Automate Radiance Workflows through Python

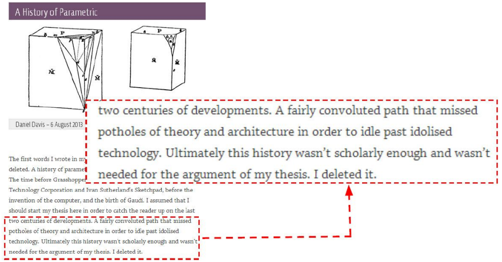
15th International Radiance Workshop
August 30, 2016

Mostapha Sadeghipour Roudsari
University of Pennsylvania

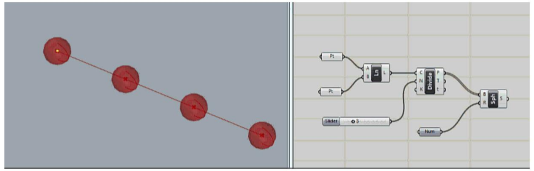
Sarith Subramaniam
Penn State University

Parametric Design

- **What is [the History of] Parametric Design?**
  Go read this link: http://www.danieldavis.com/a-history-of-parametric/

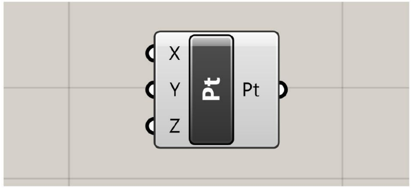


What is Grasshopper?
Why does it matter?



Visual Scripting

Pt = Rhino.Geometry.Point3D(X, Y, Z)

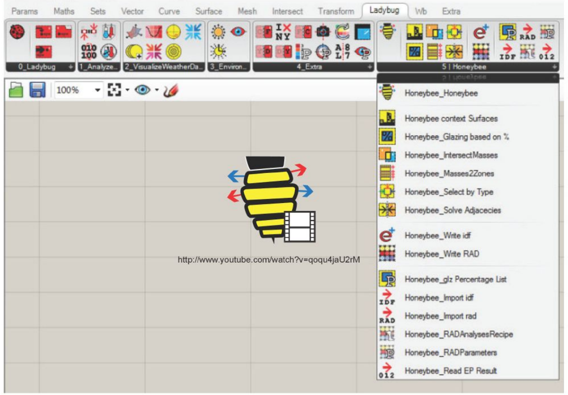


Parametric Analysis is the new !

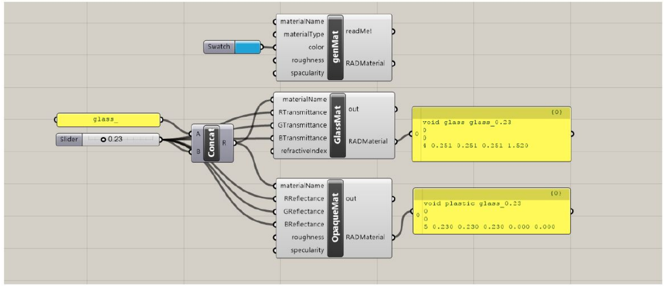
ou

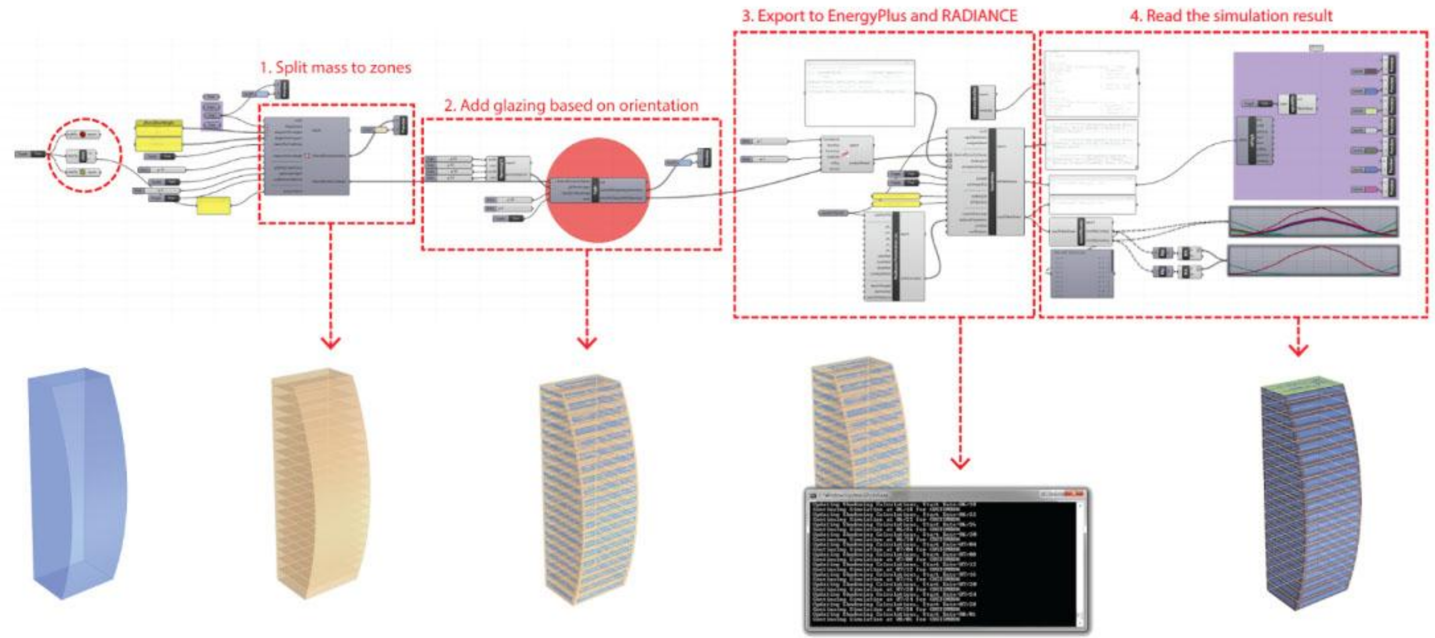
ce

Honeybee: Grasshopper <> Radiance/Daysim/EnergyPlus

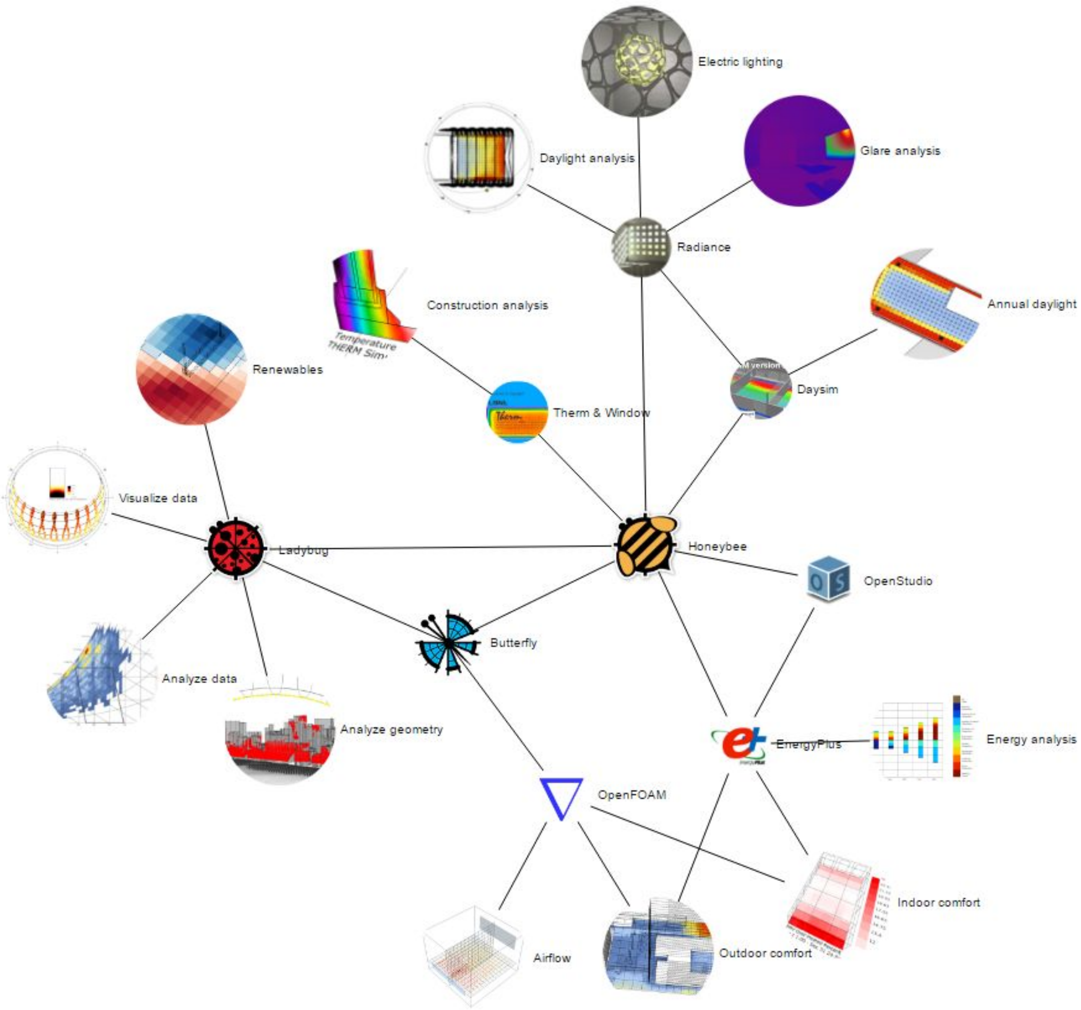


Honeybee: Multiple Material Components




1. Split mass to zones
2. Add glazing based on orientation
3. Export to EnergyPlus and RADIANCE
4. Read the simulation result

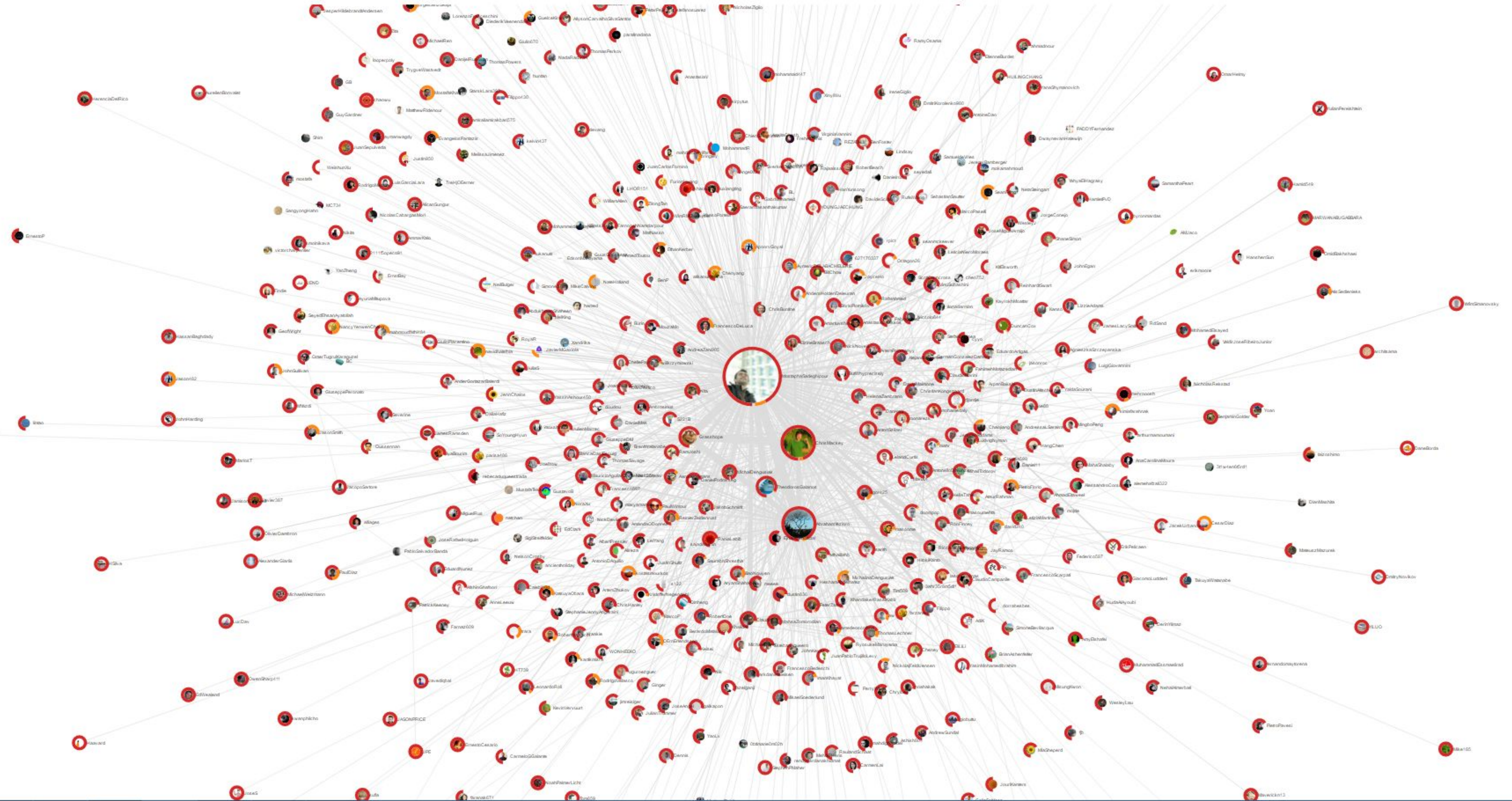
2013


Electric lighting
Daylight analysis
Glare analysis
Radiance
Construction analysis
Annual daylight
Renewables
Therm & Window
Daysim
Visualize data
Ladybug
Honeybee
OpenStudio
Butterfly
Analyze data
Analyze geometry
EnergyPlus
Energy analysis
OpenFOAM
Indoor comfort
Airflow
Outdoor comfort

2016

http://www.grasshopper3d.com/group/ladybug

. 1365 Discussions . 5029 Replies . 498 Comments . 660 Connections .

