OWL
Occupant Wellbeing through Lighting

V1.0 (15 Oct 2021)

This is the documentation of OWL: an opensource workflow in Rhino Grasshopper developed for performing spectral lighting simulations, and evaluating non-image-forming effects of light on occupant wellbeing.

This tool uses parsimonious approach with minimal inputs, for evaluating the following NIF metrics: **Melanopic metrics** (subset of alpha-opic metrics) recommended by CIE as part of CIES026 standard, and **Circadian metrics** proposed by Lighting Research Centre.

This is developed by Marshal Maskarenj at LAB/LOCI UCLouvain, along with Bertrand Deroisy and Sergio Altomonte; as part of the FNRS funded project SCALE (Shading Control Algorithms from Luminance-based Evaluations).

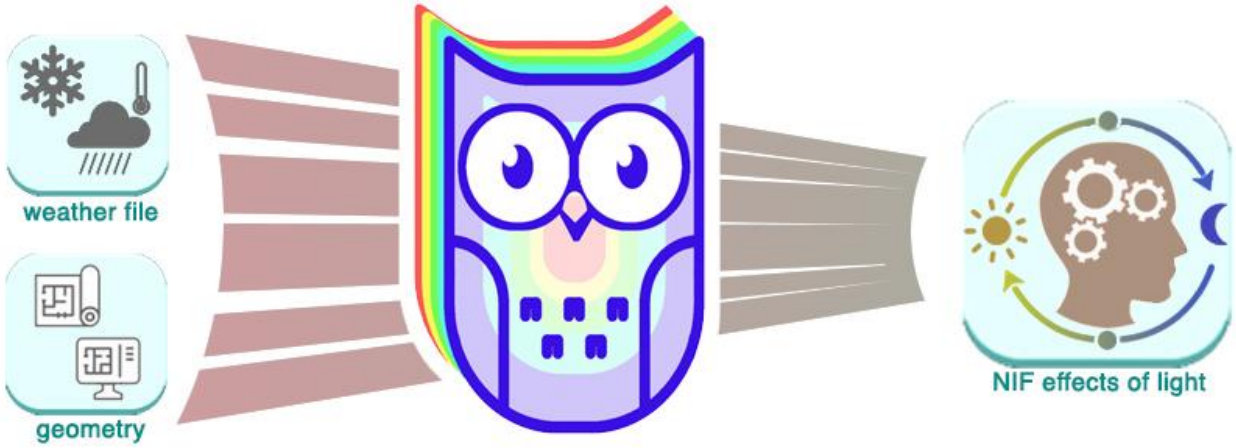
Contents

OWL: Occupant Wellbeing through Lighting

Overview

Functionality of OWL

The workflow of OWL includes multiple models for evaluating non-image forming (NIF) metrics, as indicators of the non-visual effects of light on occupant well-being. This integrated approach of combining models, requires minimal inputs from the users in evaluating the following NIF metrics: α-opic metrics recommended by CIE as part of the CIES026 standard, and circadian metrics proposed by Lighting Research Centre.



A user needs to define only the weather file (in .EPW format) and geometrical data of the scene (from Rhino viewport) as input to the workflow. Using the following combination of {either Perez or CIE Model} >> {Luminance to CCT models} >> {CIE015 standard} >> and {either of CIES026 standard or LRC's protocol}, this workflow evaluates these NIF metrics of lighting under unobstructed sky conditions. For viewing conditions indoors, the per-channel SPD weightage is evaluated for generating a Radiance sky file, and a luminance map of the scene is generated via Radiance image-based simulations with spectral sky as light source. With the application of multiple masks, the luminance is discretized into patches, and the NIF metrics are evaluated using a combination of {Discretized Luminance} >> {Luminance to CCT models} >> {CIE015 standard} >> and {either of CIES026 standard or LRC's protocol}.

In the present version, luminaires and reflected components of daylight are not considered for evaluating NIF metrics, but may be incorporated in future versions.

Structure of this document

The first part of the document presents instructions for installing prerequisites, which are needed for OWL to function on a system. The second part presents a detailed description of the originally developed components. The third part describes components based on functionality, along with their functional screen-captures on the grasshopper canvas.

Installation instructions as prerequisite to running OWL

Installing Python

1. Python 2.7 can be downloaded from https://www.python.org/downloads/release/python-2718/
2. The following need to be added to path: <<C:\Python27>> and <<C:\Python27\Scripts>>
more information on adding path variables can be accessed here: https://helpdeskgeek.com/windows-10/add-windows-path-environment-variable/
3. Using Pip, the following components must be installed: *numpy*, *scipy*, and *matplotlib*... Sample command for pip installation of these packages is like <<pip install numpy>>

Installing Radiance

1. Radiance needs to be installed on Windows. It is available at https://github.com/LBNL-ETA/Radiance/releases
The latest release as of this day is here: https://github.com/LBNL-ETA/Radiance/releases/tag/8aeb24b8
2. Radiance needs to be added to Path, if not automatically done in installation process.

Installing latest Ladybug 1.xx for access to Ladybug Image Viewer component

1. The latest Ladybug 1.x can be downloaded from https://www.food4rhino.com/en/app/ladybug-tools (latest is 1.3 as of this day)
2. With the folder unzipped, and *installer.gh* opened in Grasshopper – the installation instructions can be followed.

Installing GH_Cpython component

1. The component can be downloaded in a zip file from https://www.food4rhino.com/en/app/ghcpython
2. In Grasshopper, appropriate folder needs to be opened as File > Special Folders > Component Folders
3. From the zip file, *GH_CPython* and *FastColoredTextBox.dll* need to be pasted into Libraries folder

Modifying following locations on the GH script:

The locations defined for ‘Folder’, ‘EPW File’, ‘view masks’ (three) and ‘region mask’ need to be modified to suit appropriate locations the user’s systems.

Description of components based on cluster structure:

This workflow is divided into 4 clusters: OWL-Common, OWL-Sky, OWL-View and OWL-Spectral. OWL-Common contains preliminary components used for initiating simulations, such as data extraction from weather files, identifying solar position in the sky-dome, and evaluating sky luminance using standard models. Also included in this cluster are few components, for visualizing sky-luminance distribution on the Rhino viewport. OWL-Sky and OWL-View include components specific to unobstructed sky and to observer's view, respectively, which are needed as part of specific workflows. OWL-Spectral contains components for spectral sky evaluations, which includes protocols from literature, standards (CIE015 and CIES026) and toolboxes (CIE alpha-opic toolbox, and LRC circadian toolbox).

1. OWL-Common

The OWL-Common cluster is aimed at preliminary evaluations, and includes the following components as part of the simulation workflow: **OpenEPW_loc**, **SunPos**, **OpenEPW**, **CIE_Skygen**, **PerezSky**, **skyLum_map**, **3CspectralSky** and **HDR2DiscreteLumEx**. The components are described further:

OpenEPW_loc

This component has one input node ('epwFile') and five output nodes ('name', 'latitude', 'longitude', 'timeZone' and 'elevation').
This component reads an EnergyPlus Weather (EPW) file, and extracts location-related data from its static headers. An EPW file is a text-based file that contains 8 rows of static data corresponding to the location, design conditions and daylight-saving parameters; followed by 8760 rows of hourly datasets.

This component takes such weather file as input, parses through the text-based contents of the header, and extracts parameters of location-name, latitude, longitude, time-zone, and elevation, respectively. The Python code for this component is available here.

SunPos

This component has five input nodes ('latitude', 'longitude', 'timeZone', 'hour' and 'DOY') and two output nodes ('sunAltitude' and 'sunAzimuth').
It takes the first three inputs corresponding to location from **OpenEPW_loc**, along with user-defined Hour-of-Day and Day-of-Year; and uses this input to generate the altitude and azimuth angles of Sun -- at the given location for the specified hour.

