# OpenStudio’s Radiance Functionality – Documentation (v0.9.0)

09/28/2012

The following document details the status and usage of the Radiance integration functionality of OpenStudio. This document has been revised to cover functionality in OpenStudio version 0.9.0.

# Table of Contents

OpenStudio’s Radiance Functionality – Documentation (v0.9.0) 1

Table of Contents 1

Software Overview 1

New for 0.9.0: 2

Radiance Core 3

ModelToRad.rb 3

DaylightSim.rb 5

MakeSchedules.rb 7

OpenStudio Application Integration 8

Radiance Parameters 9

Using the Radiance Features in OpenStudio 9

# Software Overview

The OpenStudio development team has added the ability to conduct daylighting analysis of an OpenStudio model using Radiance, the high quality ray tracing software tool[[1]](#footnote-1). OpenStudio manages the translation of your OpenStudio model to a valid Radiance model, pulling model data such as room surface reflectances, glazing transmittances and geometry directly from the .osm. Point-in-time simulations can be done, as well as annual simulations using either the single phase or three phase daylight coefficient methods (similar to Daysim). Once an annual daylight illuminance profile has been generated, you can generate a lighting schedule for EnergyPlus that incorporates the Radiance daylighting data along with the occupancy schedules and daylighting control points’ setpoints from the .osm. This lighting schedule gets embedded into the .osm so that when you run EnergyPlus for the whole-building energy performance, the daylighting and electric lighting control response is based on Radiance-calculated daylight distributions rather than the internal split flux calculations internal to EnergyPlus. By using the three phase method, the user can also simulate operable blinds or shades at the hourly timestep basis.

## New for 0.9.0:

* While all of the Radiance functionality can still be accessed via the command line interface (CLI), OpenStudio 0.9.0 now features the ability to use Radiance for daylighting analysis in your regular OpenStudio application workflow as well, without ever having to visit the command line (see Figure 1). Checking the “Use Radiance…” box forces OpenStudio to use Radiance in a pre-process step, performing an annual climate-based daylight simulation of all spaces containing an illuminance map and daylighting control point, producing annual lighting schedules based on Radiance computed daylight availability, and embeds them in the OpenStudio model for use in the EnergyPlus whole building energy model. This process is discussed in depth in a later section.
* There is a new glare sensor object that can be placed in spaces to evaluate glare (using the DGPsimplified metric) for single or multiple view vectors about the glare sensor point. Again, the use of this new object is discussed in a later section.
* Daylight illuminance (annual average and timestep) and glare profiles are now written out to SQLite database tables, allowing for rapid review of daylight performance of individual spaces, using the ResultsViewer application.
* Daylight metrics (Daylight Autonomy (DA), Continuous Daylight Autonomy (cDA), and Useful Daylight Illuminance (UDI) are plotted and written out to a comma separated value (.CSV) file, for daylight availability analysis and LEED and code compliance review and documentation.