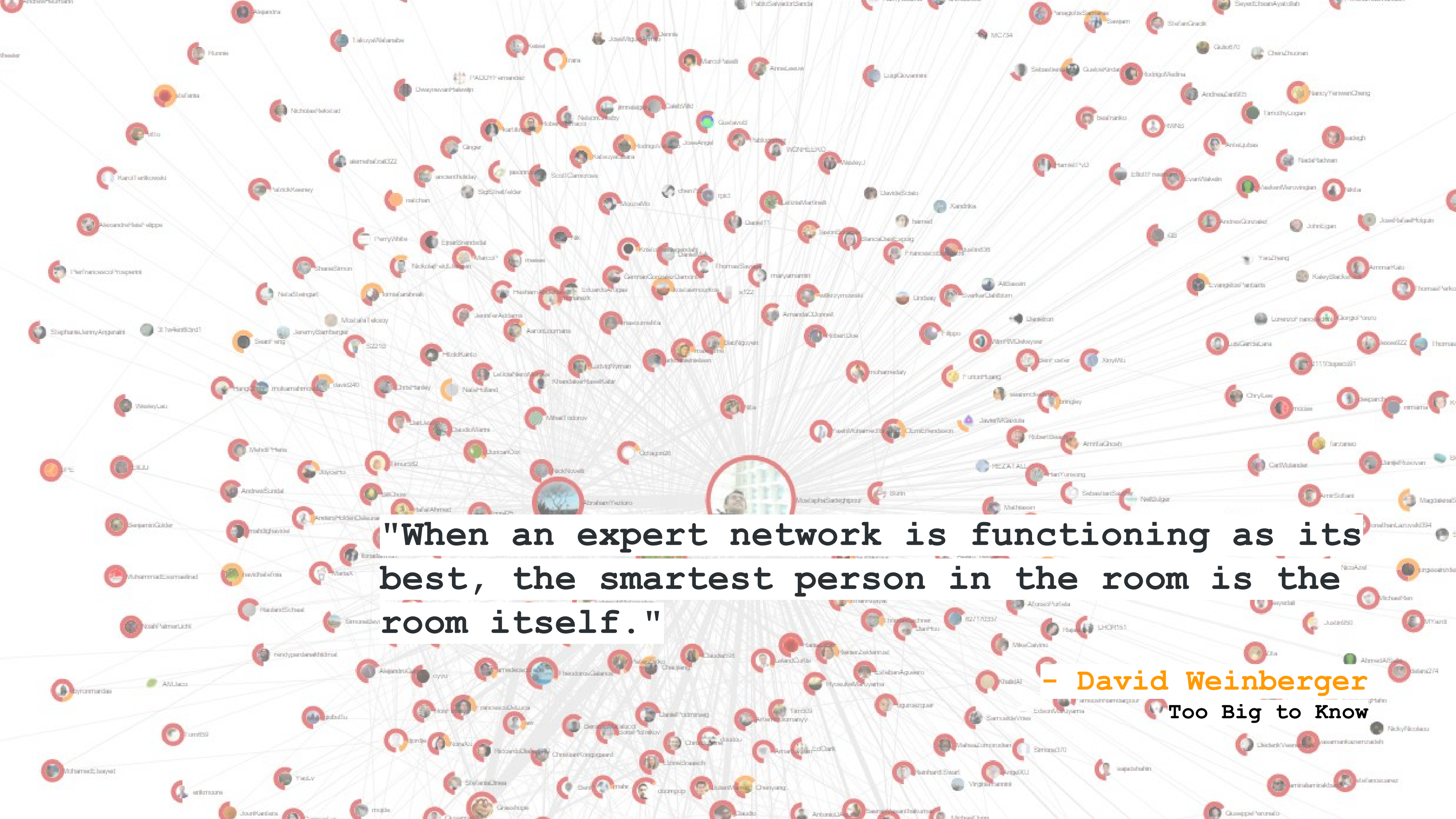


+53,000 downloads from 2014.

1365 discussions, **5029** replies and **498** comments.

78.5% of the discussions get the first reply **in less than a day**.

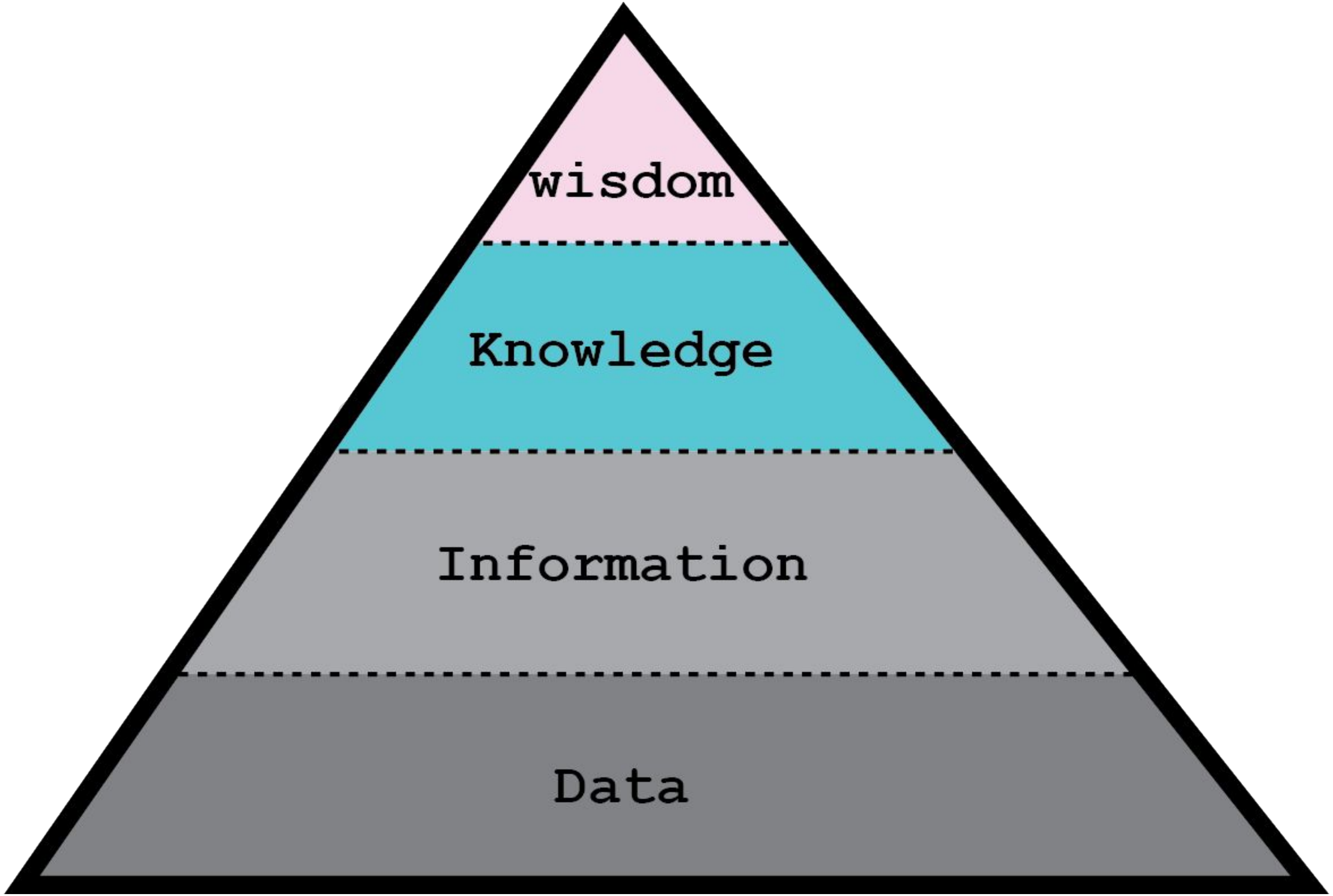
95.0% of them will get the first reply **in less than 3 days**.



"When an expert network is functioning as its best, the smartest person in the room is the room itself."

– David Weinberger

Too Big to Know

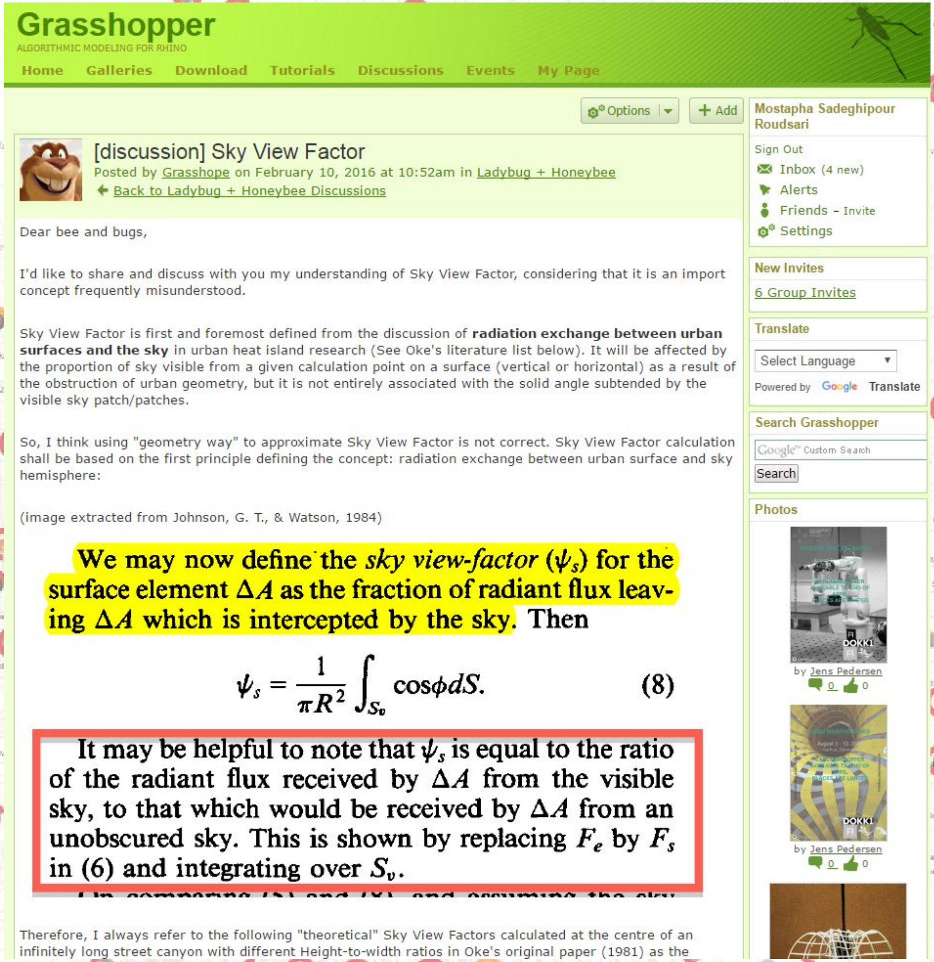

wisdom
Knowledge
Information
Data

Share Code.

Share Knowledge.

Share Examples.

Share Stories!



Grasshopper

ALGORITHMIC MODELING FOR RHINO

Home Galleries Download Tutorials Discussions Events My Page

Options | Add

[discussion] Sky View Factor

Posted by Grasshopper on February 10, 2016 at 10:52am in Ladybug + Honeybee

Back to Ladybug + Honeybee Discussions

Dear bee and bugs,

I'd like to share and discuss with you my understanding of Sky View Factor, considering that it is an import concept frequently misunderstood.

Sky View Factor is first and foremost defined from the discussion of **radiation exchange between urban surfaces and the sky** in urban heat island research (See Oke's literature list below). It will be affected by the proportion of sky visible from a given calculation point on a surface (vertical or horizontal) as a result of the obstruction of urban geometry, but it is not entirely associated with the solid angle subtended by the visible sky patch/patches.

So, I think using "geometry way" to approximate Sky View Factor is not correct. Sky View Factor calculation shall be based on the first principle defining the concept: radiation exchange between urban surface and sky hemisphere:

(image extracted from Johnson, G. T., & Watson, 1984)

We may now define the *sky view-factor* (ψ_s) for the surface element ΔA as the fraction of radiant flux leaving ΔA which is intercepted by the sky. Then

$$\psi_s = \frac{1}{\pi R^2} \int_{S_v} \cos\phi \, dS. \qquad (8)$$

It may be helpful to note that ψ_s is equal to the ratio of the radiant flux received by ΔA from the visible sky, to that which would be received by ΔA from an unobscured sky. This is shown by replacing F_e by F_s in (6) and integrating over S_v.

Therefore, I always refer to the following "theoretical" Sky View Factors calculated at the centre of an infinitely long street canyon with different Height-to-width ratios in Oke's original paper (1981) as the

Mostapha Sadeghipour Roudsari

Sign Out

Inbox (4 new)

Alerts

Friends – Invite

Settings

New Invites

6 Group Invites

Translate

Select Language

Powered by Google Translate

Search Grasshopper

Google™ Custom Search

Search

Photos

by Jens Pedersen 0 0

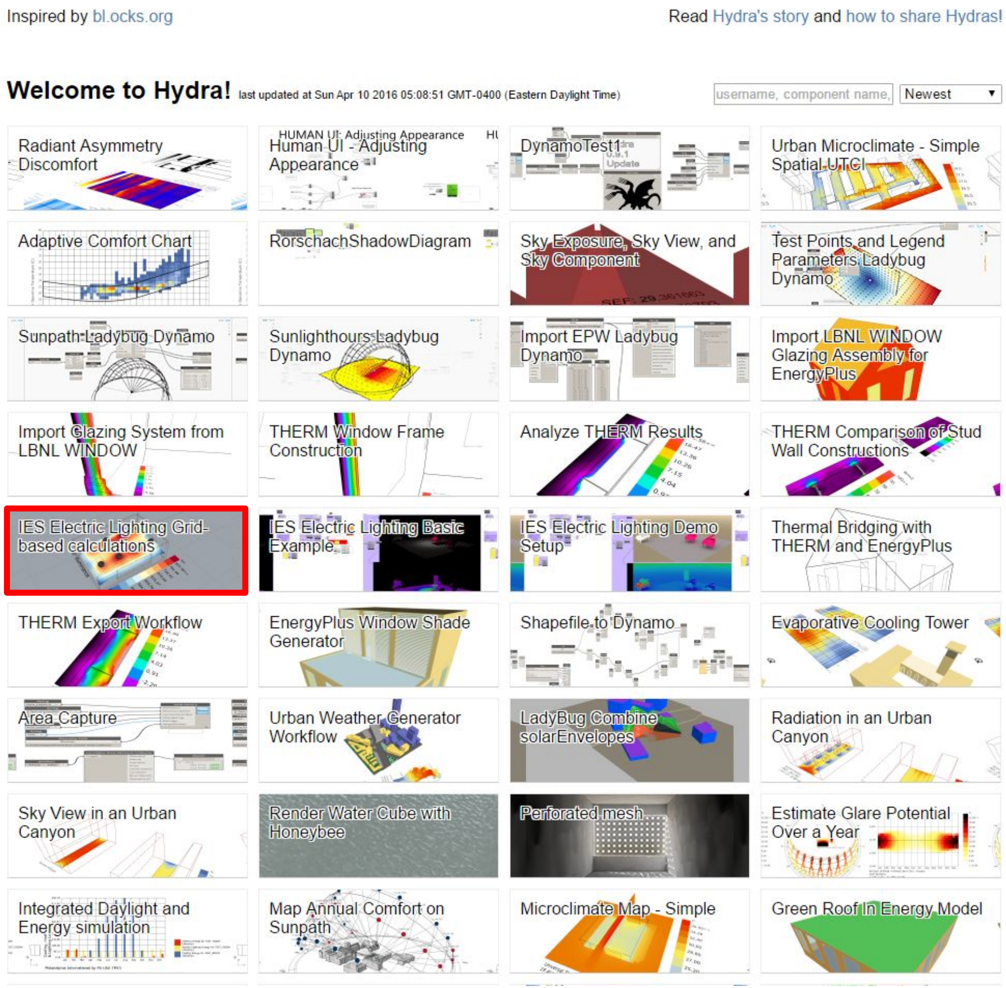
by Jens Pedersen 0 0

Share Code.