The altitude angle ('sunAltitude') corresponds to the angle of Sun relative to earth's horizon, whereas azimuth angle ('sunAzimuth') corresponds to the angular distance between projected line-of-sight of Sun on the ground, with either due South or due North. This component uses the following underlying equations to evaluate the output from the input data:

The solar altitude angle and solar azimuth can be evaluated as:

$$sunAzimuth = cos^{-1}\left(\frac{sin(delta) * cos(phi)) - (cos(delta) * sin(phi) * cos(HRA)}{\cos(sunAltitude)}\right)$$
$$sunAltitude = sin^{-1}[\sin(delta) * \sin(phi) + \cos(delta) * \cos(phi) * \cos(HRA)]$$

Here delta is the declination angle, phi is latitude angle and HRA is the hour angle calculated as below:

$$HRA = 15 * (LST - 12)$$
$$LST = HOD + \left(\frac{TC_{factor}}{60}\right)$$

Here LST is Local solar time, HOD is hour of day and TC_{factor} is time correction factor calculated as below:

$$TC_{factor} = 4 * (longitude - LSTM) + EoT$$
$$LSTM = 15 * (UTC)$$

Here LSTM is local standard time meridian, UTC is universal coordinated time, and the equation of time EoT is calculated as below:

$$EoT = 9.87 * sin\,(2 * B) - 7.53 * cos\,(B) - 1.5 * sin\,(B)$$
$$where\ B = (DOY - 81) * \left(\frac{360}{365}\right)$$

Here DOY is the day of year. The declination angle is computed as below:

$$delta = -23.45 * cos\left(\frac{360}{365}\right) * (DOY + 10)$$

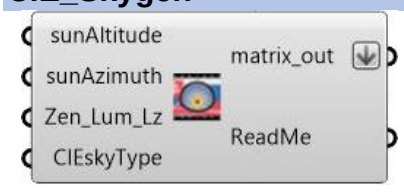
The Python code for this component is available here.

OpenEPW

This component has three input nodes ('epwFile', 'output_index', and 'hour_of_year') and four output nodes ('data', 'dtype', 'index_list' and 'ReadMe'). It extracts temporally-dynamic data from the EnergyPlus Weather (EPW) file, which is a text-based file format containing 8 rows of static data, followed by 8760 rows of datasets corresponding to every hour of the year.

Each of the hourly dataset rows are comprised of 35 environmental parameters – corresponding to various data-types of temperature, irradiance, illuminance, etc. While connected to the weather file, this component uses hour-of-year (HOY) and output-index as inputs, and parses through the dynamic datasets to extract corresponding 'data' of the specified 'dtype'. Here, 'output-index' is a numeric value between 1 and 35 corresponding to type of environmental parameter('dtype'); whereas HOY translates to a value between 1 and 8760 (365 x 24) corresponding to any given hour of any given day. For example, to extract Zenith Luminance at 10 am on 2nd Jan, the HOY is set at 34, and 'output-index' is set at 20. The Python code for this component is available here.

CIE_Skygen



This component has four input nodes ('sunAltitude', 'sunAzimuth', 'Zen_Lum_Lz' and 'CIEskyType'), and two output nodes ('matrix_out' and 'ReadMe'). It uses the CIE standard sky model for generating sky-luminance data across 145 equal-sized discrete sky-patches ('matrix_out'), at various altitudinal/azimuthal positions in the sky-dome.

The 145 positions correspond to the continuous scheme used in Daysim, adopted from Tregenza sky-subdivision scheme. 145 luminance values are generated as output, as a function of the inputs, which correspond to Solar Altitude, Solar Azimuth, Zenith Luminance, and CIE Sky-type, respectively. The first three data for a given HOY are taken from **SunPos** and **OpenEPW** modules respectively (by setting index at 20); while the CIE-Sky-type is user defined for parametric simulations at various sky-types. This parameter takes a value between 1 and 15, which corresponds to the 15 CIE sky-types for a range of overcast-, uniform-, intermediate- and clear-sky conditions. This component uses the following underlying equations to evaluate the output from the input data:

The ratio of luminance of the arbitrary sky element (or, in our case, the luminance of the 145 pre-defined positions) with respect to the luminance at zenith may be represented as a combination of gradation ratio (φZ/ φ0) and indicatrix ratio (fχ/fZs) as:

$$\frac{L_{\gamma\alpha}}{L_z} = \frac{f(\chi).\varphi(Z)}{f(Z_s).\varphi(0^\circ)}$$

Here, the function f expresses the scattering indicatrix while the function φ expresses the luminance gradation. Luminance at zenith is taken from weather file as input, and the luminance at any patch can be evaluated using this ratio. Here, χ represents the angular distance of the sky-element from the sun, and is calculated as:

$$\chi = cos^{-1}(cosZ_s . cosZ + sinZ_s . sinZ . cosAz)$$

Here Az = |α – α_s|, where α and α_s are azimuthal angles of the vertical plane of the sky element and sun position respectively. Z and Z_s represent the zenith angles of the specific patch and of the sun, respectively.

For the luminance gradation function, the gradation at any sky element at angle Z from zenith is defined as:

$$\varphi(Z) = 1 + a.exp\left(\frac{b}{cosZ}\right)$$

This is valid for 0<Z<π/2; however at the horizon, φ(π/2)=1. By definition, the Z value at zenith is 0 degrees, as such,

$$\varphi(0°) = 1 + a.exp(b)$$

The position of the sun and the angle between the sun and sky-element is included in the Scattering indicatrix function, which also accounts for the change in sun position as follows:

$$f(\chi) = 1 + c.\left(exp(d.\chi) - exp\left(d.\frac{\pi}{2}\right)\right) + e.cos^2\chi$$

Here χ is the angular distance of the sky-element from Sun. In the indicatrix function computed for the zenith, solar angular distance of the zenith is the same as zenith distance of the Sun. Hence:

$$f(Z_s) = 1 + c.\left(exp(d.Z_s) - exp\left(d.\frac{\pi}{2}\right)\right) + e.cos^2 Z_s$$

In these equations, ‘a’, ‘b’, ‘c’, ‘d’, and ‘e’ are coefficients that depend on sky-type, and their values are available in this paper here. The Python code for this component is available here.

PerezSky

This component has seven input nodes ('sunAltitude', 'sunAzimuth', 'Zen_Lum_Lz', 'HorzDiffRad', 'DirNormRad', 'OptAirMass', and 'NormExIrrad') and two output nodes ('matrix_out' and 'ReadMe'). It uses the Perez all-weather model for generating sky-luminance data across 145 equal-sized discrete sky-patches, as described previously. 145 luminance output values are generated as a function of the inputs, which are: Solar Altitude, Solar Azimuth, Zenith Luminance, Horizontal Diffuse Radiation, Direct Normal Radiation, Optical Air Mass, and Extra-terrestrial Direct Normal Radiation, respectively.