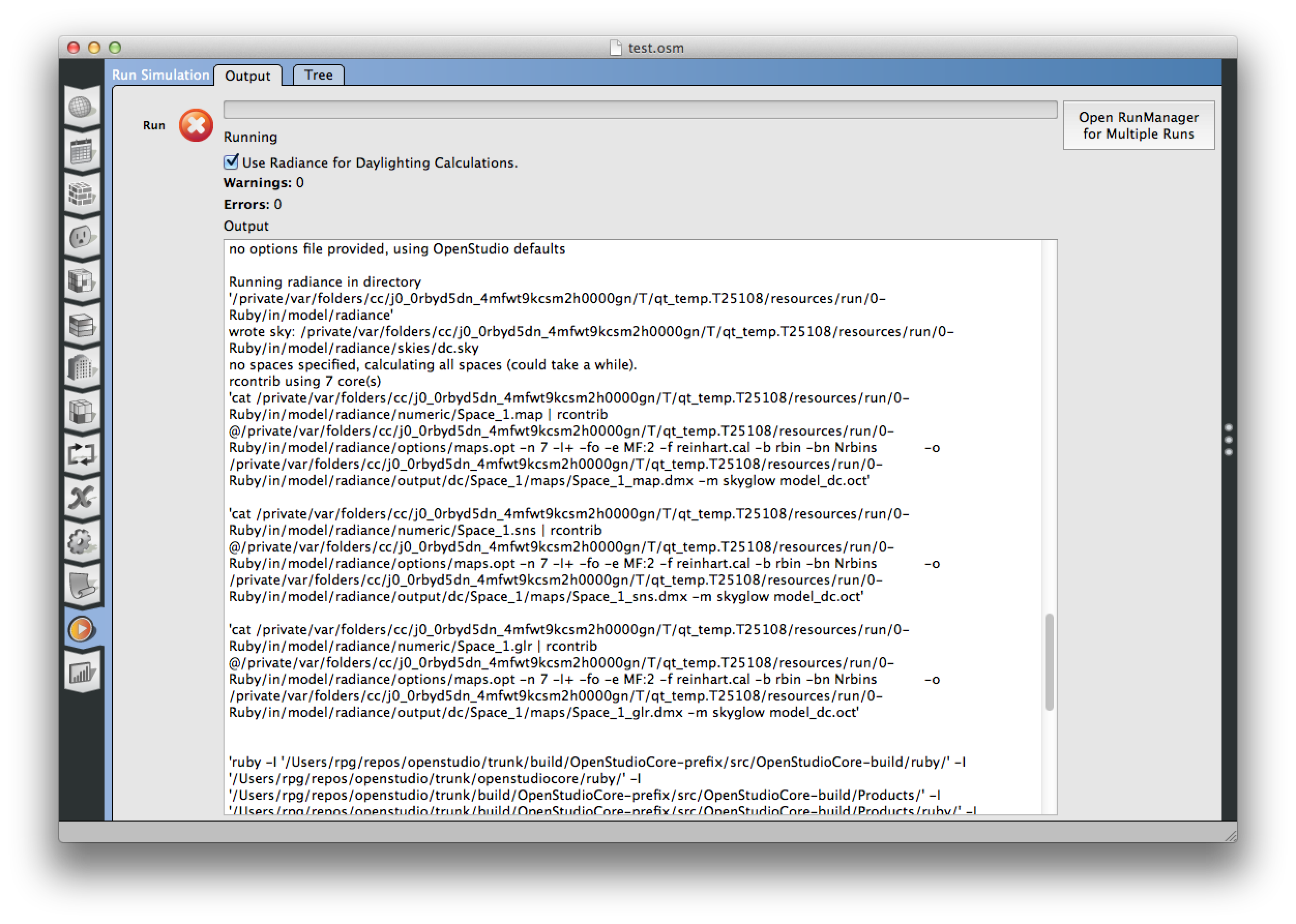


Figure : OpenStudio RunManager using Radiance for Daylighting Simulation

# Radiance Core

The core Radiance functionality is implemented in a series of Ruby scripts; their names and basic descriptions are as follows:

* ModelToRad.rb – Converts an OpenStudio model (.osm) to a Radiance model
* DaylightSim.rb – Performs a variety of operations:
  + Performs “point-in-time” simulations using classic Radiance programs (rpict and rtrace)
  + Computes daylight coefficients for illuminance maps (views partially supported)
  + Performs an annual climate based daylight simulation, optionally using the three phase method with bidirectional scattering distribution functions (BSDFs) to represent windows in a variety of configurations
* MakeSchedules.rb - Reads in an annual daylight illuminance profile (computed by DaylightSim.rb), as well as the daylighting control illuminance setpoint and the space’s occupancy schedule from the .osm, and generates a new lighting schedule based on the Radiance-computed daylight illuminance, and embeds a new lighting schedule in the original .osm. This .osm can then be passed to EnergyPlus for a whole building energy analysis, with Radiance-computed daylight performance, and electric lighting response (load based).

In addition to the previous three core programs, there are a couple of utility scripts that allow one to visualize the daylighting data:

* AnnualPlot.rb – creates an annual temporal plot of average daylight illuminance for a space’s illuminance map
* IlluminanceMapPlot.rb – produces a flood plot of an illuminance map’s daylight illuminance data for a single point in time

## ModelToRad.rb

The operation of ModelToRad.rb is fairly straightforward. Input is an OpenStudio model (.osm) and optionally an EnergyPlus eplusout.sql file. If no eplusout.sql file is given, ModelToRad.rb automatically runs EnergyPlus on the input .osm, to extract model data such as room surface reflectances, glazing transmittances, and weather data. The output is a complete Radiance model and associated support files for performing Radiance analysis of the source .osm.

Command line help is available for ModelToRad.rb, simply by calling the program with the -h or--help option:

$: [path to]ModelToRad.rb -h

Usage: ModelToRad.rb [ARGV[0] (ARGV[1]) [options]]

ARGV[0] - Path to OpenStudio Model

ARGV[1] - Optional path to EnergyPlus SQLite output, if not given will run EnergyPlus.

Optional variables:

-v, --verbose Verbose (debug) mode

Other options:

-h, --help Show this message

When a user runs ModelToRad.rb, an EnergyPlus run is started, to extract the weather information and constructions from the .osm. Simple Radiance materials are created, converting absorbptance to reflectance in the case of opaque materials, and visible light transmittance to visible light transmissivity in the case of glass materials. Clear glazing is supported at this time; if you wish to simulate diffuse materials, you must use the BSDF material and the three-phase method (described below).

The model is output to a “model” subdirectory in the .osm’s companion directory (i.e. the directory in the .osm’s directory with the same name as the model file, which contains various support files). Refer to Figure 2 for an overview of the Radiance model directory structure, which we will discuss now.

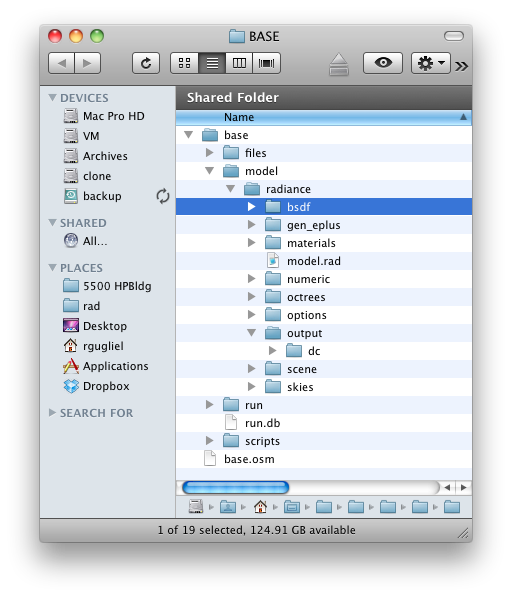


Figure : Radiance Export Directory Structure

As you can see from Figure 2, several directories are created as a result of running ModelToRad.rb on your OpenStudio model. Let’s review those here:

