: Number of lines to skip in the input file. Usually used to
            skip the header.

    Returns:
        A NumPy array.
    """
    with open(binary_file, 'rb') as reader:
        if (nrows or ncols or ncomp) is None:
            # get nrows, ncols and header line count
            nrows, ncols, ncomp, line_count = binary_mtx_dimension(binary_file)
        # skip first n lines from reader
        for i in range(line_count):
            reader.readline()

        array = np.fromfile(reader, dtype=np.float32)
        if ncomp != 1:
            array = array.reshape(nrows, ncols, ncomp)
        else:
            array = array.reshape(nrows, ncols)

    return array
```

Pollination

honeybee-radiance-postprocess

- A standardized results folder is used for annual results.
- Use class methods to calculate metrics.



```
"Calculating Daylight Autonomy using the results folder."
from honeybee_radiance_postprocess.results import Results

results = Results(folder='my_results_folder')
da, grids_info = results.daylight_autonomy(threshold=300)

```

Post-processing of Aperture Groups

- A sneak peek!
- Part of HB[+] but not ported over to the Ladybug Tools Grasshopper plugin.

Post-processing of Aperture Groups

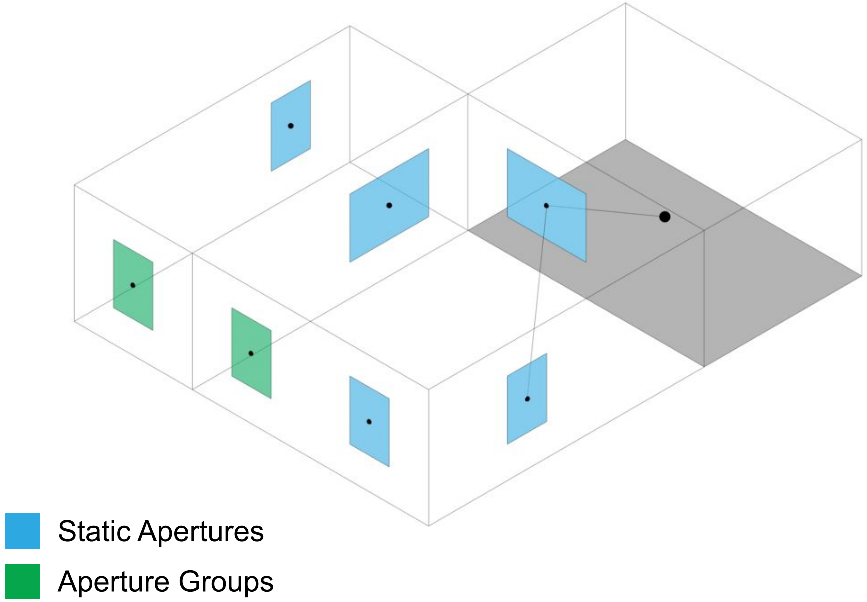
- What is an Aperture Group?
    - A group of Apertures that share a dynamic group identifier.
    - The identifier is simply a name for the group.
- Aperture Groups can have states assigned to them.
    - A state can also have additional geometry related to the Aperture, e.g., blinds, but in theory it can be any geometry.

Post-processing of Aperture Groups

- What is a light path?
  - Determines the path of light taken through interior spaces.
- Only used in the annual-daylight recipe.
  - The light paths are calculated whenever a HB Model is written to a model folder, but only used in one recipe.

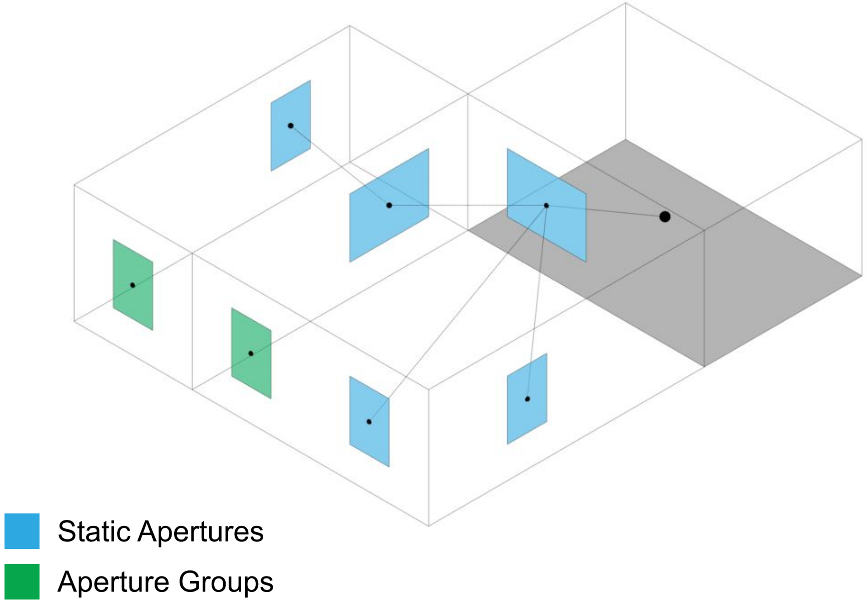
Post-processing of Aperture Groups

- Light paths are calculated for Room-based models.
- Traces from Room to exterior Apertures.
    - Includes interior Apertures and adjacent Rooms.



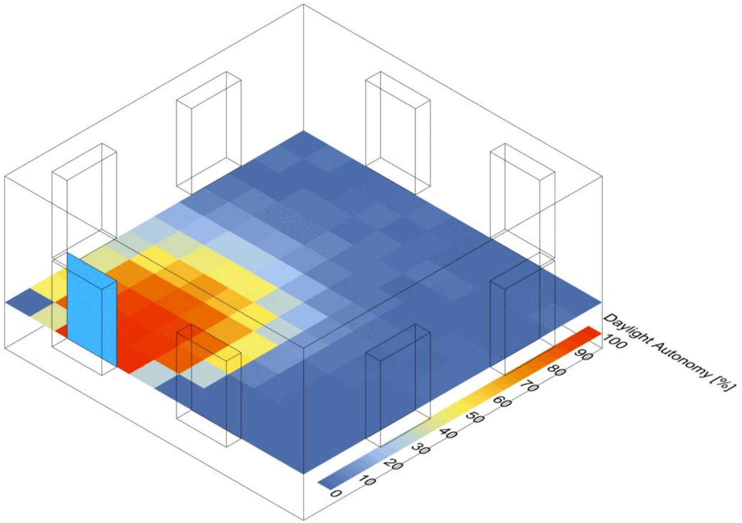
Post-processing of Aperture Groups

- All static Apertures in a model will be considered to be one light path.