The Solar altitude/azimuth data is taken from the outputs of **SunPos**, while the other environmental parameters for a given HOY are taken from **OpenEPW** by setting 'index' at: 20, 16, 15, and 12; for: Zenith Luminance, Diffuse Horizontal Radiation, Direct Normal Radiation, and Extra-terrestrial Direct Normal Radiation, respectively. The Optical Air Mass parameter is user-defined for parametric simulations at various turbidity values. This component uses the following underlying equations to evaluate the output from the input data:

The relative luminance is defined as ratio between luminance of the considered sky element and luminance of a reference sky element (which, in our case is Zenith luminance from weather file), as

$$l_v = f(Z,\chi) = [1 + a\ exp\ (b/\cos Z)] * [1 + c\ exp(d\ \chi) + e\cos^2\chi]$$

Where Z is the zenith angle of the sky element, {a, b, c, d, e} represent adjustable coefficients, and χ represents the angular distance of the sky-element from the sun, which is calculated as:

$$\chi = \cos^{-1}(\cos Z_s . \cos Z + \sin Z_s . \sin Z . \cos Az)$$

Here Az = |α – α_s|, where α and α_s are azimuthal angles of the vertical plane of the sky element and sun position respectively. Z and Z_s represent the zenith angles of the specific patch and of the sun, respectively. Since l_v represents relative luminance, when the zenith luminance is known, luminance of any patch can be evaluated as:

$$L_{patch} = L_{zenith} * \frac{f(Z,\chi)}{f(0,Z_s)}$$

This is because, at zenith, the zenith angle is 0 and the angular distance with sun is the same as its zenith angle. The significant difference between Perez Model and CIE model, is that in the CIE model, the values of the coefficients a through e are taken from tabulated values, depending on the sky-type. For 15 different sky-types ranging from overcast to intermediate to clear, the coefficients are available in literature, but the user needs to define the sky-type. In Perez model, however, the coefficients are calculated based on data from the weather file itself – and two parameters of sky's clearness (ε) and sky brightness (Δ) are precalculated from weather data for evaluating the coefficients, as:

$$\varepsilon = \frac{\left[\frac{E_{ed} + E_{es}}{E_{ed}} + 1.041\ Z^3\right]}{[1 + 1.041\ Z^3]}$$

$$\Delta = m\frac{E_{ed}}{E_{es0}}$$

Here 'm' refers to optical air mass, Z is the solar zenith angle, and E_{ed}, E_{es} and E_{es0} refer to horizontal diffuse irradiance, normal incident direct irradiance and normal incident extraterrestrial irradiance respectively, from the weather file. Based on ε and Δ, the coefficients {a, b, c, d, e} are calculated as:

$$\theta = (\theta_1(\varepsilon) + (\theta_2(\varepsilon) * Z) + delta * (\theta_3(\varepsilon) + (\theta_4(\varepsilon) * Z))$$

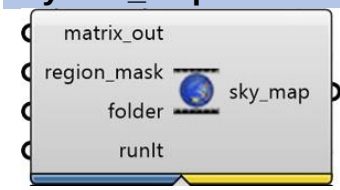
Where θ stands for a, b and e, and coefficients {a_1,a_2,a_3,a_4}, {b_1,b_2,b_3,b_4} and {e_1,e_2,e_3,e_4} are selected from tabulated values based on ε, with thresholds of ε defined in this paper.

For c and d, the same functions as earlier are valid for ε thresholds higher than 1.065; but for sky clearness value between 1 and 1.065, the following functions are used:

$$c = exp\,[(\Delta(c_1 + c_2 Z))^{c3} - c_4$$
$$d = -exp[\Delta(d_1 + d_2 Z)] + d_3 + \Delta d_4$$

Where sub-coefficients of c and d and also tabulated, and their ε thresholds are also available in literature. The Python code for this component is available here.

skyLum_map



This component has four input nodes ('matrix_out', 'region_mask', 'folder', and 'runIt') and one output node ('sky_map').

The output from **CIE_Skygen** or **PerezSky** connects to the input ('matrix_out') along with location of mask file with 145 discrete equal-sized patches ('region_mask'); and this component generates an image of skydome with 145 luminance data in the folder location specified ('folder'). The output ('sky_map') contains the location of this generated image, which can be connected to a Ladybug ImageViewer component for visualizing luminance distribution on the Grasshopper canvas. This component uses an external instance of Python 2.7, which needs to be installed on the system along with 'cpython' component in Grasshopper. The installation instruction for the pre-requisites is provided in an earlier section. The Python code for this component is available here.

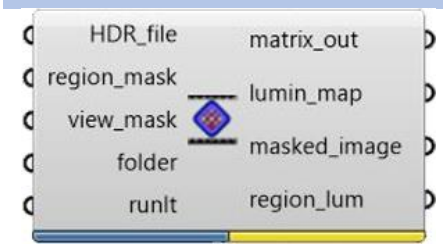
3CspectralSky

This component has thirteen input nodes, corresponding to average-SPD-per channel (from **SPD2spectral**), location (latitude, longitude, and timeZone from **OpenEPW_loc**), time of year (month, day, hour), solar altitude angle (from **SunPos**), and weather file data (direct normal radiation, and horizontal diffuse radiation from **OpenEPW**). Other inputs are user defined, such as folder (location for saving radiance sky file as output), sky_name (the name of output file to be saved), and sky-type (Radiance sky-type: -s | +s | -c | -i | +i | -u).

This takes the input parameters to generate a Radiance sky file (.sky) with the spectral skyglow parameter, which is used for Radiance image-based simulation. With the average SPD per channel data input, this Radiance skyfile serves as key input in Honeybee image-based simulation recipe, which is used for Radiance simulation in Honeybee to generate HDR luminance map.

This luminance map serves as input for the view-based workflow, where the luminance distribution is discretized into equal segments using **HDR2DiscreteLumEx,** and this data, via components of OWL-View cluster, connects to the OWL-spectral components for analyzing the NIF metrics. The Python code for this component is available here.

HDR2DiscreteLumEx

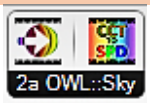


This component has five input nodes ('HDR_file', 'region_mask', 'view_mask', 'folder' and 'runIt') which correspond to location of radiance-generated HDR luminance file ('HDR_file'), location of mask file indicating region – with 145 discrete equal-sized patches ('region_mask'), location of mask file representing view – with either of binocular/human-vision/fisheye boundaries ('view_mask'), location for saving component outputs ('folder'), and a toggleable Boolean for running this component ('runIt'), respectively.

This component has four output nodes, which correspond to an array of 145 region-averaged luminance data at the view -- equivalent to outputs from **CIE_Skygen** or **PerezSky** ('matrix_out'), location of a false-color luminance map for the scene ('lumin_map'), location of luminance map factored with region-mask and

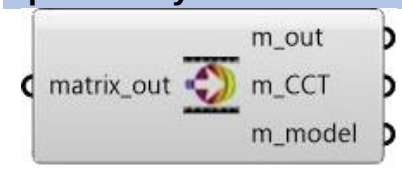
view-mask ('masked_image'), and location of a false-color luminance map with region-averaged luminance values ('region_lum'), respectively.
This component takes the HDR file, evaluates the luminance of each pixel using Radiance *pvalue* function, factors-in the view-mask and discretizes the luminance into equal sized patches. For this component to function, it needs multidimensional (.mat) files, corresponding to the 145 discrete regions, as well as boundaries for binocular vision, human vision and fisheye, respectively – which are provided with this workflow. The image-based outputs may be connected to Ladybug Imageviewer component for visualization on the Grasshopper canvas. This component uses an external instance of Python 2.7, which needs to be installed on the system along with 'cpython' component in Grasshopper. The installation instruction for the pre-requisites is provided in an earlier section. The Python code for this component is available here.

2a. OWL-Sky



The 'OWL-Sky' cluster includes the following components: **SpectralSkydome** and **CCT2SPD**, which are used for workflows specific to outdoor zenith-facing evaluation of luminance and spectral metrics. This corresponds to a situation where the observer is facing the zenith (that is, looking upwards) under an unobstructed sky dome, and is exposed to all parts of the sky dome. For workflows where spectral and luminance evaluations are done for a viewer indoors, these components may be replaced with corresponding components of the 'OWL-View' cluster.

SpectralSkydome



This component has one input ('matrix_out') and three outputs ('m_out', 'm_CCT', and 'm_model'). Each of the input and output nodes of this component correspond to arrays of 145 values, corresponding to sky-discretized distribution of respective data as previously described.