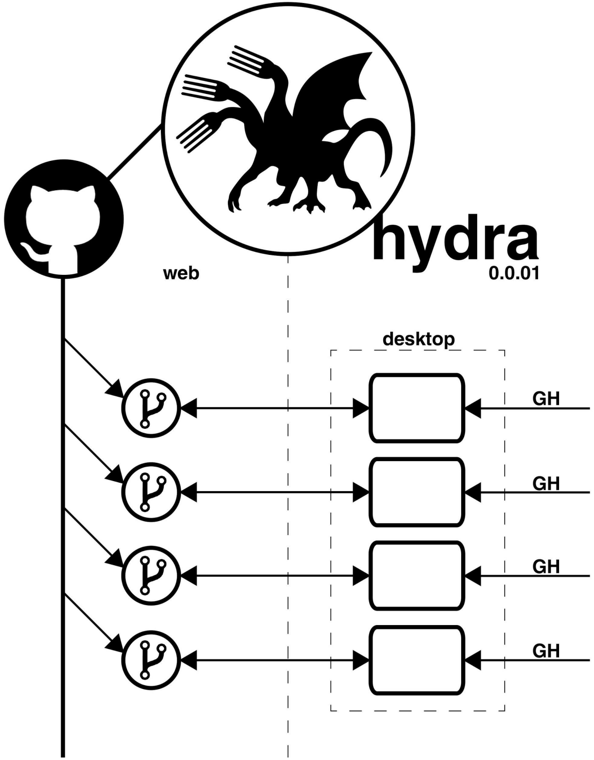
Share Knowledge.

Share Workflows.

Share Stories!

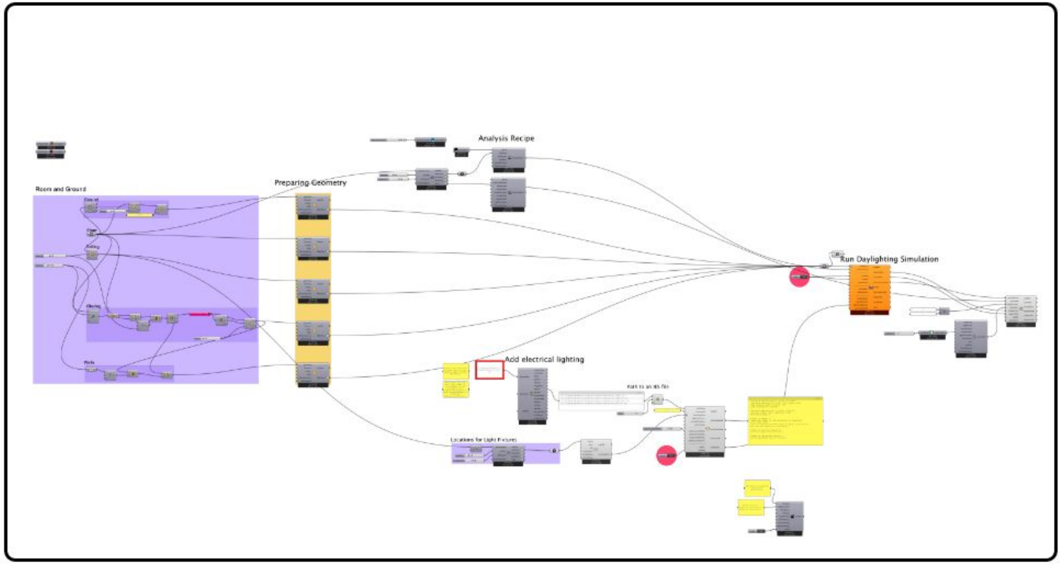


http://hydrashare.github.io/hydra/



IES Electric Lighting Grid-based calculations

 ZoomToExtent



Download

#IES #Electric #Lighting #Grid-based #calculations

Description

Indoor electric lighting calculations based on grid points

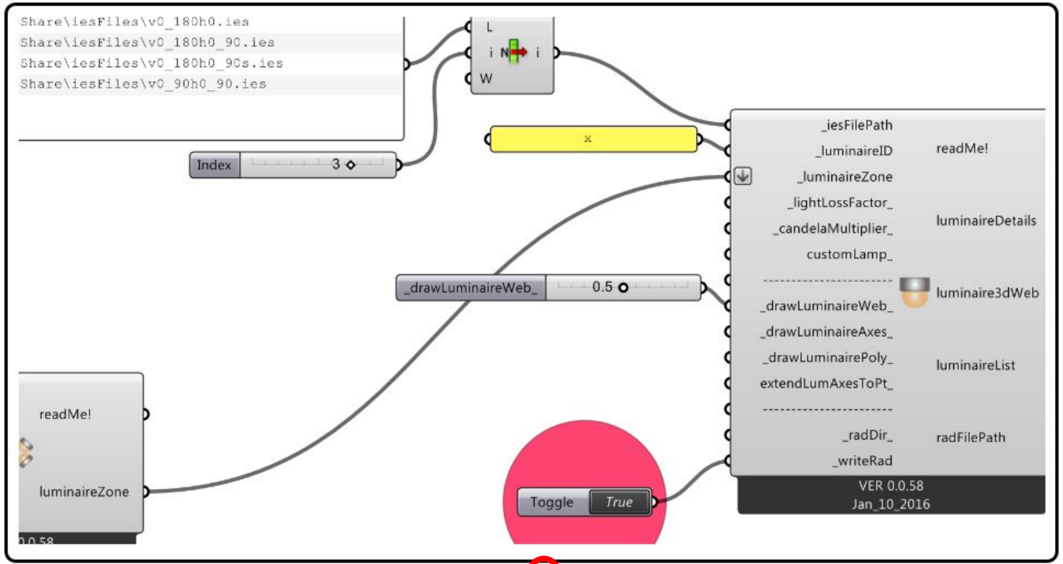
This file has been submitted by sariths

Check out this example on Hydra!

Tags

IES Electric Lighting Grid-based calculations

ZoomToExtent



Download

#IES #Electric #Lighting #Grid-based #calculations

Description

Indoor electric lighting calculations based on grid points

This file has been submitted by sariths

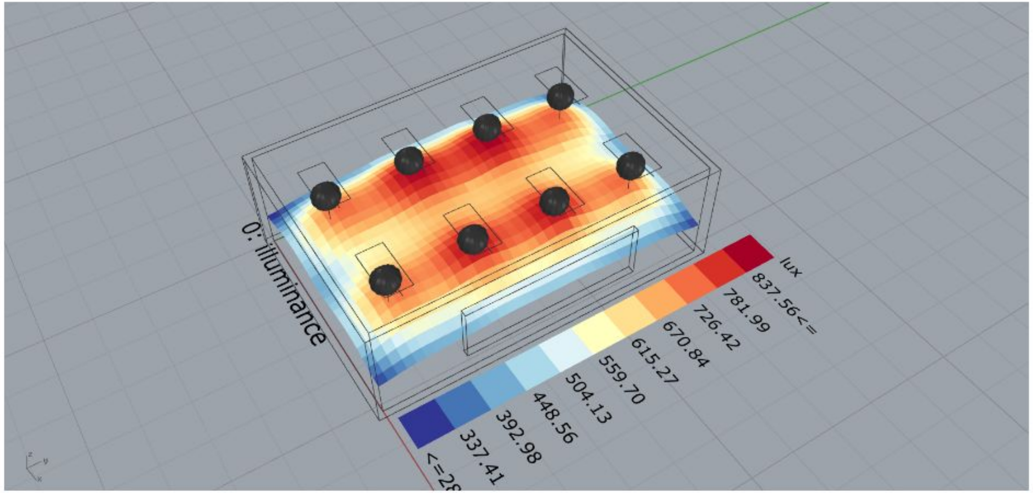
Check out this example on Hydra!

Tags

sariths's example IES_Electric_Lighting_Grid-based_calculations January 13, 2016 Return to Main Page

IES Electric Lighting Grid-based calculations

sharethis 25 Email 0 Share 0 Tweet 0 Share 0 Google + 0 ZoomToExtent



Download

#IES #Electric #Lighting #Grid-based #calculations

Description

Indoor electric lighting calculations based on grid points

This file has been submitted by sariths

Check out this example on Hydra!

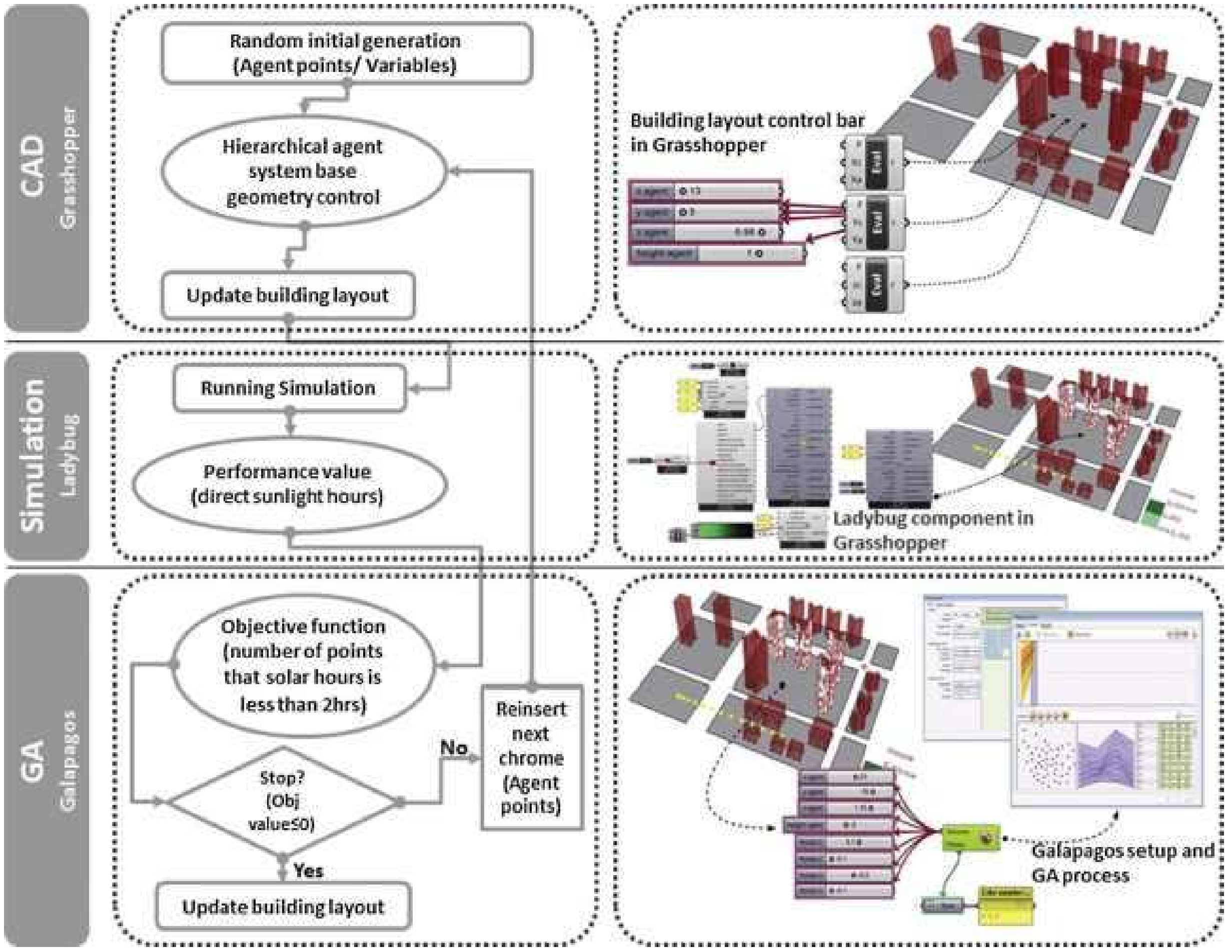
Tags

Share Code.

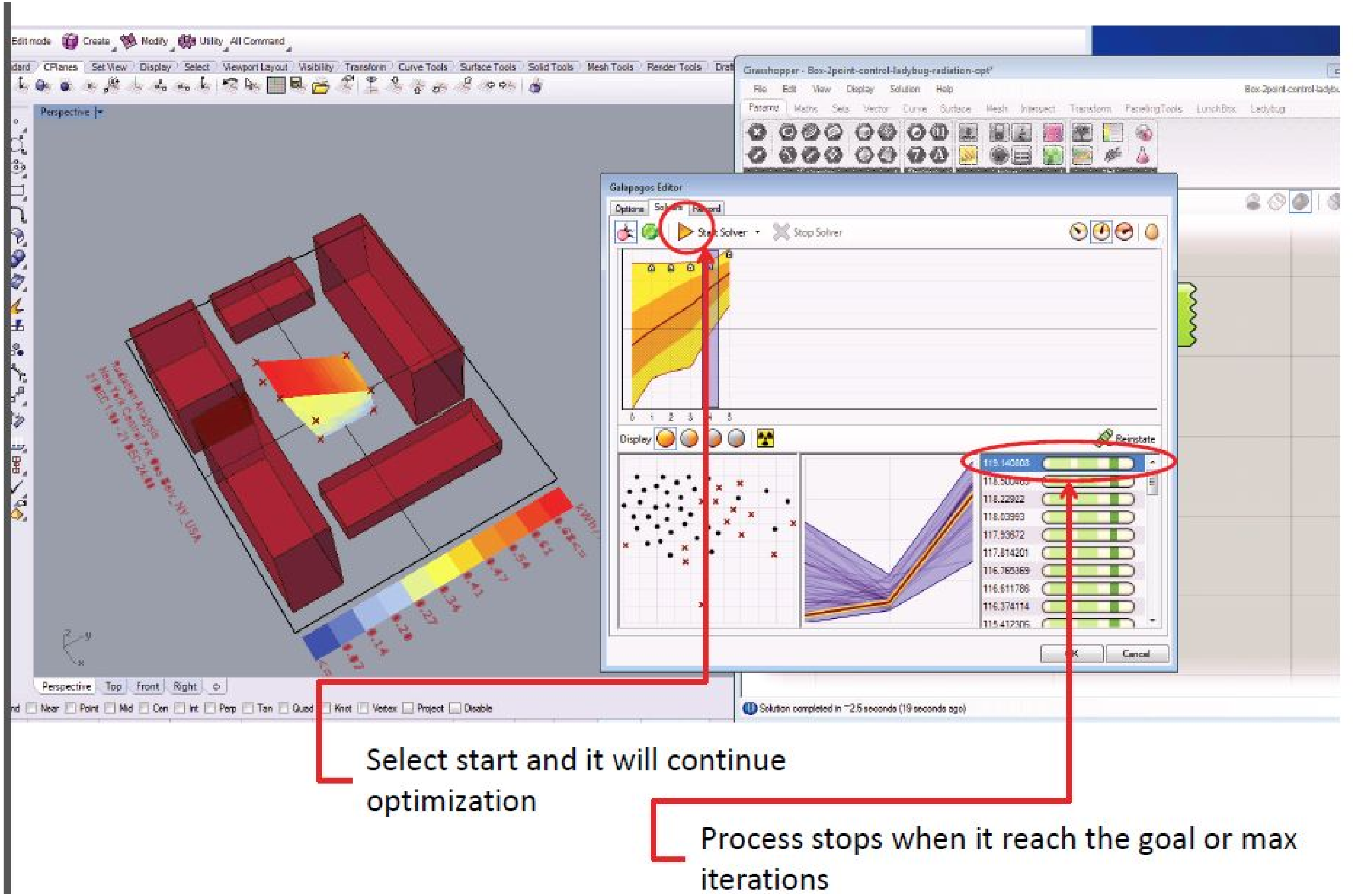
Share Knowledge.