* bsdf – this folder contains BSDF[[2]](#footnote-2) “mapping files”, which tell DaylightSim.rb which BSDF files to use for given timestep conditions. Currently, OpenStudio supports two BSDFs, the idea being that you supply the program with two BSDFs – one to describe the window, and one to describe the window with a shade or blind deployed. For each unique glazing found in the .osm (unique in azimuth and transmissivity), a line will be added by ModelToRad.rb to the mapping.rad file. For example:  
    
  glaz\_3.14\_0.47\_OS\_Space\_1.vmx,glazing.xml,glazing\_blind.xml,glaz\_3.14\_0.47\_OS\_Space\_1.dmx  
    
  The user must edit the mapping.rad file in the BSDF folder, editing the second and third fields to refer to the BSDF files that represent the unobstructed and shaded windows, respectively. The user also needs to place the BSDF files in the bsdf directory so they can be found by the DaylightSim.rb program.
* gen\_eplus – This directory stores the output of the EnergyPlus run from the model translation process
* materials – Stores a variety of material definition files, derived from the OpenStudio model
* numeric – Stores illuminance map and daylighting control point data in a Radiance-ready format
* octrees – Housekeeping directory for storing temporary octrees
* options – Location for automatically- and user-generated options files for Radiance programs
* output – Stores output from the Radiance simulations
* scene – Stores the geometry files for the Radiance model
* skies – Housekeeping directory for storing temporary sky description files

There is also a file called “model.rad” that gets created and saved in the model directory. This is a simple “collector file” that loads all the translated geometry files so that you can view them in rvu.

## DaylightSim.rb

DaylightSim.rb does a number of things, so we will break them down in order.

Command line help is available for DaylightSim.rb, simply by calling the program with the -h or--help option:

$: DaylightSim.rb -h

Usage: DaylightSim.rb [options]

Simulation variables:

--month 06,09,12 Monthly argument(s)

--day 21 Daily argument(s)

--hour 08,12,16 Hourly argument(s)

--view front.vf,plan.vf View argument(s)

--spaces core,perimeter Space argument(s)

--cores 8 Number of cores to allocate

--dims 400 Max image dimension

Simulation type switches:

--dc Generate daylight coefficients

--glare Boolean for requesting fisheye views of spaces for glare evaluation (currently disabled in v0.9.0)

--dcts Use daylight coefficients for timestep analysis

--x Use single-phase DC method

--z Use three-phase DC method

--ts Use Radiance “classic” (point-in-time) method

Other options:

-h, --help Show this message

The three main modes of operation of DaylightSim.rb are:

* Generate daylight coefficients for illuminance maps found in an .osm (--dc)
* Perform annual climate-based daylight simulation using daylight coefficients (--dcts)
* Perform point-in-time simulations using Radiance “classic” and continuous sky models (--ts)

The first two modes are used together, to perform an annual analysis. First, the user runs DaylightSim.rb with the --dc option, to generate the daylight coefficients for the illuminance maps. Next, the user runs DaylightSim.rb with the --dcts option, which uses the daylight coefficients computed in the previous step along with the weather file information, to perform an hourly simulation of the daylight performance for the entire year. When using the daylight coefficient method, the user must also specify whether they are requesting a “single-phase” or “three-phase” method. With the single-phase method, only the daylight matrix is computed. This saves time (and obviates the need for having BSDFs describing each window) but does not allow for changing the window configuration at each timestep. For hourly window configuration changes, one must use the three-phase method, which requires BSDFs and the user must modify the mapping.rad file as explained in the preceding section.

For single-phase, the user adds the --x option, and for three-phase, the user adds the --z option. One of these options must be specified at runtime with the --dc and --dcts options in order for the program to work properly. These options are not necessary if using the --ts option (see below).

The --month, --day, and --space options allow the user to simulate discrete pieces of the model or the calendar year. In order to use MakeSchedules.rb, one needs to compute a full 8760 hour (annual) schedule, however.

The --ts option uses “Radiance classic” (i.e., rtrace and/or rpict) to simulate discrete timesteps with a continuous sky model. The day and month options work the same way as they do with the daylight coefficient options. A flood plot of the illuminance maps for each timestep is automatically generated.

The output from a --dcts simulation is an annual daylight illuminance profile, stored in the radiance directory in output/ts/maps/[space\_name]/[space\_name].ill, which is passed as input to the MakeSchedules.rb program, covered in the next section.

**New for v0.9.0:** In addition to the .ill file, the Radiance daylighting results are also saved to a mySQL database (radout.sql) and the daylight metrics are saved to a comma delineated (.csv) file.

DaylightSim.rb interrogates your system for available cores, and by default uses the maximum cores your system has – minus one, to maintain system responsiveness. The user can specify a specific number of cores to use with the -c or--cores option, if using the scripts directly.

## MakeSchedules.rb

MakeSchedules.rb takes an annual daylight illuminance profile (.ill), along with an OpenStudio model (.osm) and the EnergyPlus output (eplusout.sql) and creates a lighting schedule based on: a) the hourly daylight illuminance at the lighting control point, b) the illuminance setpoint for the daylighting control point, and c) the occupancy schedule for the timestep. Currently a simple continuous dimming (100%-0%) mode is assumed, but future support for more realistic dimming capabilities and algorithms is planned.