Here, 'matrix_out' and 'm_out' correspond to sky-luminance distribution from **CIE_skygen** or **PerezSky** components, whereas 'm_CCT' and 'm_model' correspond to the calculated CCT of the respective patch evaluated by an appropriate Luminance-to-CCT model. The models in literature for evaluating sky CCT from sky luminance, along with recommendations for the most suitable model for a given range, have been reported recently by Diakite-Kortlever and Knoop. Through numerical analysis of their recommendations, the luminance thresholds are identified for application of the most suitable model across various luminance ranges, when evaluating CCT from luminance for a patch. The Chain1999 model is applicable for patch-luminance less than 3172 cd/m^2, Rusnak model for luminance between 3172 and 5200 cd/m^2, and the Chain2004 model for luminance higher than 5200 cd/m^2 for a sky-patch. For every input data-point representing a luminance value, this component identifies the suitable model following the thresholds above, and evaluates CCT for the respective patch by applying the model. The output nodes 'm_out', 'm_CCT' and 'm_model' correspond to input array of luminance, evaluated array of CCTs, and array of model-index used for the evaluation, respectively. This component uses the following underlying equations to evaluate the CCT of any sky-patch from the region-averaged luminance data as input:
When the luminance is less than 3172 cd/m^2, the Chain1999 model is applied for luminance-to-CCT:

$$CCT = \frac{10^6}{-132.1 + 59.77\ log_{10}L}$$

Here L is the region-averaged luminance of any patch. When luminance is between 3172 and 5200 cd/m^2,the Rusnak model is applied, where

$$CCT = \frac{10^6}{10.2 * L^{0.26}}$$

For luminance higher than 5200 cd/m^2, the Chain2004 model is applied for converting luminance to CCT, as:

$$CCT = \frac{10^6}{181.35233+120\,(-4.22630+log_{10}L)}$$

The Python code for this component is available here.

CCT2SPD

This component has one input ('CCT') and two outputs ('SPD' and 'Wavelength'). For any given input value between 4000 and 25000, corresponding to CCT value in Kelvins, this component emulates the models in the CIE015 standard

Using CIE015, this component generates 107 data-points corresponding to relative SPD for every 5nm wavelength in the range of 300 to 830nm. The 52nd value, corresponding to relative SPD at 560nm is always 100. The CIE015 standard document reports the protocol – which is used in this component – for reconstructing daylight's CCT to SPD, with spectral resolution of 5nm through 300-830nm range. The model restricts the CCT-input between 4000K and 25000K. For a given CCT value, the model evaluates 107 data-points corresponding to relative spectral power distribution [S(λ)] of daylight D, for every 5nm bin through 300-830nm range. The distribution is normalized for the value at 560 nm wavelength, as such the normalized SPD at 560nm is set to the value 100 (exactly) for any reconstructed SPD curve. For 145 patches, this component generates 15515 values, corresponding to 145 arrays of 107 datapoints.
This component uses the following underlying equations to evaluate the output from the input data:

For CCT values between 4000K and 25000K, the CIE015 recommended standard color value (x_D) can be calculated as:

$$x_D = \frac{-4.6070 * 10^9}{CCT^3} + \frac{2.9678 * 10^6}{CCT^2} + \frac{0.00911 * 10^3}{CCT} + 0.244063 \quad , for\ 4000K < CCT < 7000K$$

$$x_D = \frac{-2.0064 * 10^9}{CCT^3} + \frac{1.9018 * 10^6}{CCT^2} + \frac{0.24748 * 10^3}{CCT} + 0.237040 \quad , for\ 7000K < CCT < 25000K$$

From the standard color value (x_D), the CIE daylight locus (y_D) can be calculated as:

$$y_D = (-3.000 * {x_D}^2) + (2.870 * x_D) - 0.275$$

With x_D and y_D values, the factors M_1 and M_2 can be calculated as:

$$M_1 = \frac{-1.3515 - 1.7703x_D + 5.9114y_D}{0.0241 + 0.2562x_D - 0.7341y_D} \quad , and$$

$$M_2 = \frac{0.0300 - 31.4424x_D + 30.0717y_D}{0.0241 + 0.2562x_D - 0.7341y_D}$$

Along with factors M_1 and M_2, and wavelength-specific eigenvectors S_0, S_1 and S_2; the SPD data for wavelength between 300 to 830nm can be reconstructed as:

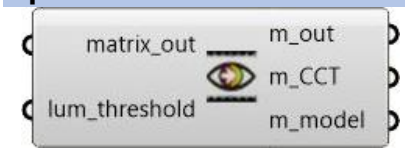
$$S(\lambda) = S_0(\lambda) + M_1 * S_1(\lambda) + M_2 * S_2(\lambda)$$

The tabulated values of wavelength-specific eigenvectors $S_0(\lambda)$, $S_1(\lambda)$ and $S_2(\lambda)$, for 5nm-separated SPD between 300 to 830nm can be referred from the CIE015 standard document. The Python code for this component is available here.

2b. OWL::View

The 'OWL-View' cluster includes the following components: **SpectralViewdome** and **viewCCT2SPD**, which are used for workflows specific to indoor evaluation of luminance and spectral metrics. This corresponds to a situation where an observer is indoors, and the luminance distribution at the observer's view are taken from Radiance simulation outputs. For workflows where an observer is facing the zenith under unobstructed sky, the spectral and luminance evaluations may be done by replacing these components with corresponding components of the 'OWL-Sky' cluster.

SpectralViewdome



This component has two input nodes ('matrix_out' and 'lum_threshold') and three output nodes ('m_out', 'm_CCT', and 'm_model').

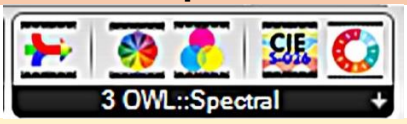
While this performs in similar fashion to **SpectralSkydome**, there are two key modifications: the input-luminance values are evaluated using Radiance image-based simulations and discretized into an array of 145-positions using **HDR2DiscreteLumEx** component; and another input node of 'lum_threshold' is used for setting luminance thresholds that separate direct-sky component from indoor/reflected component of luminance. This is because the models defined in Diakite-Kortlever and Knoop convert sky luminance to CCT, and may not be applicable to luminous exitance from walls and other indoor surfaces. Thus, for luminance values under the threshold (which loosely corresponds to reflected luminous exitance values), the luminance and also the CCT values are muted, and outputs of '0' are generated for both. The underlying models are same as that for **SpectralSkydome**. The Python code for this component is available here.

viewCCT2SPD

This component has one input ('CCT') and two outputs ('SPD' and 'Wavelength').

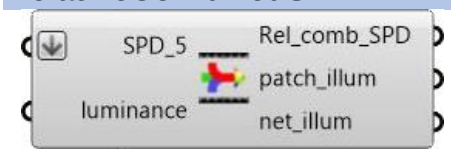
While this performs in similar fashion to **CCT2SPD**, there is a key modification. Since **SpectralViewdome** mutes out luminance (and CCT values) under the luminance threshold, this component generates arrays of zeros for patches with zero CCT value – in addition to using the CIE015 protocol for the remaining patches. For 145 patches, this component also generates 15515 values, corresponding to 145 arrays of 107 datapoints; with zeros filling in arrays with zero CCT input (for patches with internal reflected components). The underlying models are same as that for **CCT2SPD**. The Python code for this component is available here.

3. OWL::Spectral



This cluster includes components specific to spectral sky evaluations, which includes protocols from literature, standards (CIE015 and CIES026) and toolboxes (CIE alpha-opic toolbox, and LRC circadian toolbox); and includes: **RelativeCombinedSPD**, **SPD_graph**, **SPD2spectral**, **CIES026_aopic**, and **circadLight**.