Share Examples.

Share Stories!



Yun Kyu Yi, Hyoungsub Kim, Agent-based geometry optimization with Genetic Algorithm (GA) for tall apartment's solar right.



Yun Kyu Yi, Hyoungsub Kim, Agent-based geometry optimization with Genetic Algorithm (GA) for tall apartment's solar right.

Make your own tool!

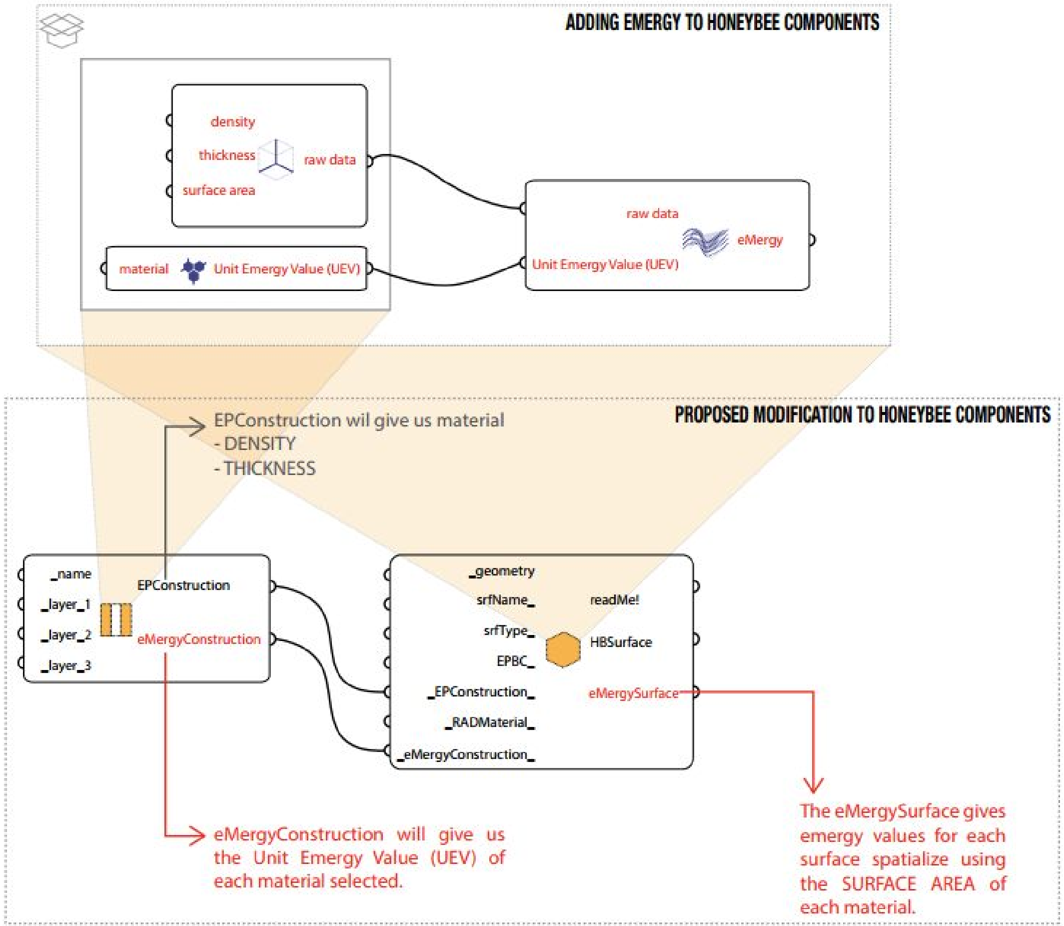
GRASSHOPPER PLUG-INs

Clark's Crow

HoneyBee

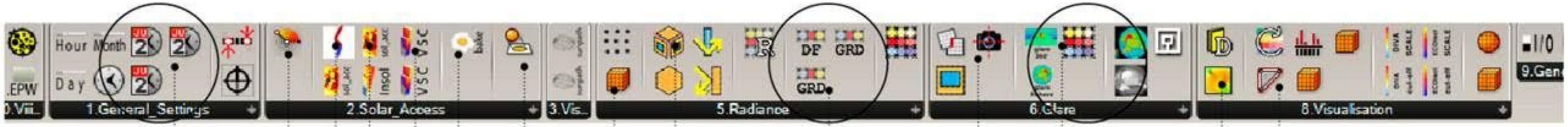
Key:

- existing component
- proposed component



Naomi Keena, Mohamed Aly Etman, Nancy Diniz, Alexandra Rempel, Anna Dyson

Center for Architecture Science and Ecology (CASE), Rensselaer Polytechnic Institute



Mar/Dec/Jun 21st for the extremely lazy people

Sun path that gives vectors for solar access simulations. (there is a second sun path there that runs for 21st June to get maximum hours for legend high)

Simple bake command that gets the layer name from the "readme" of analysis period.

3D grid creator for radiance (asign floor and ceiling and cube size)

Just a simplified set of radiance simulations for ease

Camera and viewport setup for imagebased simulations

Average period for .ill files

Solar access with and without minimum mark cutoff (+ average passing points for regulations)

Very small and full of bugs component that works like Ecotect-shading_profile.

Geometry filter (when HBSrfs crushes). **This is very usefull** for complex "dirty" models. It filters surfaces with very small sides (line-like surfaces) that make radiance crush

Glare setup for fisheye and cylinder (360°) views

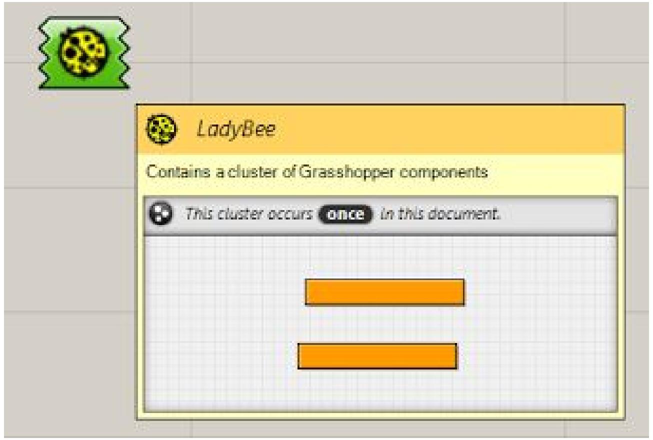
&

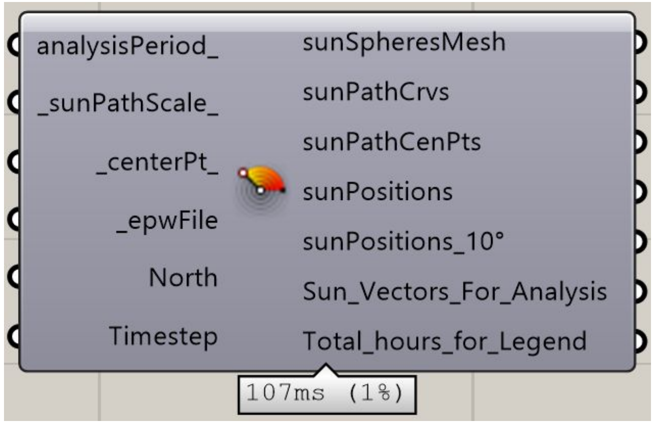
Visualization of those

Filters the mesh bellow and above target value and creates a contour line (with a smooth factor to make it visually better)

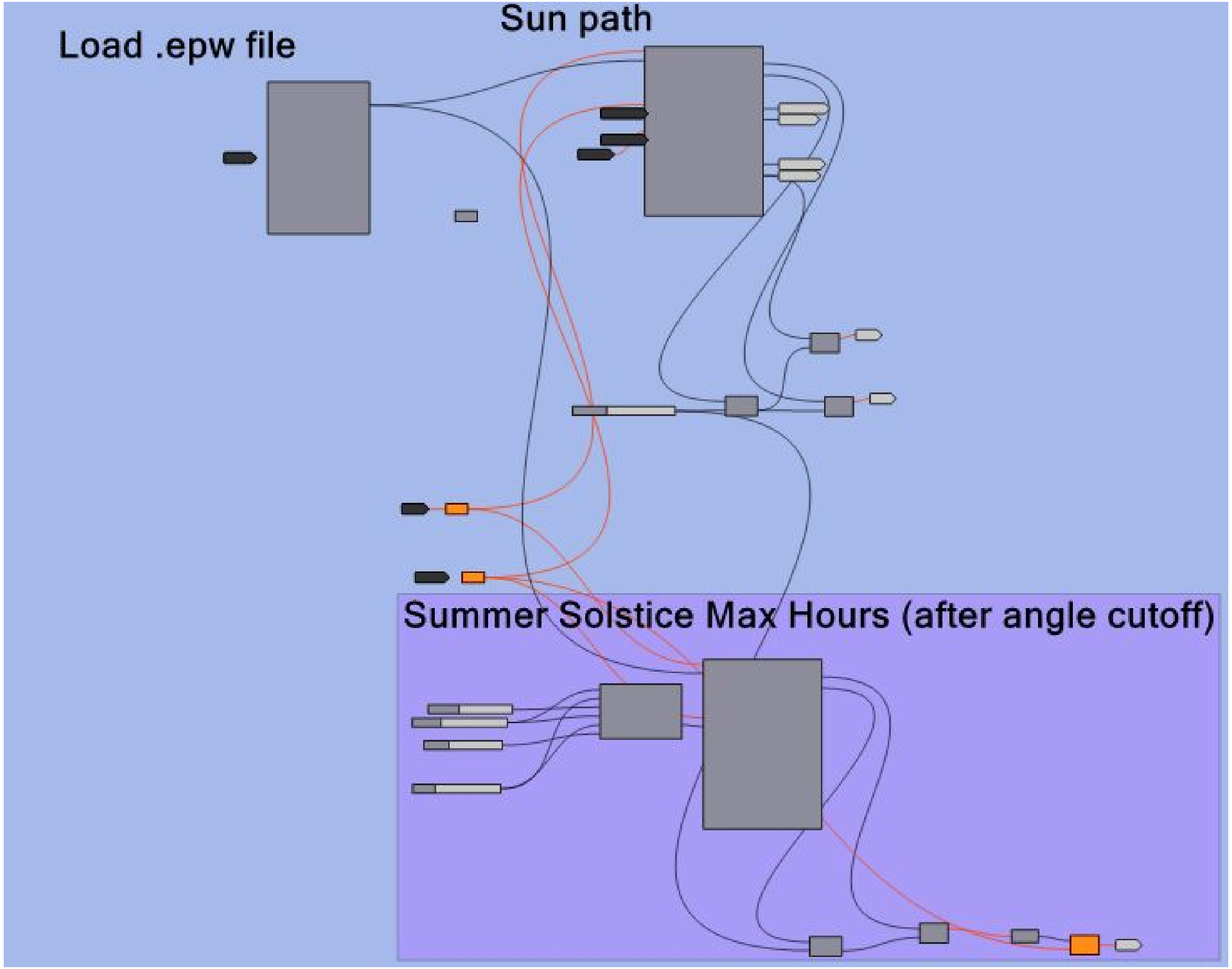
Now this is a tricky and messy one. It was created on a project where we wanted to test reflection on a specific target after solar rays reflect from a adjacent tower. It works with asigning target and context. It is very similar to the component you already have but for some reason i made a new (can't remember exactly why)

VSC only visual and then VSC for regulations were only non-horizontal surfaces are calculated and also overlapping surfaces are filtered. (this takes ages to run on more complex models)


LadyBee
Contains a cluster of Grasshopper components
This cluster occurs once in this document.


analysisPeriod_
sunPathScale
centerPt
_epwFile
North
Timestep
sunSpheresMesh
sunPathCrvs
sunPathCenPts
sunPositions
sunPositions_10°
Sun_Vectors_For_Analysis
Total_hours_for_Legend
107ms (1%)

The other side of the coin!


Load .epw file
Sun path
Summer Solstice Max Hours (after angle cutoff)

Direct sunlight hour analysis

PAUL WINTOUR
March 13, 2015 at 12:47 am

Hi

I'm wondering if there is a Dynamo node that can calculate the number of hours of direct sunlight a surface receives. Basically I want it to work like the 'Sunlight hours analysis' component found in Ladybug in Grasshopper. All I can currently see in Dynamo is daylighting and solar insolation. I would like to avoid constructing it from scratch if possible. Any thoughts?

PAUL WINTOUR
March 15, 2015 at 11:56 pm

anyone?

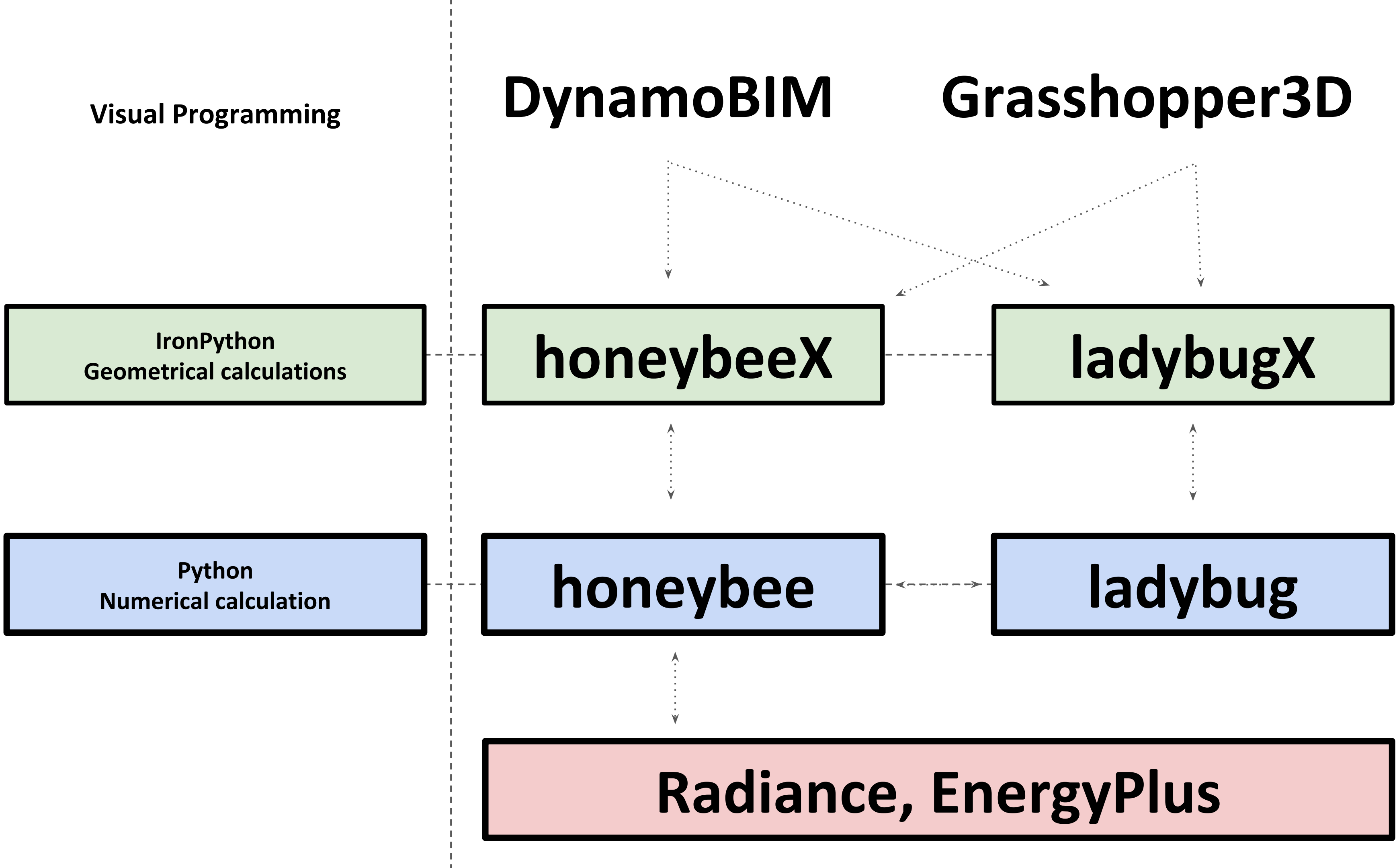
[Radiance-general] REVIT to Radiance pipeline

Shakespeare, Robert A. shakespe at indiana.edu
Thu Nov 12 06:35:11 PST 2015

At the last Radiance Workshop, there was some discussion regarding converting REVIT models to Radiance, with materials and named surfaces parsed into readable .rad datasets. As REVIT has essentially become the architectural industry modeling tool, the pipeline to Radiance is an important one. It seems that the latest version of Sketchup has some challenges with the very helpful su2rad, written quite awhile ago. I heard that others were using some other pipelines, perhaps integrating obj2rad, etc.
I would appreciate your sharing conversion pipelines which interface with the MOST CURRENT releases of REVIT, SKETCHUP, etc.
Hopefully a generous and expert code person would consider writing a direct REVIT plugin for this purpose. There might be modest financial encouragement to fill this gap. I do realize that it may be short-lived as AutoDESK releases are moving targets, yet to engage the full Radiance tool kit for lighting design and analysis on REVIT generated models, will keep Radiance on the forefront and accessible to a broader audience.
Appreciatively,
Rob Shakespeare
shakespe at indiana.edu

-------------- next part --------------
An HTML attachment was scrubbed...
URL: <http://www.radiance-online.org/pipermail/radiance-general/attachments/20151112/abe112fe/attachment.html>

Let the`bug fly free!


Visual Programming
DynamoBIM
Grasshopper3D
IronPython
Geometrical calculations
honeybeeX
ladybugX
Python
Numerical calculation
honeybee
ladybug
Radiance, EnergyPlus

What does it really mean?

```python
from honeybee.room import Room
from honeybee.radiance.material.glass import GlassMaterial
from honeybee.radiance.sky.certainIlluminance import SkyWithCertainIlluminanceLevel
from honeybee.radiance.recipe.gridbased import HBGridBasedAnalysisRecipe
```

0

```python
room = Room(origin=(0, 0, 3.2), width=4.2, depth=6, height=3.2, rotationAngle=45) # create a test room
room.addFenestrationSurface(wallName='back', width=2, height=2, sillHeight=0.7) # add a window to the back wall
glass_60 = GlassMaterial.bySingleTransValue('tvis_0.6', 0.6) # add another window with custom material.
room.addFenestrationSurface('right', 4, 1.5, 1.2, radianceMaterial=glass_60) # This time to the right wall
```

1

```python
# run a grid-based analysis for this room
sky = SkyWithCertainIlluminanceLevel(illuminanceValue=2000) # generate the sky
testPoints = room.generateTestPoints(gridSize=0.5, height=0.75) # generate grid of test points
rp = HBGridBasedAnalysisRecipe(sky=sky, pointGroups=(testPoints,), simulationType=0, hbObjects=(room,))
```

2

```python
# write and run the analysis
rp.writeToFile(targetFolder=r'c:\ladybug', projectName='room')
rp.run(debug=False)
```

3

```python
results = rp.results(flattenResults=True)
print 'Average illuminacen level in this room is {} lux.'.format(sum(results) / len(results))
```

4

```python
from honeybee.room import Room
from honeybee.radiance.material.glass import GlassMaterial
from honeybee.radiance.sky.certainIlluminance import SkyWithCertainIlluminanceLevel
from honeybee.radiance.recipe.gridbased import HBGridBasedAnalysisRecipe

# create a test room
room = Room(origin=(0, 0, 3.2), width=4.2, depth=6, height=3.2, rotationAngle=45)

# add a window to the back wall
room.addFenestrationSurface(wallName='back', width=2, height=2, sillHeight=0.7)

# add another window with custom material. This time to the right wall
glass_60 = GlassMaterial.bySingleTransValue('tvis_0.6', 0.6)
room.addFenestrationSurface('right', 4, 1.5, 1.2, radianceMaterial=glass_60)
```

0 + 1/4 Geometry

```python
...
# run a grid-based analysis for this room
# generate the sky
sky = SkyWithCertainIlluminanceLevel(illuminanceValue=2000)

# generate grid of test points
testPoints = room.generateTestPoints(gridSize=0.5, height=0.75)

# put the recipe together
rp = HBGridBasedAnalysisRecipe(sky=sky, pointGroups=(testPoints,),
                               simulationType=0, hbObjects=(room,))
# write and run the analysis
rp.writeToFile(targetFolder=r'c:\ladybug', projectName='room')
rp.run(debug=False)
```

2/4 Recipe

```
...
results = rp.results(flattenResults=True)
print 'Average illuminance level in this room is {} lux.' \
    .format(sum(results) / len(results))
```

Number of total materials: 5

Number of total surfaces: 1

Files are written to: c:\ladybug\room\gridbased

C:\Users\Administrator\Documents\GitHub\hydrashare.github.io>c:

C:\Users\Administrator\Documents\GitHub\hydrashare.github.io>cd c:\ladybug\room\gridbased

c:\ladybug\room\gridbased>c:\radiance\bin\oconv -f c:\ladybug\room\gridbased\Uniform_CIE_2000.sky

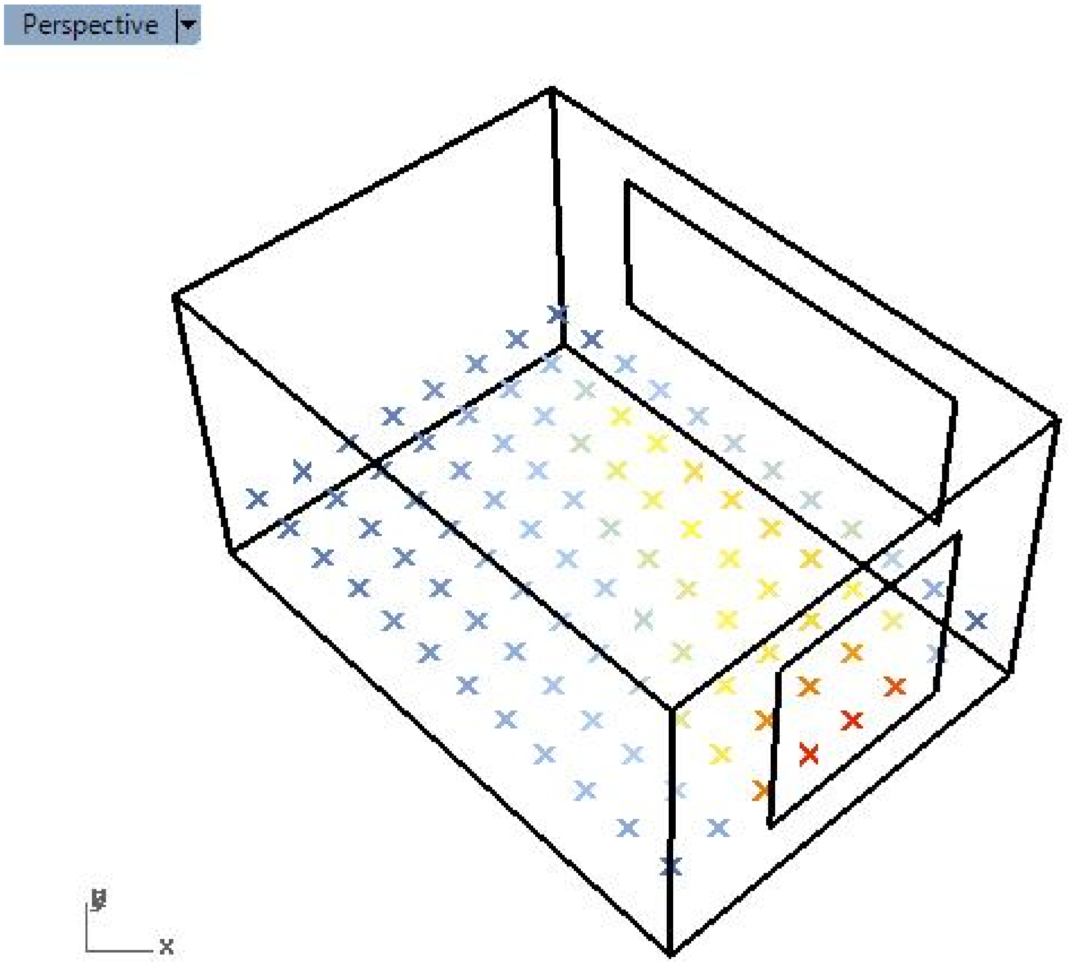
c:\ladybug\room\gridbased\room.mat c:\ladybug\room\gridbased\room.rad 1>room.oct

c:\ladybug\room\gridbased>c:\radiance\bin\rtrace -aa 0.25 -ab 2 -dj 0.0 -ad 512 -ss 0.0 -h -dc 0.25 -st 0.85 -lw 0.05 -as 128 -ar 16 -lr 4 -I -dt 0.5

-dr 0 -ds 0.5 -dp 64 -e error.txt c:\ladybug\room\gridbased\room.oct 0<c:\ladybug\room\gridbased\room.pts 1>room.res

Average illuminance level in this room is 158.901223225 lux.

3+4 /4 Results


Perspective
x

Dynamo

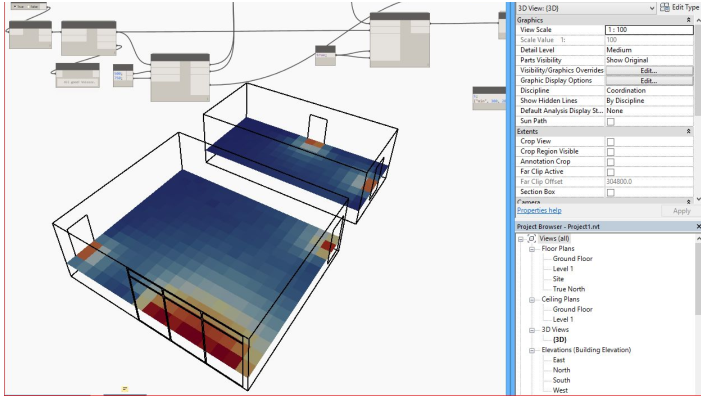
Edit Python Script...

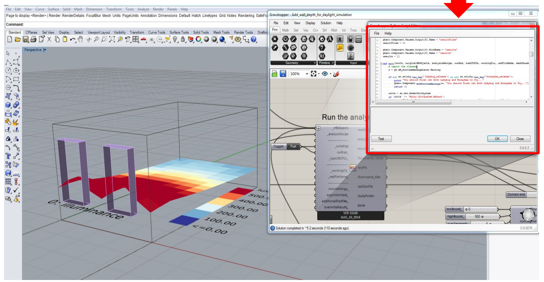
```python
import sys
sys.path.extend(IN[0])

try:
    from honeybeex.radiance.recipe.gridbased import GridBasedAnalysisRecipe
    from honeybeex import wrapper
except ImportError:
    raise ImportError("Failed to import Honeybee libraries.\n" + \
                      "Set-up path to Honeybee libraries in honeybee_honeybee component")


_HBSky, _testPoints, ptsVectors_, _simType, _radParameters_ = IN[1:]

if _HBSky and _testPoints and ptsVectors_:
    if _radParameters_:
        _radParameters_ = _radParameters_.unwrap()

    analysisRecipe = GridBasedAnalysisRecipe(
        _HBSky.unwrap(), _testPoints, ptsVectors_, _simType, _radParameters_
        )

    OUT = wrapper.Wrapper(analysisRecipe)
```

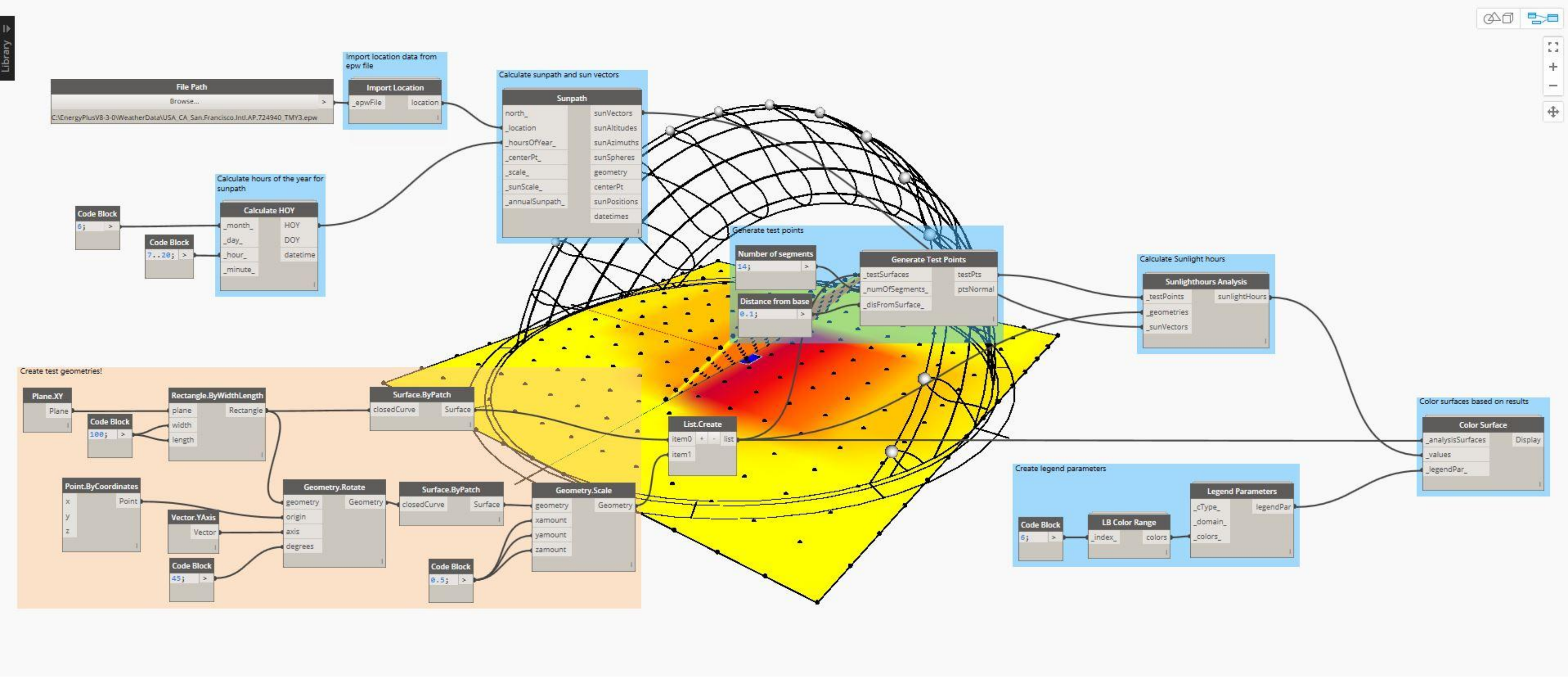
Grasshopper Python Script Editor

File Help

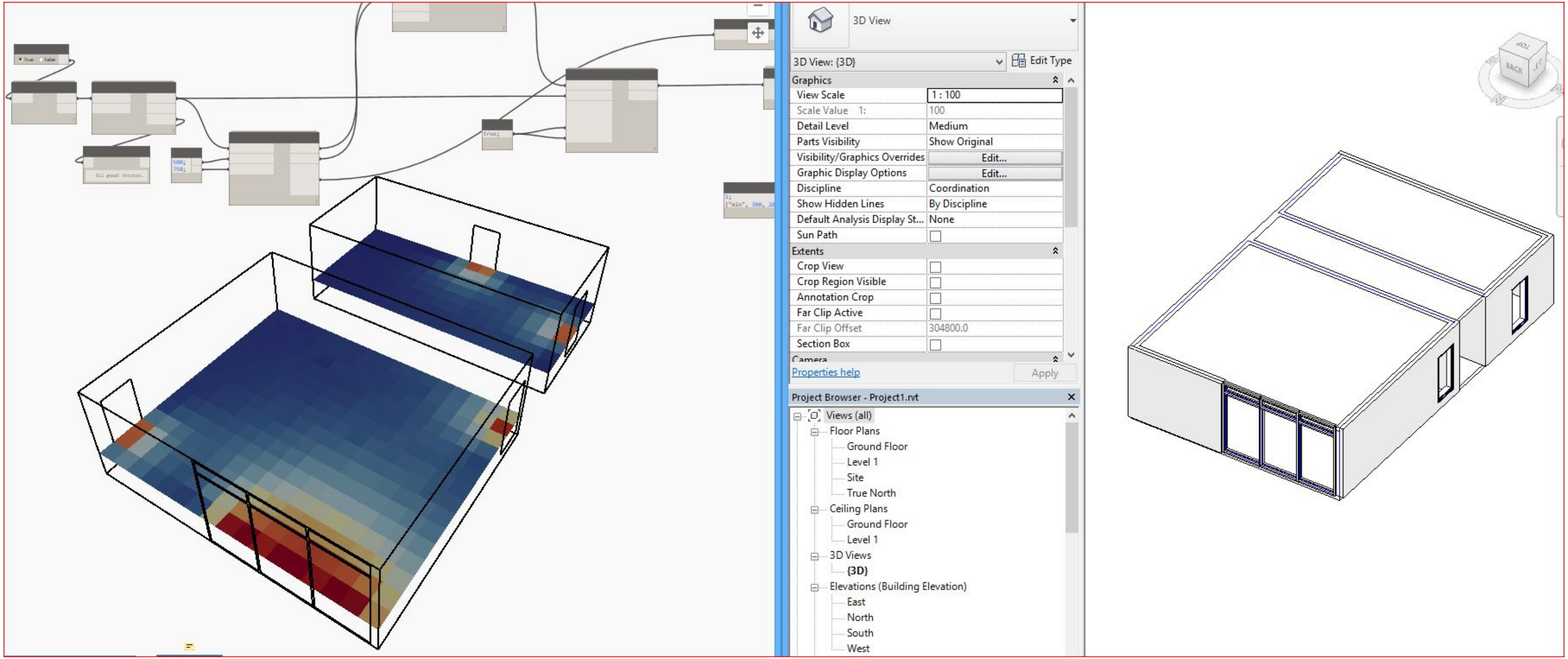
```python
import sys
from scriptcontext import sticky

if "hb_lib_path" in sticky:
    sys.path.append(sticky["hb_lib_path"])

try:
    from honeybeex.radiance.recipe.gridbased import GridBasedAnalysisRecipe
    from honeybeex import wrapper
except ImportError:
    raise ImportError("Failed to import Honeybee libraries.\n" + \
                      "Set-up path to Honeybee libraries in honeybee_honeybee component")

if _HBSky and _testPoints and ptsVectors_:
    if _radParameters_:
        _radParameters_ = _radParameters_.unwrap()

    analysisRecipe = GridBasedAnalysisRecipe(
        _HBSky.unwrap(), _testPoints, ptsVectors_, _simulationType_, _radParameters_
        )

    analysisRecipe = wrapper.Wrapper(analysisRecipe)
```

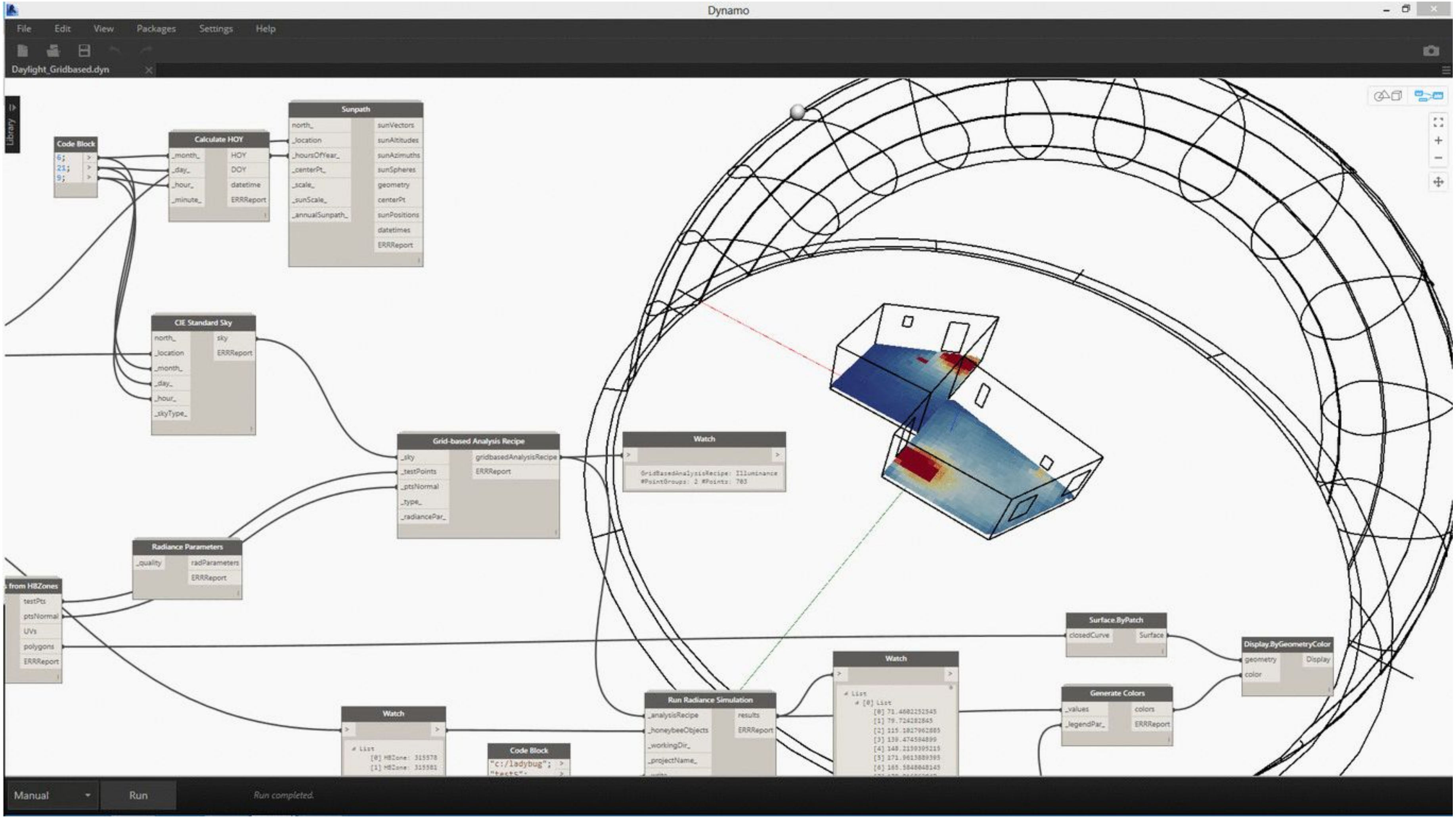






Library
File Path
Browse...
C:\EnergyPlusV8-3-0\WeatherData\USA_CA_San.Francisco.Intl.AP.724940_TMY3.epw
Import location data from epw file
Import Location
_epwFile
location
Calculate sunpath and sun vectors
Sunpath
north_
_location
hoursOfYear
centerPt
scale
sunScale
annualSunpath
sunVectors
sunAltitudes
sunAzimuths
sunSpheres
geometry
centerPt
sunPositions
datetimes
Calculate hours of the year for sunpath
Code Block
6;
Code Block
7..20;
Calculate HOY
month
day
hour
minute
HOY
DOY
datetime
Generate test points
Number of segments
14;
Distance from base
0.1;
Generate Test Points
_testSurfaces
numOfSegments
disFromSurface
testPts
ptsNormal
Calculate Sunlight hours
Sunlighthours Analysis
_testPoints
_geometries
_sunVectors
sunlightHours
Create test geometries!
Plane.XY
Plane
Code Block
100;
Rectangle.ByWidthLength
plane
width
length
Rectangle
Surface.ByPatch
closedCurve
Surface
Point.ByCoordinates
x
y
z
Point
Vector.YAxis
Vector
Code Block
45;
Geometry.Rotate
geometry
origin
axis
degrees
Geometry
Surface.ByPatch
closedCurve
Surface
Code Block
0.5;
Geometry.Scale
geometry
xamount
yamount
zamount
Geometry
List.Create
item0
item1
list
Color surfaces based on results
Color Surface
_analysisSurfaces
_values
legendPar
Display
Create legend parameters
Code Block
6;
LB Color Range
index
colors
Legend Parameters
cType
domain
colors
legendPar


3D View
3D View: {3D}
Edit Type
Graphics
View Scale
1 : 100
Scale Value 1:
100
Detail Level
Medium
Parts Visibility
Show Original
Visibility/Graphics Overrides
Edit...
Graphic Display Options
Edit...
Discipline
Coordination
Show Hidden Lines
By Discipline
Default Analysis Display St...
None
Sun Path
Extents
Crop View
Crop Region Visible
Annotation Crop
Far Clip Active
Far Clip Offset
304800.0
Section Box
Camera
Properties help
Apply
Project Browser - Project1.rvt
Views (all)
Floor Plans
Ground Floor
Level 1
Site
True North
Ceiling Plans
Ground Floor
Level 1
3D Views
{3D}
Elevations (Building Elevation)
East
North
South
West

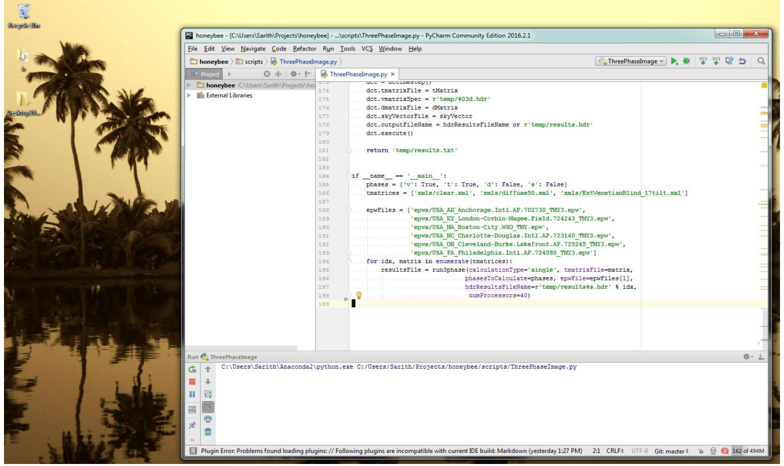

Dynamo
File
Edit
View
Packages
Settings
Help
Daylight_Gridbased.dyn
Code Block
6;
21;
9;
Calculate HOY
month
day
hour
minute
HOY
DOY
datetime
ERRReport
Sunpath
north_
_location
hoursOfYear
centerPt
scale
sunScale
annualSunpath
sunVectors
sunAltitudes
sunAzimuths
sunSpheres
geometry
centerPt
sunPositions
datetimes
ERRReport
CIE Standard Sky
north_
_location
month
day
hour
skyType
sky
ERRReport
Grid-based Analysis Recipe
_sky
_testPoints
_ptsNormal
type
radiancePar
gridbasedAnalysisRecipe
ERRReport
Watch
Radiance Parameters
_quality
radParameters
ERRReport
from HBZones
testPts
ptsNormal
UVs
polygons
ERRReport
Surface.ByPatch
closedCurve
Surface
Display.ByGeometryColor
geometry
color
Display
Generate Colors
_values
legendPar
colors
ERRReport
Run Radiance Simulation
_analysisRecipe
_honeybeeObjects
workingDir
projectName
results
ERRReport
Code Block
"c:/ladybug";
Manual
Run
Run completed.

Goals

Goals

Enable seamless Cross-OS compatibility

Microsoft Windows

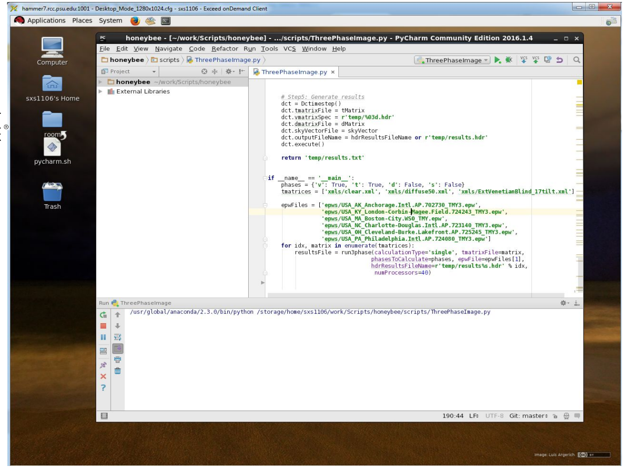


ubuntu

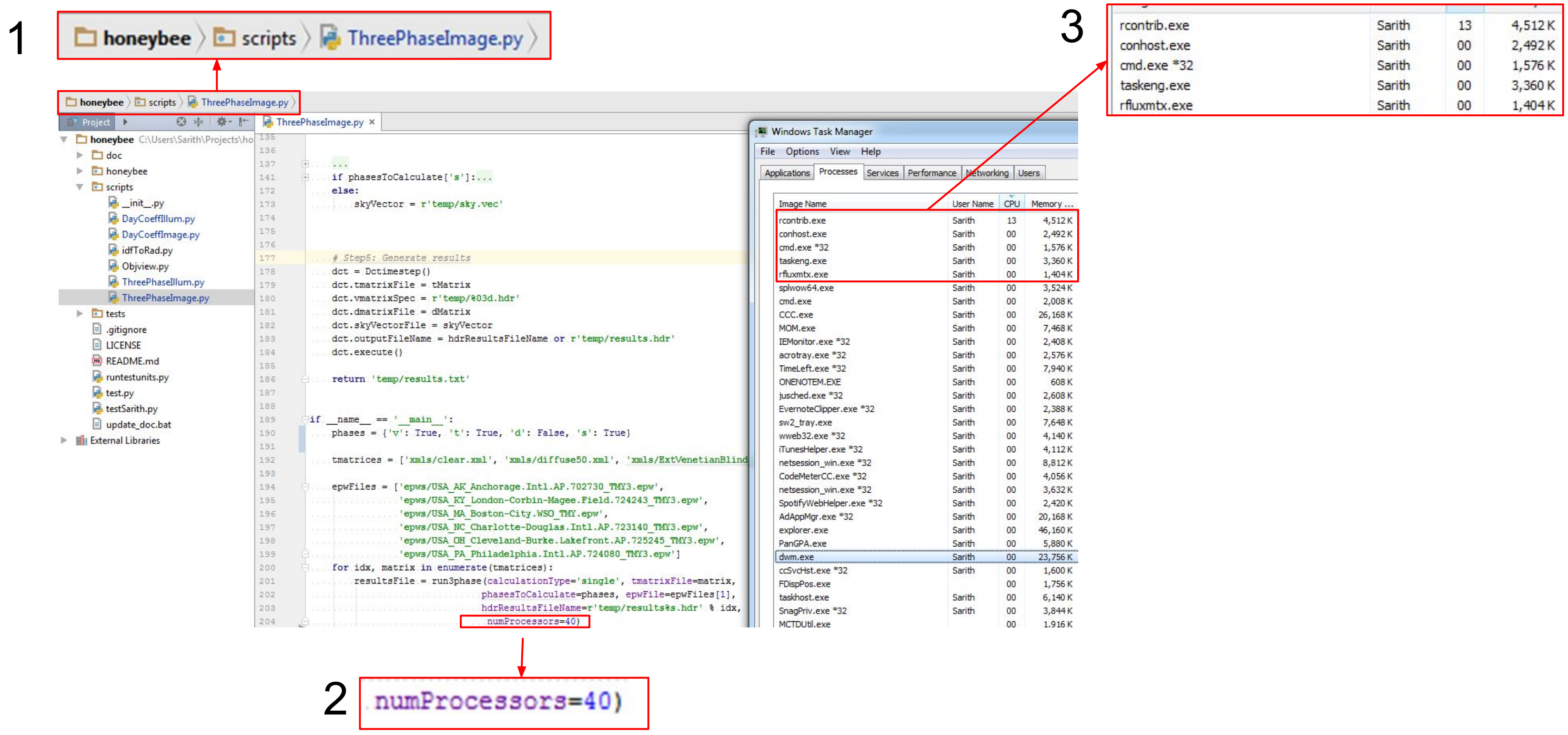
redhat LINUX

Mac OS X

amazon webservices

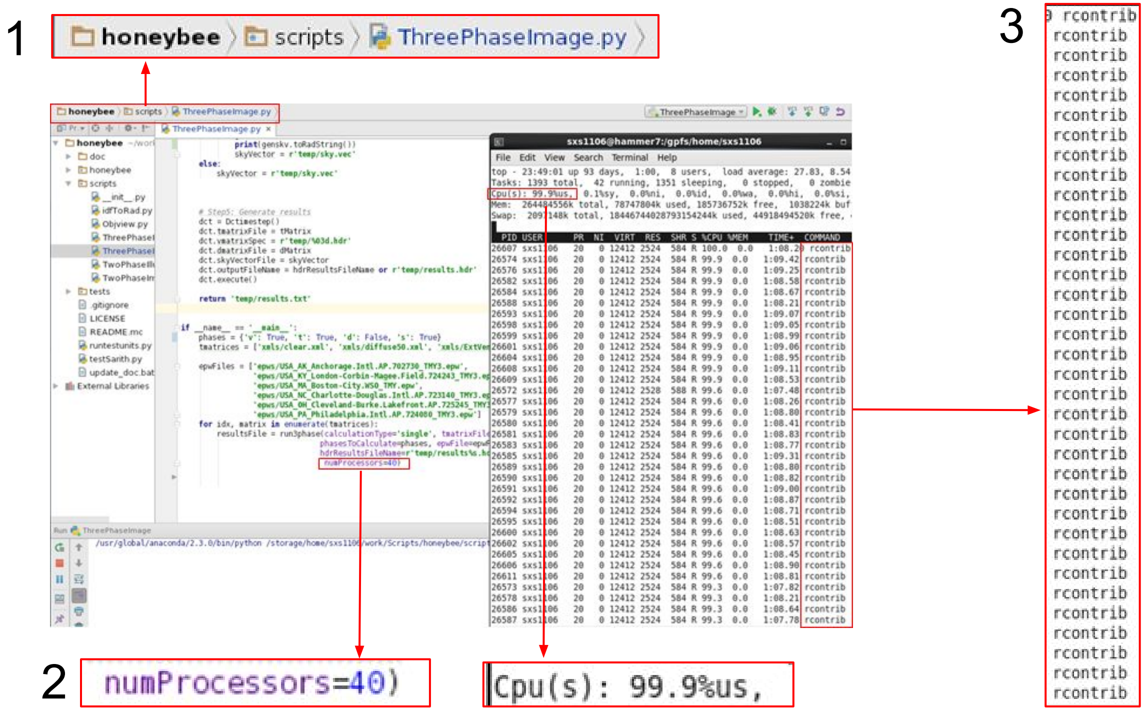


A 3 Phase Method Workflow in Windows



System Config: Remote Cluster, Xeon, 40-Core

The same script in a Unix environment



System Config: Desktop, i7, 8-core

Behind the scenes: genskyvec(.pl)

```
genskv = Genskyvec()
genskv.inputSkyFile = r'temp/sky.rad'
genskv.outputFile = r'temp/sky.vec'
genskv.skySubdivision = 4
genskv.execute()
```

"C:\Program Files\OpenStudio 1.11.0\strawberry-perl-5.16.2.1-32bit-portable-reduced\perl\bin\perl.exe" "C:\Program Files\OpenStudio 1.11.0\share\openStudio\Radiance\bin\genskyvec.pl" -m 4 < temp\sky.rad > temp\sky.vec

/gpfs/home/sxs1106/work/Radiance/bin/genskyvec -m 4 < temp/sky.rad > temp/sky.vec

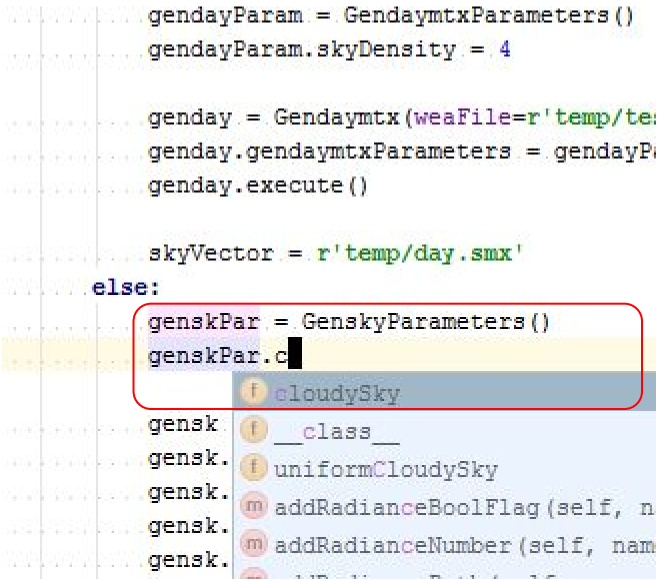
PERL Interpreter Genskyvec Inputs

Goals

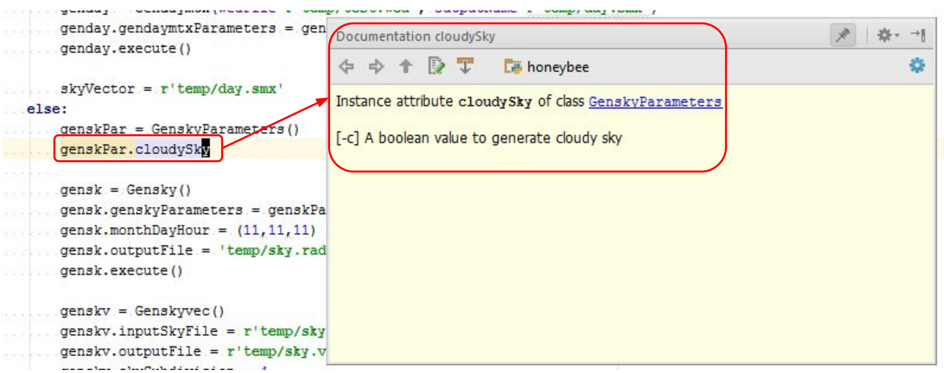
Enable seamless Cross-OS compatibility

Simplify syntax with abstractions

Human readable inputs for infrequently used commands



Code completion on compatible IDEs.
(API remembers and prompts for inputs)



Look-up class documentation while coding workflows.

A self-documenting API

Index

Sub-modules

- `honeybee.radiance.command`
- `honeybee.radiance.datatype`
- `honeybee.radiance.filemanager`
- `honeybee.radiance.geometry`
- `honeybee.radiance.material`
- `honeybee.radiance.parameters`
- `honeybee.radiance.postprocess`
- `honeybee.radiance.properties`
- `honeybee.radiance.recipe`
- `honeybee.radiance.runmanager`
- `honeybee.radiance.view`

`honeybee.radiance` module

Honeybee Radiance libraries.

SHOW SOURCE ≡

Sub-modules

`honeybee.radiance.command`

`honeybee.radiance.datatype`

Descriptors, factory classes etc for the Radiance library.

`honeybee.radiance.filemanager`

A collection of auxiliary funtions for working with radiance files and objects.

`honeybee.radiance.geometry`

A collection of methods for writing Radiance geometry file.

`honeybee.radiance.material`

`honeybee.radiance.parameters`

`honeybee.radiance.postprocess`

`honeybee.radiance.properties`

http://ladybug-analysis-tools.github.io/honeybee/doc/radiance/index.html

Goals

Enable seamless Cross-OS compatibility

Simplify syntax with abstractions

Type and Error Checking

Gensky Command

```
genskPar = GenskyParameters()
genskPar.turbidity = 0.01
gensk = Gensky()
gensk.genskyParameters = genskPar
gensk.monthDayHour = (11,11,11)
gensk.outputFile = 'temp/sky.rad'
gensk.execute()
```

Gensky Manpage

radsite.lbl.gov/radiance/man_html/gensky.1.html

"Values less than 1.0 are physically impossible."

Error Message

```
Traceback (most recent call last):
  File "C:/Users/Sarith/Projects/honeybee/scripts/ThreePhaseIllum.py", line 153, in <module>
    phasesToCalculate=phases,epwFile=epwFiles[1])
  File "C:/Users/Sarith/Projects/honeybee/scripts/ThreePhaseIllum.py", line 113, in run3phas
    genskPar.turbidity = 0.01
  File "C:\Users\Sarith\Projects\honeybee\honeybee\radiance\parameters\_frozen.py", line 53,
    object.__setattr__(self, key, value)
  File "C:\Users\Sarith\Projects\honeybee\honeybee\radiance\datatype.py", line 368, in __set
    raise ValueError(msg)
ValueError: The specified input for t (turbidity) is 0.01. This is below the valid range. Th
```

ValueError: The specified input for t (turbidity) is 0.01. This is below the valid range.

Goals

Enable seamless Cross-OS compatibility

Simplify syntax with abstractions

Type and Error Checking

Leverage Python to reduce effort

```
rfluxPara = RfluxmtxParameters()
rfluxPara.I = True
rfluxPara.aa = 0.1
rfluxPara.ad = 4096
rfluxPara.ab = 12
rfluxPara.lw = 0.0000001
rflux = Rfluxmtx()
skyFile = rflux.defaultSkyGround(r'temp/rfluxSky.rad',skyType='r4')
rflux.receiverFile = skyFile
rflux.sender = '-'
rflux.rfluxmtxParameters = rfluxPara
rflux.radFiles = [r"room.mat",r"room.rad"]
rflux.pointsFile = r"indoor_points.pts"
rflux.outputMatrix = r"temp/dayCoeff.dc"
rflux.execute()
```

Radiance

```
rfluxmtx" -y 6 -aa 0.1 -ab 12 -ad 4096 -lw 1e-07 -I - temp\rfluxSky.rad room.mat room.rad < indoor_points.pts > temp\dayCoeff.dc
```

Automate predictable tasks: Create a sky definition

```
rfluxPara = RfluxmtxParameters()
rfluxPara.I = True
rfluxPara.aa = 0.1
rfluxPara.ad = 4096
rfluxPara.ab = 12
rfluxPara.lw = 0.0000001
rflux = Rfluxmtx()
skyFile = rflux.defaultSkyGround(r'temp/rfluxSky.rad',skyType='r4')
rflux.receiverFile = skyFile
rflux.sender = '-'
rflux.rfluxmtxParameters = rfluxPara
rflux.radFiles = [r"room.mat",r"room.rad"]
rflux.pointsFile = r"indoor_points.pts"
rflux.outputMatrix = r"temp/dayCoeff.dc"
rflux.execute()
```

1.Create sky definition

2.Add rfluxmtx comments

rfluxSky.rad

```
#@rfluxmtx h=u u=Y
void glow ground_glow
0
0
4 1 1 1 0

ground_glow source ground
0
0
4 0 0 -1 180

#@rfluxmtx h=r4 u=Y
void glow sky_glow
```

```
rfluxmtx" -y 6 -aa 0.1 -ab 12 -ad 4096 -lw 1e-07 -I - temp\rfluxSky.rad room.mat room.rad < indoor_points.pts > temp\dayCoeff.dc
```

3. Assign sky as "sender"

Automate predictable tasks: Count a points-file

```
rfluxPara = RfluxmtxParameters()
rfluxPara.I = True
rfluxPara.aa = 0.1
rfluxPara.ad = 4096
rfluxPara.ab = 12
rfluxPara.lw = 0.0000001
rflux = Rfluxmtx()
skyFile = rflux.defaultSkyGround(r'temp/rfluxSky.rad',skyType='r4')
rflux.receiverFile = skyFile
rflux.sender = '-'
rflux.rfluxmtxParameters = rfluxPara
rflux.radFiles = [r"room.mat",r"room.rad"]
rflux.pointsFile = r"indoor_points.pts"
rflux.outputMatrix = r"temp/dayCoeff.dc"
rflux.execute()
```

1.Count the number of points

indoor_points.pts

```
2.0000	0.5000	0.8000	0.0000	0.0000	1.0000
2.0000	1.5000	0.8000	0.0000	0.0000	1.0000
2.0000	2.5000	0.8000	0.0000	0.0000	1.0000
2.0000	3.5000	0.8000	0.0000	0.0000	1.0000
2.0000	4.5000	0.8000	0.0000	0.0000	1.0000
2.0000	5.5000	0.8000	0.0000	0.0000	1.0000
```

```
rfluxmtx" -y 6 -aa 0.1 -ab 12 -ad 4096 -lw 1e-07 -I - temp\rfluxSky.rad room.mat room.rad < indoor_points.pts > temp\dayCoeff.dc
```

2. Assign the number of points in the command

Make the workflow more flexible

```
rfluxPara = RfluxmtxParameters()
rfluxPara.I = True
rfluxPara.aa = 0.1
rfluxPara.ad = 4096
rfluxPara.ab = 12
rfluxPara.lw = 0.0000001
rflux = Rfluxmtx()
skyFile = rflux.defaultSkyGround(r'temp/rfluxSky.rad',skyType='r4')
rflux.receiverFile = skyFile
rflux.sender = '-'
rflux.rfluxmtxParameters = rfluxPara
rflux.radFiles = [r"room.mat",r"room.rad"]
rflux.pointsFile = r"indoor_points.pts"
rflux.outputMatrix = r"temp/dayCoeff.dc"
rflux.execute()
```

Inputs can be in any order.

```
rfluxmtx" -y 6 -aa 0.1 -ab 12 -ad 4096 -lw 1e-07 -I - temp\rfluxSky.rad room.mat room.rad < indoor_points.pts > temp\dayCoeff.dc
```

The ordering of inputs needs to follow a certain order.

rfluxmtx [−v][rcontrib options] { sender.rad | - } receivers.rad [-i system.oct] [system.rad ..]

Goals

Enable seamless Cross-OS compatibility

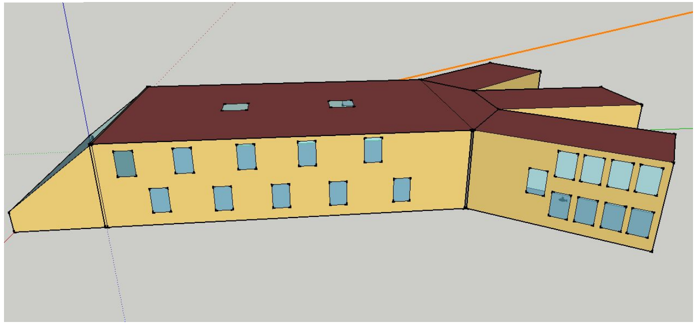
Simplify syntax with abstractions

Type and Error Checking

Leverage Python to reduce effort

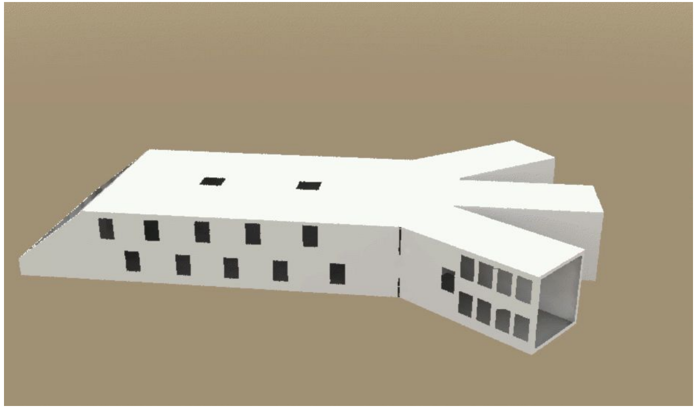
Interoperability between simulation platforms

```python
from honeybee.energyplus import filemanager, geometryrules
from honeybee.hbsurface import HBSurface
from honeybee.hbfensurface import HBFenSurface
from honeybee.hbshadesurface import HBShadingSurface
from honeybee.hbzone import HBZone
import os


def idfToRadString(idfFilePath):
    """Convert an idf file geometery to a radiance definition.

    Radiance materials are assigned based on surface types and not from
    EnergyPlus materials or construction. You can create and map your
    own Radiance materials by adding a few number of lines to the code.
    """
    objects = filemanager.getEnergyPlusObjectsFromFile(idfFilePath)

    # if the geometry rules is relative then all the points should be added
    # to X, Y, Z of zone origin
    geoRules = geometryrules.GlobalGeometryRules(
        *objects['globalgeometryrules'].values()[0][1:4]
    )

    hbObjects = {'zone': {}, 'buildingsurface': {}, 'shading': {}}

    # create zones
    for zoneName, zoneData in objects['zone'].iteritems():
        # create a HBZone
        zone = HBZone.fromEPString(",".join(zoneData), geometryRules=geoRules)
        hbObjects['zone'][zoneName] = zone

    # create surfaces
    for surfaceName, surfaceData in objects['buildingsurface:detailed'].iteritems():
        surface = HBSurface.fromEPString(",".join(surfaceData))
        surface.parent = hbObjects['zone'][surfaceData[4]]
        hbObjects['buildingsurface'][surfaceName] = surface
```



IDF

Radiance



HDR

Goals

Enable seamless Cross-OS compatibility

Simplify syntax with abstractions

Type and Error Checking

Leverage Python to reduce effort

Interoperability between simulation platforms

Apply Object Oriented Programming (OOP) where required

Radiance syntax alludes to OOP patterns

NAME

rcontrib - compute contribution coefficients in a RADIANCE scene

SYNOPSIS

rcontrib [−n nprocs][−V][−c count][−fo | −r][−e expr][−f source][−o ospec][−p p1=V1,p2=V2][−b binv][−bn nbins] { −m mod | −M file } [$EVAR] [@file] [rtrace options] octree
rcontrib [options] −defaults

NAME

rfluxmtx - compute flux transfer matrix(es) for RADIANCE scene

SYNOPSIS

rfluxmtx [−v][rcontrib options] { sender.rad | - } receivers.rad [-i system.oct] [system.rad ..]

DESCRIPTION

NAME

vwrays - compute rays for a given picture or view

SYNOPSIS

vwrays [-i -u -f{a|f|d} -c rept | -d] { view opts .. | picture [zbuf] }

DESCRIPTION

Vwrays takes a picture or view specification and computes the ray origin and direction corresponding to

Honeybee “wraps” Radiance syntax through OOP

Base Class

Implements rtrace options ab, ad, as etc.

```
@frozen
class GridBasedParameters(AdvancedRadianceParameters):
    """Radiance Parameters for grid based analysis...."""

    def __init__(self, quality=None):
```

Subclass

Implements rcontrib.

```
@frozen
class RcontribParameters(GridBasedParameters):
    """Radiance Parameters for rcontrib command including rtrace parameters.

    def __init__(self, modFile=None):
```

Subclass

Implements rfluxmtx.

```
@frozen
class RfluxmtxParameters(GridBasedParameters):

    def __init__(self, sender=None, receiver=None, octree=None, systemFile
        """Init parameters."""
```

Goals

Enable seamless Cross-OS compatibility

Simplify syntax with abstractions

Type and Error Checking

Leverage Python to reduce effort

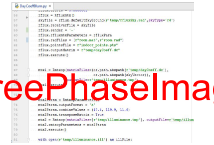
Interoperability between simulation platforms

Apply Object Oriented Programming (OOP) where required

Recipes !

Use the API to create “recipes” for common workflows

ThreePhaseIllum.py



ThreePhaseImage.py

DayCoeffImage.py

DayCoeffImage.py

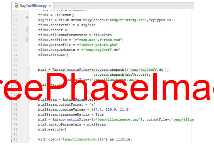
PhotonMapImage.py

FivePhaseImage.py

```python
def run3phase(phasesToCalculate={'v':True,'t':True,'d':True,'s':True},
              calculationType='annual',epwFile=None,tmatrixFile=None):


    if phasesToCalculate['v']:
        # Step1: Create the view matrix.
        rfluxPara = RfluxmtxParameters()
        rfluxPara.I = True
        rfluxPara.aa = 0.1
        rfluxPara.ab = 10
        rfluxPara.ad = 65536
        rfluxPara.lw = 1E-5

        #step 1.1 Invert glazing surface with xform so that it faces inwards
        xfrPara = XformParameters()
        xfrPara.invertSurfaces = True

        xfr = Xform()
        xfr.xformParameters = xfrPara
        xfr.radFile = 'glazing.rad'
        xfr.outputFile = 'glazingI.rad'
        xfr.execute()

        rflux = Rfluxmtx()
        rflux.sender = '-'

        #Klems full basis sampling and the window faces +Y
        recCtrlPar = rflux.ControlParameters(hemiType='kf',hemiUpDirection='+Z')
        rflux.receiverFile = rflux.addControlParameters('glazingI.rad',
                                                        {'Exterior_Window':recCtrlPar})
        rflux.rfluxmtxParameters = rfluxPara
        rflux.pointsFile = 'indoor_points.pts'
```

Use “recipes” independently or through 3D Environments

ThreePhaseIllum.py



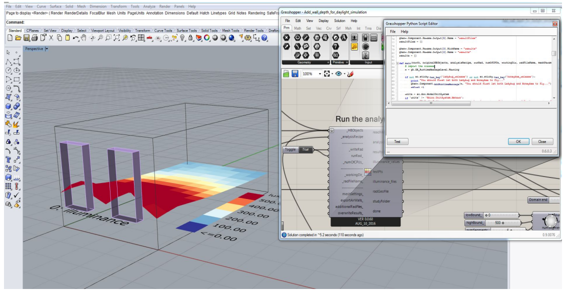
ThreePhaseImage.py

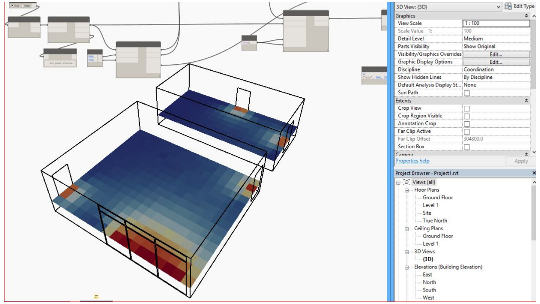
DayCoeffImage.py

DayCoeffImage.py

PhotonMapImage.py

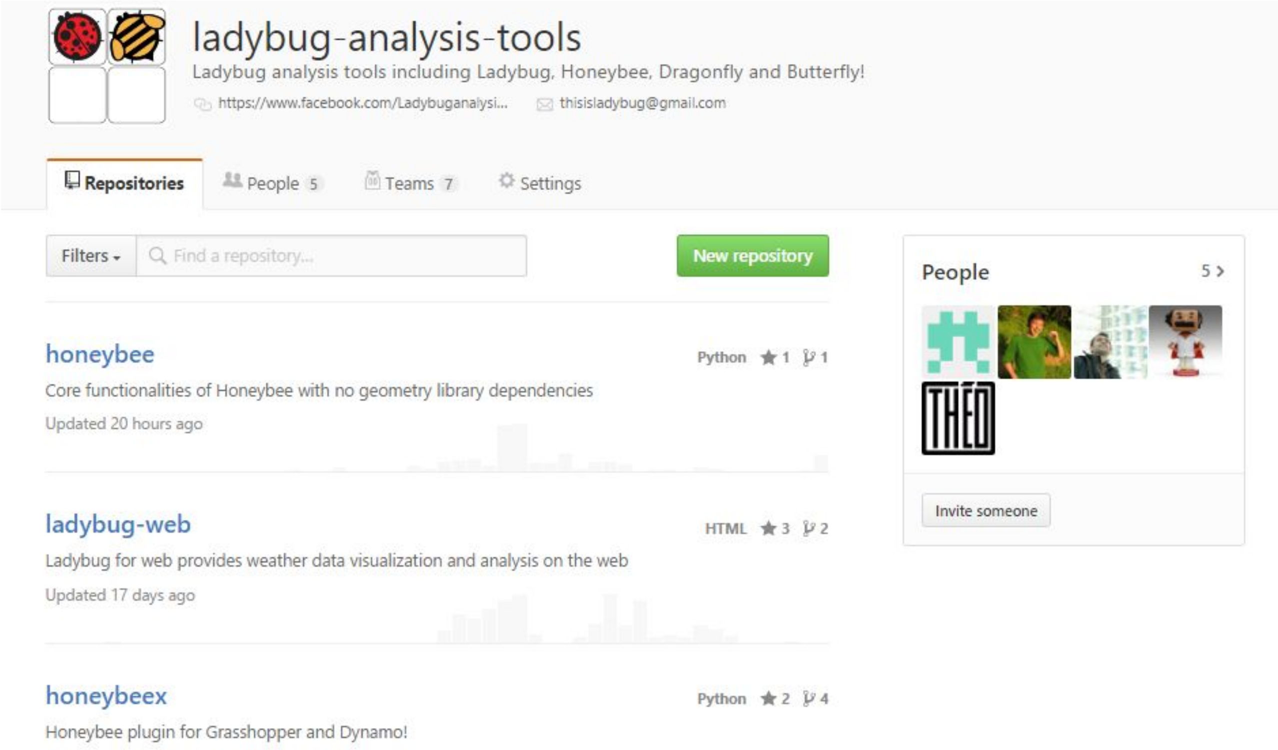
FivePhaseImage.py





Live Demo

How to get involved?



https://github.com/ladybug-analysis-tools

Thank You!