The program evaluates, at each timestep, the space occupancy, daylight illuminance, and dimming setpoint at the sensor and adjusts the lighting power fraction accordingly. Finally, the program embeds a new lighting schedule in the .osm, and removes the existing schedule. Users can then run the .osm with OpenStudio for a whole building energy analysis, incorporating Radiance-informed daylighting performance.

Users can elect to keep the original .osm input file (and its original lighting schedule) with the -k or --keep option. Users can also base the lighting setpoint on the average workplane illuminance rather than the point illuminance at the daylighting control point with the -a or --average option.

Support for more realistic Photosensors (placement and spatial sensitivity) is planned for the v1.0.0 release, at the end of 2012.

Command line help is available for MakeSchedules.rb, simply by calling the program with the -h or--help option:

$: [path to]MakeSchedules.rb -h

Usage: ruby MakeSchedules.rb 'C:\path\to\model.osm' 'C:\path\to\eplusout.sql’

Options:

-k, --keep Keep original osm file and schedule

-a, --average use workplane average illuminance for setpoint input

-v, --verbose Verbose mode

-h, --help Show this message

# OpenStudio Application Integration

As mentioned in the introduction, OpenStudio can now use Radiance for the daylighting simulation of your model and integrate the lighting energy savings directly into the OpenStudio workflow, so that they are considered in the whole building energy model (EnergyPlus). This is done, quite simply, by checking the “Use Radiance for Daylighting Calculations” option in the run manager tab in the OpenStudio application, as shown in Figure 3.

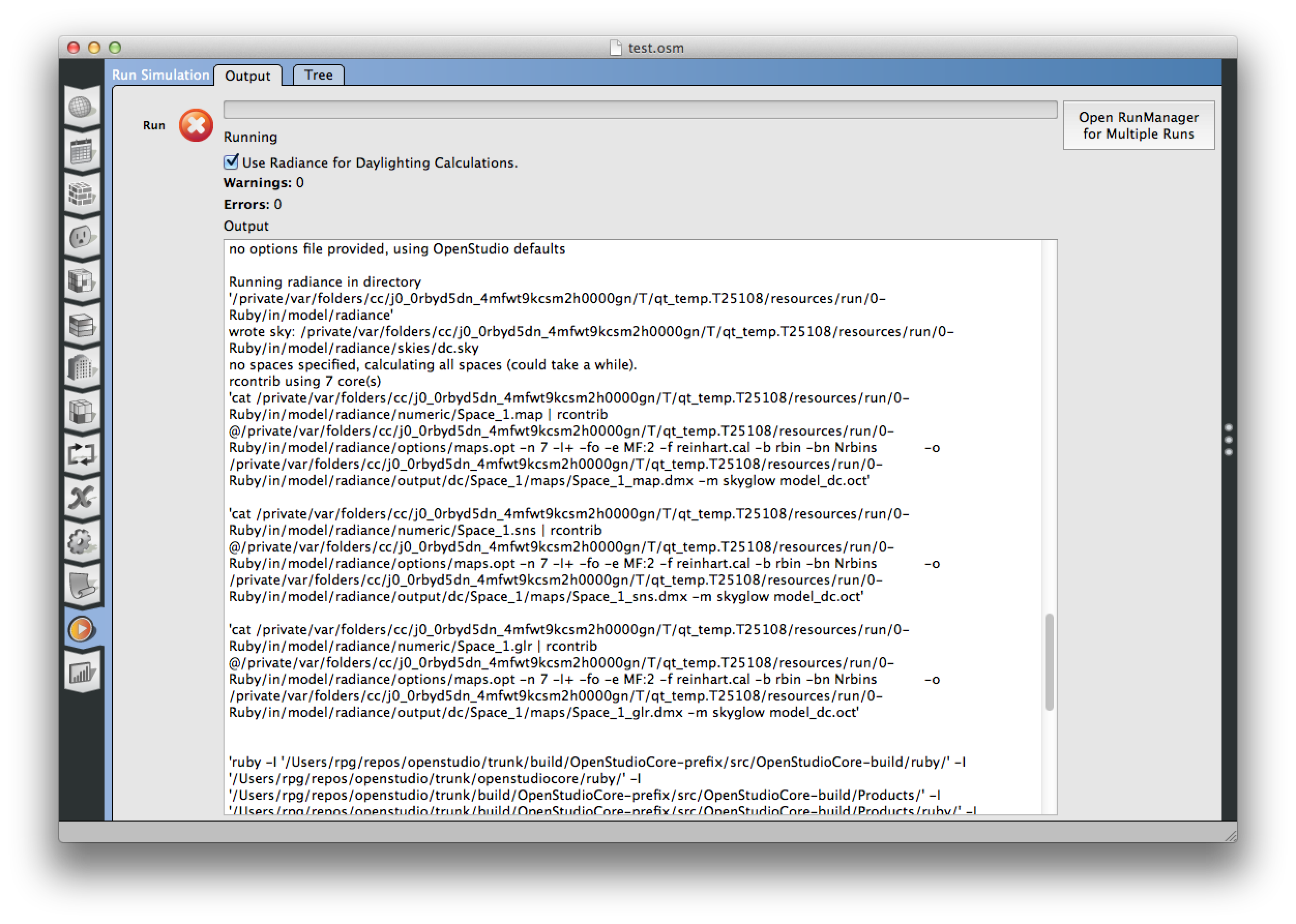


Figure : The "Use Radiance" Option

Checking this box tells the OpenStudio application to use many of the scripts that have already been described (namely, ModelToRad.rb, DaylightSim.rb --dc, DaylightSim.rb --dcts, and MakeSchedules.rb) to perform a “round-trip” Radiance daylight simulation of your OpenStudio model, in advance of the EnergyPlus building simulation. Specifically, this entails the following actions:

* Convert currently open OpenStudio model (.osm) to Radiance format
* Compute daylight coefficients for all illuminance maps found in model
* Perform an annual climate-based daylight simulation for all illuminance maps found in model, using weather information from the model-attached weather (.epw) file
* Generate lighting schedules (schedule:compact EnergyPlus objects) for all spaces, based on occupancy, daylighting setpoint, and daylight availability (as calculated by Radiance)
* Remove daylighting controls from the EnergyPlus model (.idf), and embed the Radiance-informed lighting schedules into the OpenStudio model

Following that process, OpenStudio passes the (updated) model to EnergyPlus, where a whole building energy performance simulation is performed, using the lighting schedule based on Radiance-computed performance.

In addition to the building energy performance output (contained in the eplusout.sql file), there are two additional daylighting-related outputs, radout.sql and metrics.csv:

* radout.sql – contains Radiance-based daylighting performance information, stored in a series of data tables that can be reviewed and plotted in OpenStudio ResultsViewer.
* metrics.csv – a data file of daylight metrics data for the model, based on the annual simulation.

## Radiance Parameters

Radiance is a very flexible and open software tool that allows for tremendous variation in the requested rigor – and resulting accuracy – of the calculations it performs. The OpenStudio development team has made a series of default choices for the settings of the various Radiance programs, reflected in the DaylightSim.rb script and the “maps.opt” file that is automatically generated by ModelToRad.rb. The user has complete freedom to modify the settings if they wish, using the scripts at the command line. If using the “Use Radiance…” checkbox (or the scripts’ defaults) however, here are the relevant default Radiance parameters:

* rcontrib: -ab 10 -ad 4000 -as 50 -dt 0 -dc 1 -ds 0.05 -lw 0.0001
* Daylight coefficient sky subdivision scheme: -M:2 (Reinhart; 581 patches including ground)

***These defaults were chosen as a balance between accuracy and speed of simulation execution; it cannot be overstated that every building model is different, every design investigation has different sensitivities, and these affect the relationship between simulation parameters and the quality of the results.***

The user has the option to change the Radiance parameters when using the command line scripts directly, and the OpenStudio team plans to add the ability to change these parameters from within the OpenStudio application GUI in a future release.

## Using the Radiance Features in OpenStudio

There are several steps to setting up an OpenStudio model for Radiance analysis. For each space you wish to analyze, you need:

* Illuminance Map (analysis grid)
* Daylight Sensor Point
* Glare Analysis Point (optional)

Figure 4 points out the necessary objects on the main OpenStudio toolbar.



Daylight Sensor

Illuminance Map

Glare Sensor

Figure : Daylighting Toolbar Objects

Figure 5 Shows an example model with the required objects for radiance daylighting analysis placed.

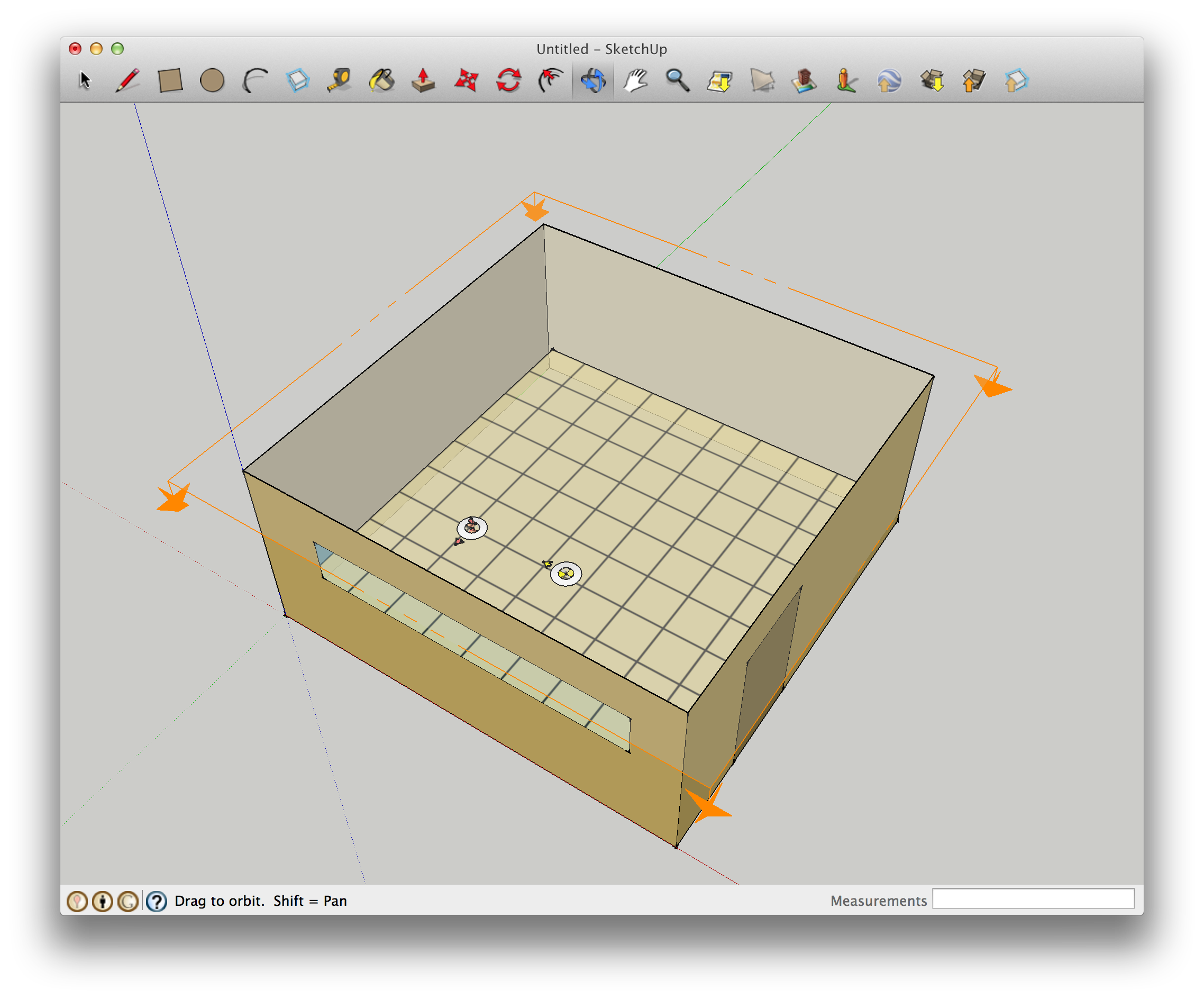


Figure : Example Model

You use the SketchUp plugin to place these objects; generally, you want to place an illuminance map object that covers the entire space square footage, leaving a small “wall offset” – generally 12-14” – from the edge of the map and the space ‘s walls. Place a daylight sensor on the workplane in a location that best matches the critical point – the point that best correlates to the lighting control’s “view” of the daylighting for the space. Lastly, place a glare sensor object in a location for glare evaluation. You should aim the glare sensor’s primary view vector (shown as an arrow on the sensor object) toward the primary view. In the classroom example above, the glare sensor is aimed to the west, where the whiteboard is. You can add additional view vectors to your glare sensor, which allows for rotational freedom to be factored into the glare calculation[[3]](#footnote-3).

Make sure your space is associated with a thermal zone, and that that thermal zone is associated with that space’s daylighting control point and illuminance map. This is also done in the inspector, in the plug-in.

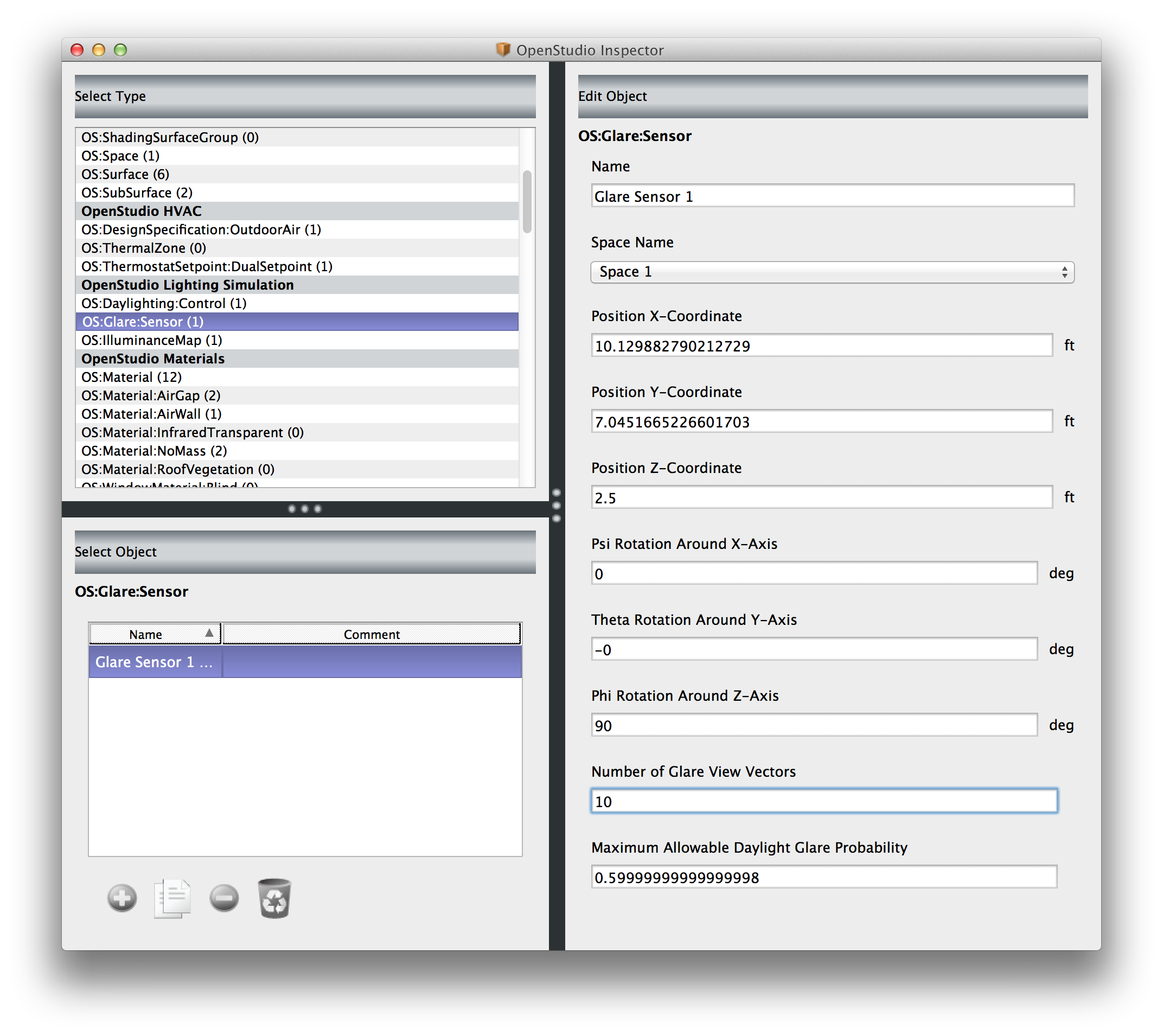


Figure : Glare Sensor Options in the Inspector

Once all of the objects have been placed and “wired up” to their respective spaces and zones, simply check the “Use radiance…” box in the run tab on the OpenStudio application, and click “run”. The OpenStudio Application will handle the entire simulation workflow from there. Radiance will be used to predict the annual daylight illuminance, and lighting schedules will be adjusted accordingly prior to the EnergyPlus run. The final result will be an eplusout.sql data file with Radiance-informed electric lighting usage, and a radout.sql file that contains several daylighting performance datasets:

* Daylight Sensor Illuminance
* Exterior Direct Normal Illuminance
* Exterior Global Horisontal Illuminance
* Illuminance Map Illuminance
* Daylight Glare Probability (Min, Max, and Mean)
* Mean Illuminance Map Illuminance

Using ResultsViewer, one can easily visualize this data as flood or line plots; Figure 7 shows the annual average illuminance map for a space, giving the user a simple means to evaluate the daylight availability in the space for the entire year. Clicking on the results tab in the App and then clicking the “Open Results Viewer…” button will load both the EnergyPlus data as well as the Radiance data.

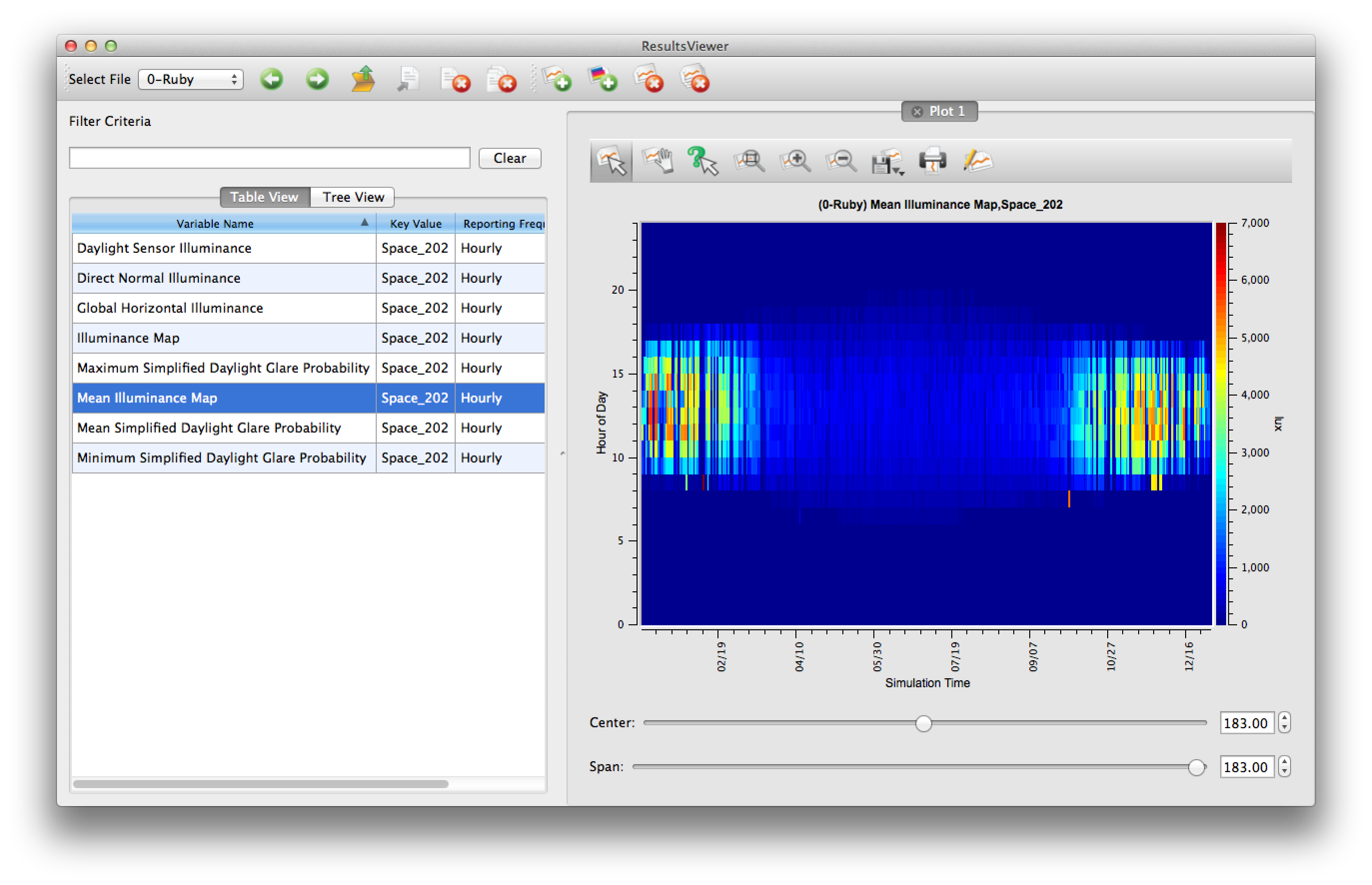


Figure : Radiance Results in Results Viewer

Lastly, several daylight metrics are calculated from the radiance-calculated daylight illuminance. These are saved to a comma-separated value file (.CSV) in your model’s run/0-Ruby directory, called “DaylightingMetrics.csv”. The output is shown in Figure 8.

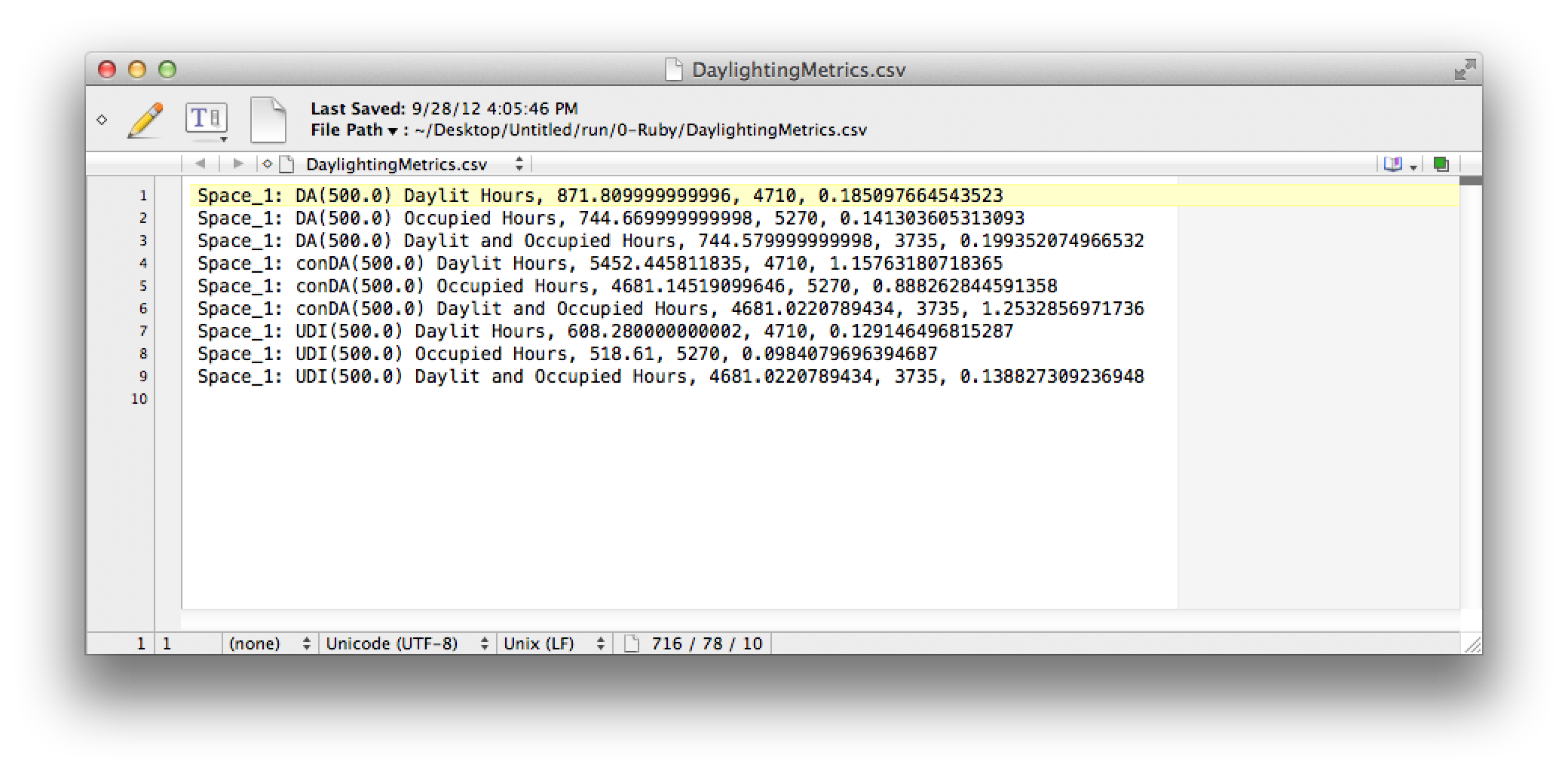


Figure : Daylight Metrics Report

OpenStudio reports the daylight metrics for all daylit hours, occupied hours, and occupied+daylit hours. There is currently no consensus on which method to use in practice; the OpenStudio development team believes they all have merit, so we report all three, and leave it up to the user to decide which one(s) to use.

1. www.radiance-online.org [↑](#footnote-ref-1)
2. Bidirectional Scattering Distribution Function, used to characterize complex fenestration devices [↑](#footnote-ref-2)
3. See Jacubiec’s “Adaptive Zone” Paper for details http://lrt.sagepub.com/content/44/2/149 [↑](#footnote-ref-3)