RelativeCombinedSPD

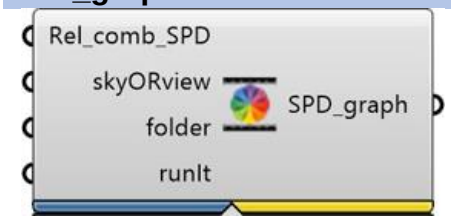


This component has two inputs ('SPD_5' and 'luminance') and three outputs ('Rel_comb_SPD', 'patch_illum' and 'net illum').

The inputs correspond to 5-nm separated arrays of SPD data, and luminance from any equal-sized patch, respectively; whereas the outputs correspond to the evaluated Relative combined SPD from the combination of sources, illuminance contribution of each equal-sized patch on a surface, and net illuminance from all patches in the exposed hemisphere, respectively.

This component is used to merge multiple SPD contributions from various positions (such as the 145 sets of SPD corresponding to the 145 patches, as generated by **CCT2SPD** or **viewCCT2SPD**) into one singular SPD dataset, which may be used for evaluating alpha-opic or circadian metrics, or be used as a starting spectral input for LARK. For this, the approach of LRC's worksheet (Circadian Light Combined Calculator) is replicated, and relative-combined-SPD for the entire sky-dome is calculated by considering each equal-sized sky patch as individual light-source. The average luminance for each patch is calculated by **PerezSky**, **CIE_Skygen**, or **HDR2DiscreteLumEx**, and the illuminance-contribution of each patch is identified by factoring their subtended solid angle. Each of the 145 patches have an average solid angle of 0.0433 steradians (cone with 13.5° apex angle), which is further factored with the sine of their altitude angles: which for patches 1-30, 31-60, 61-84, 85-108, 109-126, 127-138, 139-144 and 145; subtends by 6°, 18°, 30°, 42°, 54°, 66°, 78° and 90° respectively, with respect to the horizon. Patch-luminance factored with 0.0433 steradians, along with sine of individual altitude angles, yields the illuminance contributions for each patch. In the workflows, this component reduces the 15515 datapoints corresponding to 145 sets of 107 values, into one single set of 176 values corresponding to 2-nm separated SPD data over the entire hemisphere, in the 380-730nm range. For more information, this publicly available toolbox can be referred. The Python code for this component is available here.

SPD_graph



This component has four input nodes ('Rel_Comb_SPD2', 'skyORview', 'folder', and 'runIt') and one output node ('SPD_graph'). For the selection between sky or view (where Boolean False represents sky-case and True represents view-case), this component plots SPD data between 380-730nm range with respective wavelength, and saves the plot in the specified folder.

The output contains the location of the saved plot, and needs to be connected to Ladybug Imageviewer component for visualizing the plot on grasshopper canvas. this component uses an external instance of Python 2.7, which needs to be installed on the system along with 'cpython' component in Grasshopper. The installation instruction for the pre-requisites is provided in an earlier section. The Python code for this component is available here.

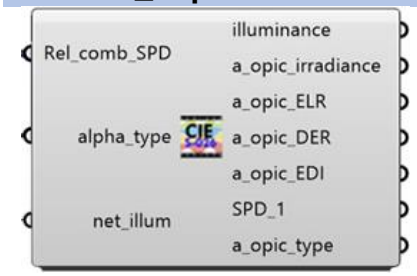
SPD2spectral

This component has one input ('Rel_comb_SPD') and one output ('channel_output').

The input corresponds to 2nm separated SPD data in the 380-730nm range, and channel type (3-channel or 9-channel); whereas the output corresponds to SPD averaged-per-channel. This has a functionality similar to LARK's 'SpectralMaterials' component. Here, the input SPD-data is binned to red, green and blue regions of the wavelength, and binned-averages are normalized with SPD peak-value to generate average-SPD-per-channel. The 586-780nm range is considered the red (R) region, the 498-586nm range is considered the green (G) region, and the 380-498nm range is considered the blue (B) region of the visible spectrum. 10nm-separated SPD data in the 300-830nm range is taken as input, and data in the R-G-B domains is isolated and interpolated to 1-nm interval. Further, data in each of the three domains is averaged, the bin-averages are normalized with SPD peak, and average SPD for each of the R-G-B channels is

evaluated as output. This output is key to generating radiance sky file with appropriate spectral contribution, in **3CspectralSky** component. The Python code for this component is available here.

CIES026_aopic



This component has three inputs ('Rel_comb_SPD', 'alpha_type', and 'net_illum') and seven output nodes ('illuminance', 'a_opic_irradiance', 'a_opic_ELR', 'a_opic_DER', ' a_opic_EDI', 'SPD_1', and 'a_opic_type'). The inputs correspond to 2nm separated SPD between 380-730nm range (from **RelativeCombinedSPD**), type of alpha-opic (from among s-cone-opic, m-cone-opic, l-cone-opic, rhod-opic and melan-opic), and net illuminance on the surface, respectively.

This component evaluates the outputs using the approach in the CIES026 toolbox, and these outputs: net Illuminance, α-opic irradiance, α-opic Efficacy of Luminous Radiation (ELR), α-opic Daylight Efficacy Ratio (DER), and α-opic Equivalent Daylight Illuminance (EDI) represent the alpha-opic metrics (or melanopic metrics, when alpha-type is set at melanopic) recommended by CIE. Other outputs include the interpolated 1nm separated SPD in the 300-830nm range, and the category of a_opic used for simulation.

When alpha-type is set to melan-opic, this component uses the following equations to evaluate the melanopic outputs from the input data (and by setting other α-opic types, the respective α-opic outputs are calculated):

The melan-opic radiant flux (Φ_{mel}) (W) is calculated as:

$$\Phi_{mel} = \int \Phi_{e,\lambda}(\lambda) s_{mel}(\lambda) d\lambda$$

OUT2: The melan-opic irradiance (E_{mel}) (W/m^2) is calculated as:

$$E_{mel} = \int E_{e,\lambda}(\lambda) s_{mel}(\lambda) d\lambda$$

OUT3: The melanopic efficacy of Luminous Radiation (mELR) ($K_{mel,v}$) (W/lm) is calculated as:

$$K_{mel,v} = \frac{\Phi_{mel}}{\Phi_v} = \frac{\int \Phi_{e,\lambda}(\lambda)\ \ s_{mel}\,(\lambda) d\lambda}{K_m \int \Phi_{e,\lambda}(\lambda)\ V\,(\lambda) d\lambda}$$

Here, K_m equals 683.002 lm/W.

OUT4: The melan-opic daylight efficacy ratio (mDER) is defined as:

$$\gamma_{mel,v}^{D65} = \frac{K_{mel,v}}{K_{mel,v}^{D65}}$$

OUT5: The melan-opic equivalent daylight (D65) illuminance (m-EDI) (lux) is defined as:

$$E_{mel,v}^{D65} = \frac{E_{mel}}{K_{mel,v}^{D65}}$$

The denominator value in mDER and mEDI equations is the mELR for D65 radiation, calculated as:

$$K_{mel,v}^{D65} = \frac{\Phi_{mel}^{D65}}{\Phi_v^{D65}} = \frac{\int \Phi_{e,\lambda}^{D65}(\lambda)\ \ s_{mel}\,(\lambda) d\lambda}{K_m \int \Phi_{e,\lambda}^{D65}(\lambda)\ V\,(\lambda) d\lambda}$$

Since D65 has a constant SPD, this value is always 1.3256.

For testing the validity of the mathematical models used in this component, this publicly available toolbox can be referred. The Python code for this component is available here.

circadLight

This component has two input nodes ('Rel_comb_SPD' and 'net_illum') and two output nodes ('CLa' and 'CStim').

The inputs correspond to 2-nm separated SPD data for the entire hemisphere in 380-730nm range, and net illuminance at the surface from the entire exposed hemisphere, respectively; and these nodes take data from the outputs of **RelativeCombinedSPD** component. This component emulates the LRC toolbox (Circadian Light Combined Calculator) to evaluate the outputs of Circadian Light and Circadian Stimulus from the inputs.