- Each Aperture Group is its own light path and it can be combined with interior Apertures.
- Interior Aperture Groups will not be simulated as of now due to the increased complexity it adds.

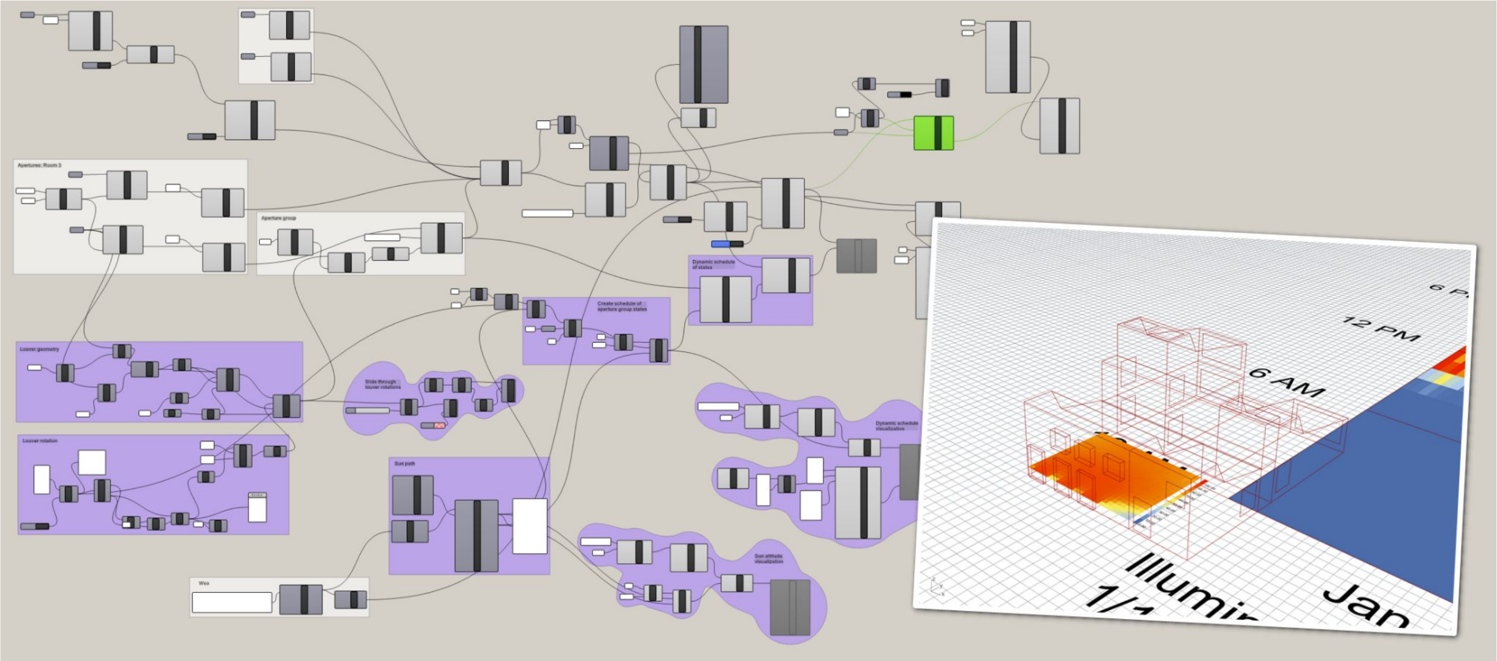


Post-processing of Aperture Groups

- The light paths are important for post-processing (and ray tracing) of states for Aperture Groups.
- Ray tracing for each light path individually.
  - All other light paths (static Apertures and Aperture Groups) are blacked out during the process.
- Reduces the workload and avoids a lot of sensor points showing 0 illuminance.



Post-processing of Aperture Groups (annual-daylight)


Apertures: Room 3
Aperture group
Dynamic schedule of states
Create schedule of aperture group states
Louver geometry
Slide through louver rotations
Louver rotation
Sun path
Dynamic schedule visualization
Sun altitude visualization
Wea
6 P
12 PM
6 AM
Illumin
1/1
Jan

Pollination

Splitting the grid

- All grid-based recipes split the grids and restructure the results in the end.

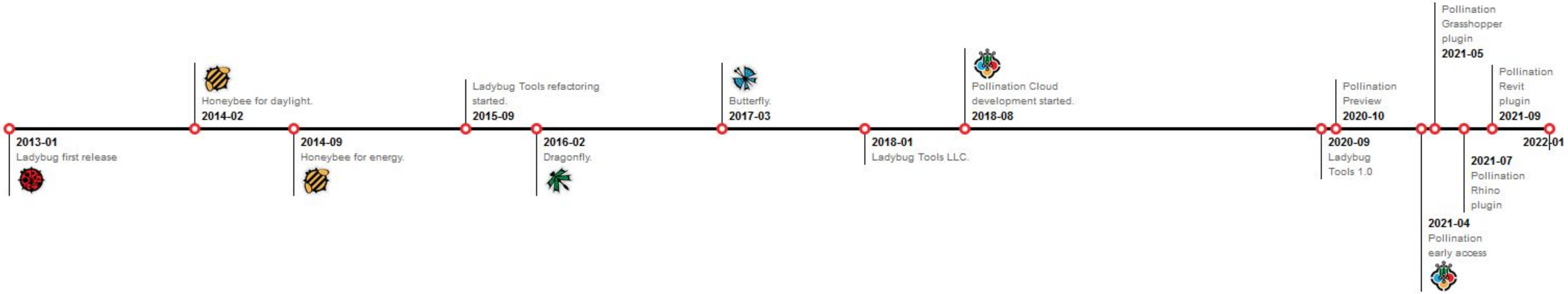
Splitting the grids in “model\grid”:

```
> honeybee-radiance grid split-folder model\grid grids 19 pts --grid-divisor 3 --min-sensor-count 200
```

Restructuring the results (illuminance) in “input_folder”:

```
> honeybee-radiance-postprocess grid merge-folder .\input_folder .\output_folder ill --dist-info
dist_info.json
```

Pollination



2013-01
Ladybug first release

2014-02
Honeybee for daylight.

2014-09
Honeybee for energy.

2015-09
Ladybug Tools refactoring started.

2016-02
Dragonfly.

2017-03
Butterfly.

2018-01
Ladybug Tools LLC.

2018-08
Pollination Cloud development started.

2020-09
Ladybug Tools 1.0

2020-10
Pollination Preview

2021-04
Pollination early access

2021-05
Pollination Grasshopper plugin

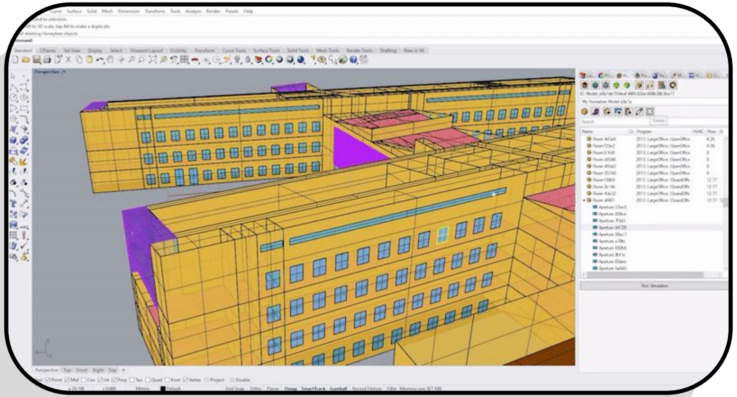
2021-07
Pollination Rhino plugin

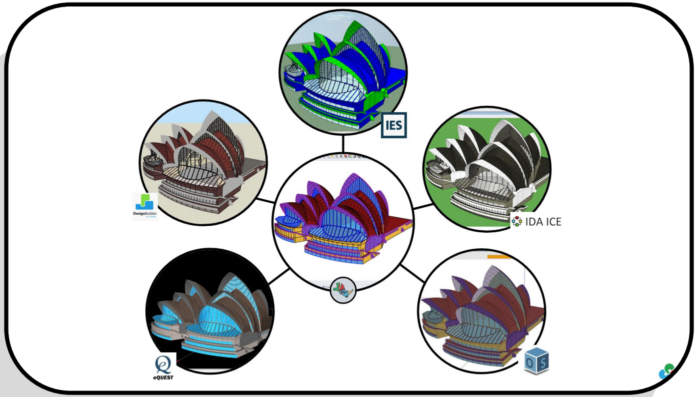
2021-09
Pollination Revit plugin

2022-01

CAD Plugins

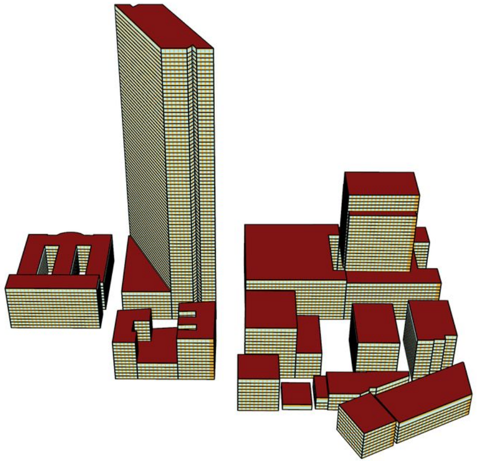
- Plugins for Rhino and Revit.
- Create and simulate analytical models in Rhino and Revit.
- Can be used as a stand-alone application, but also in combination with the Grasshopper plugin and cloud-computing.





Grasshopper Plugin

- Enables you to set-up and run simulations on Pollination.
- You do not have to leave the Grasshopper interface.
- Compatible with Ladybug Tools (v1.3.0 and above).
- Ladybug Tools Grasshopper scripts can be reused to run simulations on the cloud.



Radiance studies in the cloud

- Previously at the Radiance Workshop 2019, New York.
  - Andy McNeil did a 3½ hour workshop/tutorial.
- You can run Radiance recipes in the cloud on Pollination with just a few extra components.

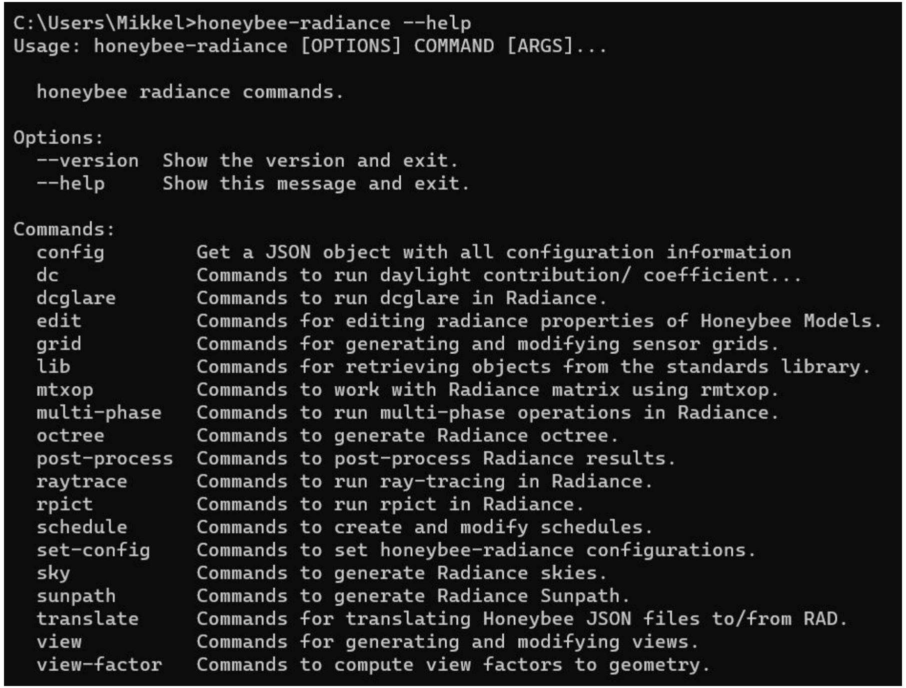
Running a study on Pollination

- Set up the model.
- Schedule a study on the cloud.
- Bring back the results to Grasshopper.

honeybee-radiance CLI

Overview

- The command line interface of honeybee-radiance is the backbone of the recipes.
- Most of the commands in the post-process module have been moved to honeybee-radiance-postprocess.



```
C:\Users\Mikkel>honeybee-radiance --help
Usage: honeybee-radiance [OPTIONS] COMMAND [ARGS]...

  honeybee radiance commands.

Options:
  --version  Show the version and exit.
  --help     Show this message and exit.

Commands:
  config        Get a JSON object with all configuration information
  dc            Commands to run daylight contribution/ coefficient...
  dcglare       Commands to run dcglare in Radiance.
  edit          Commands for editing radiance properties of Honeybee Models.
  grid          Commands for generating and modifying sensor grids.
  lib           Commands for retrieving objects from the standards library.
  mtxop         Commands to work with Radiance matrix using rmtxop.
  multi-phase   Commands to run multi-phase operations in Radiance.
  octree        Commands to generate Radiance octree.
  post-process  Commands to post-process Radiance results.
  raytrace      Commands to run ray-tracing in Radiance.
  rpict         Commands to run rpict in Radiance.
  schedule      Commands to create and modify schedules.
  set-config    Commands to set honeybee-radiance configurations.
  sky           Commands to generate Radiance skies.
  sunpath       Commands to generate Radiance Sunpath.
  translate     Commands for translating Honeybee JSON files to/from RAD.
  view          Commands for generating and modifying views.
  view-factor   Commands to compute view factors to geometry.
```

Example: Daylight Factor Study

```
> honeybee-radiance translate model-to-rad-folder daylight_factor.hbjson model

> honeybee-radiance sky illuminance 100000.0 --ground 0.2 --cloudy --name sky.sky

> honeybee-radiance octree from-folder model --output scene.oct --include-aperture --default --add-before sky.sky

> honeybee-radiance raytrace daylight-factor scene.oct model\grid\Room_3.pts --rad-params "-ab 2 -aa 0.1 -ad 2048 -ar 64" --rad-params-locked "-I -h" --sky-illum 100000 --output Room_3.df
```

Example: Sky

```
> honeybee-radiance sky cie -alt 52.1476158264 -az 173.649363239 -type 0 -g 0.2 --name cie_innsbruck.sky
```

... or

```
> honeybee-radiance sky cie 28 Aug 12:00 -lat 47.27 -lon 11.35 -tz 1 -type 0 -g 0.2 --name cie_innsbruck.sky
```

cie_innsbruck.sky

```
!gensky -ang 52.147616 -6.350637 +s -g 0.200

skyfunc glow sky_glow
0
0
4 1.000 1.000 1.000 0
sky_glow source sky
0
0
4 0 0 1 180

skyfunc glow ground_glow
0
0
4 1.000 1.000 1.000 0
ground_glow source ground
0
0
4 0 0 -1 180
```

Example: Skydome

```
> honeybee-radiance sky skydome --sky-density 4 --name skydome.dome
```

skydome.dome

```
#@rfluxmtx h=u u=Y
void glow ground_glow
0
0
4 0.200 0.200 0.200 0
ground_glow source ground
0
0
4 0 0 -1 180

#@rfluxmtx h=r4 u=Y
void glow sky_glow
0
0
4 1.000 1.000 1.000 0
sky_glow source sky
0
0
4 0 0 1 180
```

Run recipes with lbt-recipes CLI

- The recipes from lbt-recipes are used in the LBT Grasshopper plugin.

```
> lbt-recipes run daylight-factor daylight_factor_inputs.json --project-folder . --workers 16
```

daylight_factor_inputs.json

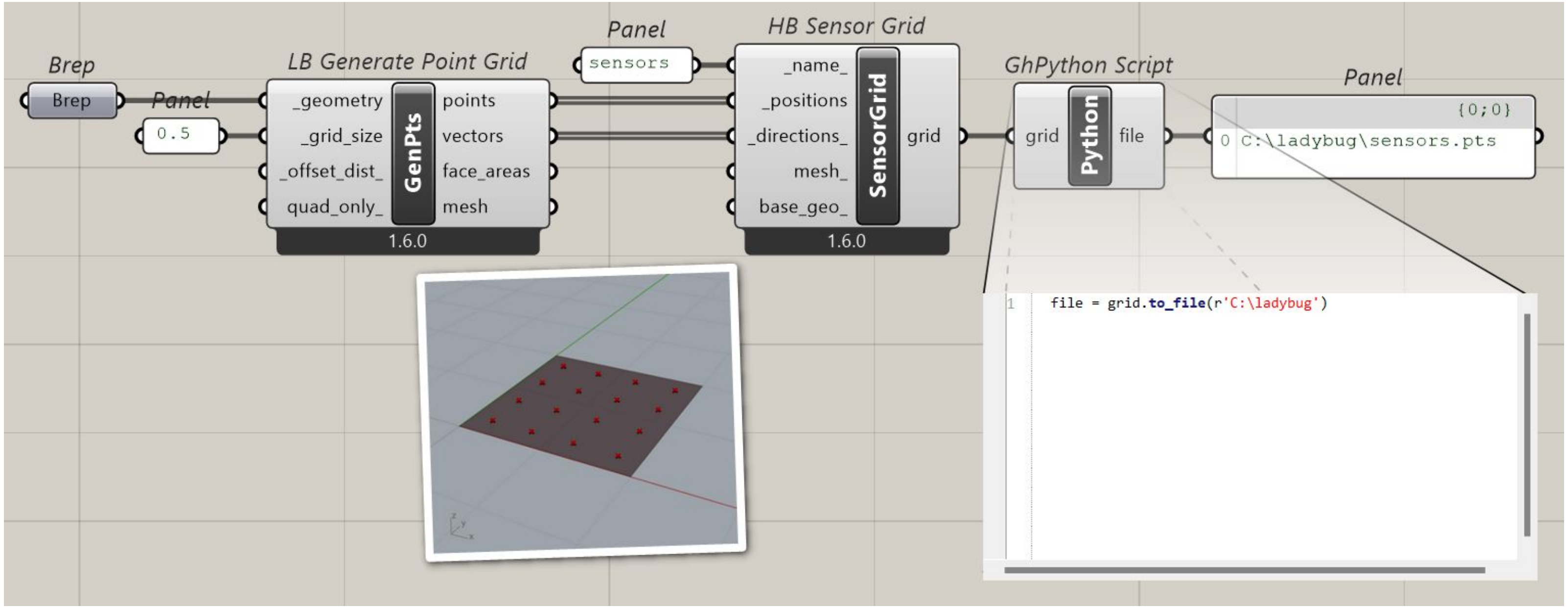
```
{
	"cpu-count": 8,
	"grid-filter": "*",
	"min-sensor-count": 200,
	"model": "0.0_daylight_factor.hbjson",
	"radiance-parameters": "-ab 2 -aa 0.1 -ad 2048 -ar 64"
}
```

Other useful utilities

… that might be useful for Radiance users who do not want to run simulations in the Ladybug Tools & Pollination ecosystem

Pollination

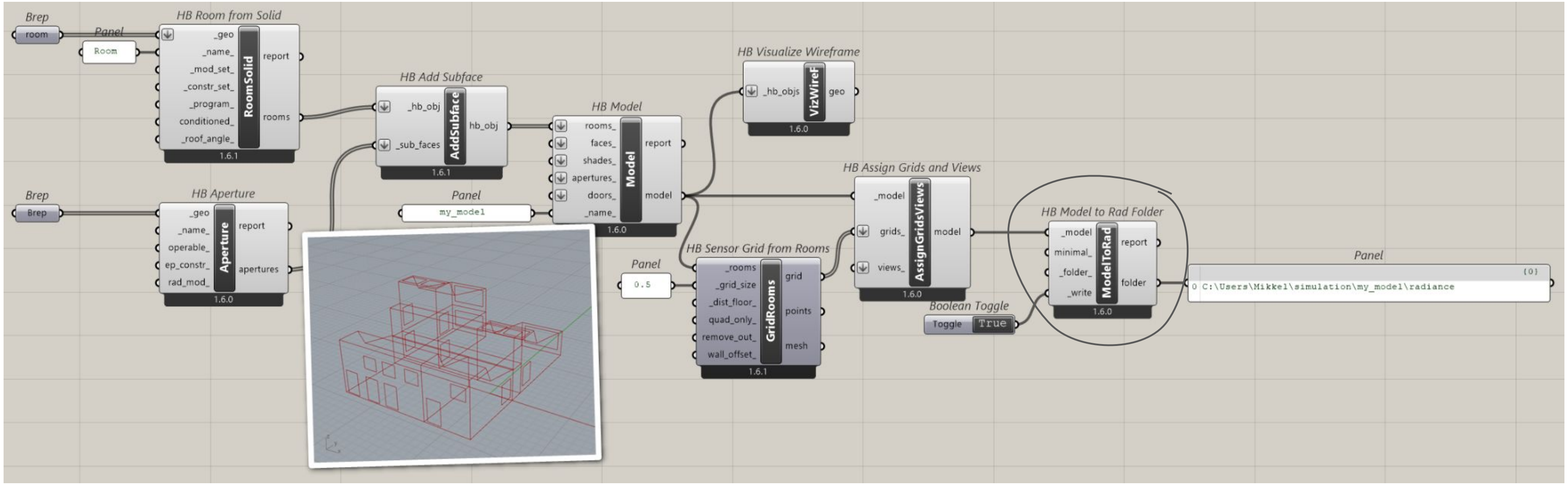
Creating sensor points



```
{0;0}
0 C:\ladybug\sensors.pts
```

```python
file = grid.to_file(r'C:\ladybug')
```

Writing rad and mat files



```
0 C:\Users\Mikkel\simulation\my_model\radiance
```