OUT1: Circadian Light is evaluated as:

$$CLa = 1547.9\left[\int Mc_\lambda E_\lambda d\lambda + 0.7\left(\left(\int Sc_\lambda E_\lambda d\lambda - 0.2616\int Vc_\lambda E_\lambda d\lambda\right) - 3.3\left(1 - e^{\frac{\int sco_\lambda E_\lambda d\lambda}{6.5215}}\right)\right)\right]$$

Here Mc_λ, Sc_λ, Vc_λ,and sco_λ, represent melanopic sensitivity factor, s-cone sensitivity factor, photopic luminous efficiency function, and scotopic sensitivity factor, respectively. E_λ representing the light source spectral irradiance function is calculated as:

$$E_\lambda = \int rcSPD_\lambda * \frac{illuminance}{683\int P_\lambda\, rcSPD_\lambda\, d\lambda}\, d\lambda$$

Here, rcSPD represents the relative combined SPD data between 380-730nm separated by 2nm interval – an input data, illuminance is another input data, and Pλ represents the photopic sensitivity factor.

OUT2: Circadian Stimulus can be evaluated from Circadian Light as:

$$CS = 0.7 * \left(1 - \frac{1}{1 + \left(\frac{CLa}{355.7}\right)^{1.1026}}\right)$$

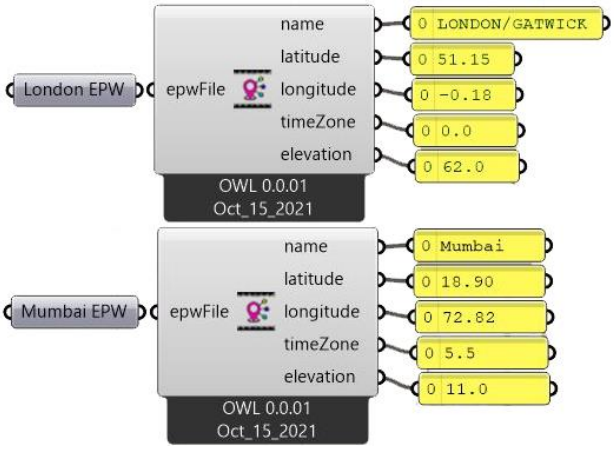
For testing validity of the mathematical models used in this component, this publicly available toolbox can be referred. The Python code for this component is available here.

Describing functionality of the various components in the workflow

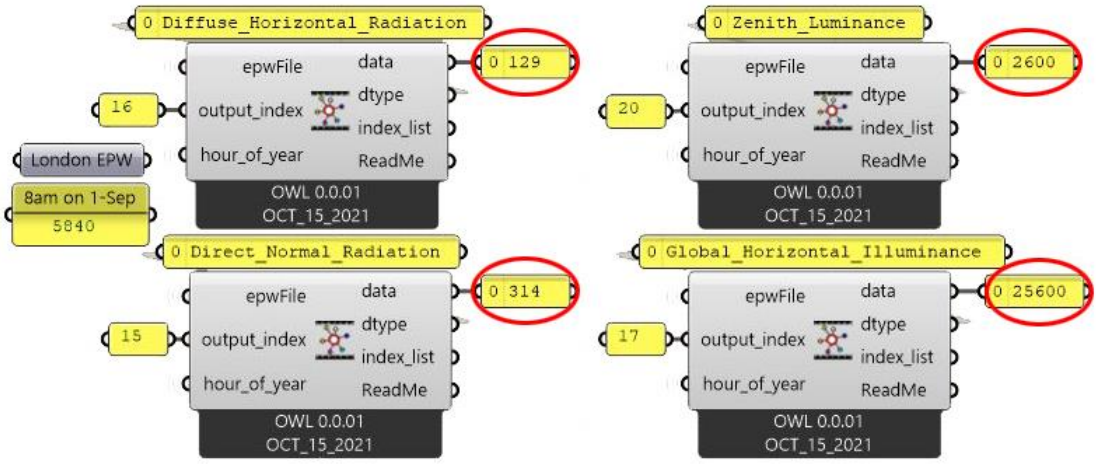
1. Extracting relevant parameters from the weather file for the point in time

The components named *OpenEPW_loc* and *OpenEPW* are designed to extract static headers and temporally dynamic data from the EnergyPlus Weather (EPW) file.

The *OpenEPW_loc* component takes the weather file as input, and extracts location name, latitude, longitude, time-zone, and elevation data, respectively.

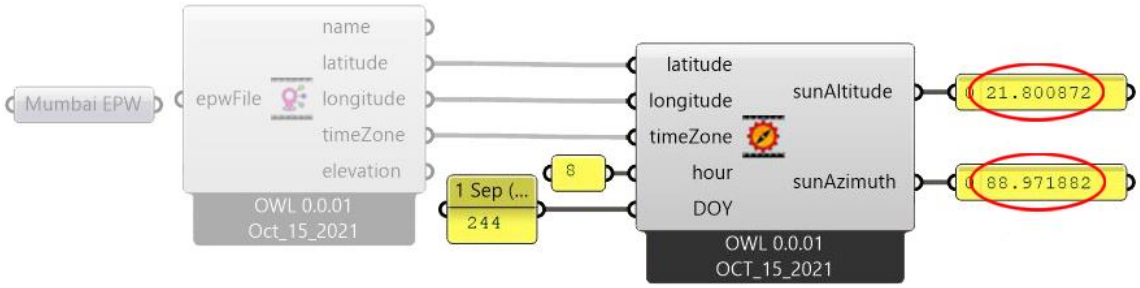


While connected to the weather file, the *OpenEPW* component uses hour-of-year (HOY) and output-index as inputs for extracting corresponding data; where 'output-index' is a numeric value between 1 and 35 corresponding to type of environmental parameter.



2. Solar Altitude and Azimuth at a point in time, for a location

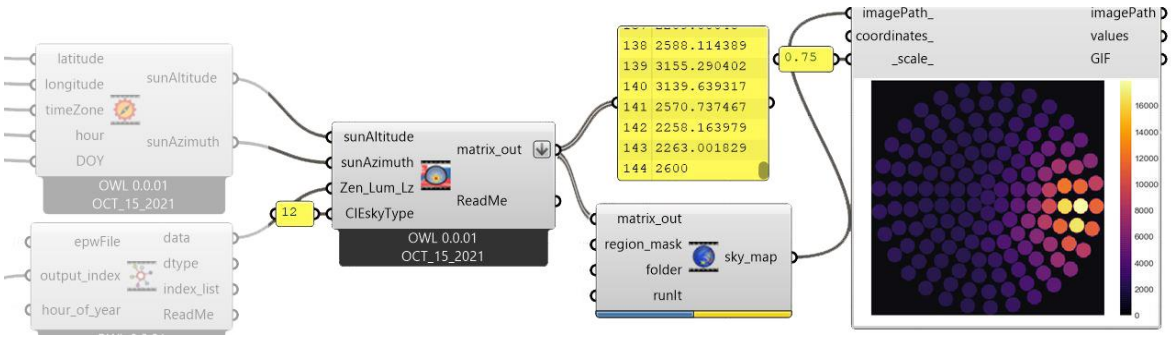
The component *SunPos* uses the location data from *OpenEPW_loc,* along with user-defined Hour-of-Day and Day-of-Year; to generate the solar altitude and solar azimuth at the location at the specified time



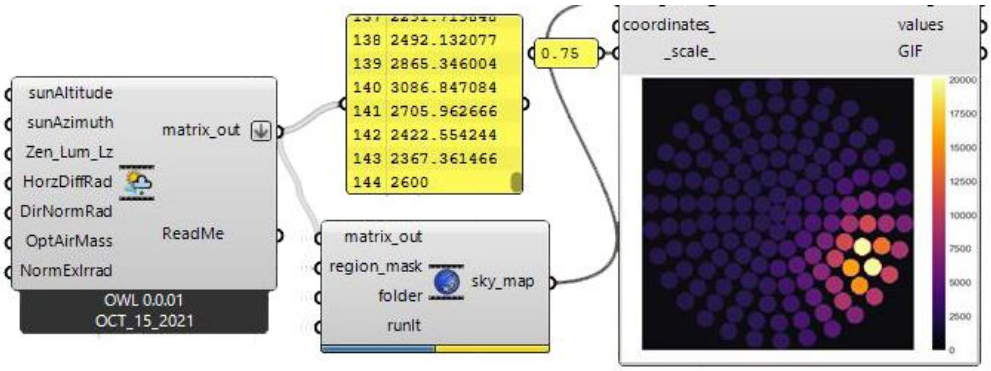
3. Generating sky-luminance for various sky patches

The components *CIE_Skygen* and *PerezSky* use the CIE standard sky model and Perez all-weather model, respectively, for generating sky-luminance data across various altitudinal/azimuthal positions of the sky.

The *CIE_Skygen* module generates 145 luminance values as output, as a function of: Solar Altitude, Solar Azimuth, Zenith Luminance, and CIE Sky-type; the first three data for a given HOY are taken from *SunPos* and *OpenEPW* modules respectively, while the CIE-Sky-type is user-defined for simulating various sky-types. This parameter takes a value between 1 and 15, which corresponds to the 15 CIE sky-types for a range of overcast-, uniform-, intermediate- and clear-sky conditions. The component *skyLum_map* takes the 145 datapoints as input along with a mask of 145 patches in a *.mat format, to generate a luminance map for viewing on the GH canvas.

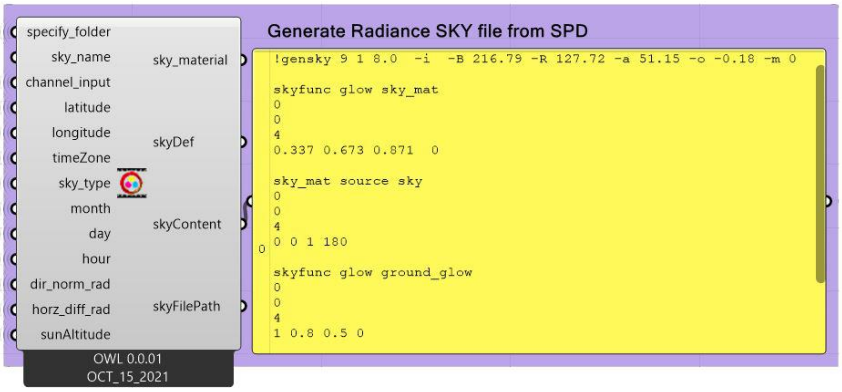


PerezSky also generates 145 luminance datapoints as output, as a function of: Solar Altitude, Solar Azimuth, Zenith Luminance, Horizontal Diffuse Radiation, Direct Normal Radiation, Optical Air Mass, and Extra-terrestrial Direct Normal Radiation. The Solar altitude/azimuth data is taken from the outputs of *SunPos,* while the other environmental parameters for a given HOY are taken from *OpenEPW* by setting 'index' at: 20, 16, 15, and 12; for: Zenith Luminance, Diffuse Horizontal Radiation, Direct Normal Radiation, and Extra-terrestrial Direct Normal Radiation, respectively. The Optical Air Mass parameter is user-defined for parametric simulations at various turbidity values.



4. Generating Radiance skyfile for Image based simulation

The component *3CspectralSky*, channel data (from *SPD2spectral*) and generates a radiance sky-file for Radiance image-based simulations. This component takes added inputs, such as location information (from *OpenEPW_loc*), Radiance sky-type, Time information (Hour/Day/Month), solar position (from *SunPos*), and environmental parameters of Direct Normal Radiation and Diffuse Horizontal Radiation (from *OpenEPW*), to generate a Radiance sky file. The average-SPD per channel data comprises the '*skyfunc glow sky_mat'* parameter of the sky-file, which is one of the key inputs while generating luminance maps through Honeybee's *'Run Daylight Simulation*' and *'Image based simulation*' components.

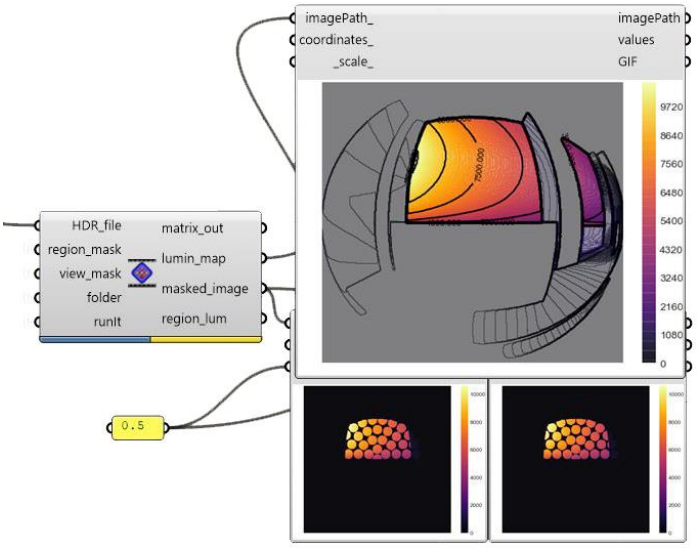


5. Per-pixel reading of the luminance data, and discretizing luminance at various view positions

The luminance maps generated by '*Honeybee Run Daylight Simulation*' component are in high dynamic range image (HDRi) format. Through post-processing, the per-pixel data from HDR files can be clustered into 2-dimensional regions, which can then be averaged to reduce the data resolution and increase computational speed. This way, the luminance detail from a user's view is not compromised, and the CCT of each region can be separately calculated from discretized luminance data, akin to the protocol used for sky-luminance distribution.

HDR2DiscreteLumEx is designed to read HDR luminance maps and extract actionable data. This component discretizes the HDR-luminance map to 145 constituents, representing equal sized regions in an observer's hemispherical view, where the center represents the foveal point. Additional inputs to this component are: a multilayered .mat file of masks for each of the discretized regions; and a single-layer mask file representing field of view: fisheye/binocular/human-vision.

The generated outputs of this component include maps of masked luminance, region-average luminance map, and a matrix of 145 averaged luminance values to be used for the *spectral recipe* presented in the next section.



6. Spectral recipes under sky and at the viewpoint

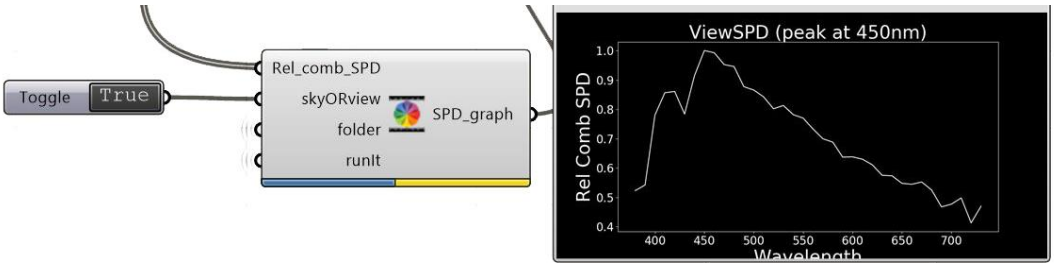
Spectral recipes involve multiple components, and are tailored for evaluating NIF metrics either under the sky, or at the view positions. Sequentially, either recipe includes components for converting Luminance of different patches to respective CCT, converting CCT of each patch further to SPD; cumulating SPD over the hemisphere, and using this cumulative spectral data for evaluating the NIF metrics. Here, hemisphere can refer either to the sky-dome with zenith at the center and horizon at the periphery; or it could refer to fisheye view with foveal position at the center and peripheral at the edges. For evaluating NIF under unobstructed sky, luminance distribution can be evaluated by *CIE_Skygen* or *PerezSky* components, while for the view-analysis, the matrix output from *HDR2DiscreteLumEx* is used.

For sky-cases, *SpectralSkydome* component converts luminance data to respective CCT output for daylight, by emulating luminance-related spectral sky models. For every input data-point representing a luminance value, this component identifies the most-suitable model based on thresholds, evaluates CCT by applying the model; and provides CCT and model-index as output. For view-cases, *SpectralViewdome* performs a similar functionality, with the addition of providing a luminance threshold for separating direct-sky component of daylight from the internally and externally reflected components. This is because luminance-to-CCT models are applicable only for daylight.

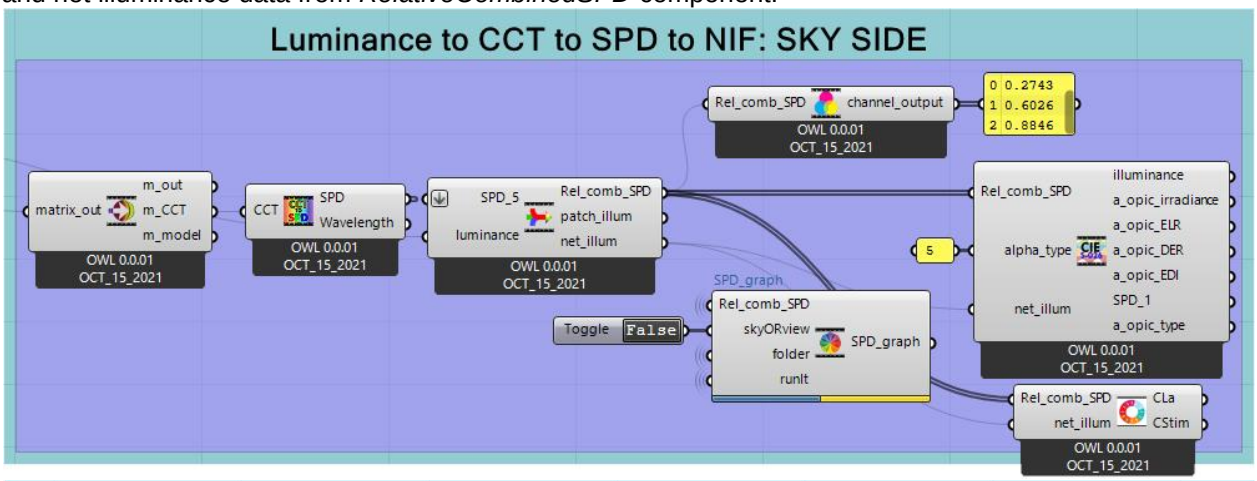
CCT2SPD emulates the models in the CIE015 standard: a protocol for reconstructing daylight's CCT to SPD, with spectral resolution of 5nm through 300-830nm range. The model is applicable for CCT value between 4000K and 25000K; and for a given CCT value, the model evaluates 107 data-points corresponding to relative spectral power distribution [S(λ)] of daylight D, for every 5nm bin through 300-830nm range. The distribution is normalized for the value at 560 nm wavelength, as such the normalized SPD at 560nm is set to the value 100 (exactly) for any reconstructed SPD curve. For view-case, patches with luminance under threshold represent reflected component of daylight, and *SpectralViewdome* generates 0 CCT as data output for such patches. Further, a specifically tailored component *ViewCCT2SPD* nullifies SPD contributions from patches with 0 CCT. The modification in these two components for view-cases *(SpectralViewdome + ViewCCT2SPD)* from those used for sky-cases *(SpectralSkydome + CCT2SPD)* is that only patches representing direct-sky component are considered for luminance-to-CCT-to-SPD generation, while those representing internal or external bounces are neglected.

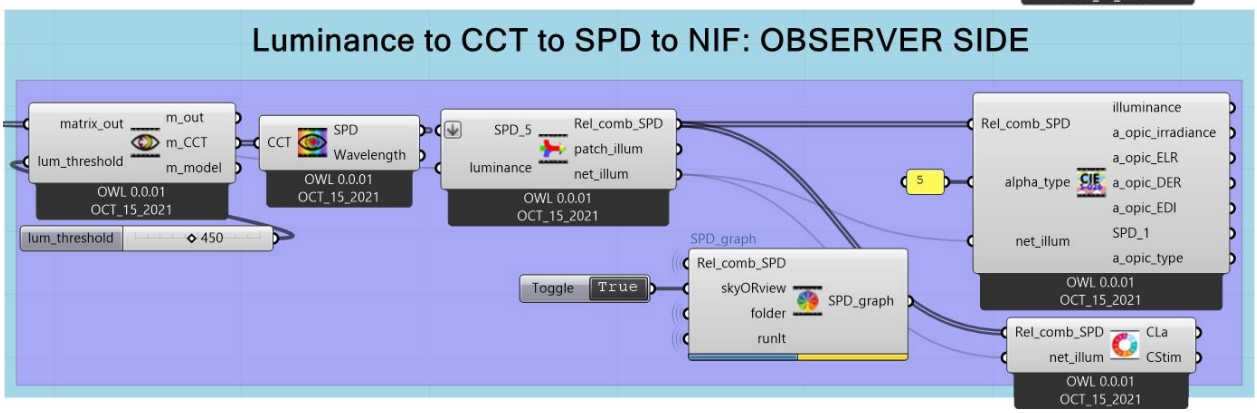
RelativeCombinedSPD cumulates the individual contributions of SPD and illuminance from each hemispherical patch the entire hemisphere, for it to be useful in either NIF evaluating toolbox, or as the spectral input for LARK. This component replicates the approach of LRC's worksheet, and relative-combined-SPD for the entire sky-dome is calculated by considering each equal-sized sky patch as individual light-source. Each of the 145 patches have an average solid angle of 0.0433 steradians (cone with 13.5°

apex angle), which is further factored with the sine of their altitude angles: which for patches 1-30, 31-60, 61-84, 85-108, 109-126, 127-138, 139-144 and 145; subtends by 6°, 18°, 30°, 42°, 54°, 66°, 78° and 90° respectively, with respect to the horizon. *RelativeCombinedSPD* takes 5nm-separated SPD data from *CCT2SPD* (15515 values representing all 145 patches), along with luminance from each of the 145 patches from *PerezSky* or *CIE_Skygen*, as inputs and evaluates the relative-combined-SPD (176 datapoints through 380-730nm range with step-size of 2nm) and net-illuminance as output. *SPD_graph* plots relative combined SPD across the wavelength range, between 380 to 730nm.



CIES026_aopic emulates the CIES026 toolbox for evaluating α-opic metrics. The term 'a-opic' represents either of the following: s-cone-opic, m-cone-opic, l-cone-opic, rhodopic, or melanopic. This component takes incident-SPD and net-illuminance as inputs along with an indicator of the alpha-type, to evaluate the α-opic metrics. By factoring α-opic and photopic sensitivity curves with the input data, the following α-opic metrics are evaluated: net Illuminance, α-opic irradiance, α-opic Efficacy of Luminous Radiation (ELR), α-opic Daylight Efficacy Ratio (DER), and α-opic Equivalent Daylight Illuminance (EDI). The component *circadLight* emulates the protocol in the LRC toolbox (Circadian Light Combined Calculator), and evaluates the outputs of Circadian Lighting and Circadian Stimulus. As input, this component takes in the Relative combined SPD and net illuminance data from *RelativeCombinedSPD* component.





7. Dashboard of components

A user dashboard is created using panels and sliders for input, and with panels and ImageViewer for output, as seen below: