
Tools 4 SketchUp Documentation

Release 0.0.1

Kevin Hartwell, Thomas Leduc

May 09, 2017

Contents

1	Introduction	1
1.1	What is t4su ?	1
1.2	Software	1
1.3	Related Scientific Publishings	1
1.4	Citing T4SU	2
2	Installation Procedures	3
2.1	Under Windows	3
2.2	Under Linux	4
2.3	Installing T4SU through the Ruby console	4
3	Loading your data into SketchUp	7
3.1	Importing CSV Files	8
3.2	Importing Solene Files	12
4	Preparing Your Data	15
4.1	Extruding Heights from your Geometries	15
4.2	Checking the Normal Vector of your Geometries	17
4.3	Recreating a Terrain with Contour Lines	18
4.4	Projecting Along the Z Axis	21
5	General Tips to Start	23
5.1	Selecting Geometries	23
5.2	Batch Processing	26
5.3	Running Times and Geometrical Precision	27
5.4	Using Buffers as Visual Information	33
5.5	Basic Example of Data Visualization	35
6	Sampling Methods	37
6.1	Sampling the Bounding box	37
6.2	Creating Point and Face Samples of your Buildings	38
6.3	Sampling A Path	40
6.4	Triangulating Geometries with Gmsh	41
7	Drawing Sun Paths	45
7.1	Drawing Paths of Points	45
7.2	Geode Sun Paths	48
7.3	Specific Sun Paths	50
8	Ray Casting and Isovists	53
8.1	The Science Behind : Isovists	53
8.2	Ray Casting : an Introduction to Visibility Studies	54

8.3	What is an isovist ?	55
8.4	Creating Partial 2D Isovists	58
8.5	3D Isovists and Hedgehogs	60
9	Sky Maps	63
9.1	3D Sky Maps	63
9.2	3D Sky Maps and Building Facades	65
9.3	2D Sky Maps and Vegetation Modelization	67
9.4	Sunny Sky Maps	69
10	View Factors	73
10.1	The Science Behind : Sky View Factors	73
10.2	Calculating the Sky View Factor	74
10.3	The Science Behind : Solar Radiation	77
10.4	Sun View Factors	78
10.5	Viewing Direct Solar Irradiation along a Path	80
11	T4SU Ruby API	85
11.1	Readers:	85
11.2	Sampling:	85
11.3	Listing Samples :	85
11.4	ProjectAllong Z Axis :	86
11.5	Make Layer Invisible :	86
11.6	Remove Layer :	86
11.7	Load Trees :	86
11.8	Create a Partial 3D Isovist :	86
11.9	Iterating over a Point :	86
11.10	Create a line at a point, pointing downwards:	87
11.11	Create a 3D isovist:	87
11.12	Partial 3D Isovist	87

Welcome to t4su's documentation manual.

What is t4su ?

T4SU (Tools for SketchUp) is a SketchUp plugin, implemented in Ruby, that gives you the capability:

- To couple SketchUp (the well-known CAAD tool) with the SOLENE simulation software developed in the CERMA laboratory.
- To map out visual-based morphological indicators permitting the analysis of the urban fabric.

Software

The latest version of the plugin is hosted at Renater, France's National Research and Education Network.

Related Scientific Publishings

Developped by Thomas Leduc, (c)UMR 1563 AAU / CRENAU (CNRS/MCC/ECN), 2014-2016

T4SU has already been used in the following publications :

- *Signorelli, V., & Leduc, T. (2016). De l'influence de la végétation et du relief dans la simulation de la ville à l'échelle du quartier. In J.-P. Goulette & B. Ferries (Eds.), SCAN'16 (pp. 125–134). Toulouse, France: Presses Universitaires de Nancy.
- *Signorelli, V., Leduc, T., & Chauvat, G. (2016). How far is far enough? Towards an adaptive and "site-centric" modelling integrating co-visibility constraints for optimal land use. In J.-S. Bailly, D. Griffith, & D. Josselin (Eds.), 12th Spatial Accuracy Assessment in Natural Resources and Environmental Sciences (pp. 233–240). Montpellier, France.
- *Signorelli, V., Leduc, T., & Tourre, V. (2016). Ego-city - Automatic textual description of urban ambiances' factors. In 3rd International Congress on Ambiances - Ambiances, tomorrow. Volos, Greece.

T4SU was also presented during the Journées Solene 2016. You can find more information on <https://solene2016.sciencesconf.org/data/pages/ResumesSeminaireSolene2016.pdf>

Citing T4SU

If you use T4SU in the context of your research, please cite the following paper :

*K. Hartwell and T. Leduc. T4SU : analyses et représentations des vues du ciel , du soleil et des saillances paysagères dans le contexte d'un outil de CAO. In S. Bimonte, T. Devogele, and A. Hassan, editors, Atelier session démonstration – Conférence Spatial Analysis and Geomatics – SAGEO 2016, page 7, Nice, France, 2016.

Installation Procedures

There are multiple ways to install T4SU. Choose the procedure depending on your OS. You can also directly upload it from inside SketchUp.

Under Windows

The following installation procedure assumes that you already have installed a recent release of the [SketchUp software](#) (Make or Pro). You simply have to:

- Download the latest Ruby script file (a file with the .rbs extension) available on sourcesup.renater.fr.
- Install the plugin by placing this Ruby script file into the appropriate folder (the Windows default location is: C:\Users\YOUR USERNAME\AppData\Roaming\SketchUp\SketchUp 2016\Tools),
- Restart SketchUp.

Once you have restarted SketchUp, you should see that various new commands have been added to the menus bar.

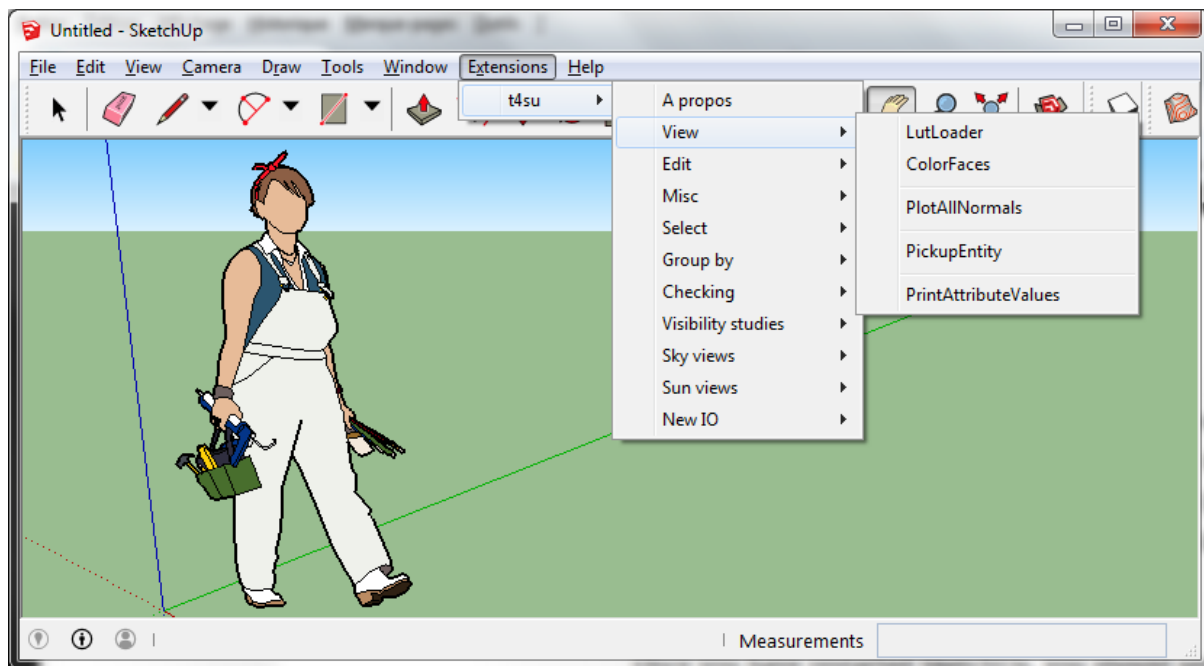


Fig. 2.1: Various new commands have been added to the menus bar

Under Linux

Sketchup doesn't propose a native Linux distribution. Nonetheless, many people use [wine](#). There are many tutorials already online explaining how to install sketchup in an Ubuntu-based environment, so we're not going to go through that again. Nonetheless, if you are having problems installing, here is a nifty little [github project called SkipI](#) that can make your life a little easier. Many Sketchup installation configurations under Linux require disabling the Ruby API, which we need. This method will ensure you can keep the API enabled by automatically installing a stable version of Sketchup8. SkipI will also configure a desktop shortcut for you.

First remove any earlier attempts at installing Sketchup. Then open a new terminal and hit :

```
wget https://raw.githubusercontent.com/wilfm/SkipI/master/skipi.sh
chmod +x skipi.sh
./skipi.sh
```

This will install Sketchup8, which isn't the newest version, but it is at least stable on Linux. Once in Sketchup, you may realize that .rbs folders don't work in older Sketchup versions : you'll need a .rbz file. Making a .rbz is fairly straightforward, as it is actually just a .zip file. So send your T4su extension to a compressed file, then simply change the end of it's name from .zip to .rbz. You're now ready to go to **Window > Preferences > Install Extensions...** and select your t4su_version.rbz file.

Next, download the plugin from [renater](#). Once in Sketchup, you may realize that .rbs folders, which work on Windows, don't work in older Sketchup versions : you'll need a .rbz file.

Making a .rbz is fairly straightforward, as it is actually just a .zip file. So send your T4su extension to a compressed file, then simply change the end of it's name from .zip to .rbz.

You're now ready to go to Window > Preferences > Install Extensions... and select your t4su_version.rbz file.

Installing T4SU through the Ruby console

The following installation procedure assumes that you already have installed a recent release of the SketchUp software (Make or Pro). You simply have to:

- Download the Ruby script file (a file with the .rbs extension) available on sourcesup.renater.fr;
- Copy it in the folder you want (let's assume we choose: D:myScripts);
- Open ruby's Console via the **Window > Ruby Console** menu entry;
- Install this new script file copying-pasting the following Ruby instruction into the command line interface (the console):

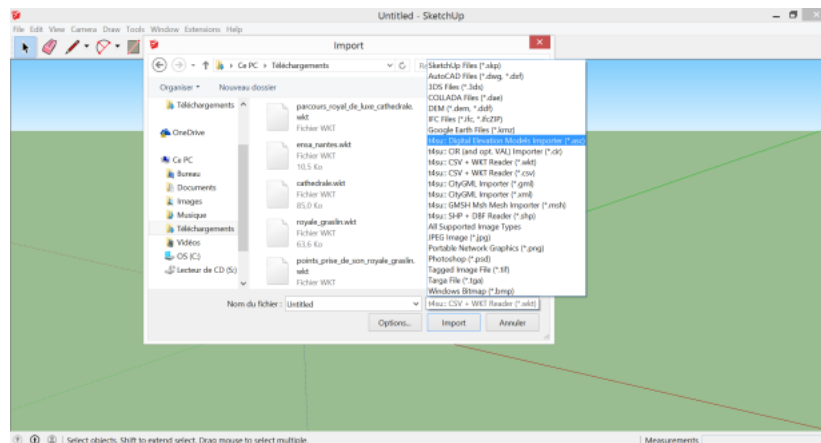
```
Sketchup::require('D:/myScripts/t4su-20161017.rbs')
```

You should not have to restart SketchUp to see that various new commands have been added to the menus bar.

CHAPTER 3

Loading your data into SketchUp

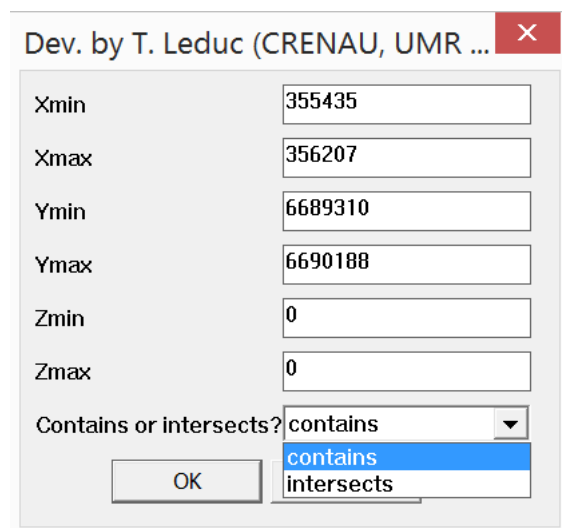
Once you have installed t4su, click on **File>Import....** As you click on the available file types, you can notice that several new types of imports are available :



You can import geometries :

- ASCII Grid file types (*.asc)
- Solene output files (*.cir)
- WellKnownText (*.wkt) / Comma-Separated Values (*.csv)
- Geography Markup Language (*.gml) / Exentible Markup Language (*.xml)
- Mesh Files produced by Gmsh (*.msh)
- Shapefiles (*.shp)

Once you've selected your file and clicked **import**, a first window will inform you of the number of geometries found. A second window will then appear: this is the Bounding Box.



The X, Y and Z values correspond to the minimum and maximum coordinates of the geometry. By leaving the default values, the geometry is re-positioned by taking the lower-left corner of the imported geometry and setting it at the lower-left corner of the Sketchup local coordinate system.

Importing CSV Files

“Comma-Separated Value” files store tabular data in the form of a plain text file. These files are very common in GIS, partly because they can be combined with the Well-Known Text format, which translate geometries into text. Here’s an example of a CSV file opened in Qgis :

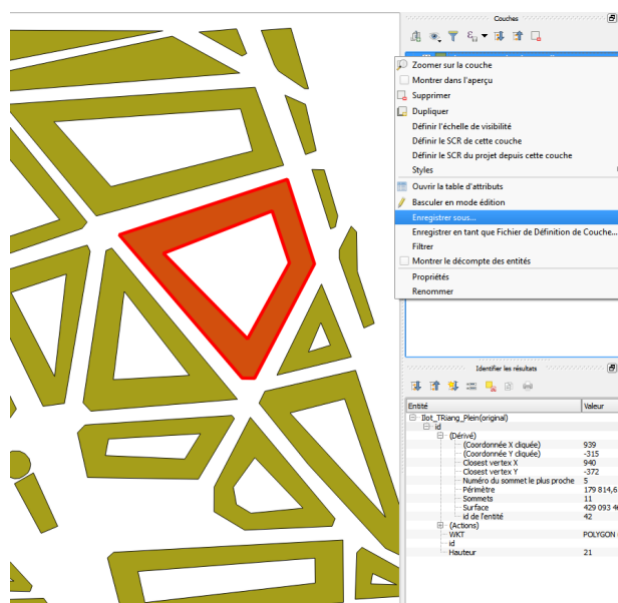


Fig. 3.1: Csv File opened in QGIS.

To import csv/wkt files :

Open the file with a traditional GIS software. If need be, refine your selected area, then save your selection as csv.

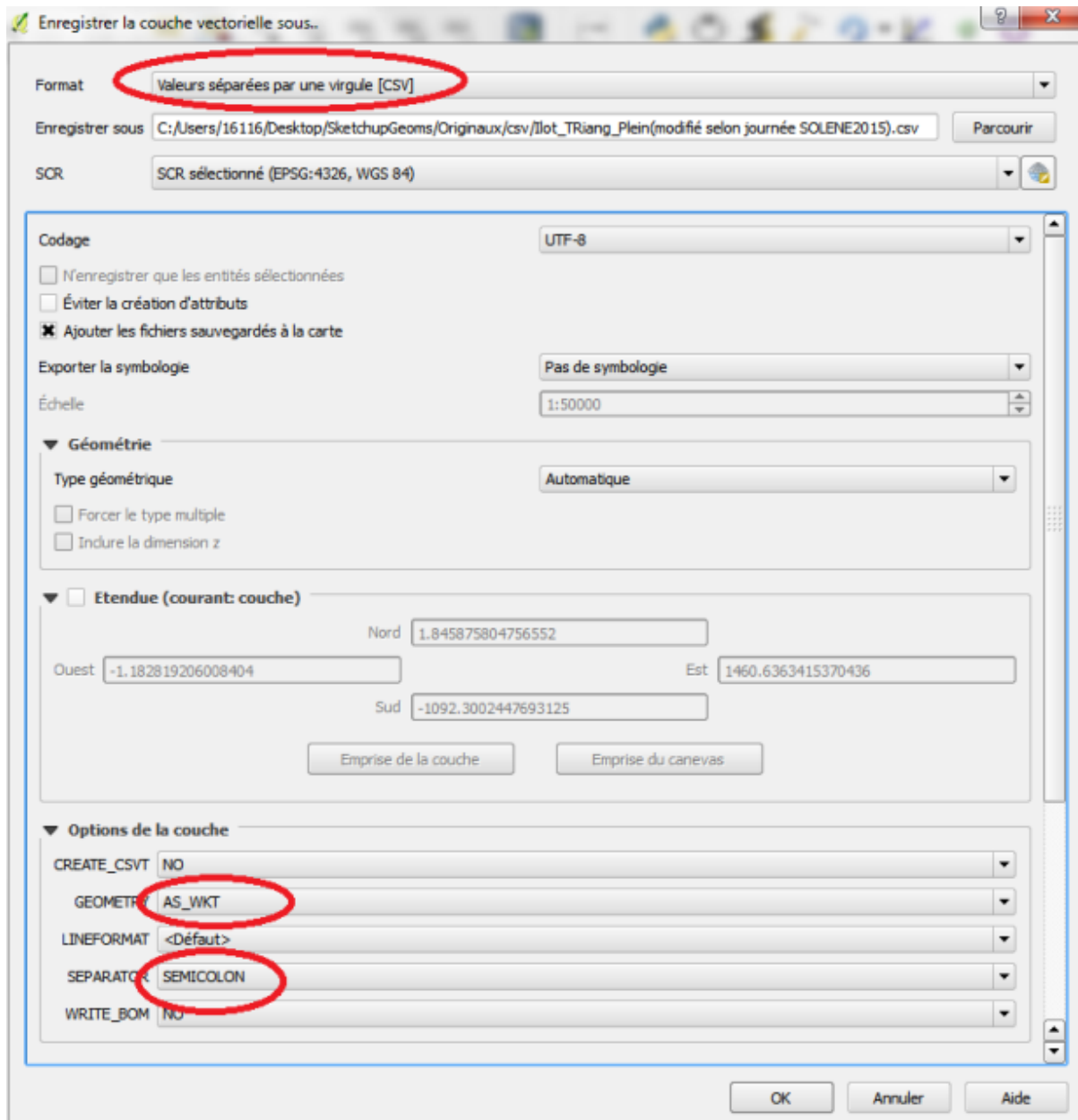


Fig. 3.2: Saving a CSV file.

Chose the following Save Options :

CREATE_CSVT: NO

GEOMETRY: AS_WKT

LINEFORMAT: Default

SEPARATOR: SEMICOLON

WRITE_BOM: NO

Open your new CSV file in a text editor. In a regular csv file, your first few lines should resemble something like this :

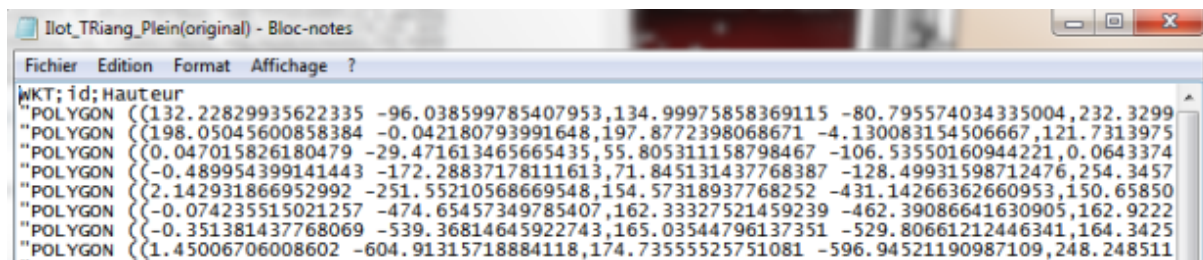


Fig. 3.3: Original QGIS export file

SketchUp needs to know the type of each attribute to properly read csv files, so replace the first line from this :

WKT;id;elevation

to this :

the_geom:Polygon; id:int; elevation:float

Of course, Polygon should be replaced accordingly by Linestring or Point. You may also have other attributes other than these basic ones, including boolean and string types.

Delete the “” at the start and end of each line. The fastest way to do this is by clicking Edit > Replace, entering ” in the find line and leaving the replace by line empty, and clicking Replace All.

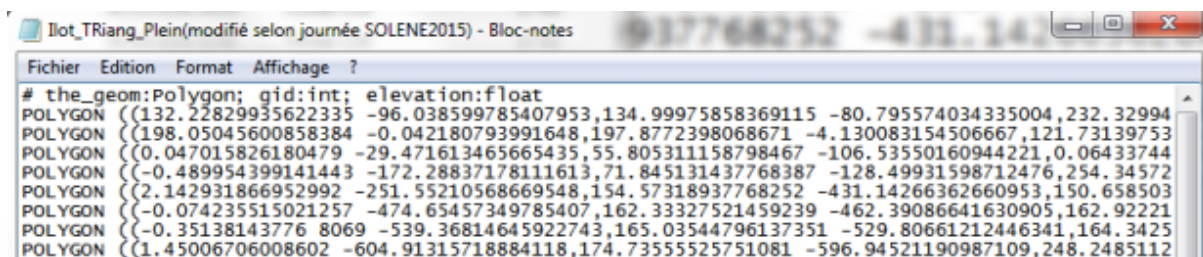


Fig. 3.4: Modified SketchUp import file.

You can then import your file into SketchUp.

Let's change it up a bit by calculating the sky view factor on the sampled bounding box :

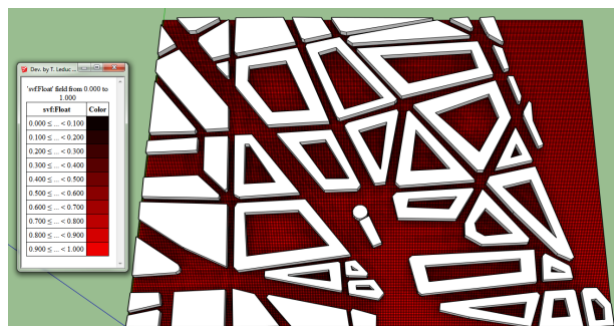


Fig. 3.5: Sky View Factor on Sampled Bounding Box.

Now that we've found what we're looking for, how can we import this file back to our standard cartography software? To export CSV files:

Click on t4su > New I/O > CsvWktWriter. The following command box will appear :

Fig. 3.6: CsvWktWriter Command Box.

You can choose to import all (*) or a single Layer at a time. In our example, we're just interested in our new bounding box. You will also need to supply the geometry type to correctly instantiate the WKT portion of the csv file. In this example, we're just going to export our bounding box as the other geometry layer hasn't been modified.

If you've decided to import all your layers (*), you can choose create a single document for all your geometries. Be warned though, this will not work if you have conflicting geometry types : a single csv file cannot contain more than one geometry type. Here's what our exported bounding box looks like :

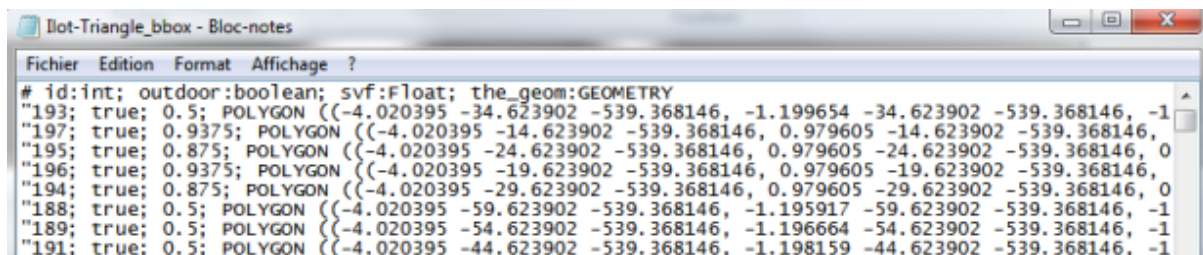


Fig. 3.7: Original Sketchup Output File

Once again, you'll have to change the first line of your new CSV file : remove the attribute types and replace #the_geom:GEOMETRY by WKT so as to mimic the original file.

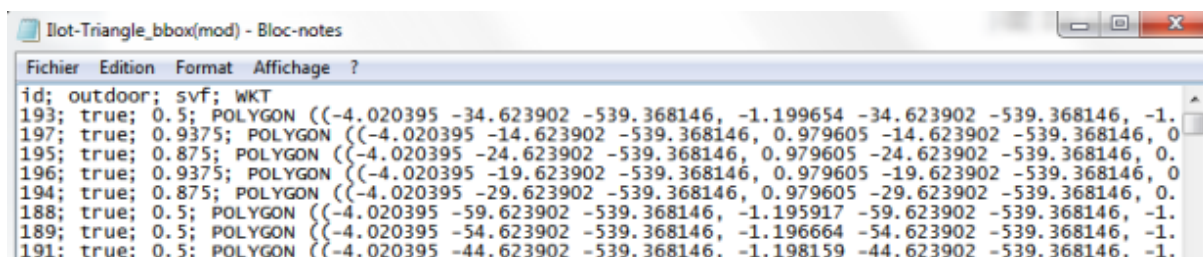


Fig. 3.8: Modified Qgis Input File

If your file is part of a larger map, select yes at Back to world coordinates. This will reset the geometries to their original geographic location. This way, any geometries or features we have added on SketchUp can be found again at the correct location and coordinate system when re-exporting to a tradition GIS system. To view the Sky Viw Factor in Qgis, **right-click on the layer name > Properties > Style > Categorized > svf > classify.**

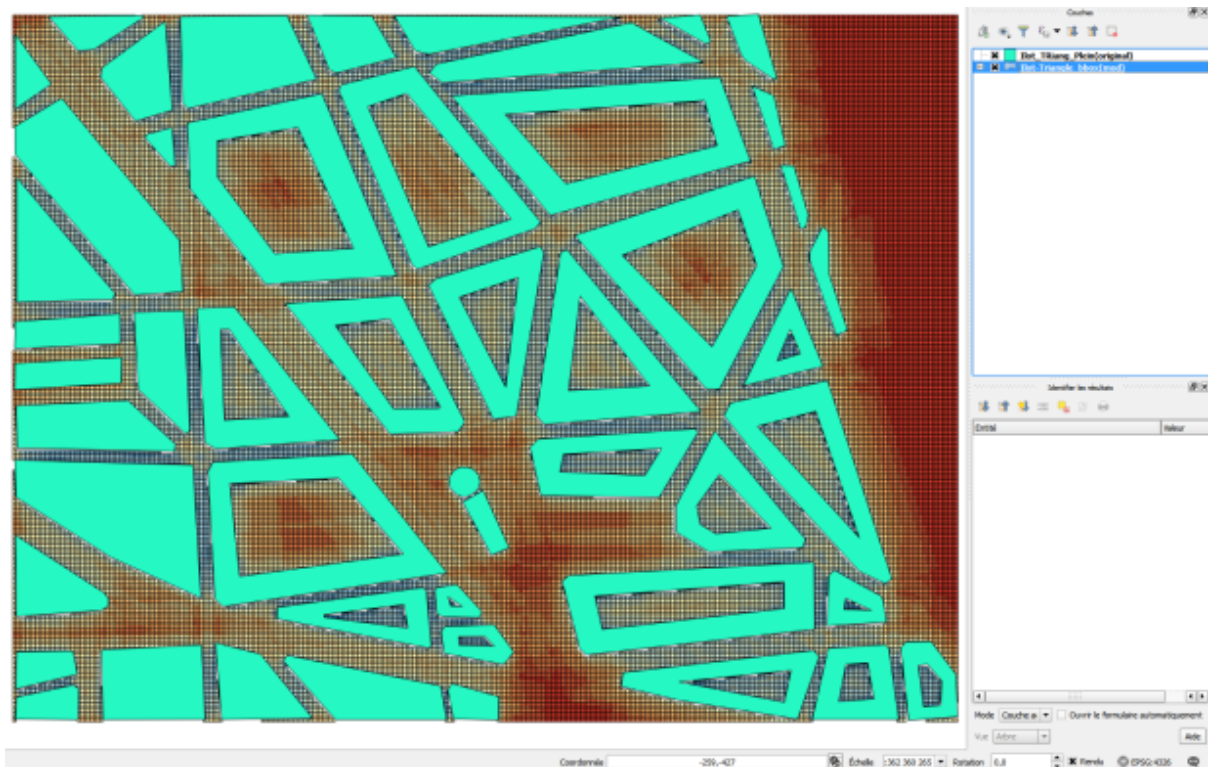


Fig. 3.9: Bounding Box with Sky View Factor viewed in QGIS.

My thanks to Houda Belgacem for letting me use her geometry file and for her help with .csv file conversion.

Importing Solene Files

.cir files are Solene's geometry files. Solene is T4su's standalone counterpart, so switching from one software to another is pretty straightforward in this case. Skp2Solene export.png Exporting a Layer to Solene

You can only import one layer at a time, so make sure everything you need is in the correct one. You can further choose to keep your geometries in a local coordinate system, or use the coordinate system used during the original file import. Solene also uses local coordinates, so choose no unless you're planning on exporting your Solene file back into real world coordinates further on. You can also choose whether or not to have a unique output file if you're importing several layers at a time. Unlike other I/O options, this command box does not ask the geometry type because Solene only uses faces.

Fig. 3.10: Solene Export Command Box

Once you hit OK, choose your export path and file name to finalize the .cir file creation. Once in Solene, create a

new project, then click on the icon with the red box, file and arrow to import your .cir file.

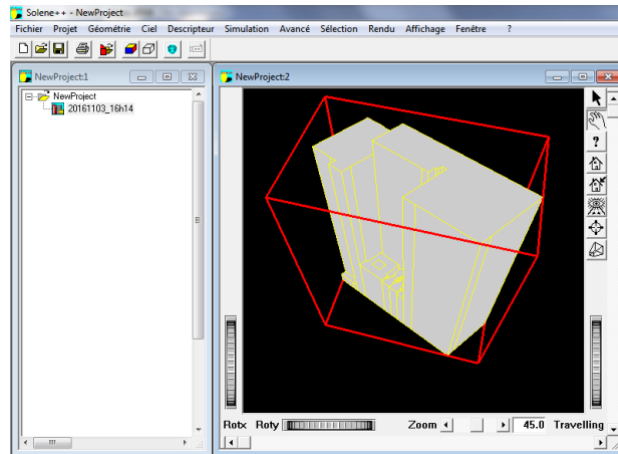


Fig. 3.11: Layer's Geometry Imported in Solene

CHAPTER 4

Preparing Your Data

Learn to pre-process your geometries before performing any analysis.

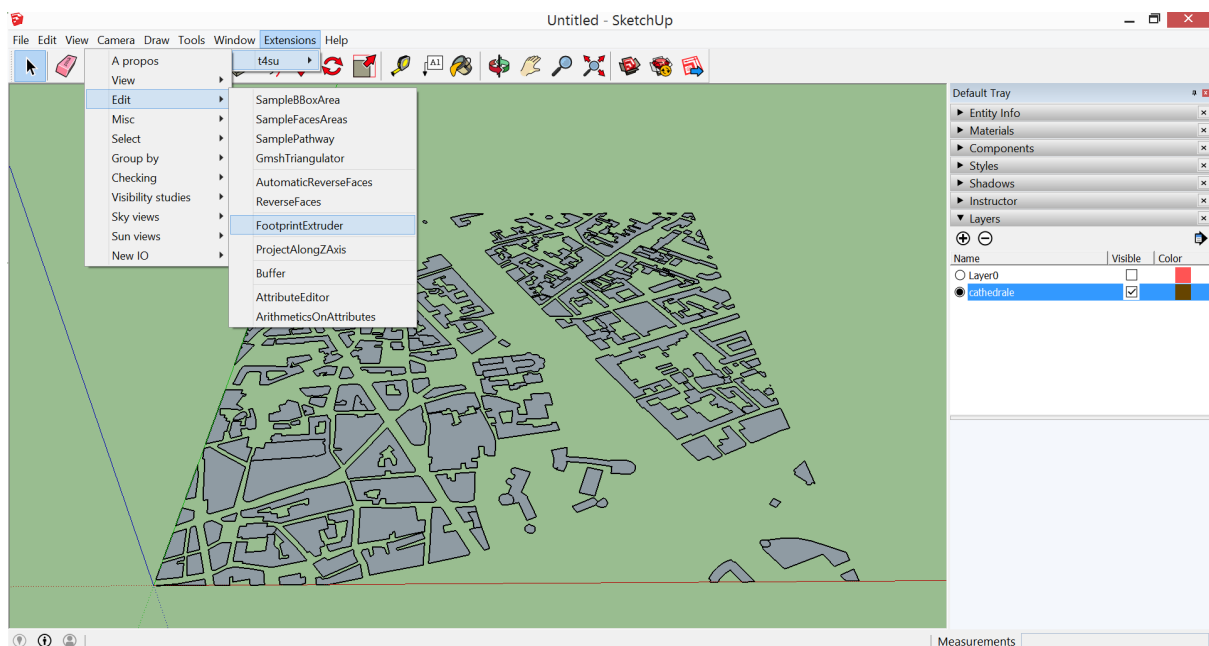
Extruding Heights from your Geometries

Once you have loaded your data, it will appear on your screen as well as in your Layers. The geometries are, for now, flat. To analyse our data, we must first extrude the geometries to their corresponding heights.

See also:

Loading your data into SketchUp

To do this, click **Extensions > t4su > Edit > FootprintExtruder**



A new window will open, asking for the attribute's name. Select the name of the column where your heights are stored.



If you are not sure which column it is, open your file in an external application and check. Here is an example of a WellKnown-Text file : the attribute we are looking for is named “elevation:double”.

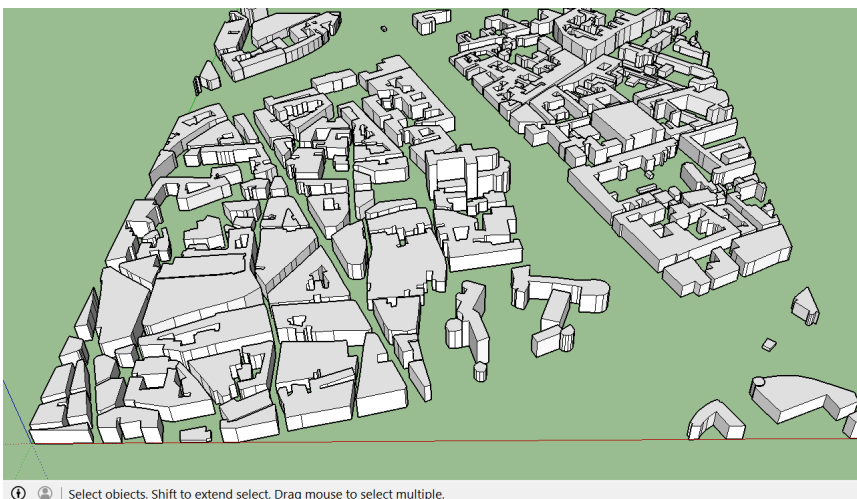
	A	B	C
1	#gid:int	the_geom:Polygon	elevation:double
2	1	POLYGON ((355980.6689874 0, 355974.4 6689871 0, 3559	11
3	2	MULTIPOLYGON (((356206.6120044511 6689627.166823:	17
4	3	POLYGON ((355997.0612293882 6690187.153400643 0, 3	24
5	4	POLYGON ((356191.6288474575 6690187.153400643 0, 3	13
6	5	MULTIPOLYGON (((355954.76257182774 6690187.153400	21
7	6	POLYGON ((355454.6008749507 6690187.153400643 0, 3	32
8	7	POLYGON ((355937.3 6690123.7 0, 355899.2 6690103.5 0	19
9	8	POLYGON ((356146.3 6690116.4 0, 356136.9 6690128.8 0	13
10	9	POLYGON ((355956.6689927.9 0, 355964.4 6689936.1 0, 3	19
11	10	POLYGON ((355953.6 6690105.2 0, 355927.2 6690091.5 0	14
12	11	POLYGON ((356206.6120044511 6690020.14575894 0, 35	27
13	12	POLYGON ((356175.7 6689840.7 0, 356150.2 6689828.8 0	13
14	13	POLYGON ((355932.4 6689833.3 0, 355907.6689833.6 0, 3	19
15	14	POLYGON ((355515.6 6690060.9 0, 355505.1 6690078.8 0	10
16	15	POLYGON ((356206.6120044511 6689720.822970828 0, 3	11
17	16	POLYGON ((356206.6120044511 6689326.924768041 0, 3	23
18	17	POLYGON ((356182.8 6690101.2 0, 356172.9 6690109 0, 3	7
19	18	POLYGON ((355960.4 6690096.1 0, 355979 6690069.9 0, 3	18
20	19	POLYGON ((355933.5 6690165.6 0, 355930 6690165.2 0, 3	5
21	20	POLYGON ((355913.2 6690086 0, 355903.5 6690079.7 0, 3	15
22	21	POLYGON ((356027.22751663846 6690187.153400643 0,	30
23	22	POLYGON ((356206.6120044511 6689903.910648798 0, 3	25

Select whether you would like the geometries to be extruded Upwards (Z direction) or Downwards (-Z direction) and click **Okay**.



Fig. 4.1: Top Image : The imported geometries are rooftops. Bottom Image : Rooftops have been extruded downwards to create buildings

Warning: If you do not select a specific object, the extrusion will be carried over all geometries. You may also select specific geometries before extruding, in which case only the selected figures will be affected by the extrusion.



Checking the Normal Vector of your Geometries

For any simulation to run smoothly, you must make sure that all of your two-dimensional geometries face the correct direction.

To check this, just click **Extensions > t4su > View > PlotAllNormals**. The following box will appear. Select the correct layer name, and size of the vectors :

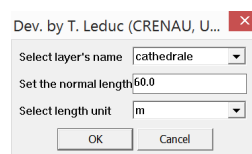


Fig. 4.2: Checking Normals Comand Box

Once you click **OK**, a new layer will be created resembling this screenshot:

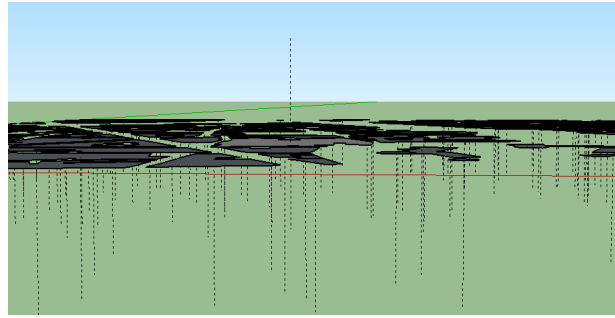
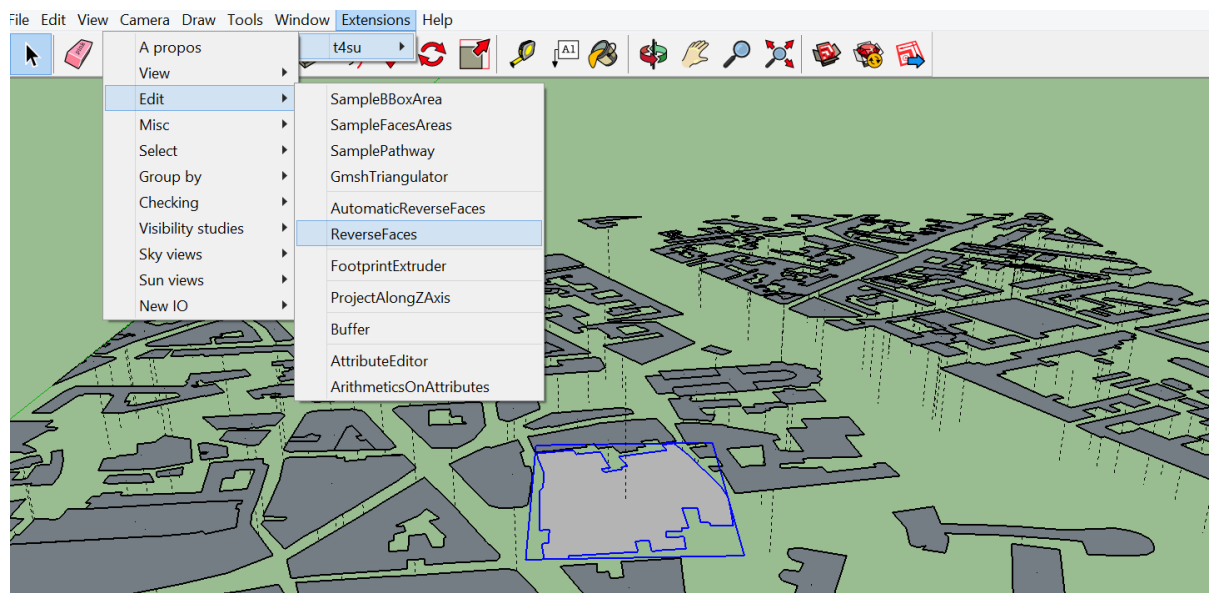


Fig. 4.3: Incorectly Positioned Normal Vectors.

As you can notice, one of the geometries is facing the wrong way. Let's change that :

- select the faces you wish to reverse
- click on **Extensions > t4su > Edit > ReverseFaces**



- Delete the obsolete PlotAllNormals layer
- Repeat **Extensions > t4su > View > PlotAllNormals** to see if everything worked.

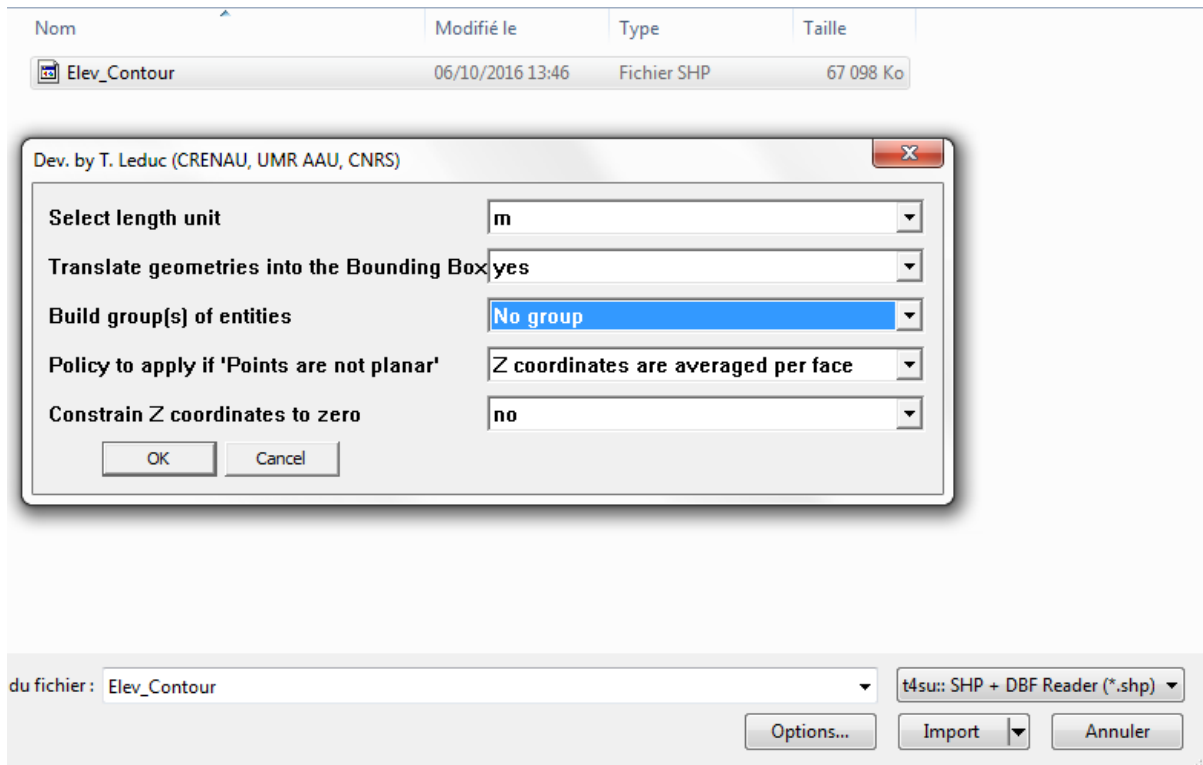
Recreating a Terrain with Contour Lines

On many topographical maps, altitude levels are frequently represented by contour lines, or “isohypes” : lines representing equal elevation. We can turn these isolines into a usable digital terrain model. Start by importing your lines.

See also:

Loading your data into SketchUp

This case has an **extra step during the import process** : click on **Options..** and change the **Build group(s) of entities** line to “No group”.



Next, select everything (**Ctrl+A**) and click on **Draw > Sandbox > From Contours**. If when selected, your geometries appear individually bounded by selections boxes, it means you've forgotten to import them as **No Group**. Instead of re-importing your geometries, you can **right-click** on your selected geometries, then hit **Explode**.

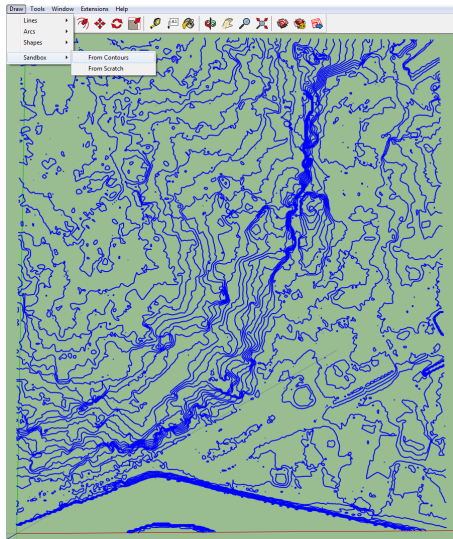


Fig. 4.4: Correct importation of isolines.

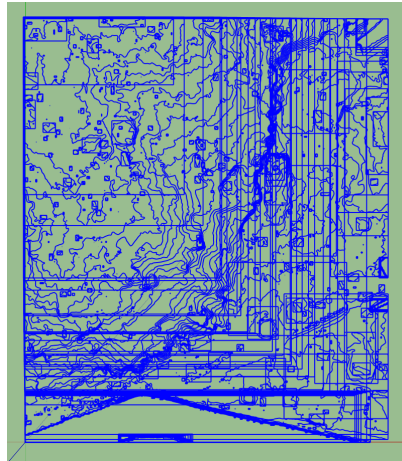


Fig. 4.5: Incorrect importation of isolines.

You should end up with a layer that resembles this:

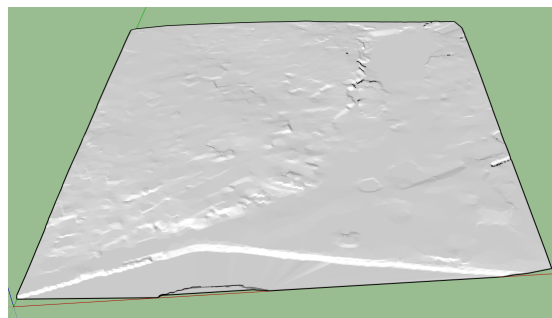


Fig. 4.6: Simplified TIN Model.

Visibility, including sky view factors, changes on a rough terrain: our sky view is much more restrained at the bottom of a valley than at the top of a hill. Adding precise altitudes to your map will greatly aid in the accuracy of your calculations.

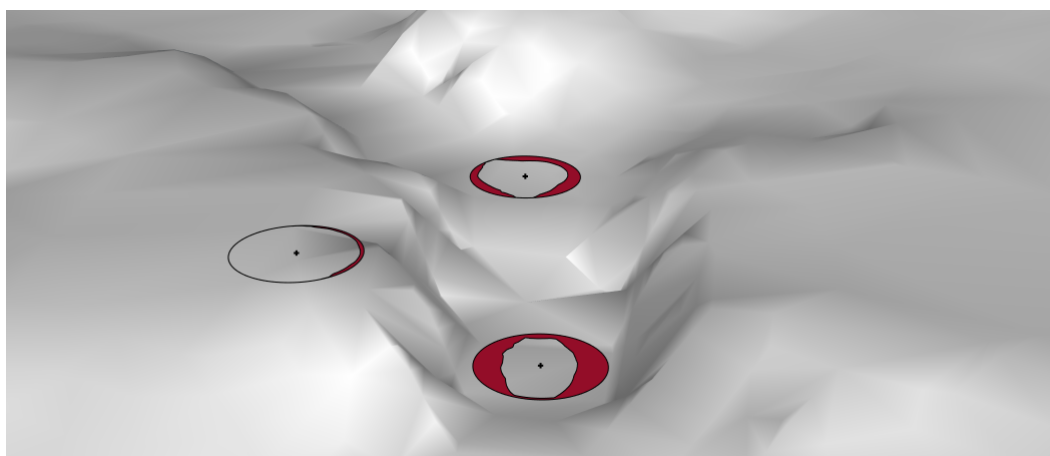


Fig. 4.7: Close-up of three “2D Sky-Maps” on a rough terrain : areas in dips and valleys “see” less of the the sky than the tops of hills.

Projecting Along the Z Axis

Projecting along the Z-axis is really helpful in plotting many objects, like trees or people, on a uneven terrain. The concepts is pretty straightforward : we first import or create a layer of points, then project them on our target surfaces. This technique also works for line segments. So first, import your uneven ground layer and the layers to be projected:

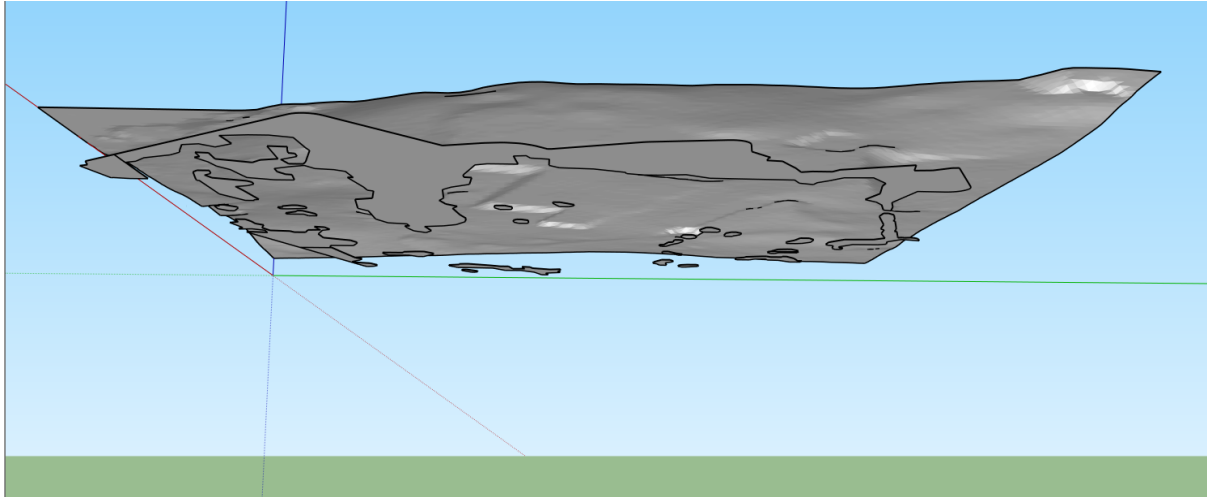


Fig. 4.8: Rough Terrain Layer above a Layer Representing Wooded Areas.

Below our ground layer, we can see another set of geometries which correspond in this case to forested areas. That is the layer to perform a ‘point sample on.

See also:

Creating Point and Face Samples of your Buildings

Once you have your points, you can delete the previous layer : we’ll only be needing the points from now on. Hit **t4su > Edit > ProjectAlongZAxis**.

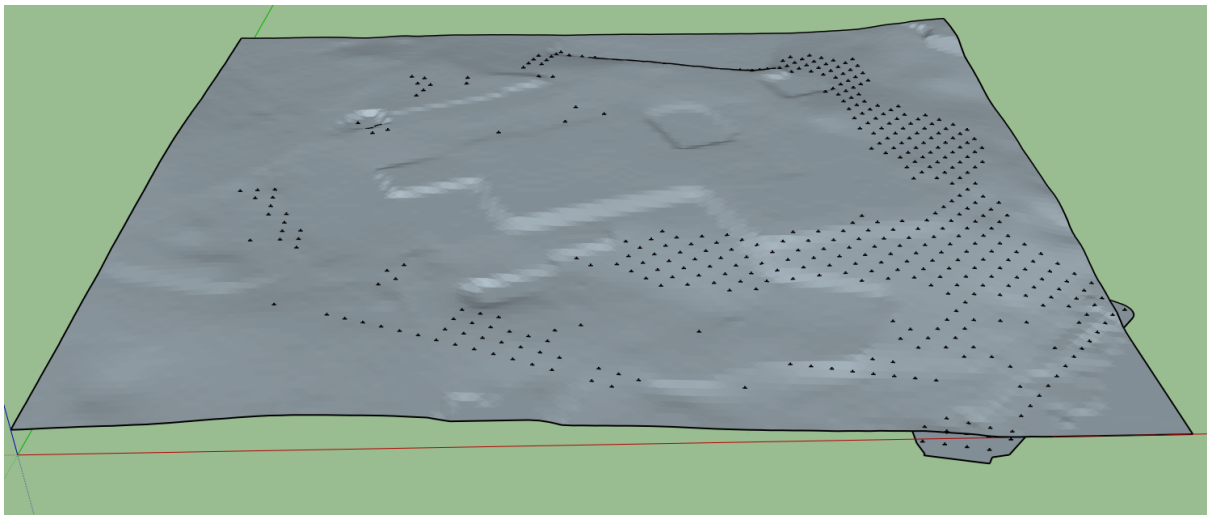


Fig. 4.9: The “Wooded Area” Faces are Sampled into Points, which are then Projected onto our Terrain Layer.

Once the command box appears, select the layer containing your sampling points. Next, decide which direction to give to the projection : you may choose between **upwards** and **downwards**. Since our points were created from a layer *below our ground layer*, we are going to **project upwards**. If of course, your point layer is situated above

the ground layer, chose **downwards projection**. This will result in your points “travelling” in the Z direction until they hit a face (in our case, the ground face). We can then use the projected sampling points as locations for more complex geometries.

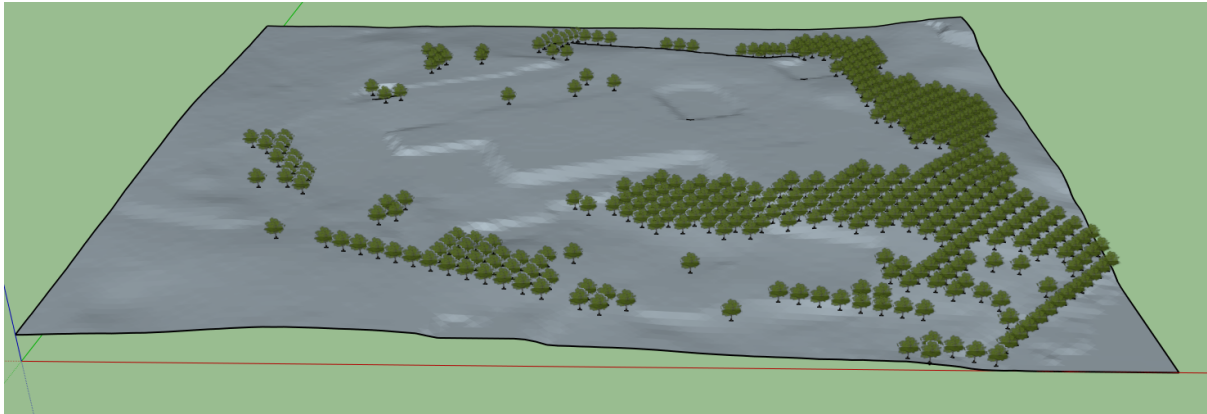


Fig. 4.10: Example of the Use of Z-Axis Projection

See also:

2D Sky Maps and Vegetation Modelization

General Tips to Start

T4SU offers many features that allow for an automatic selection, modification, creation or visualization of information, for a fast and easy way to treat and analyse our data. Many of these techniques will be used again in examples throughout the manual.

Selecting Geometries

One of the many features of T4su is that it will automatically recognize selected objects when running a process over a layer, and will run that process only on the selection. So if you want to run a process on an entire layer, make sure to not have anything selected. The T4su extension includes a very extensive assortment of selection tools, each oriented towards a specific usage. You can find this collection when clicking **Extensions > Select**. Regardless of the type of selection, you will have the same three **post-process after selection** choices:

- simply selecting the geometries;
- automatically deleting the geometries;
- moving the geometries to a new layer. The layer will be named [NameOfSelectionProcess]-[Date and Time]

Types of Selection Processes :

- SelectRoofFaces : this will select the top-most geometries
- SelectWallFaces : this will select all the vertical faces.

These two selection features are pretty good at distinguishing one from the other. To test this, I've modified my original shapefile, adding a pointed roof and a small dormer on top of the original roof. The selection module still works :

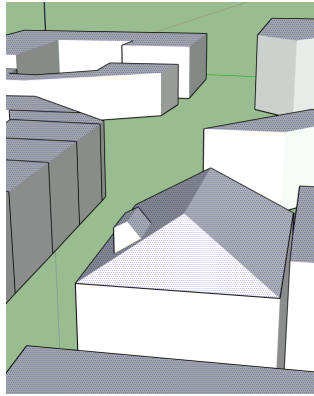


Fig. 5.1: Roof Selection

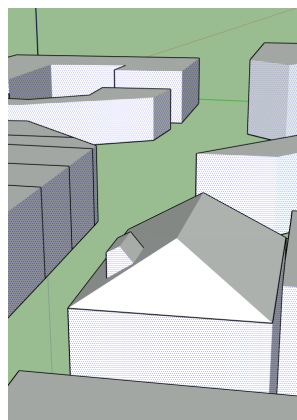


Fig. 5.2: Wall Selection

- **SelectGroundLevelFaces** : Selects all the faces with a z0 value of 0. When performing this on a building layer, this selects the ground floor. This will also work for ground-level Sampled Faces and Bounding Box with a Z-value of 0.
- **SelectAllConnectedFaces** : used for export towards Salome
- **Select UnnecessaryEdges** : selects Tin edges of coplanar surfaces to simplify a rough terrain by grouping coplanar faces. This is mainly used used directly with the **remove geometries** option. You may have to play around with the **threshold value**, as there's a strong chance some of your faces will disappear. This is because your threshold value is too low, meaning your margin of error is too large when selecting coplanar faces. Conversely, choosing too small a value will hardly reduce your number of faces, if any at all.

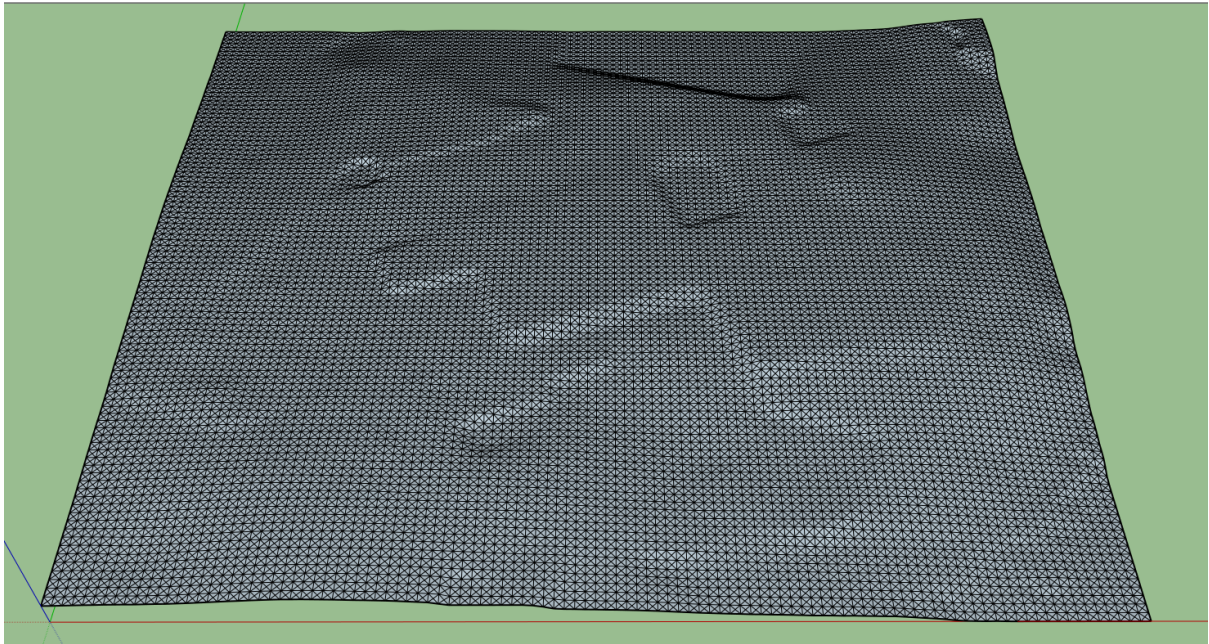


Fig. 5.3: Original Terrain

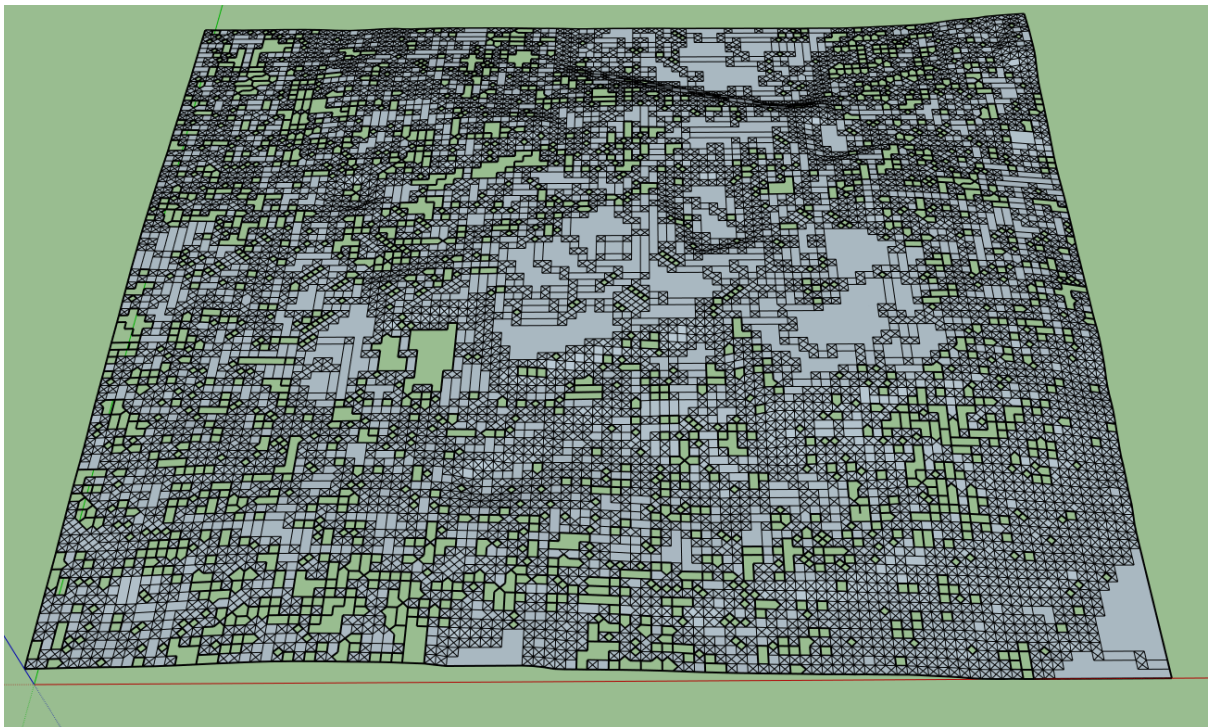


Fig. 5.4: Deleting Unnecessary Edges with Too Low a Threshold will Puncture the Terrain

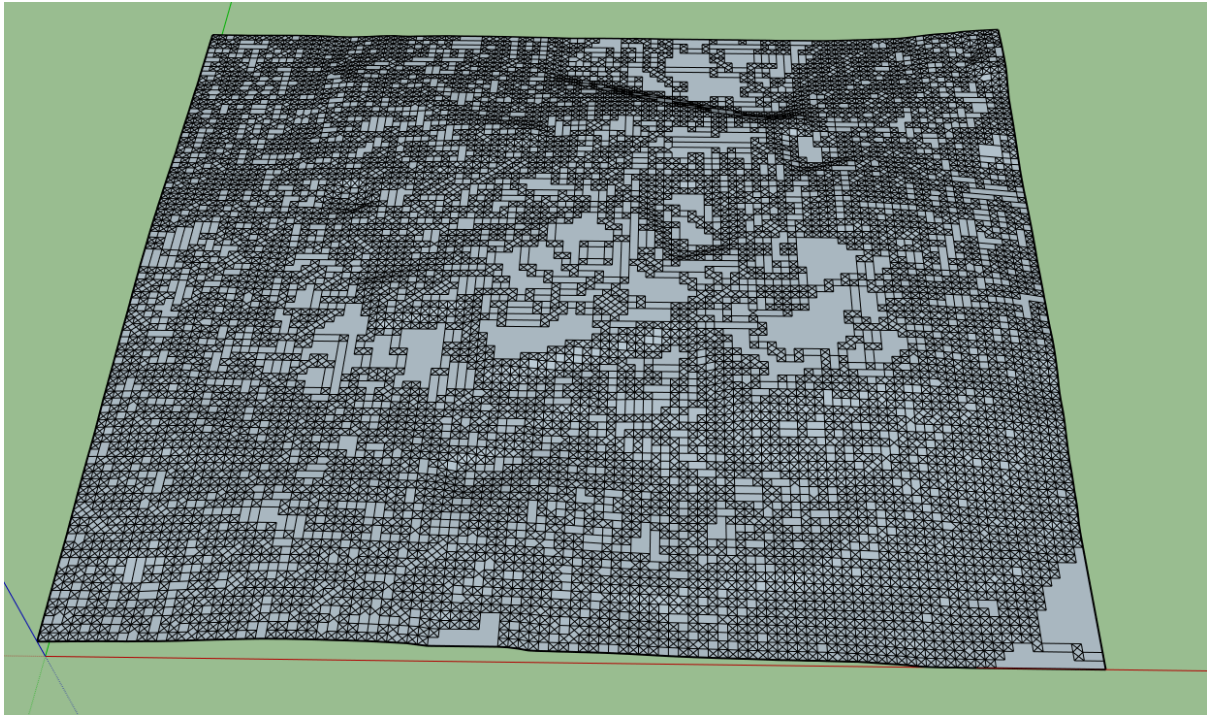


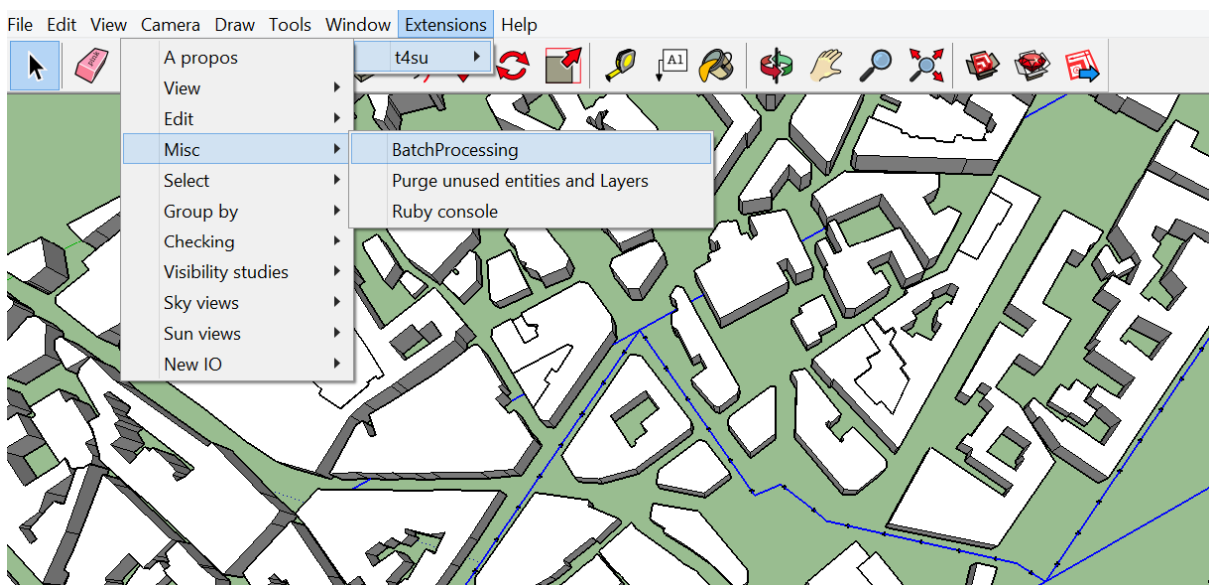
Fig. 5.5: Result of Selecting a Correct Threshold for Unnecessary Edges

- `SelectGroupsAccordingToVolumeProperties` : Whether the 3D geometry is closed (ie, hermetic, bounded by faces on all sides), or open (missing a face, or with window or door openings).

An interesting feature is that these types of selections will bypass geometry Groups made in sketchUp.

Batch Processing

Batch Processing allows you to automate a number of processes available in t4su, rather than working with point-and-click.



You first need to plot any points of interest, through a sampling method, or by importing a layer or geo-localized points. Using predefined data for your calculations allows for a more precise result than using your mouse for

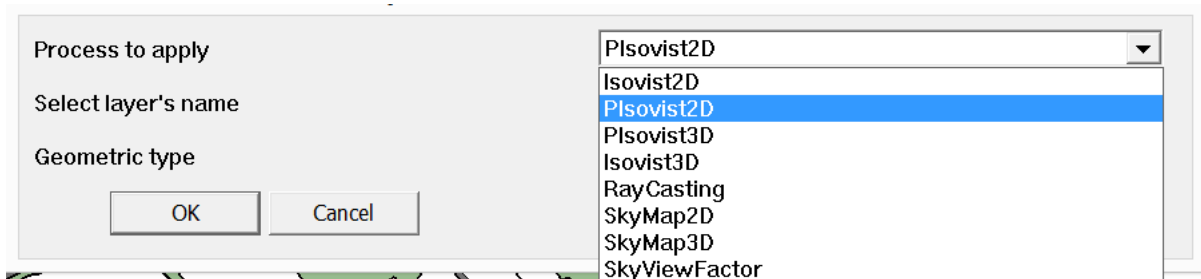
each individual point, and may be time-saving depending on the number of points you have.

See also:

Sampling Methods

Loading your data into SketchUp

Once you click **Extensions > Misc > BatchProcess**, the following box will appear :



Select your layer containing your data points. These will be at the center of your disks (if you work in 2D) or spheres (for 3D Isovists or Sky Mapping), or the origin of your rays (in the case of Ray Casting).

Running Times and Geometrical Precision

Most modules in T4SU allow you to choose your precision. There are general guidelines for choosing the correct. First, we can notice that the precision “p” follows the equation:

What is the concrete impact of augmenting precision ? Here is a first concrete example using 3D Sky Map:

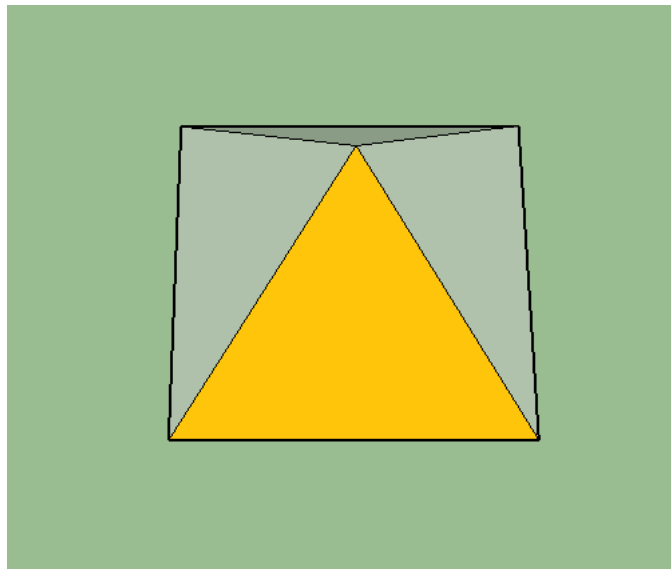


Fig. 5.6: 3D Skymap (p=4)

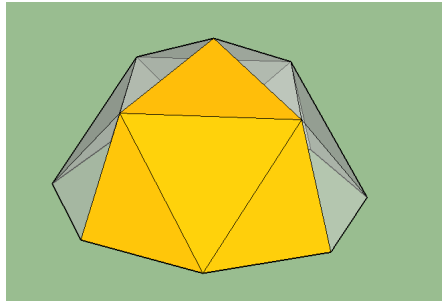


Fig. 5.7: 3D Skymap (p=16)

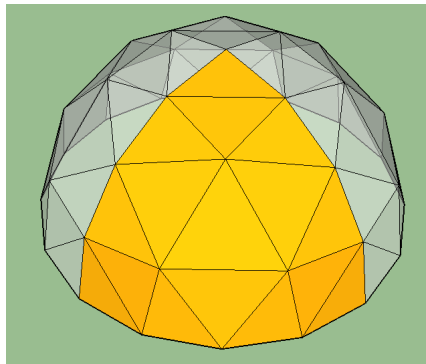


Fig. 5.8: 3D Skymap (p=64)

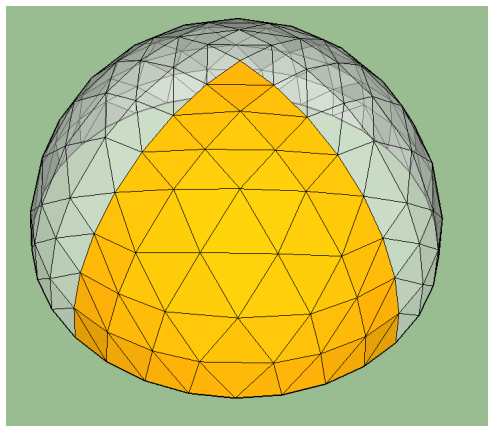


Fig. 5.9: 3D Skymap (p=256)

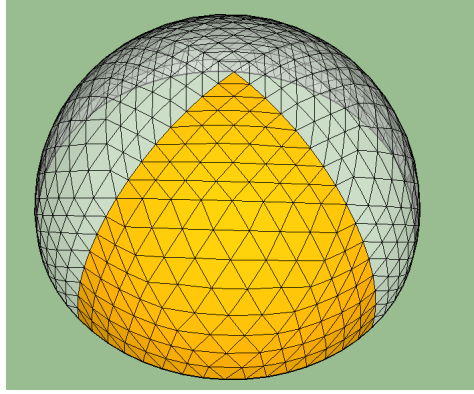


Fig. 5.10: 3D Skymap (p=1024)

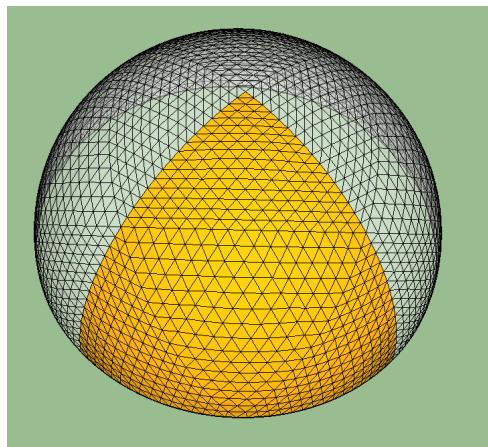


Fig. 5.11: 3D Skymap (p=4096)

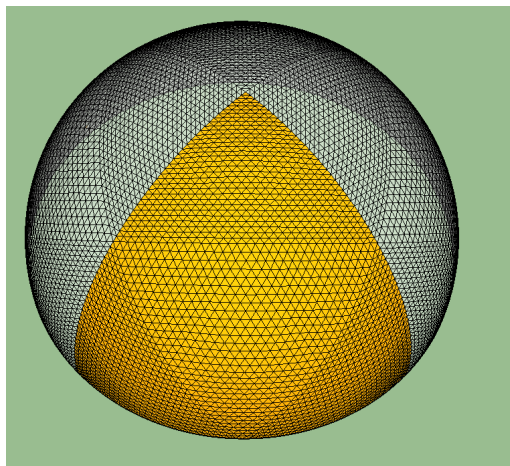


Fig. 5.12: 3D Skymap (p=16384)

For $n=1$, the most basic form is a four-sided pyramid. For $n=2$, the number of faces is $4^2=16$. For $n=3$, the number of faces is $4^3=64$ and so on. To calculate running times, each type of 3D SkyView has been repeated ten times, leaving the Ruby Console window on to show the elapsed time after each run. Although computation times will change according to the hardware used, here is a general idea of how long each type of 3D Sky View will take; As you can see, the last construction took 19 minutes (1150 seconds) to finish!

Number of faces Number of iterations		4	16	64	256	1024	4096	16384
Iteration 1		0,004	0,006	0,011	0,046	0,417	7,522	128,404
Iteration 2		0,005	0,008	0,018	0,088	0,84	14,027	234,236
Iteration 3		0,005	0,012	0,028	0,133	1,178	20,272	346,776
Iteration 4		0,007	0,111	0,038	0,195	1,483	27,705	458,882
Iteration 5		0,008	0,014	0,047	0,281	1,87	35,628	564,334
Iteration 6		0,008	0,016	0,058	0,3	2,341	41,246	680,333
Iteration 7		0,009	0,019	0,075	0,408	2,775	50,292	792,557
Iteration 8		0,01	0,022	0,086	0,439	3,131	55,449	934,421
Iteration 9		0,012	0,026	0,122	0,492	3,521	62,527	1067,693
Iteration 10		0,013	0,032	0,112	0,598	4,022	79,601	1150,123
Average		0,0081	0,0266	0,0595	0,298	2,1578	39,4269	635,7759

Fig. 5.13: Time (in seconds) taken for each 3D SkyView construction depending on its number of faces

Precision is also important when *constructing isovists*. As with *Sky Views*, we have the a choice of 4:sup: 'n precision. In this case, we are subject to 4, 16, 64 (...) rays to build our isovist. Too few rays, and the negative space will not be entirely mapped out :

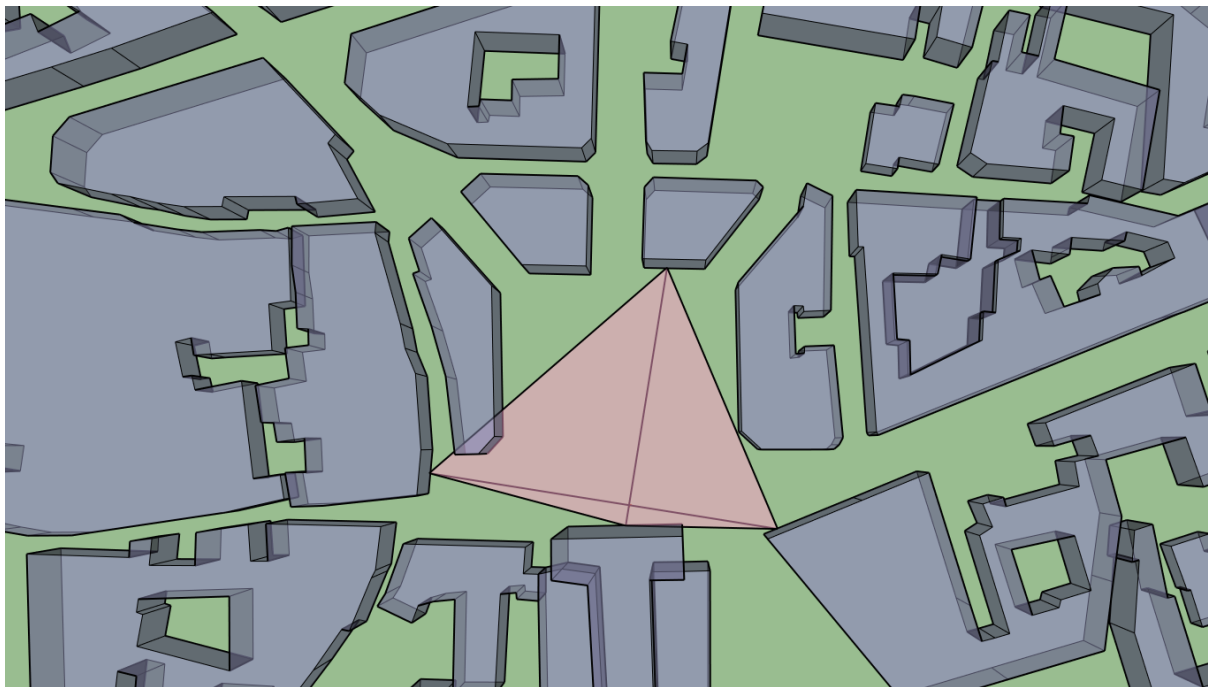


Fig. 5.14: 2D isovist built with 4 Rays, also shown.

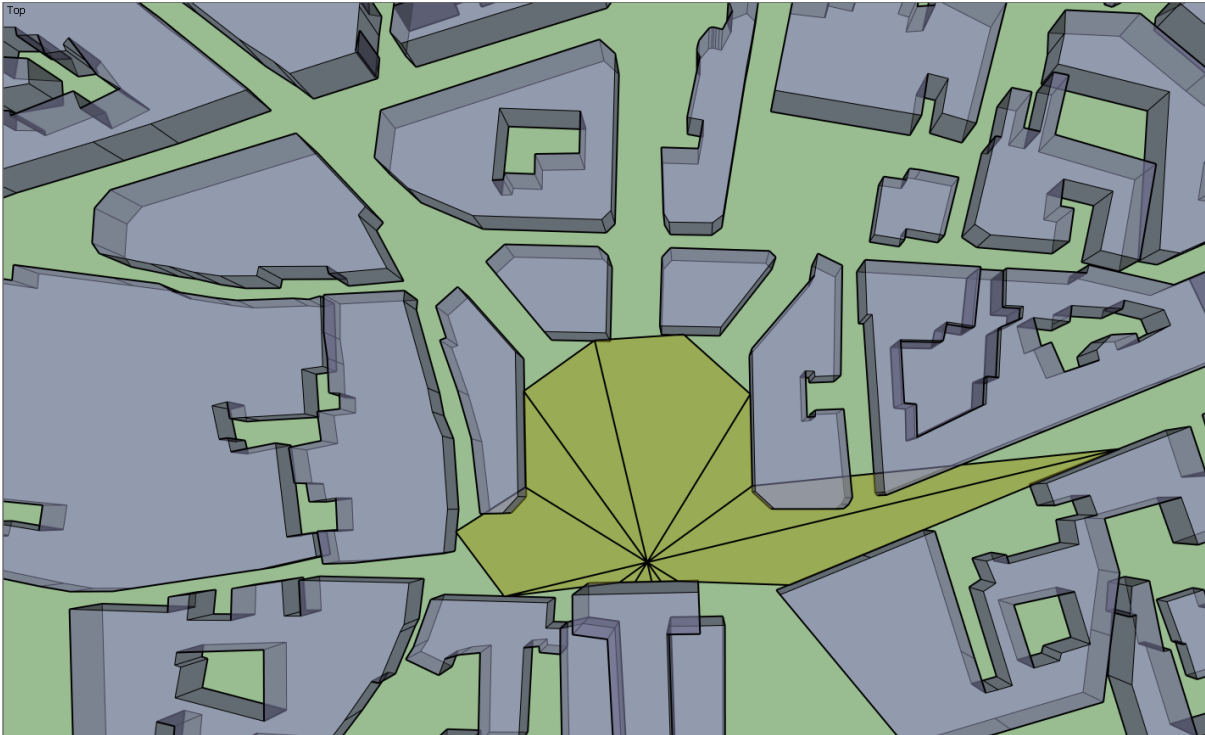


Fig. 5.15: 2D isovist built with 16 rays, also shown

In the pictures above ($n=1,2$ respectively), because the precision is too low, parts of the isovist traverse the buildings. You can verify this by clicking **View > Face Style > X-ray**. When moving to a precision of 256 ($n=3$), we can still notice a small error:

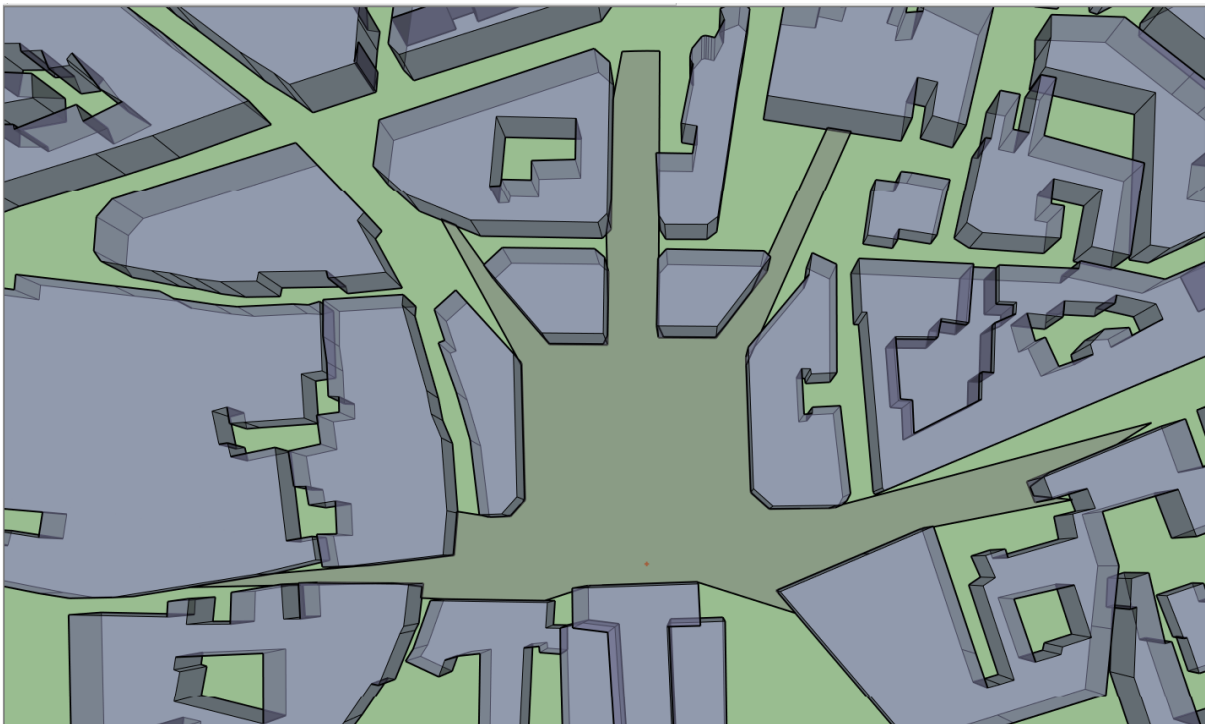


Fig. 5.16: 2D Isovist Constituted of 256 Rays.

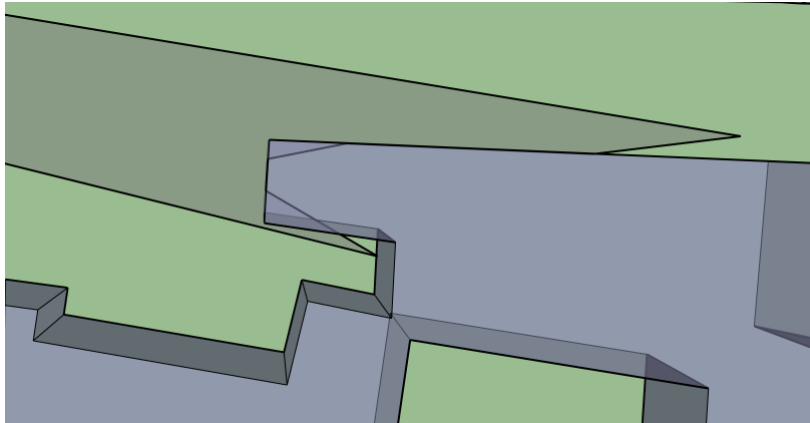


Fig. 5.17: Close-Up of the 256-Ray Isovist.

Moving to 1024 or 4096 rays correct this problem: they are almost identical, but the first one takes approximately 0.3 seconds whilst the second one take 1.3 seconds.

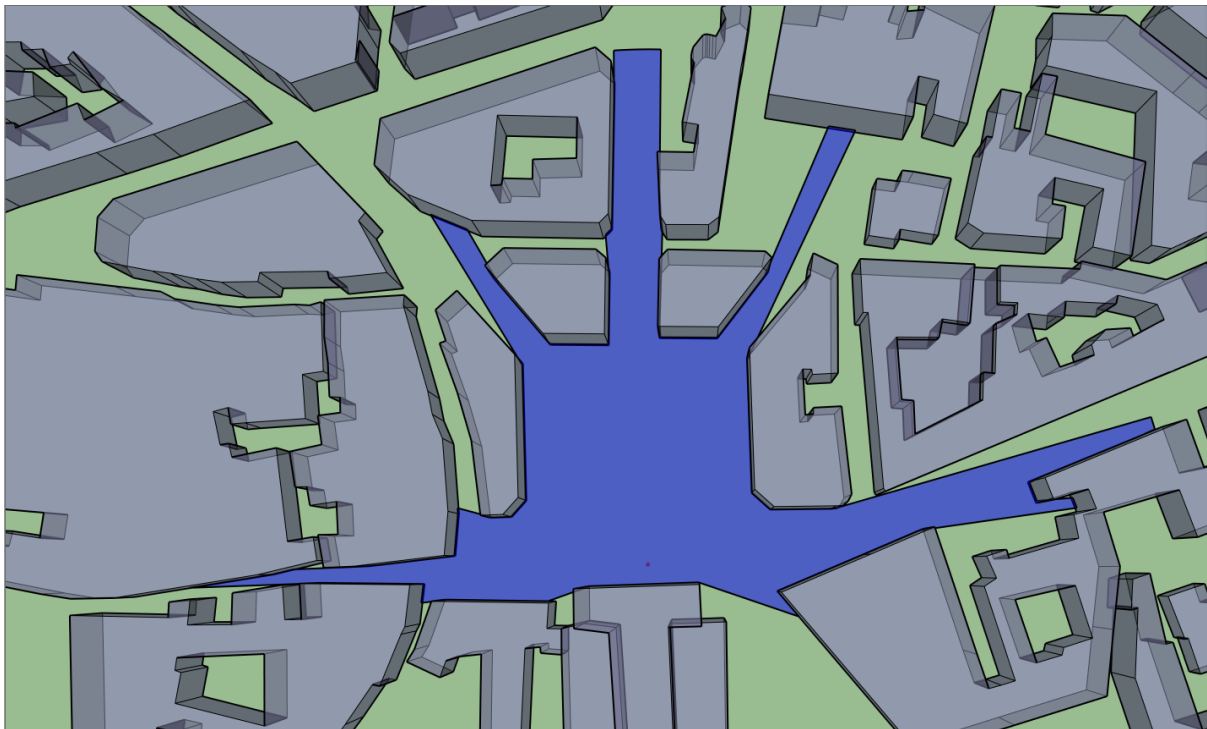


Fig. 5.18: 2D Isovist (1024)

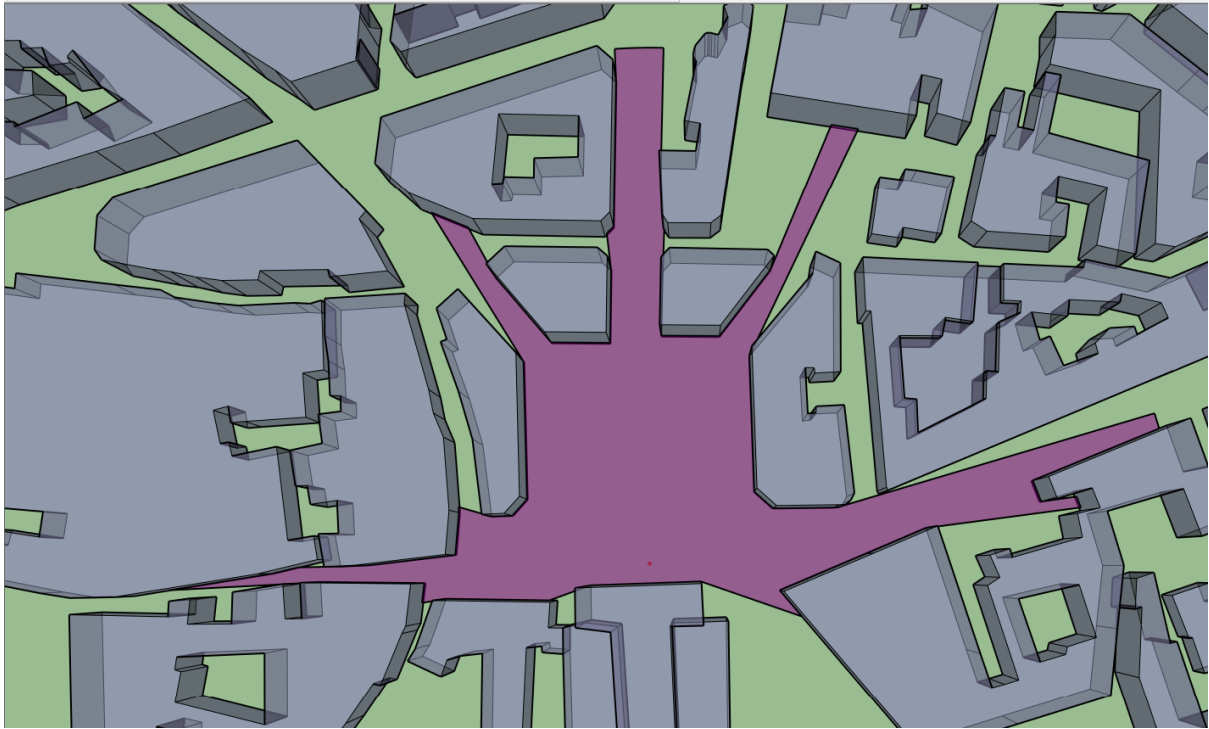


Fig. 5.19: 2D Isovist (4096)

In this case, the isovist with a precision of 1024 rays has the best precision to calculation time ratio.

See also:

What is an isovist ?

Using Buffers as Visual Information

T4su allows you to create buffers around points: a polygon enclosing a point at a fixed distance. You can access it by clicking **Extensions > t4su > Edit > Buffer**.

Select layer's name	SamplePathway_parcours_royal_de_luxe_cathedrale
Select entities' type	Sketchup::ConstructionPoint
Select buffer's type	Mitre buffer
Set the buffer size	1.0
Select length unit	m
Set the dimension	2D
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

The buffer's form can be either **Round** or **Mitre**, meaning square.

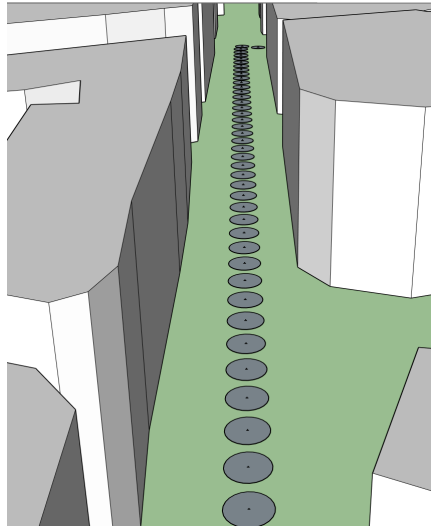


Fig. 5.20: Round 2D Buffers

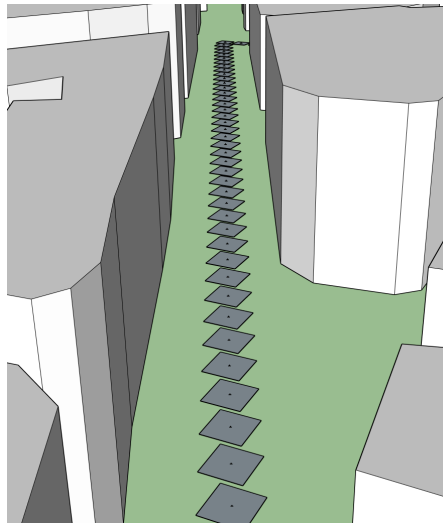


Fig. 5.21: Mitre 2D Buffers

The buffer's command box also allows you to set your buffer in 2D or 3D space. If you have selected a mitre buffer, the 3D result will be a cube, the size of the sides being your **set buffer size**. If you've selected a round buffer, your **set buffer size** will represent the length of the radius of a sphere.

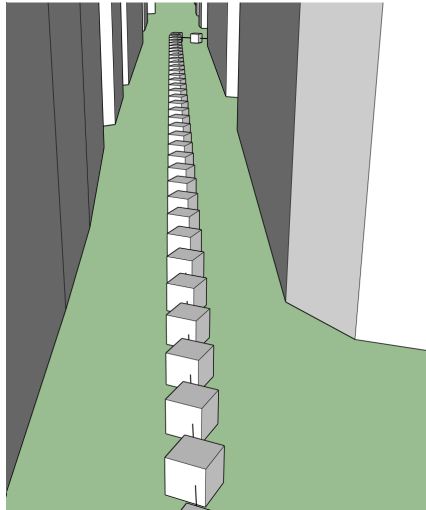


Fig. 5.22: Cubic Buffer

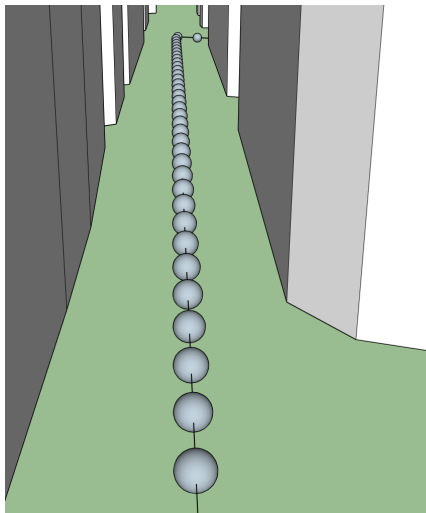


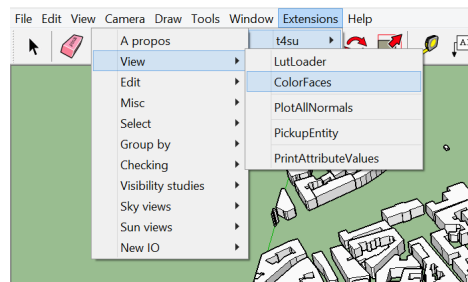
Fig. 5.23: Spheric Buffer

Warning: These buffers are regular geometries in your field. If they are left apparent while running a module that requires ray casting, your buffers will be taken into account and **will interfere with the calculations**.

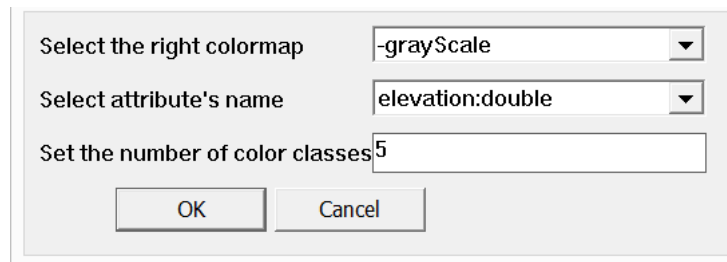
Basic Example of Data Visualization

T4su allows for an easy and rapid way of determining the range of building heights.

First, click **Extensions > t4su > View > ColorFaces**

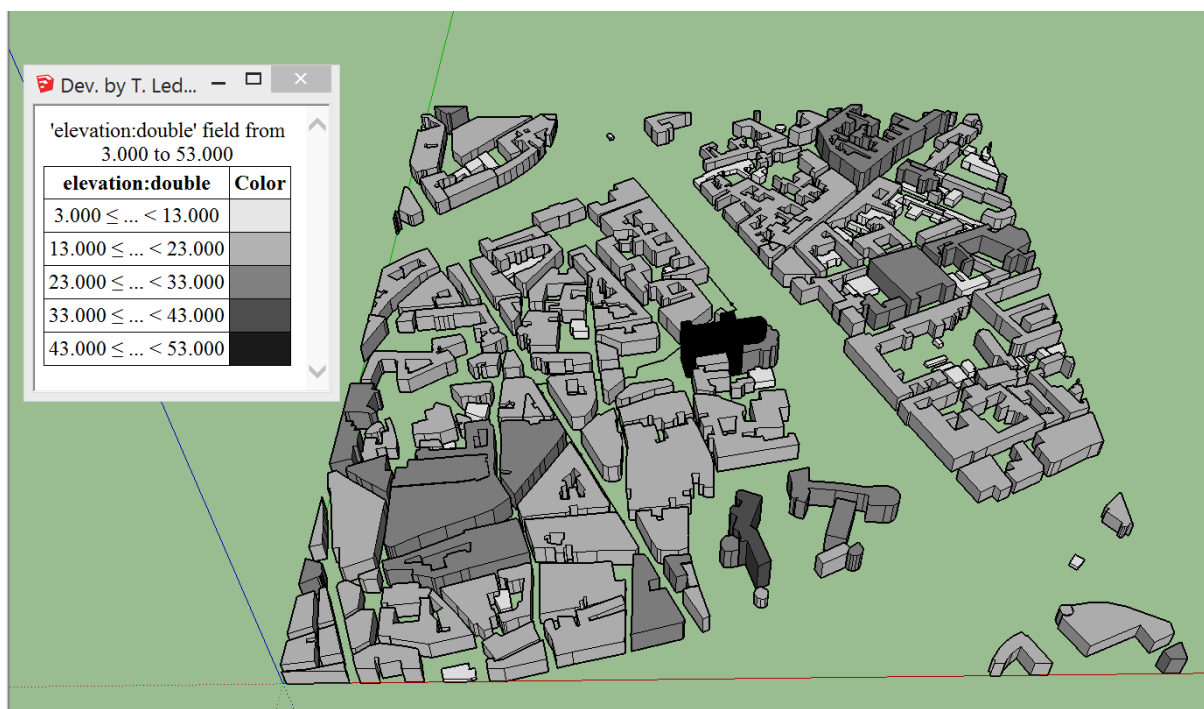


The following box will appear :



- Select your color scheme. You will notice that all colors appear twice, one of each containing a “-” in front (see picture above). Selecting a color scheme without “-” will allocate darker colors to smaller buildings and lighter colors to taller buildings. The “-” will produce the opposite : darker colors for higher buildings, lighter colors for smaller buildings.
- Select the correct attribute amongst the list. Here we have selected building heights (elevation:double) but you could choose another criteria (as long as it is float or integer).
- Select the number of classes you wish to create. This is entirely up to you, but keep in mind that above 5 classes, the difference in color may be difficult to distinguish. If you do, we recommend you use “fire”, “ice” or “GreenRed” which allows for a change of hue instead of a change in lightness / darkness. Be careful to select a color scheme that [corresponds to the nature of your data](#).

Here is an example of the end result :



Sampling Methods

T4su offers a four ways to sample your geometries. Sampling is very useful whenever a phenomena may be better studied with spatial discretization. whether they be flat surfaces or lines. They can be found by clicking **Extensions > t4su > Edit :**

Sampling the Bounding box

As with lines and geometries, the “ground floor” of your SketchUp can also be sampled into points or faces.

See also:

Sampling Methods

Start by clicking **Extensions > t4su > Edit > SampleBBoxArea.**

Set the X sampling distance (less than ~ 771,03m)	38
Set the Y sampling distance (less than ~ 877,25m)	43
Set z0 value	1.6
Select length unit	m
Sample into	Sketchup::ConstructionPoint
Select indoor/outdoor	Outdoor
	Indoor
	Outdoor
	Both

OK Cancel

Fig. 6.1: Sample BBox Command Box

- Set the X sampling distance : that is the direction of the red line in Sketchup. The “(less than [...m])” gives you an information on how large the bounding box will be.
- Set the Y sampling distance : that is the length of the sample in the direction of the green line.
- You can set the z0 value. You may want to change this for several reasons: if you are studying at which intersections a driver would be directly blinded by the incoming sunlight, it would be best to set the z-value

at eye-level when sitting down. If you are looking at how much direct sunlight hits the ground, setting a z0 value to 0 or 0.1 is more appropriate.

- You can select your unit length, which should not change throughout your project, and be in concordance with your original SketchUp model.
- The “**Sample into**” is very important:
 - If you select **SketchUp::ConstructionPoint**, points will be created. In this case, X and Y distances represent the distance between points (every Xmeters in the “red direction”, every Y meters in the “green direction”).
 - If you select **SketchUp::Face**, rectangles or squares will be created. In this case, X and Y distances represent the length of the sides of the sampling area.
- Finally, selecting **indoor or outdoor** will sample either the area inside of your geometries (indoor) exclusively, or use the building faces as a limit to sample only the outside area (outdoor). If you choose **both**, the bounding box will be evenly discretized, ignoring any limits set by your buildings/geometries.

See also:

Calculating the Sky View Factor

Sun View Factors

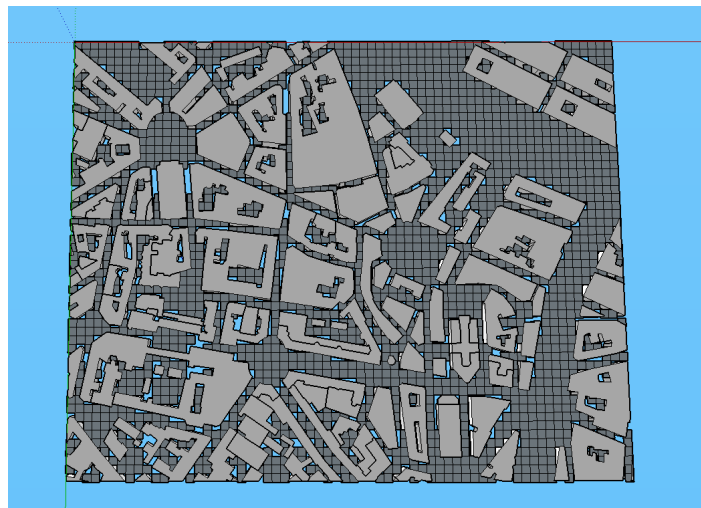


Fig. 6.2: Up-facing view with sampling distances X and Y at 1m

Creating Point and Face Samples of your Buildings

As we have seen in previous posts, we can sample a line or your bounding box into points or faces.

See also:

Sampling Methods

Creating Partial 2D Isovists

Sampling the Bounding box

We can perform the same type of task on your geometry faces. Here is how :

- Click **Extensions > t4su > Edit > SampleFacesAreas**

You will be asked to fill in the following command box:

Select layer's name	cathedrale
Set the sampling distance	5.0
Select length unit	m
Set the distance to faces	0.1
Sample into	Sketchup::ConstructionPoint
	Sketchup::ConstructionPoint
	Sketchup::Face
OK	Cancel

Fig. 6.3: SampleFaces Command Box

- Select the layer to sample.
- Select the sampling distance : unlike **SampleBBoxArea**, you only have one sampling distance : you can only create squares, or points separated by an equal distance in both (X and Y) directions.
- Select your unit length
- You can also select the **distance to face**. Similar to the z0 value, the distance is calculated as the normal to the surface. Setting a small distance (eg:0.1) is helpful in avoiding pesky overlapping sketchup geometries.
- Again, you can choose to create points or faces. This really depends on what information you wish to visualize over the discretized space.

Here are two examples of the difference between Sample Points and Sample Faces, both with a 5.0m precision and 0,1m distance from the original geometries.

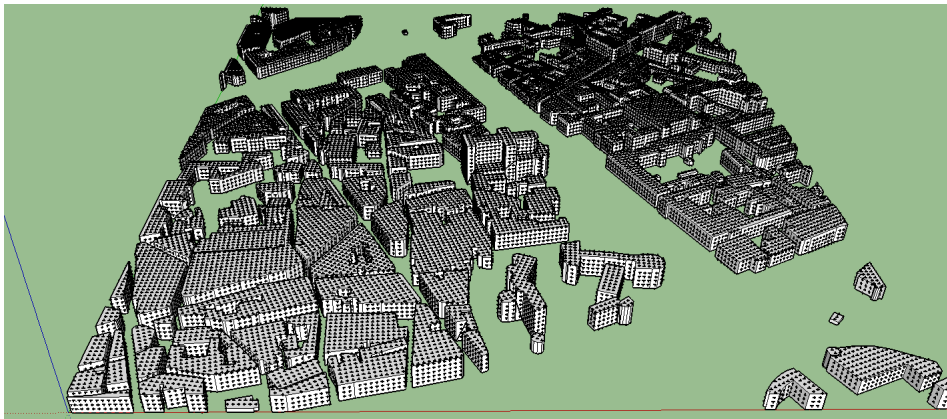


Fig. 6.4: Result of Face Sampling

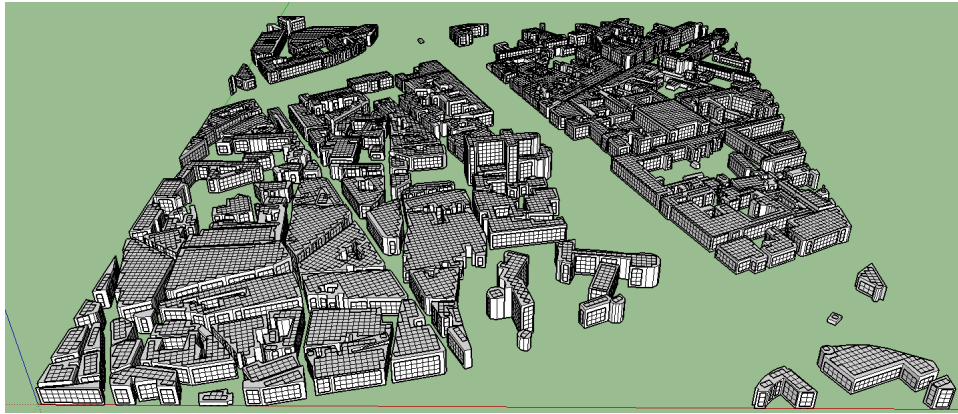


Fig. 6.5: Result of Point Smapling

Sampling A Path

As We've seen, sampling methods will grid your your area of study evenly through point or line sampling, whethere they be your bounding box, or your geometry faces. You can also perform this operation on line segments : the command box for SamplePathway is comparatively very simple :

See also:

Sampling Methods

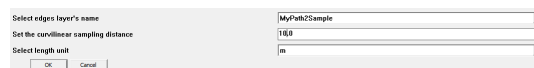


Fig. 6.6: "SamplePathway" Command Box

- Select the layer containing your edge geometry.
- Set the sampling distance. Curvilinear is a fancy way to say we are leaving cartesian coordinates: the distance taken into account only follows the line segment, not the X/Y coordinates. If a line is 10m long and is sampled every 3m, three points will be made along the line at the 3m, 6m, and 9m marks.
- Always keep the same unit length throughout of your project.

You now have a new Layer called SamplePathWay_[original_file_name], containing points spread along your line.

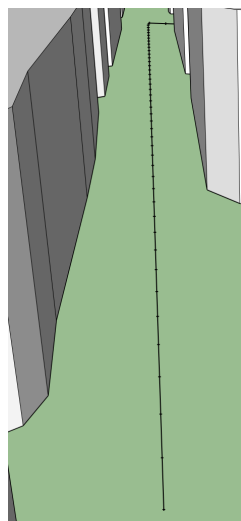


Fig. 6.7: Line Segment Sampled every 5m.

See also:*Using Buffers as Visual Information**Creating Partial 2D Isovists**Viewing Direct Solar Irradiation along a Path*

Triangulating Geometries with Gmsh

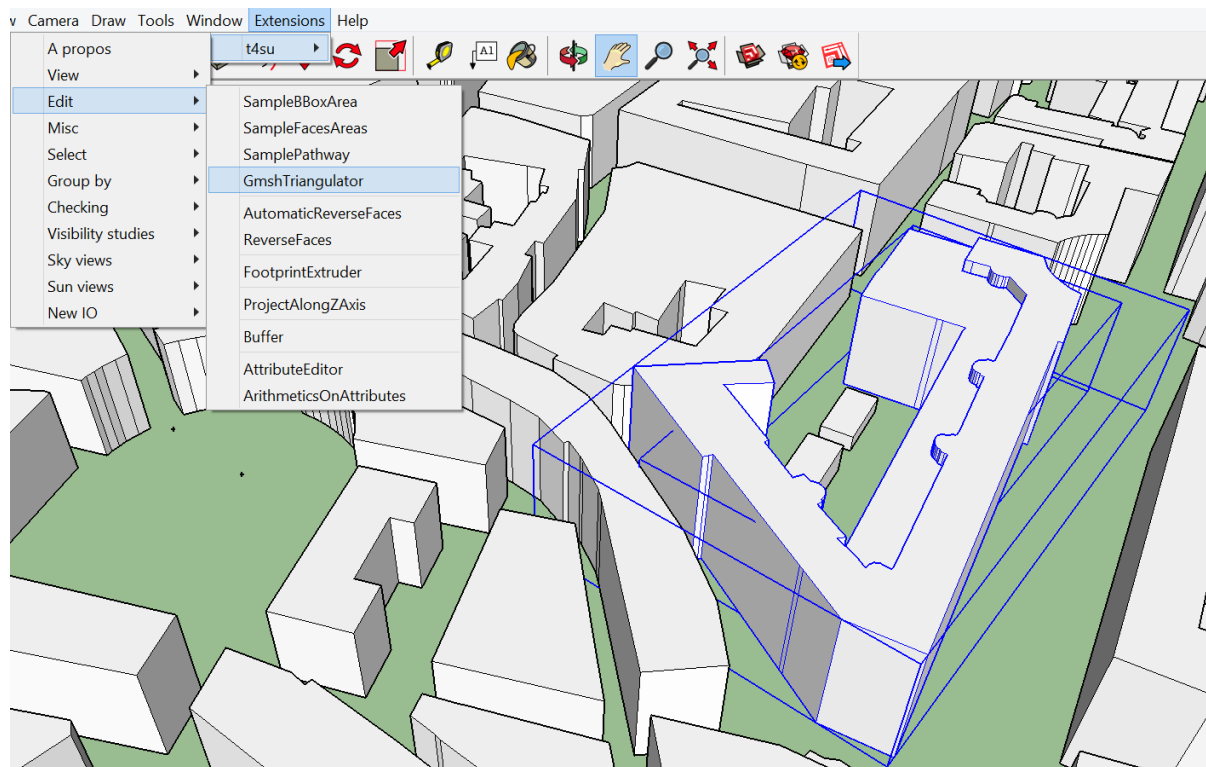
Unlike other sampling methods which result in point samples, or quadrangular (square or rectangle) tessellations, Gmsh allows for a complete subdivision of your building geometries into triangles ; it creates smaller triangles where shapes are more variable, and larger ones on flatter surfaces.

See also:*Sampling Methods*

The first step of the process is to [download Gmsh](#). Keep in mind where you decompress the .zip file ; you will need to copy/paste that location further on.

<input type="checkbox"/> Nom	Type	Taille compressée	Protégé par ...	Taille	Ratio	Modifié le
demos	Dossier de fichiers					
tutorial	Dossier de fichiers					
CREDITS	Document texte	4 Ko	Non	8 Ko	61 %	18/08/2016 20:22
<input checked="" type="checkbox"/> gmsh	Application	26 883 Ko	Non	73 626 Ko	64 %	18/08/2016 20:59
LICENSE	Document texte	8 Ko	Non	20 Ko	63 %	18/08/2016 20:22
onelab	Python File	6 Ko	Non	20 Ko	73 %	29/03/2016 00:30
README	Document texte	1 Ko	Non	1 Ko	37 %	18/08/2016 20:22

Next, click on **Extensions > t4su > Edit > GmshTriangulator**



The command box that follows allows you to connect Sketchup to Gmsh :

A dialog box titled "Gmsh Command Box (1)". It contains four input fields: "Select layer's name" with a dropdown menu showing "royale_graslin", "GMSH filename" with a text box containing "\loads\gmsh-2.13.2-Windows64\gmsh-2.13.2-Windows\gmsh.exe", "Temporary directory" with a text box containing "C:/Users/kevin/AppData/Local/Temp", and "Default characteristic length" with a text box containing "10.0". At the bottom are "OK" and "Cancel" buttons.

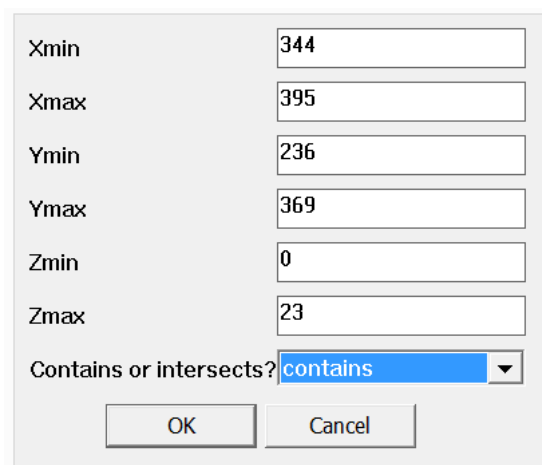
Select layer's name	royale_graslin
GMSH filename	\loads\gmsh-2.13.2-Windows64\gmsh-2.13.2-Windows\gmsh.exe
Temporary directory	C:/Users/kevin/AppData/Local/Temp
Default characteristic length	10.0

OK Cancel

Fig. 6.8: Gmsh Command Box (1)

- Select the correct layer. In this example, we've previously selected a building in our layer, the triangulation will therefore be performed only on that specific geometry. If nothing is selected, the triangulation will be operated over the entirety of the layer.
- In the text box below, paste the location of wherever you extracted the Gmsh zipfile, and add \gmsh.exe to the end.
- The triangulation will be saved in a temp file, you can decide upon its location in the third text box.
- Finally and most importantly, select the default size for your triangles, though some will be much smaller depending on the actual shape of your building. You may be tempted to create very small triangles for better precision, but keep in mind this will have a great impact on computation times, especially once you use them to analyse Sun View Factors for example.

The next window will tell you the number of triangles that were created. After clicking **OK**, A new window will appear that is very much like the one that appeared when you imported your geometry into Sketchup. Minimum and maximum x,y,z values express the recalibration of the coordinate system.

A dialog box titled "Gmsh Command Box (2)". It contains six input fields: "Xmin" with "344", "Xmax" with "395", "Ymin" with "236", "Ymax" with "369", "Zmin" with "0", and "Zmax" with "23". Below these is a dropdown menu labeled "Contains or intersects?" with "contains" selected. At the bottom are "OK" and "Cancel" buttons.

Xmin	344
Xmax	395
Ymin	236
Ymax	369
Zmin	0
Zmax	23

Contains or intersects? contains

OK Cancel

Fig. 6.9: Gmsh Command Box (2)

Once you click **OK**, a new layer will be added, named as a series of numbers. Within it, you may find a copy of your original buildings whose faces have been triangulated.

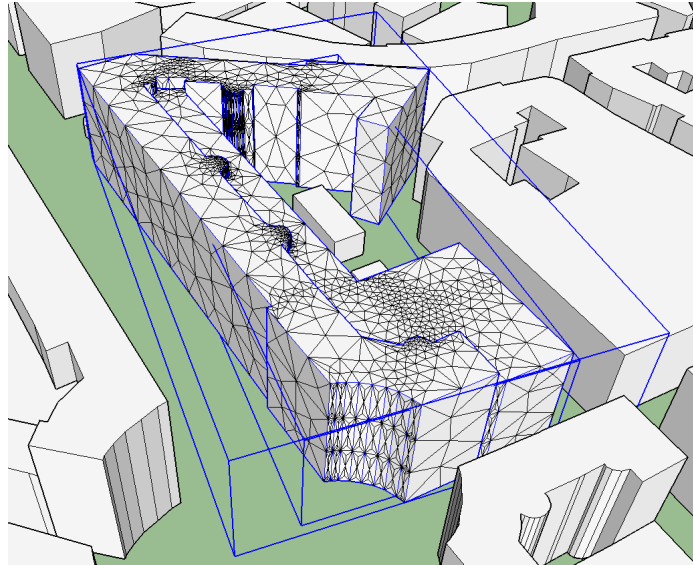


Fig. 6.10: Building Triangulated with Gmsh.

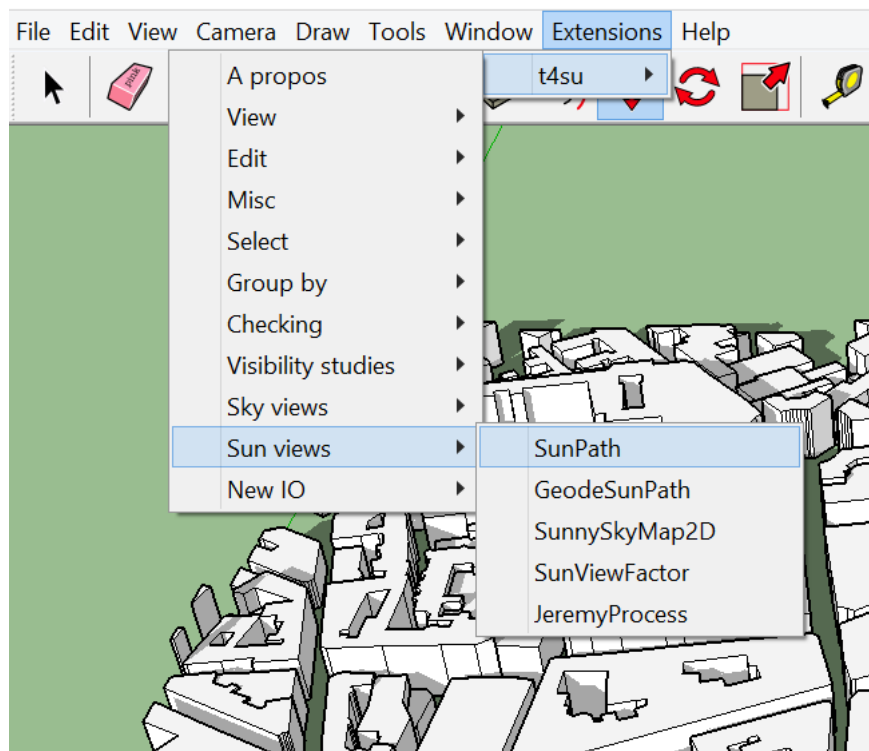
Drawing Sun Paths

Drawing Paths of Points

Tools4SketchUp offers very different possibilities than the built-in Shadow feature or other plugins available.

The first step in your solar analysis should be to set the location of your project, as well as setting the time-period for your analysis. This will be automatically translated into the appropriate Sun Paths.

First, click on **Extensions > t4su > Sun views > SunPath** :



The next window allows you to enter the necessary information for the creation of sun paths :

- Set the correct latitude.

- Set the start date and time
- Set the end date and time

It is sometimes practical to select dates that are of special interest for solar paths such as the solstices : the 21st of June and 21st of December represent the highest and lowest paths of the year, respectively.

- The daily sun path is a continuous line in the sky, which is here represented by a series of points. You can therefore decide on the interval of time (in minutes) at which the position of the sun is saved.
- If your start and end dates are not on the same day, you will have a series of sun paths. If you don't wish to trace a path for every single day, you can change this parameter : setting "2" will trace one path every two days. Setting "30" will trace a path roughly every month.

Set the latitude	47.0
Set the date (DD/MM)	21/09
Set the start time (HH:MM)	00:00
Set the date (DD/MM)	21/09
Set the stop time (HH:MM)	23:59
Set the slice time value (in minutes) per day	5
Set the slice time value (in days) between two consecutive dates	1
Plot?	yes
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

Fig. 7.1: Sun Path Command Box

Once your sun paths have been traced and placed in a Layer, as well as triangulated your geometries, you will be ready to run analyses such as Sun View Factor.

See also:

Triangulating Geometries with Gmsh

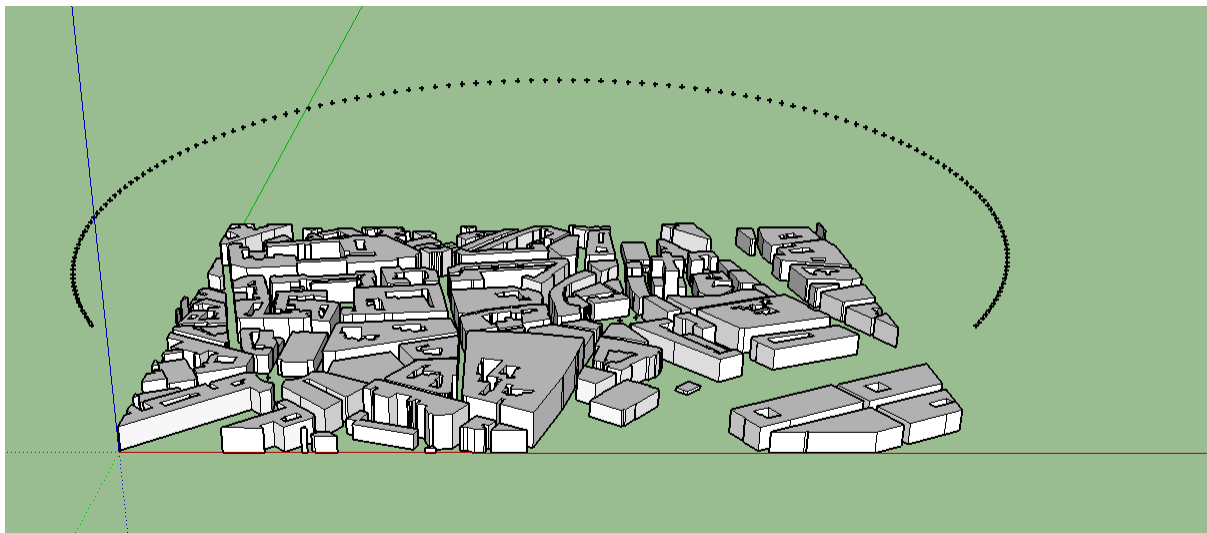


Fig. 7.2: Sun Path Result for the 21st of September

Set the latitude	47.0
Set the date (DD/MM)	21/09
Set the start time (HH:MM)	00:00
Set the date (DD/MM)	21/10
Set the stop time (HH:MM)	23:59
Set the slice time value (in minutes) per day	10
Set the slice time value (in days) between two consecutive dates	2
Plot?	yes
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

Fig. 7.3: Setting Sun Paths from September 21st to October 21st every two days, plotting every 10 minutes.

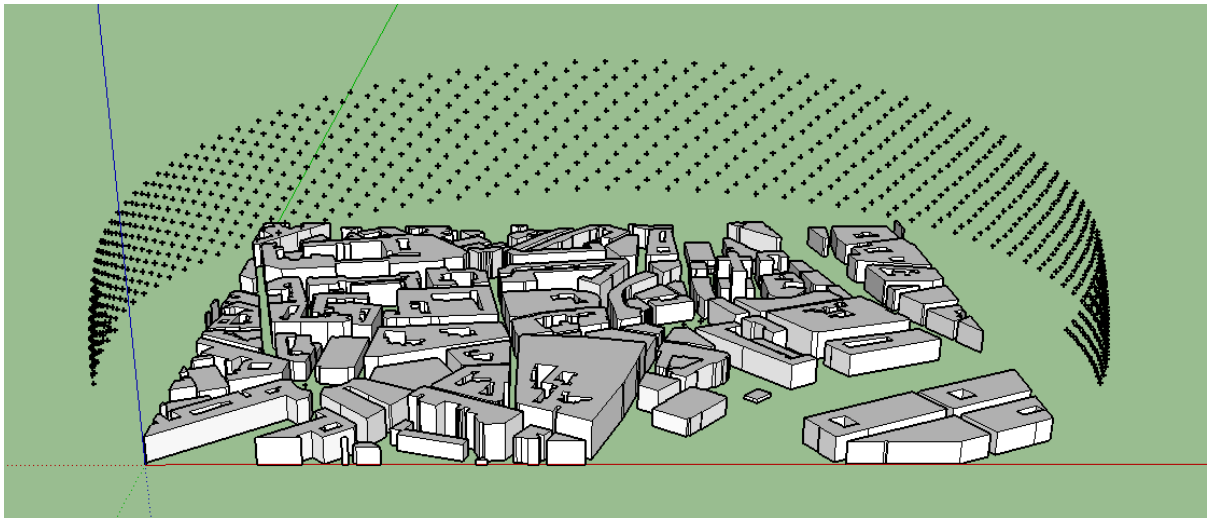


Fig. 7.4: Resulting Solar Paths from September 21st to October 21st.

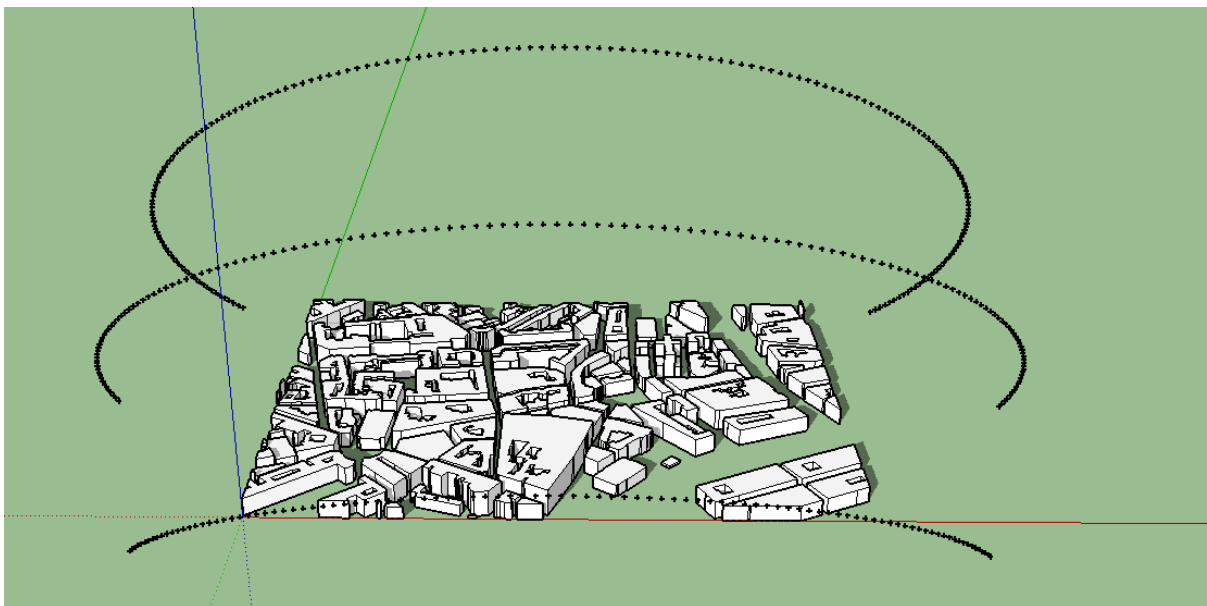


Fig. 7.5: Resulting Solar Paths from the 21st of June to the 21st of December, every 90 days, with 5-minute intervals.

Geode Sun Paths

Once you've drawn your Sun Paths, you may realize your calculation times for Sun View Factors are too much to handle. There is a way to drastically reduce time, but it come at the price of precision : Geode Sun Paths.

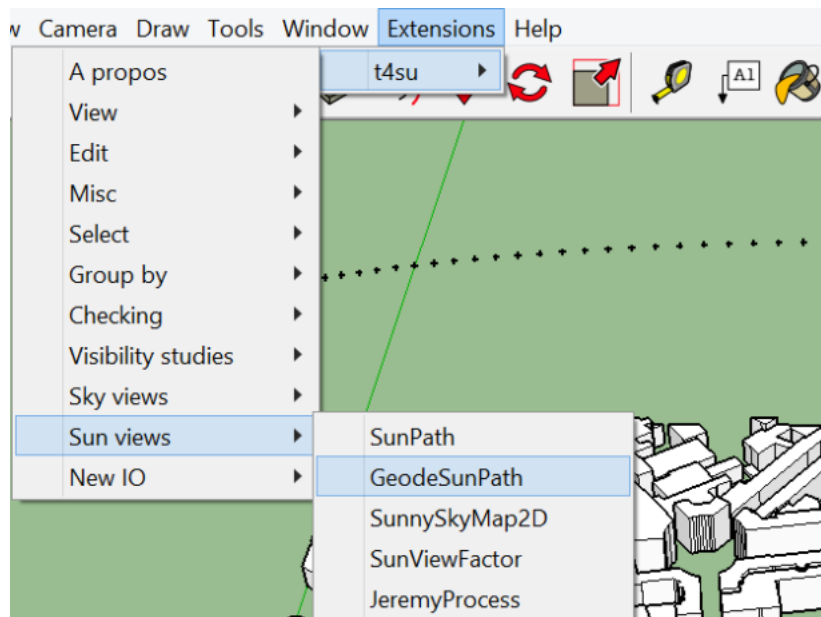


Fig. 7.6: Reaching the Geode Sun Path

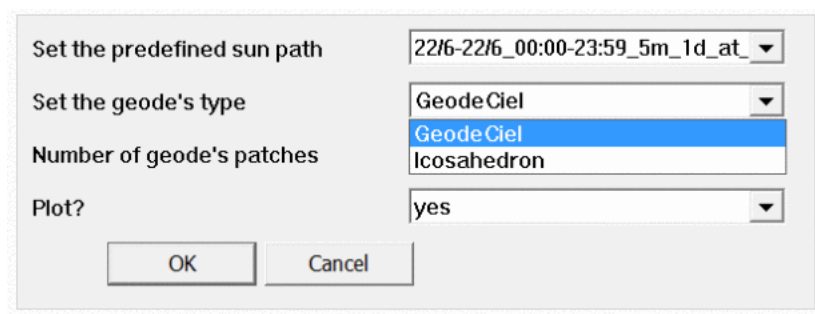


Fig. 7.7: Geode Sun Path Command Box

The “sun points”, representing the location of the sun at a specific time, travel over a sky vault. By intersecting our sun path points with the sky vault, we can find which facets of the sky could replace our sun path. geodsunpath-isoca-resulthigh-nb-patches Geode Sun Path with very high resolution : as each face is allocated to only one point, calculation times may stay similar but precision will be lost



Fig. 7.8: Geode Sun Path : each face covering multiple points, reducing calculation time

You can use the Geode Sun Path for processes such as the Sun View Factor. In that case, the center of each face is taken into account : the bigger the tessellation and the further away their center is from the original sun path, the less precise the model become.

Here is an exaggerated view of the problem (the model has been designed to maximize the discrepancy) :

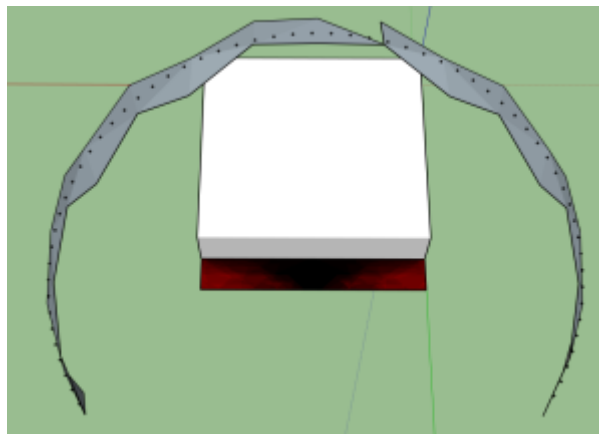


Fig. 7.9: Geode Sun Path result : 83.4 seconds.

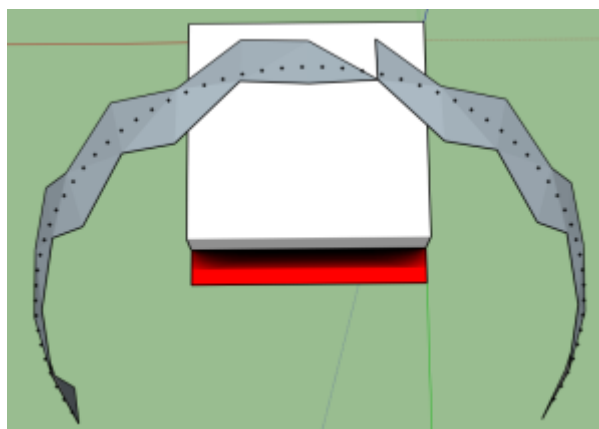


Fig. 7.10: Point Sun Path : 346 seconds

Specific Sun Paths

Now that we've seen how to create sun paths, you may want to study solar radiation during specific time periods.

See also:

Drawing Sun Paths

You can, for example, decide to only plot for specific hours. Here is an example for plotting the sun's position at noon once a month:

The input goes as such :

- Set the start date at 21/06 (summer solstice)
- set your start time at 12:00 (noon)
- set the end date to 21/12 (winter solstice)
- leave the end time at 23:59
- Set the time slice per day to 1440 (24hours)
- Set the time slice between dates to 30.

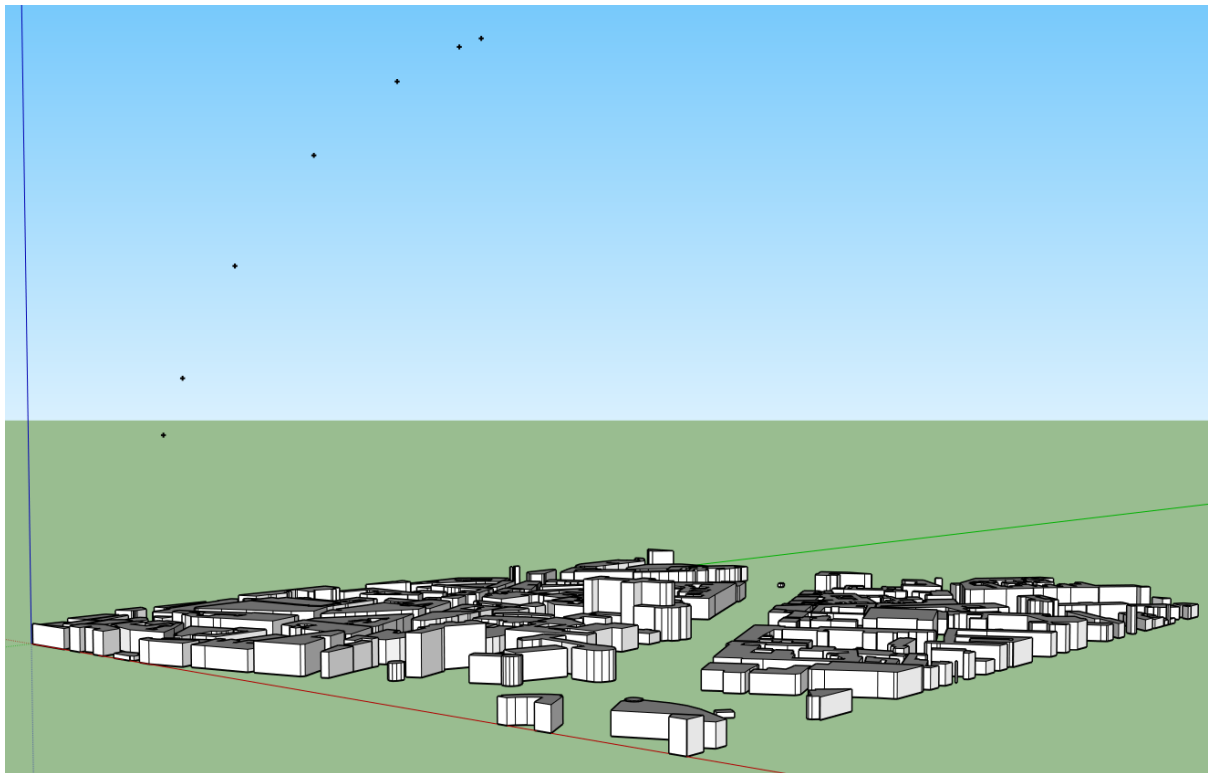


Fig. 7.11: Position of the Sun at Noon every 21st, from June to December at 24.7° latitude.

You may also want to have a more precise idea of the course of the sun through the entire year. Here is the appropriate input :

- Set the start date at 21/06 (summer solstice)
- Set your start time at 00:00
- Set the end date to 21/12 (winter solstice)
- Leave the end time at 23:59
- Set the time slice per day to 10 minutes

- Set the time slice between dates to 30.

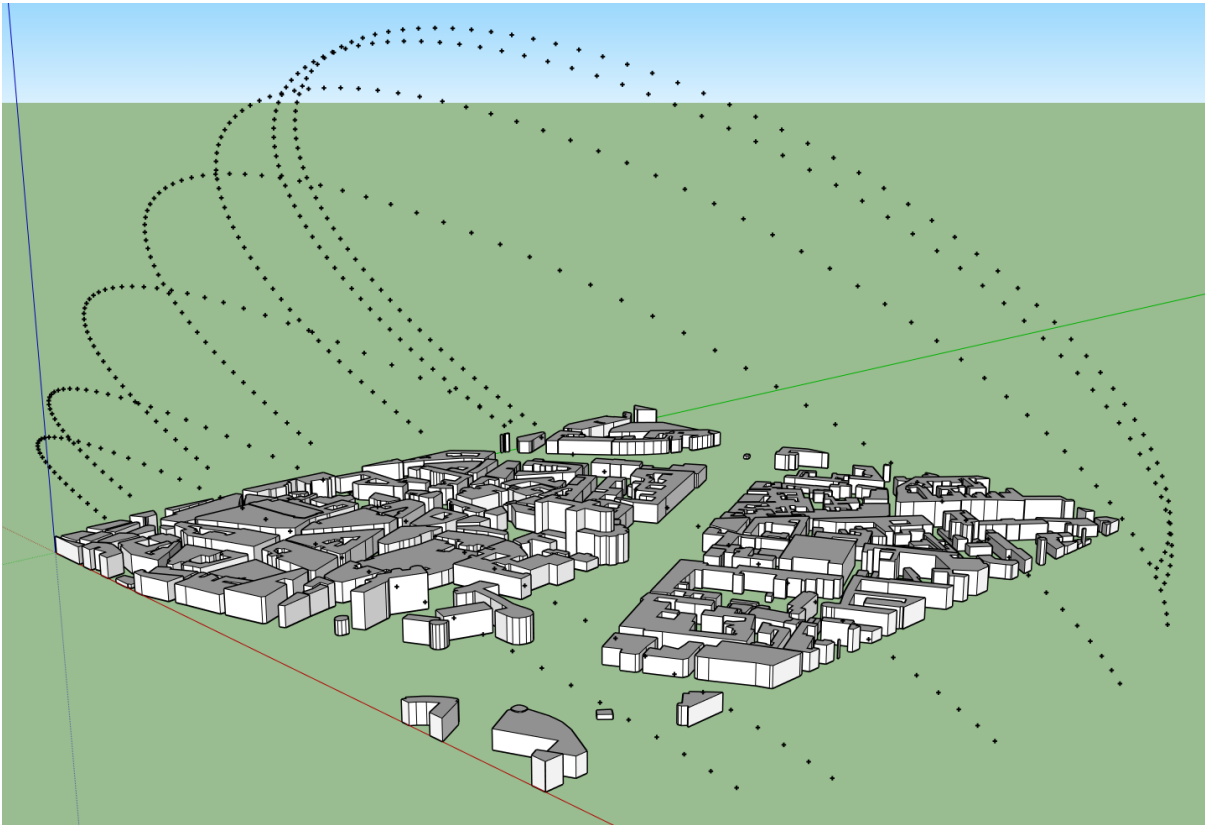


Fig. 7.12: Daily Sun Paths every 21st, from June until December, 10-minute intervals.

Ray Casting and Isovists

The Science Behind : Isovists

A Short History

Although isovists are considered to have been first set in theory by Benedikt in 1979¹ (a first mention of them was made by the same author in 1977²), their first appearance in scientific literature was a paper by Tandy³, itself based on writings from on AC Hardy⁴. This concept sprouted from JJ Gibson's work⁵ on "optic arrays". Benedikt and Burnham⁶ give a good explanation of the difference between optic rays and isovists :

:: Imagine a rectangle (representing a room) and the space within it. Draw a straight line between two points on the rectangle; draw another, and another, until the space of the rectangle is effectively filled with criss-crossing lines between all pairs of points. Now chose an point within the rectangle and you find that a large number of lines pass through that point, one of which is shared with another different set of lines passing through a point nearby. If we consider these lines to represent light rays (i.e., photon streams) of varying wavelength and intensity scattered and bounced by the edges of the rectagle (i.e., the walls of the room), then you have an optic array - a set of rays that pass through the point of interest. But where does the otic ray start and end? Is a ray that happens to pass through our point the same ray that it was before it bounced off the wall? If it is, then almost every ray in the room qualifies as belonging to our optic array, as it would eventually pass through our point. Clearly this isn't going to work. The solution lies in considering a ray only after its "last bounce", that is, in concidering only the set of lines – rays – joining our point to the nearest light-scattering surface in every direction. Now, the isovist is simply this optic array, with wavelength and intensity information omitted.

In other words,

:: To every bearing (omega, phi) from a point of observation x in the world there corresponds a distance l, from the point to the nearest surface such that we have a unique ordered set

Several equations derived from isovists and isovist fields (that is, when caculating the isovist for each point in space), mainly from it's perimeter and area, which an be discretized. We can produce a number of morphological indicators from isovists, including :

¹ ML Benedikt, 1979, "To Take Hold of Space : Isovists and Isovist Fields", Environment and Planning B, vol 6, pp. 46 - 65

² ML Benedikt, 1977, "Path-dependence and position-dependence in isovist fields", research report for the Council of Advanced Transportation Studies, University of Texas Austin, Austin, Texas.

³ Tandy, 1966

⁴ University of Newcastle, unpublished

⁵ Lynch 1976

⁶ Gallagher, 1972

- Mean Depth
- Compactness fields
- Occlusivity, occlusivity fields
- Visible Perimeter Fields

Isovist Moments can be derived into three types :

M1 represents the derivation from the mean of the perimeter's distance to x. M2 represented the derivation from the variance of the perimeter's distance to x. M3 represents the derivation from the skewness of the perimeter's distance to x. This last formulae can be used to find areas which can see well but

Ray Casting : an Introduction to Visibility Studies

Ray Casting involves choosing a point in your urban space, from which rays will spread at equal distance from each other. The ray stops when it hits either an obstacle or its set limit. It is the simplest and quickest way to assess visibility, but doesn't give as much information as isovists or hedgehog. First off, click on **Extensions > t4su > Visibility studies > RayCasting**. A command box will appear, giving you the following possibilities:

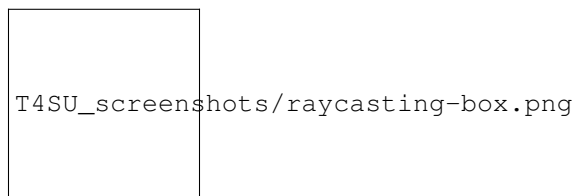


Fig. 8.1: Ray Casting Command Box

- Choose the number of rays. This will affect the results' precision, but taking the maximum isn't necessarily better : if you select a small visible range, you may not need too many rays.
- The visual range is the maximum length of the rays ; any obstacle after said range will not be picked up during the process
- Set the height (z0 value). If you're using ray casting for visibility purposes, keep a z-value around eye-level.
- Make sure the units you are working with are correct.
- Finally, you can choose to cast rays in a 2D or 3D space. This is entirely up to the type of information you want to collect. If you are only interested in street visibility and do not need information on the impact of building heights for example, a 2D Ray Casting is much more legible and should suffice.

Once you have click **OK**, you can click anywhere on your map to cast your rays, as many times as you need (you do not need to repeat the procedure to plot a second RayCast).

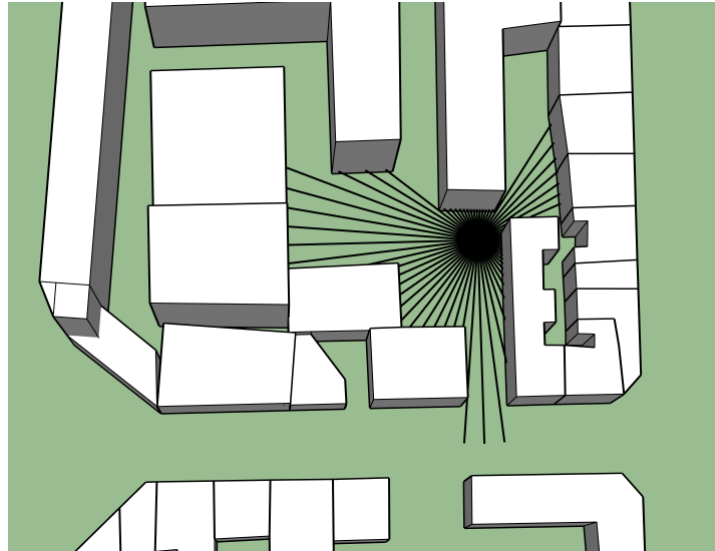


Fig. 8.2: 2D Ray Casting with 256 rays at a distance of 40m

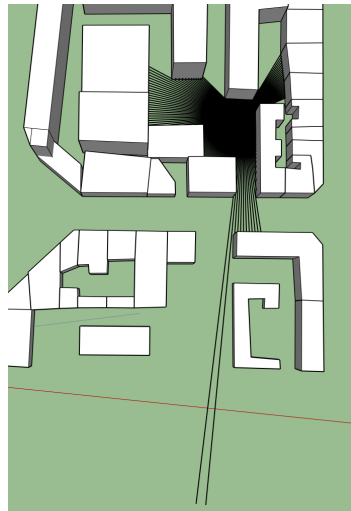


Fig. 8.3: 2D Ray Casting with 1024 rays at a distance of 150m

If you are not interested in the rays themselves, but rather the point of impact they share with the facades of your buildings, check out [CloudOfPoints](#).

What is an isovist ?

Isovists represent the entirety of the visible plain (in 2D) or volume (in 3D) at a specific point in space, bounded by the built environment. Because isovists are built upon a point, the associated form changes as it is displaced, just as our notion of urban space changes as we walk through it. They are frequently used in many fields of visibility studies : wireless network design, landscape management and analysis, pedestrian access or security.



Fig. 8.4: Isovist at a Point Close to the Cathedral.

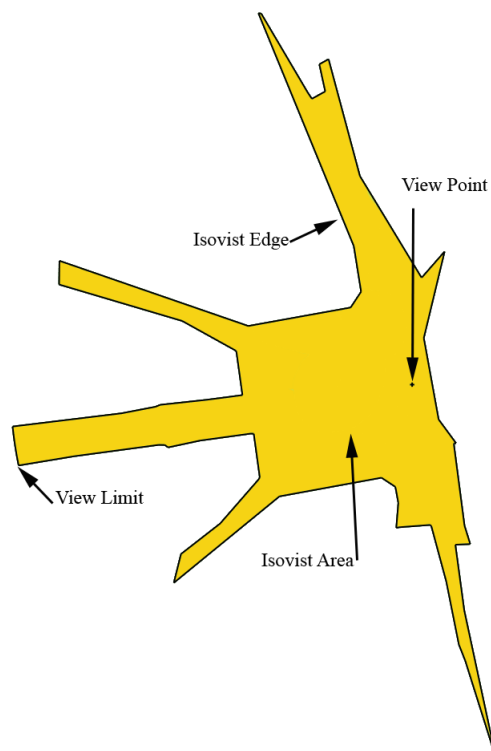
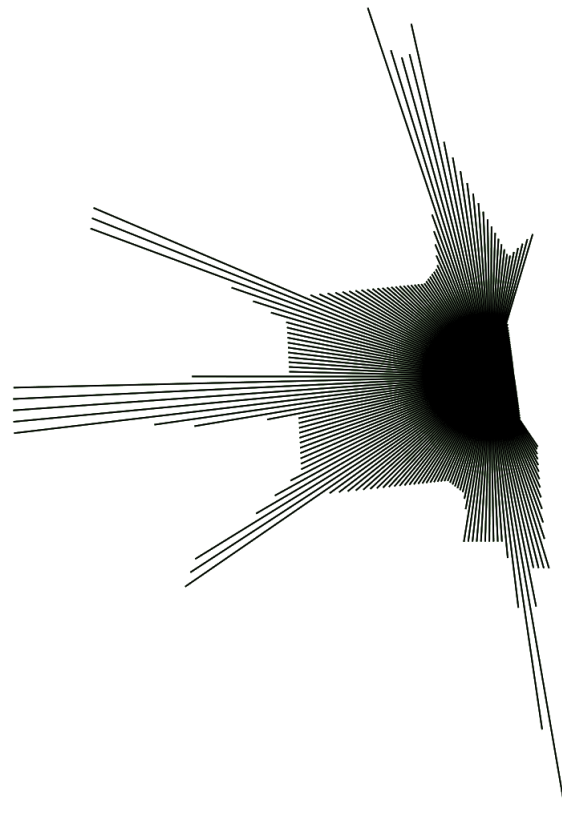


Fig. 8.5: Isovist at an Other Point in Space.

So how are isovists built? Originating from our point of origin (**view point**), rays are cast into the open space, up to a certain arbitrary distance (**view limit**). The resulting points of impact between each ray and its barriers (either obstacles or the view limit) are joined to form the **isovist's edge**. The rays are then replaced by an **isovist area**.

See also:

Ray Casting : an Introduction to Visibility Studies



Isovists can also be used to study a space's [convexity](#), that is, a region for which every pair of points in said region can be linked by a straight line also within the region. Here is a simple example of convex and non-convex spaces:

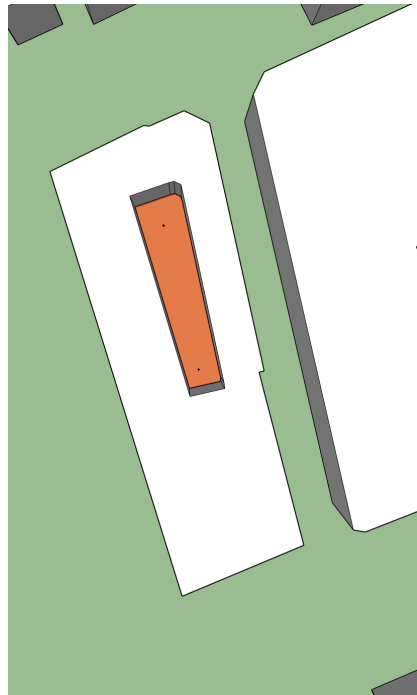


Fig. 8.6: In a Convex Space, Isovists have the Same Shape no Matter the Location of the View Point

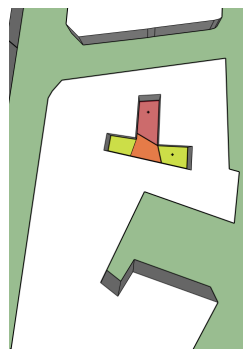


Fig. 8.7: In a Non-Convex (Concave) Space, more than one Isovist is Needed to Map the Totality of the Area.

We can notice the link between ray casting and isovist formation when playing with its precision.

See also:

Running Times and Geometrical Precision

Creating Partial 2D Isovists

Partial isovists are isovists that are constrained by a certain aperture. You can automate their process using Batch-Process, or set your positions and orientation manually by clicking **Extensions > t4su > Visibility Studies > PIsovist 2D**. If you're using the manual method, you will be redirected to the **Partial Isovist Command Box**.

First, import a linear geometry (ie: made of lines or multilines) into your Layers, or draw one yourself using the Line Tool.

See also:

Loading your data into SketchUp



Next, you'll want to sample your pathway, then use a Batch Process.

See also:

Sampling A Path

Batch Processing

Select **PIsovist2D** (which stands for Partial 2D Isovist) in the **Process to apply** and select the correct layer (starting with SamplePathway[...] if you've created your points through sampling). Finally, select **Sketchup::ConstructionPoint** as Geometry type.

If don't want to use a Batch Process or would rather select the points manually, you can do so too. Simply **click to set the point of origin and drag to set the orientation** of your partial isovists.

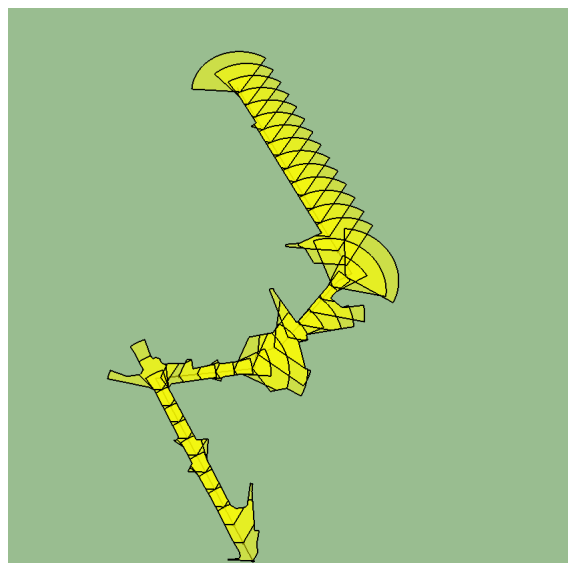
Whether you're plotting manually or automatically, you will have to fill in this command box :

Aperture half-angle (in degrees)	30.0
Number of rays	256
Set ray length	590.0
Set z0 value	1.6
Select length unit	m
Set the transparency value	0.5
Select color	Yellow
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

- Select the aperture of your circular sectors (ie: how wide you wish the isovist to be)
- Select the number of rays : a higher number will give a more precise outline, but a lower number will greatly accelerate computation time. For example, setting the aperture to 30 and the number of rays to 60 will result in a ray every 0.5 degrees.
- Select ray length. Some trial-and-error may be needed to adjust this parameter : too short, they may not cover enough ground. Too long will result in confusing overlapping geometries and longer computation times.

- Setting the z0 value allows you to decide at which height the partial isovists are calculated. By default, they are at 1m60, an average person's eye-height.
- You may also change the appearance of the isovists : transparency (from 0 to 1, 1 being the most opaque) as well as the color.
- When you are ready, click **OK**

What do we see ?



The partial isovists are constrained by the extruded building geometries. Their profile mimics our visibility as we advance along the path we have decided upon. We can notice the gradual enlargement of the isovist as we advance towards an plaza or an intersection.

3D Isovists and Hedgehogs

Like 2D Isovists, 3D Isovists and hedgehogs use ray casting to map the surrounding urban space. Unlike its 2D counterpart though, 3D Isovists will only output assembled rays that effectively touch a surface.

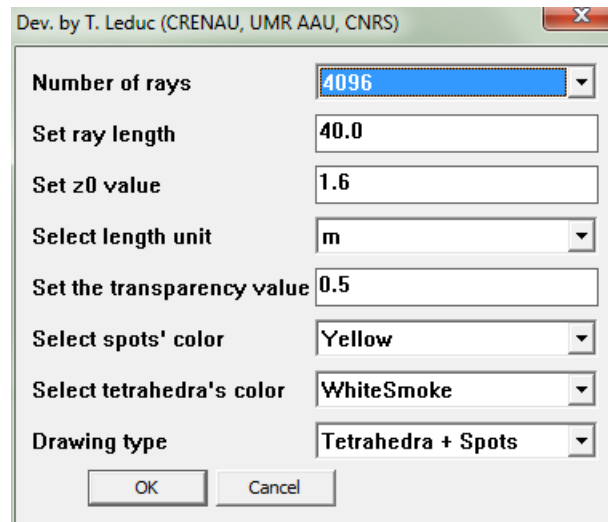


Fig. 8.8: Isovist Command Box

When creating these 3D objects, you will have to decide whether to draw “tetrahedra or spots” (or both). By choosing tetrahedra, the cast rays are drawn, set in a group and shown. Spots on the other hand, draw out the surfaces visible from our point of interest (i.e., the faces’ areas touched by the tetrahedra).

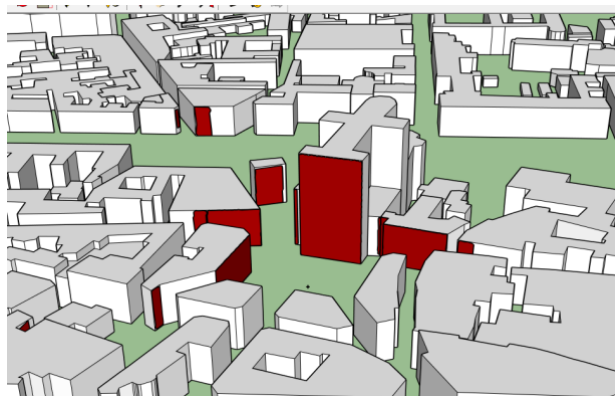


Fig. 8.9: 3D Isovist, Spots Only

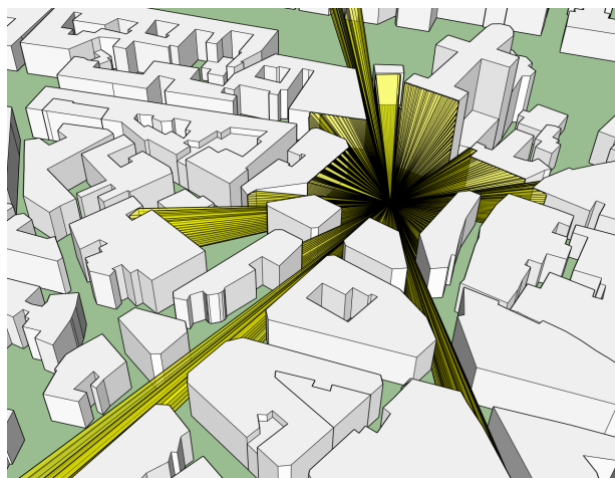


Fig. 8.10: 3D Isovist, Spots and Tetrahedra

The surfaces are created by a minimum of three points, meaning if only two rays reach a surface, that surface will not be taken into account in the Isovist/Hedgehog rendering. You will also get the following error:

The difference between a hedgehog and a 3D isovist lies in whether or not these batches of three rays form individual tetrahedra/spots, or are assembled into one. Here's an illustration of a Hedgehog result: compare it with the Isovist result above (resolution was slightly lowered).

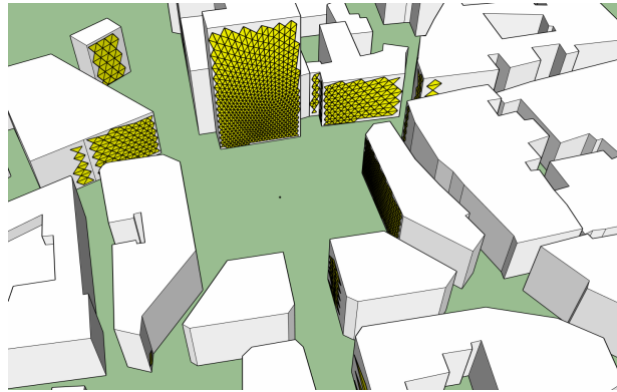


Fig. 8.11: Hedgehog Result, Spots Only.

As with many other features, it is possible to run a batchprocess of 3D isovists. In this last example, the cathedral building wall faces were selected and sampled into points. The following result allows us to know which building facades have a potential view of any side of the cathedral.



Fig. 8.12: Building Facades with Direct View of the Cathedral

Sky Maps are exactly what they're named after : maps of the visible sky. Sky Maps work in a similar fashion as Ray Casting : as rays are sent from a point of origin, they either reach the sky vault or they don't : Sky maps only take into account the rays that do reach the sky. They can be projected onto a dome, or reprojected onto a disc for a 3D or 2 visualization.

3D Sky Maps

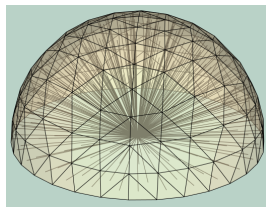
See also:

3D Sky Maps and Building Facades

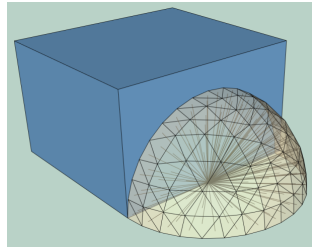
Ray Casting : an Introduction to Visibility Studies

What's a 3D Sky Map ?

Let's imagine for starters that we're at a point on a completely flat surface, without anything blocking our view. Our result would be something like this :



Of course in an urban setting, some of our rays would be blocked, and the associated dome faces would not appear :



How do I create 3D Sky Maps?

You can place 3D Isovists through the point-and-click method by clicking **t4su > Sky views > 3D Sky Map**, or through a BatchProcess if you already have your locations set out.

See also:

Batch Processing

What Can I Use Them For?

On their own and at a first glance, these domes help you assess the openness of an area, the local skyline and roughly estimate the sky view factor. But they hold much more interesting information when used, for example, to assess solar radiation. As we've seen, we have methods to calculate the Sun View Factor after having drawn Sun Paths. What happens when we reproduce this procedure on our 3D Sky Maps? To try this out, first create your domes, then click **t4su > Edit > AutomaticReverseFaces** because half of your dome's faces are facing inwards. Then proceed to draw your Sun Paths and calculate your Sun View Factor over the layer containing your 3D Sky Maps. Finally, use ColorFaces to depict the differences in energy received by each face:

See also:

Sun View Factors

Drawing Sun Paths

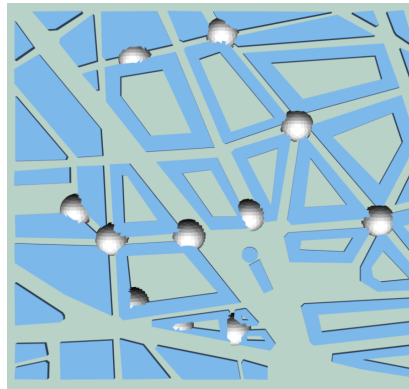
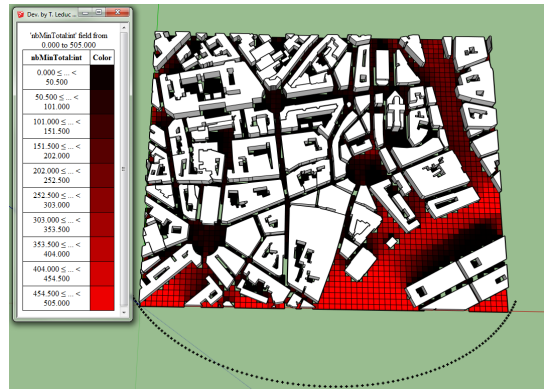


Fig. 9.1: 3D Sky Maps Created by Point-And-Click, Colored by Direct Solar Irradiation Value

We can see a lot more information at particular points in space than when doing a regular 'SunViewFactor over a tessellated area:



We can see what parts of the sky contribute most (and least) to our direct solar irradiation. Naturally, North-facing faces receive the least sunlight, and Southern faces at a roughly 45° vertical angle receive the most. We can thus find which sections of the sky contribute most to the points' solar energy gains.

3D Sky Maps and Building Facades

3D Sky Maps and Building Facades both start from a half-dome, at the center of which is the point of interest. Rays start from the center point, pass through the center of each dome face and continue on, either hitting a facade or not.

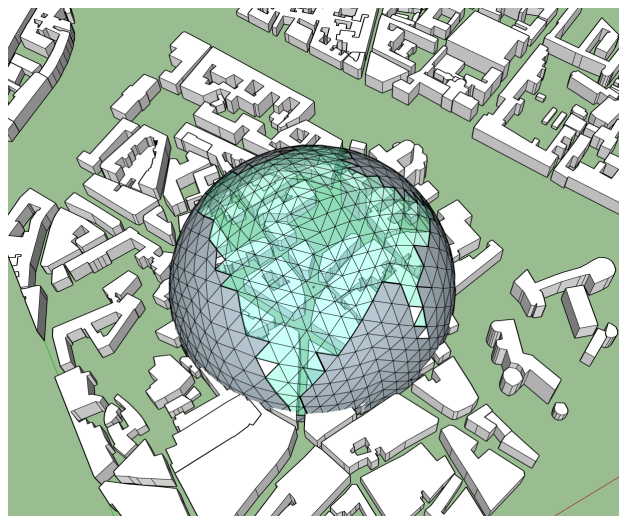


Fig. 9.2: A combination of SkyMap 3d (clear blue) and BuildingFacade (grey) at the same point in space

See also:

Running Times and Geometrical Precision

Whilst Building Facades will only show the dome faces who's rays have hit an obstacle, 3D Sky Maps will only show those that have not : They show complementary information. So when is one or the other more appropriate?

Sky maps are much more useful in showing which parts of the sky are visible at a certain point. They hold an advantage over 2D Sky Maps, in that the sky is not projected over a 2D surface: they are not deformed by any necessary spherical projection.

See also:

Calculating the Sky View Factor

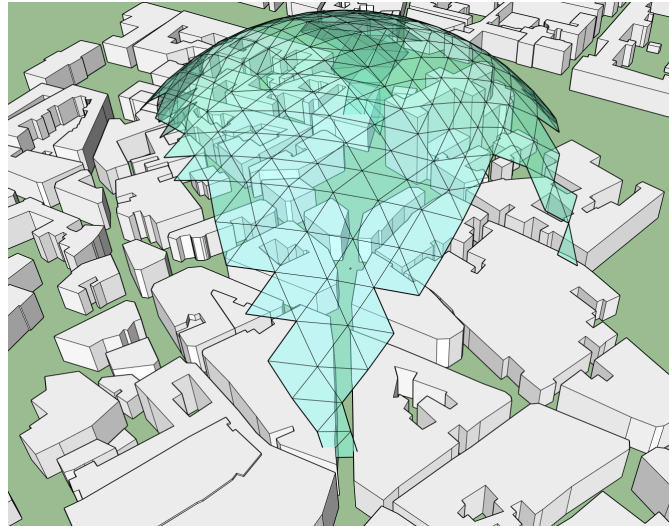
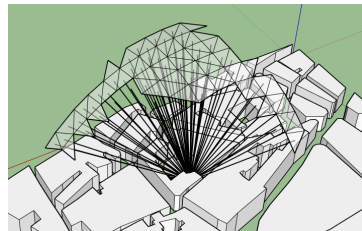


Fig. 9.3: Sky-Map 3D over an intersection



The BuildingFacade module works great for street-level visibility : it highlights the skyline, the openings in the urban space represented by “dips” in the dome formation, allowing you to assess the relative confinement of urban space.

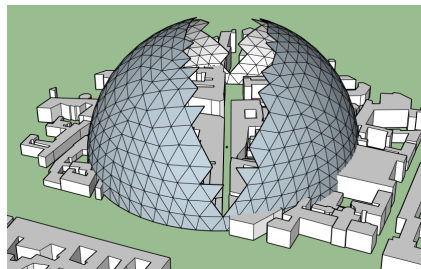
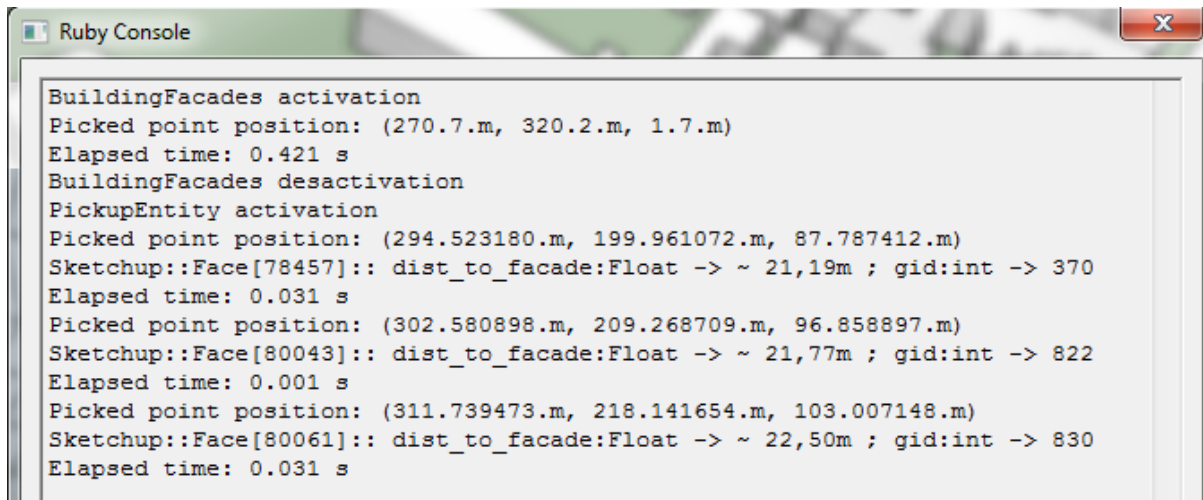


Fig. 9.4: Typical BuildingFacade profile for a narrow street

Once built, the dome faces also hold an attribute, “dist_to_facade”(float). This feature expresses the distance between the point of interest and the obstacle represented by the dome face.



```

BuildingFacades activation
Picked point position: (270.7.m, 320.2.m, 1.7.m)
Elapsed time: 0.421 s
BuildingFacades deactivation
PickupEntity activation
Picked point position: (294.523180.m, 199.961072.m, 87.787412.m)
Sketchup::Face[78457]:: dist_to_facade:Float -> ~ 21,19m ; gid:int -> 370
Elapsed time: 0.031 s
Picked point position: (302.580898.m, 209.268709.m, 96.858897.m)
Sketchup::Face[80043]:: dist_to_facade:Float -> ~ 21,77m ; gid:int -> 822
Elapsed time: 0.001 s
Picked point position: (311.739473.m, 218.141654.m, 103.007148.m)
Sketchup::Face[80061]:: dist_to_facade:Float -> ~ 22,50m ; gid:int -> 830
Elapsed time: 0.031 s

```

Fig. 9.5: Ruby console showing attributes when using PickupEntity

You can view it by clicking **View > ColorFaces** and choosing the `dist_to_facade` attribute.

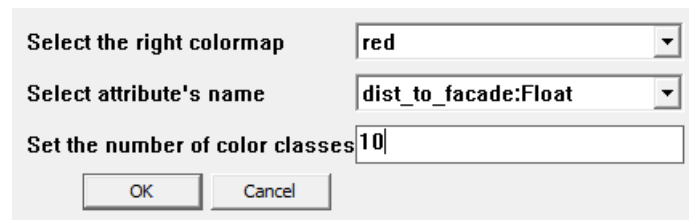


Fig. 9.6: ColorFaces Command Box

Originally, the faces all turn inwards: the results may be more visible if the faces were reverted. This can be easily achieved by selecting **Edit > AutomaticReverseFaces**, and selecting your BuildingFacade layer. Sometimes, intersecting faces are not automatically reversed. For an optimal result, render your building layer invisible before reversing your BuildingFacade layer.

2D Sky Maps and Vegetation Modelization

As we have seen, we can easily cover large areas by multiplying a single object, such as a person to make a crowd or a tree to make a forest. There are more than [1400 trees to choose from](#) in the 3D Warehouse, you really have a large choice. So are all SketchUp trees the same? What types are there and how do they affect the results of our visibility studies? Let's take two concrete examples.

First Tree : the FaceMe Tree

If you're familiar with Sketchup, you may know about the [faceMe](#) technique. Basically, a 2D Object is set to automatically turn to always face your camera, no matter which way you look at it (unless you look from above).



Second Tree : the Carboard Cutout

The Cardboard Cutout also uses the faceMe method for it's vertical face, but also has an added horizontal face.



Comparison :

Let's now compare the impact these two different trees would have on our visibility studies. For starters, are the sky maps the same? If they're not, what else would that change? To find out, import both types of trees into different layers, and place them at identical spots. To make sure we compare the same points in space, we can create a path, sample it and project the sampling points over our terrain. We can then launch a 2D Sky Map Batchprocess over our sampled path twice ; once for each tree layer type.

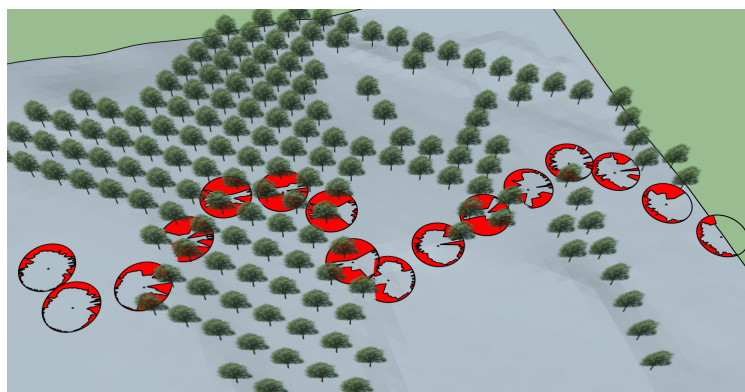


Fig. 9.7: Sky view Factor with Tree 1.

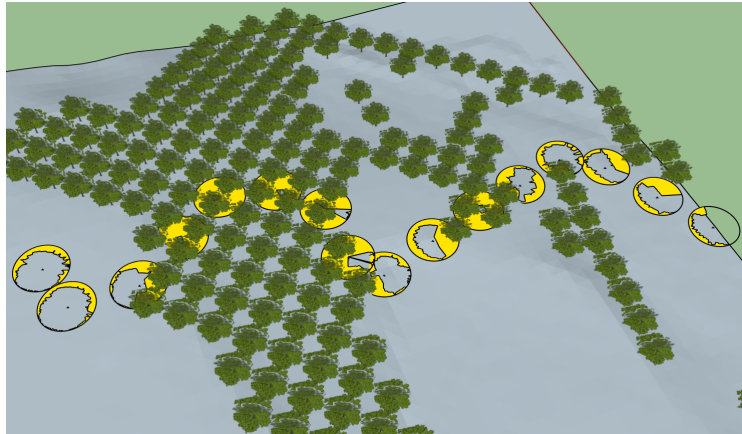


Fig. 9.8: Sky view Factor with Tree 2.



Fig. 9.9: Comparison of Both Sky View Factors.

Clearly, trees composed of only a vertical face allow for a much greater sky view factor. In this particular case, a greater sky view also means greater direct solar radiation.

See also:

Sampling A Path

Batch Processing

Conclusion:

Choosing the correct type of tree is very important : they're part of every urban setting and need to be taken into account when analysing urban energy balances. Vegetation's correct integration in such simulations is still at its premise, mainly because their modelization induces calculating variable opacity over many complex shapes. Comparatively, this SketchUp method may not be as precise as other microclimatic models, but it's definitely much faster. In any case, it's vital to know what type of geometries we're working with and anticipate the types of problems that could be encountered, whether it be a 2D, 2D+ or 3D tree model.

Sunny Sky Maps

We can project given sun paths onto our 2d sky map. This allows us to quickly understand buildings' impact on solar availability at a point in space, over different periods of time.

First, create the sun paths you wish to have on your sky view graph.

See also:

Drawing Sun Paths

In this example, we will be using the sun paths of Winter and Summer solstices, as well as an equinox, at a latitude of 47° . This allows us to see the range over an entire year.

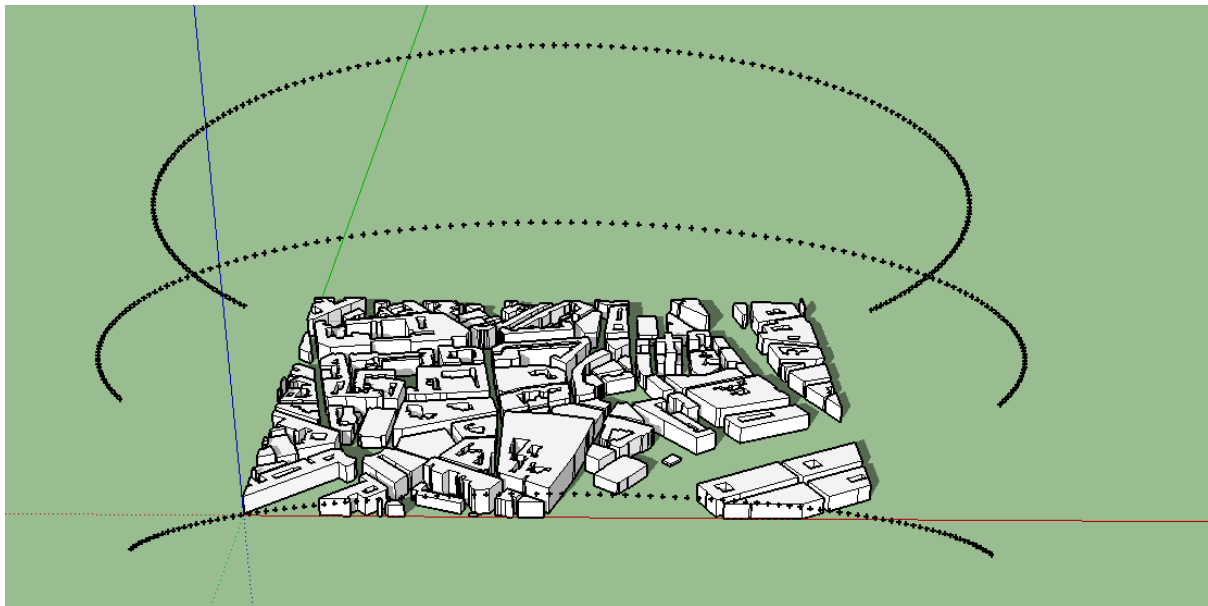


Fig. 9.10: Sun Paths during the Solstices in Nantes, France.

Next, click on **Extensions > Sun views > SunnySkyMap2D** :

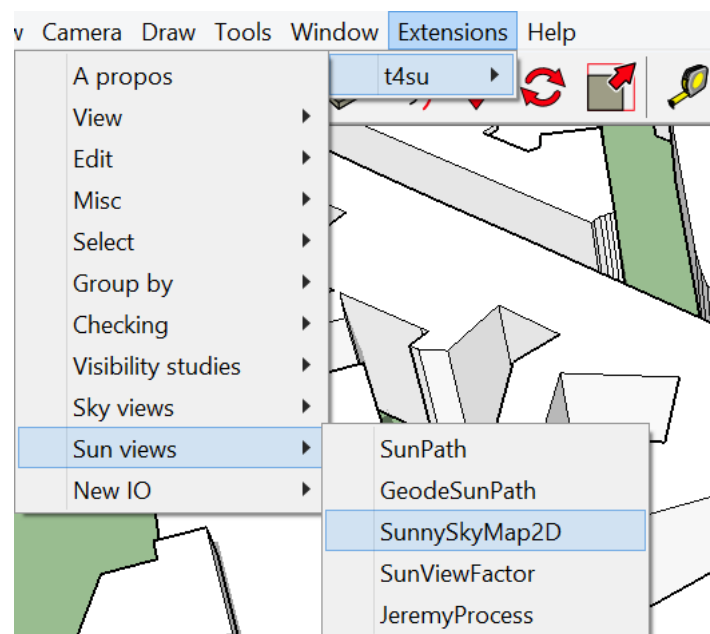


Fig. 9.11: Reaching SunnySkyMap

In the following command box, first select the layer containing your sun path:

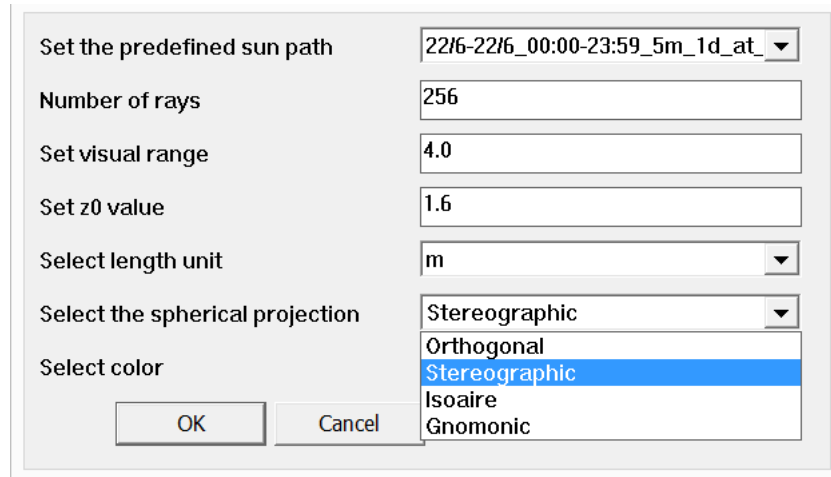


Fig. 9.12: SunnySkyMap Command Box

- Enter the number of rays used for the calculation, ie: its precision in finding the contours of the surrounding buildings
- Set the visual range : this is the size of your graph. This depends your project and your personal preferences : the amount of space you have available to plot the graph, how far out you would like to zoom and still read it...
- Select the height at which you would like to calculate your sky view factor (z0). If you would like to visualize projections at ground floor, 0.0 or 0.1 are an acceptable inputs.
- Select your unit of length. Be sure to always keep the same units throughout the entirety of the project!
- Next, select the type of projection you would like to use. If you're not sure which one fits your needs best, check [this page](#) for more ample information.
- Finally, you can change the color of the building projections.

Once you hit **OK**, you can start clicking anywhere on your project to insert an graph at that location.

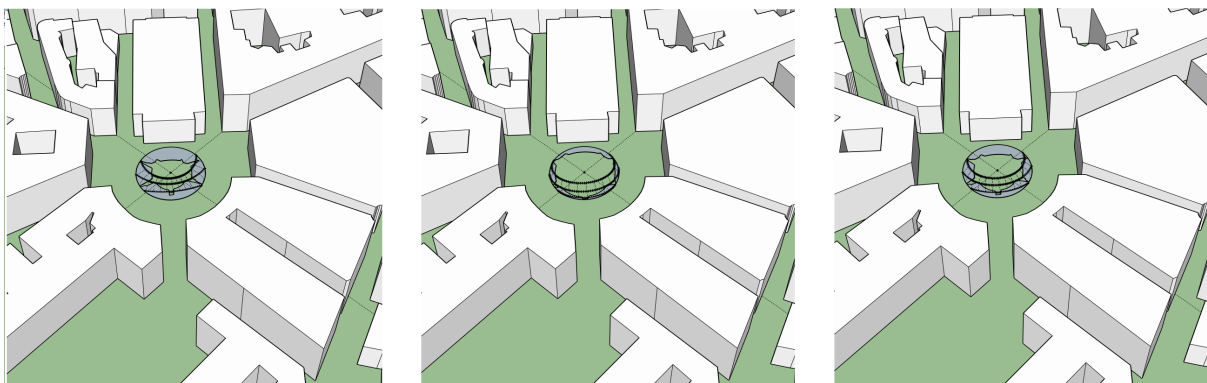


Fig. 9.13: From left to right : Stereographic, Orthogonal and Isoaire (Equal-Area) projections at the same location.

Reading such graphs are relatively simple. The disc represents a 360-view of the sky: the ground floor is projected at the circumference, whereas the center of the disk is the point in the sky directly above it (what happens in between depends on your type of projection). The filled-in parts of the graph represent parts of the sky that are blocked off by obstacles.

If a point of the Sun Path is contained in the filled in areas of the graph, it means that at that specific time and place the point you are studying is in the shade. Conversely, a sun path point that is not superimposed over a filled in area of the graph means there is direct sunlight at that specific time and place.

The Science Behind : Sky View Factors

A Short History

One of the earliest SVF calculations were made with the help of digital cameras mounted with Fisheye lens. The masks were then delineated manually and the SVF is calculated with Steyn's method¹. Other raster-based calculations were later developed, using pixel counting methods^{2,3}. To overcome this somewhat manual and time-consuming method⁴

See also:

The Science Behind : Solar Radiation

Sky View Factor and Radiation

A Sky View Factor (SVFs) represents the ratio at a point in space between the visible sky and a hemisphere centered over the analyzed location (Oke 1981) :

At a point where the SVF = 0, the entire sky is blocked from view by obstacles. If there is an obstacle blocking parts of the sky view, the SVF will be greater than 0.

- for SVF = 0 :
 - there is no short-wave reflection
 - there is no long-wave nocturnal interference

According to Oke⁵, cited by many papers including Hammerle et al.⁶, the energy budget (in terms of flux) can in this case be considered *ideal*

¹ Steyn, D. G. (1980). The calculation of view factors from fisheyelens photographs: Research note. Atmosphere-Ocean, 18, 254–258. <https://doi.org/10.1080/07055900.1980.9649091>

² MMatzarakis, A., Rutz, F., & Mayer, H. (2007). Modelling radiation fluxes in simple and complex environments: Basics of the RayMan model. International Journal of Biometeorology, 51, 323–334. <https://doi.org/10.1007/s00484-006-0061-8>

³ Rzepa & Gromk 2006, Gal et al 2007

⁴ Hämmerle, Gal, Unger, Matzarakis 2001,

⁵ Oke, T. R. (1982). The energetic basis of the urban heat island. Quarterly Journal of the Royal Meteorological Society, 108(455), 1–24. <https://doi.org/10.1002/qj.49710845502>

⁶ Hämmerle, Gal, Unger, Matzarakis, 2011, Comparison of Models Calculating the Sky View Factor used for Urban Climate Investigations, Theoretical and Applied Climatology, Oct. 2011, Vol. 105, n. 3, pp. 521-527

Where Q^* is the total energy balance, Q_G is the energy stored in soil and/or buildings, Q_H is the sensible heat (air temperature) and Q_E is the latent heat (evaporated water).

- for $SVF > 0$:
 - incoming day-time short-wave reflection increases during the day
 - outgoing night-time long-wave radiation is reduced
 - incoming night-time long-wave radiation is increased
 - altered soil heat flux

Note: For more information on the different types of radiation, see *Night-time Radiation*

Because of its influence on energy balances, the SVF holds an importance in various fields of climatology (urban⁷, forest⁸, human biometeorology⁹ ...) but its most widespread use concerns the heat island phenomena.

See also:

The Science Behind : Solar Radiation The Science Behind : Isovisits

Sky View Factor and Heat Island Phenomena

One of the earliest studies demonstrating the link between sky view factor and local urban temperatures was made by Yamashita et al.¹⁰ (1985).

View Factors reduce the information given by sky maps and sunny sky maps to a single numerical value, allowing for a more traditional cartography of a discretized space. Sky View Factors represent the ratio of visible sky at a point, which we could find the Sky View Factor alternatively by attributing to a Sky Map Disc's center the ratio between the disc's area and the area representing visual obstacles. Similarly, Sun View Factors can be found by comparing the number of points of a sun path not blocked by obstacles with the total amount of points in the sun path.

Calculating the Sky View Factor

The Sky View Factor (SVF) is used in the evaluation of the impact of urban geometry on the micro-climate, specifically temperature and the Urban Heat Island phenomenon. Sky View Factors help in the comprehension of an area's potential direct sunlight in a given amount of time. The SVF indicates the ratio between :

- the radiation received / emitted by a surface from/to the sky

and

- the theoretical total hemispheric radiating environment.

Direct "visibility" between the sky and a plane means radiation can escape the urban setting into the atmosphere, whereas radiation between said plane and another plane keeps radiation trapped. Higher SVFs therefore indicate spaces that would theoretically cool down (emit more radiation towards the sky) than a space with low SVF.

Let's see how we assess the Sky View Factor in an urban setting. First off, follow the instructions to create a grid layout of the urban space. For this example, we're using a 10x10 grid, with $z=0.1m$.

Next, select the Sky View Factor in the BatchProcessing options.

See also:

⁷ Watson, I. D., & Johnson, G. T. (1987). Graphical estimation of sky view-factors in urban environments. *Journal of Climatology*, 7(2), 193–197. <https://doi.org/10.1002/joc.3370070210>

⁸ Holmer et al., 2001

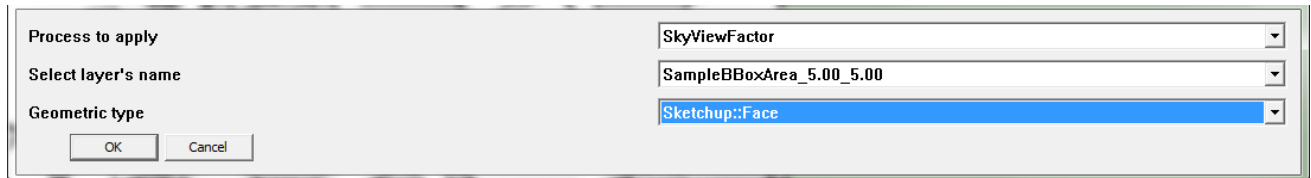
⁹ Matzarakis, 2001

¹⁰ Yamashita, 1985, On Relationships Between Heat Island and Sky View Factor in the Cities of Tama River Basin, Japan. *Atmospheric Environment* Vol. 20, n°4, pp. 681-685 1986 Pergamon Press Ltd.

*Sampling the Bounding box**Batch Processing*

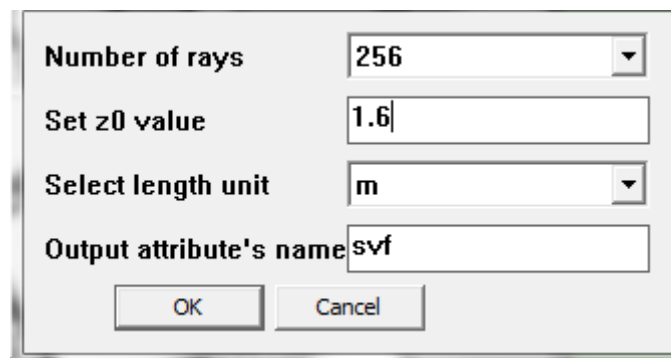
Fill in the next two command boxes accordingly. In the first box:

- Select SkyViewFactor as process
- Select the layer containing your sampled bounding box
- Select “Sketchup:Face”



The second command allows you to control the SVF parameters:

- Select the number of rays used to calculate the SVF, ie: its precision. Be careful though, this process is repeated for each sample and will impact your computing time dramatically
- Set the z0 value to 0.1, meaning we will check the SVF at ground level.
- Select your unit length
- Finally, select the name under which the values are stored. We recommend you keep the default name “svf” to keep from any confusion.



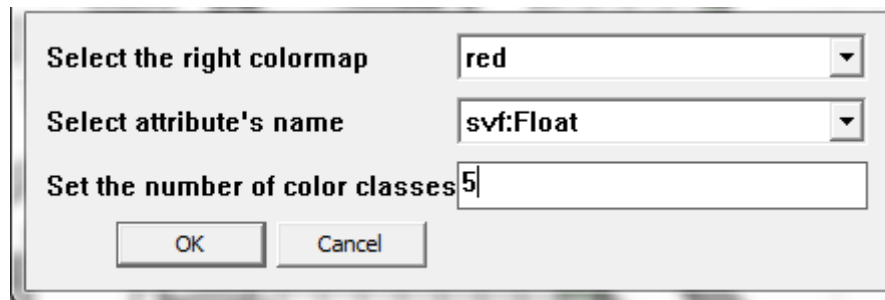
Once that is done, you can check your new feature by selecting **Extensions > View > PickUpEntity** to view individual squares, or by going to **Extensions > View > PrintAttributeValues** ** and by selecting ****svf:Float** in the following command box to show the values of every sample.

But there is a much better way of viewing and presenting the data, much like what we’ve done when viewing building heights.

See also:

visualizing-building-heights

Select **Extensions > t4su > View > ColorFaces**.



- In the next command box, select the color range you wish to use.
- In the drop-down menu next to “select attribute’s name”, find and select “svf:Float”
- Select the number of classes you want for your map.
- Hit **OK**

Your grid should be colored to resemble something like this :



Fig. 10.1: Sky View Factor over Cathedral Sector, Nantes

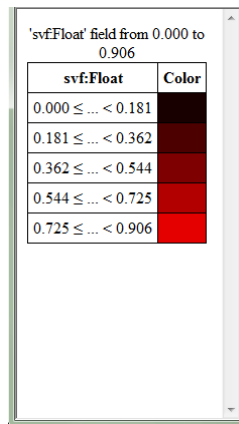


Fig. 10.2: Associated Legend of the SVF Map.

Unless you've taken a color range with a "-" in front, lighter values express areas with a high SVF, meaning areas which "view" much of the sky. Darker areas, mostly in small streets and inside building blocks, do not "see" much sky directly : most of the radiation is captured and recaptured by adjacent walls.

The Science Behind : Solar Radiation

Initial Definitions :

Solar Energy is the amount of energy sent by the sun, meaning the maximum energy received by the Earth, without taking into account climatic conditions (clouds), date or time of day which impact the angle of the surface relative to the sun.

Solar Irradiance (or insolation) is the energy received by a given area of the Earth's surface. It's measured in Watts per square meter. As the angle between Sun and surfaces' normal grows, the solar energy is spread over a larger surface, reducing solar radiance. Solar Irradiation is Solar Irradiance intergrated over time.

Solar Angle and Irradiance

Both place and time come into play when calculating the the solar angle. Place concerns where we are : latitude and longitude :

There are two conjoined earthly movement to be taken into account. - The rotation of the Earth on itself :

- The rotation of the Earth around the Sun :

Finally, we must keep in mind that the Earth's rotation axe is inclined at $23^{\circ}27'$ from the ecliptic plane. This tilt is at the origin of our yearly seasons : we consider the Earth tilted at $+23^{\circ}27'$ during summer (Souther hemisphere) and $-23^{\circ}27'$ during winter, the tilt beeing equal to 0 during the equinoxes.

Time concerns The day of the year and the time of day. All of these factors need to be taken into account before any more local analyses.

For Solar irradiation, we need to factor in time :

Atmorsphere's Interference

Radiation reaches the Earth's surface in a more or less direct way, depending on the atmospheric interference. Water, dust and pullutants floating in the air will refract rays, dispersing sunlight in multiple directions. The higher the interference, the more diffused the sunlight will be when reaching the surface. The unit of measurement to describe the amount of cloud cover is called an Okta, where 0 Okta is a completely clear sky and 8 Okta is a completely cloudy sky.

Night-time Radiation

Any radiation received by the ground (or buildings) will start releasing heat the moment the air surrounding it turns colder than the soil, typically during night-time.

The radiation emitted has changed to long-wave radiation. In a given urban model, for there to be energy loss (ie, heat loss), the radiation needs to escape back to the atmosphere. This implies a covisibility between the surfaces and the sky vault. More information on [Sky View Factor and Radiation](#)

Local weather will greatly affect solar radiation, especially the presence of clouds. As the sun passes through them, the light gets reflected and defused.

Sun View Factors

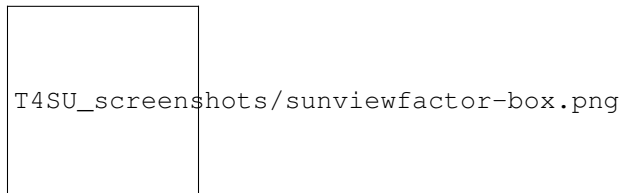
Once you've created a Sun Path, you can use it to calculate the **Sun View Factor**, which works a little like the Sky View Factor : instead of checking covisibilities between geometry faces and the sky dome, it will check the covisibility between a geometry and each point of your Sun Path.

See also:

[Drawing Sun Paths](#)

[Calculating the Sky View Factor](#)

Clicking on **Extensions > t4su > Sun Views > SunViewFactor** will enable the following command box :



- Select the two corresponding layers : your SunPath layer and your geometry layer.
- You can choose to alterate the results slightly by choosing a specific type of sky (from “pure” to “cloudy”).

Once that is done, you will find that your geometries now have extra attributes:

- `directSolarIrradiance:Float` represents the cumulative theoretical radiative energy received by the geometry from the sun at each point, which depends on their covisibility and the angle of the sun's rays.*
- `nbHitsTotal:Int`, the number rays between the geometry and the SunPath points
- `nbMinTotal:Float`, the number of rays between the geometry and SunPath, multiplied by the frequency of the plot of the Sun Path (eg: for a plot every 5 minutes if `nbHitsTotal=2`, `nbMinTotal=10`)
- `ratioTotal:Float`

You can then use **ColorFaces** to see the geographical variations of each of these new attributes. Below, we've drawn two different maps of the same phenomena : The number of minutes a defined space will receive direct sunlight on two opposing dates : 21st of June and 21st of December, the longest and shortest days of the year respectively.

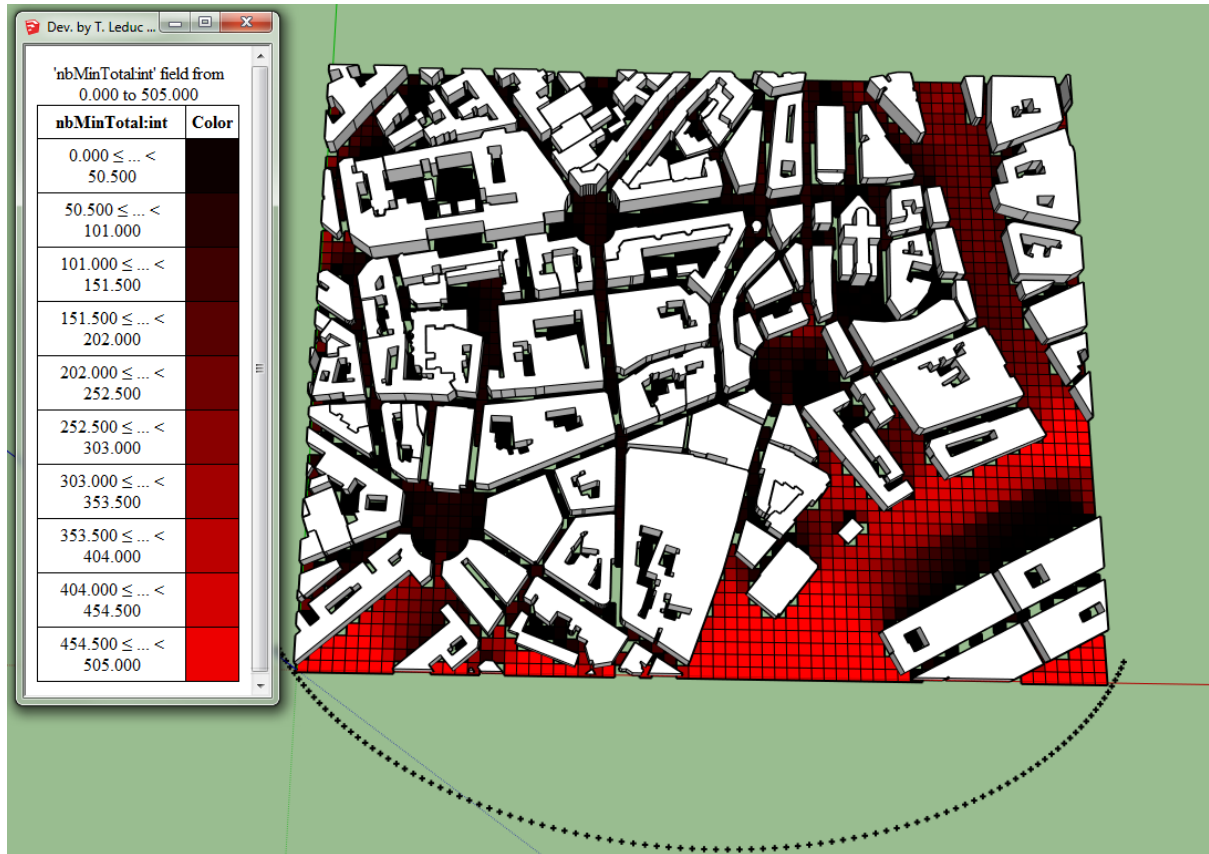


Fig. 10.3: Number of Minutes of Sunlight for the 12st of December

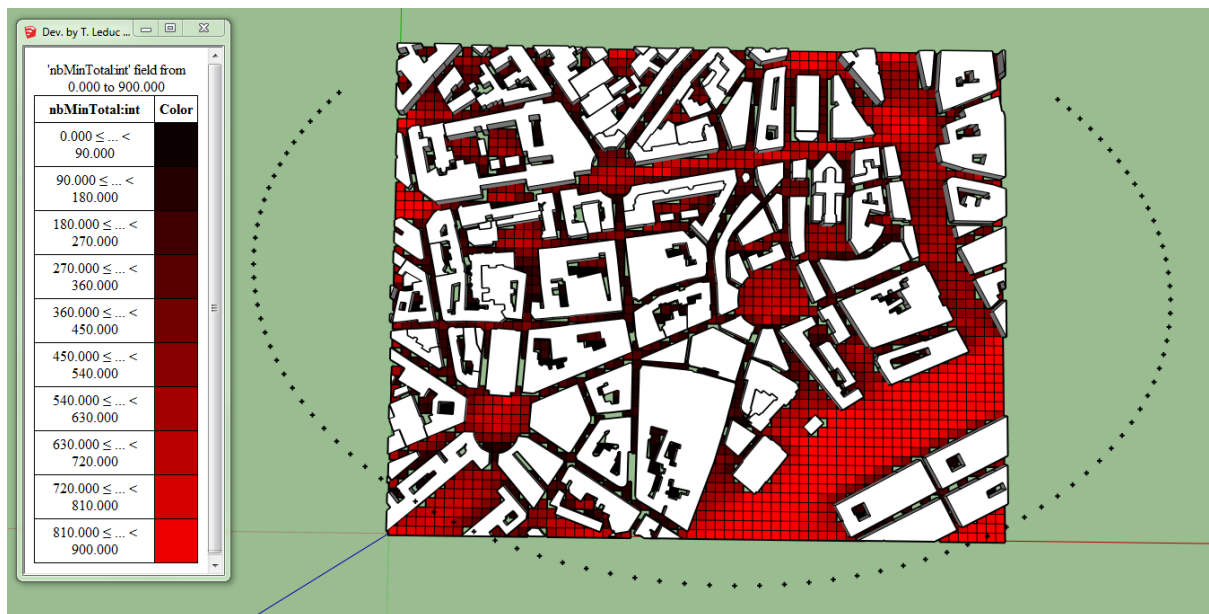


Fig. 10.4: Number of Minutes of Sunlight for the 21st of June

Wouldn't it be nice to combine these two into a single map, and be able to visualize the difference? Learn how with `ArithmeticsOnAttributes`.

See also:

Viewing Direct Solar Irradiation along a Path

*Basic Example of Data Visualization**Sampling the Bounding box*

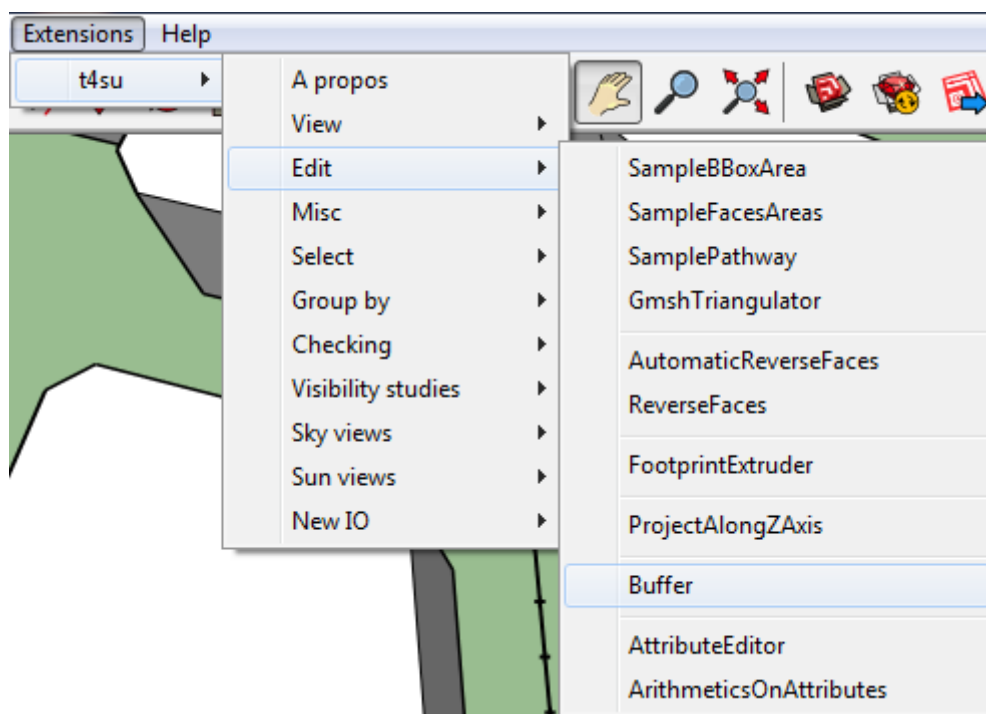
Viewing Direct Solar Irradiation along a Path

The city landscaping team has a new shipment of flowers, and they must decide where to plant them. They already have a general idea along a new pathway, using planters that reach two meters high so they can be seen from afar. These plants are very fragile and burn easily in the direct sun.

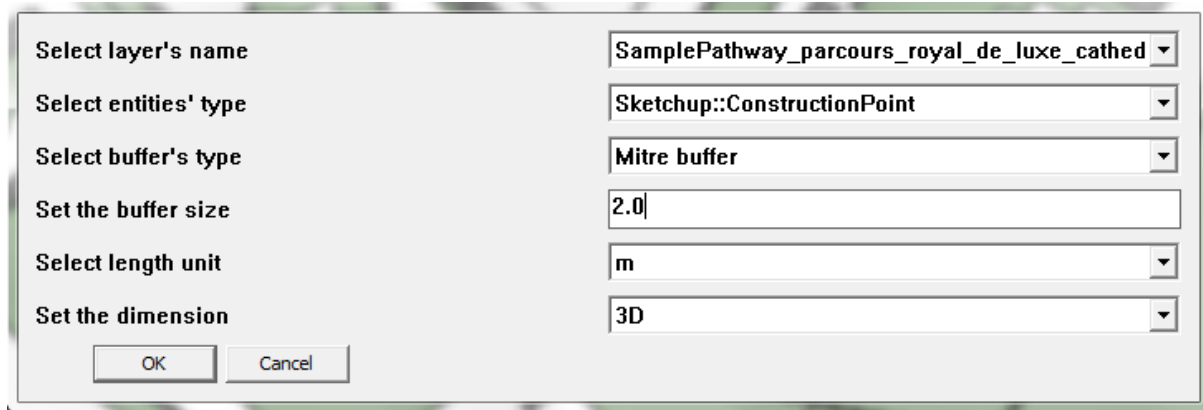
To find the ideal spot, we first have to recreate the path with line segments that you can import or draw with the SketchUp tool (we will be using the one we've used before to explain sampling points). We then proceed to the sampling of line or multi-line into points for example every 5m.

See also:*Creating Partial 2D Isovists*

Then, we need to emulate the planters. To do so, we can create a 3D buffer. Click on **Extensions > Edit > Buffer**



Select the name of the layer containing your sampled pathway and the geometry of the samples (Construction-Points). Next, you have the choice to create Round buffers or Mitre buffers (ie, with sharp corners). Select its size : we want 2m because we want to know the conditions at the top of our pedestals, not at ground floor.

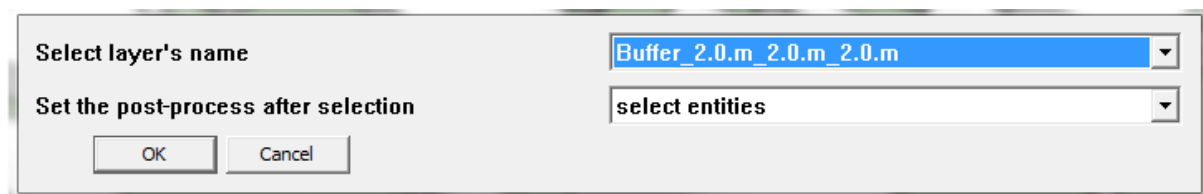


Next, we would like to know how much direct solar radiation each pedestal top would receive throughout the year. Create the appropriate sun paths.

See also:

Specific Sun Paths

Our plants are here to stay, so let's create sun paths for the entire year. Go to **Extensions > t4su > Select > SelectRoofFaces** and select your buffer layer.



We are finally ready to calculate the direct solar irradiation. Click on **Extensions > t4su > Sun views > Sun-ViewFactor**. Select your buffer layer in the drop-down list, which still has the "rooftops" selected, as well as your Sunpath layer. You also have the choice of the type of sky. Let's take the sky will give the highest radiation results : "Pure". If you click on **Extensions > t4su > View > PickupEntity**, you can select each buffer to see the new attributes that have been added. Click on **Extensions > t4su > View > ColorFaces** and select DirectSolarIrradiance:Float as your classification criteria. You shouldn't need more than four or five classes. The results show that

areas close to intersections generally receive slightly more radiation. It would be best to place the plants along the narrowest streets.

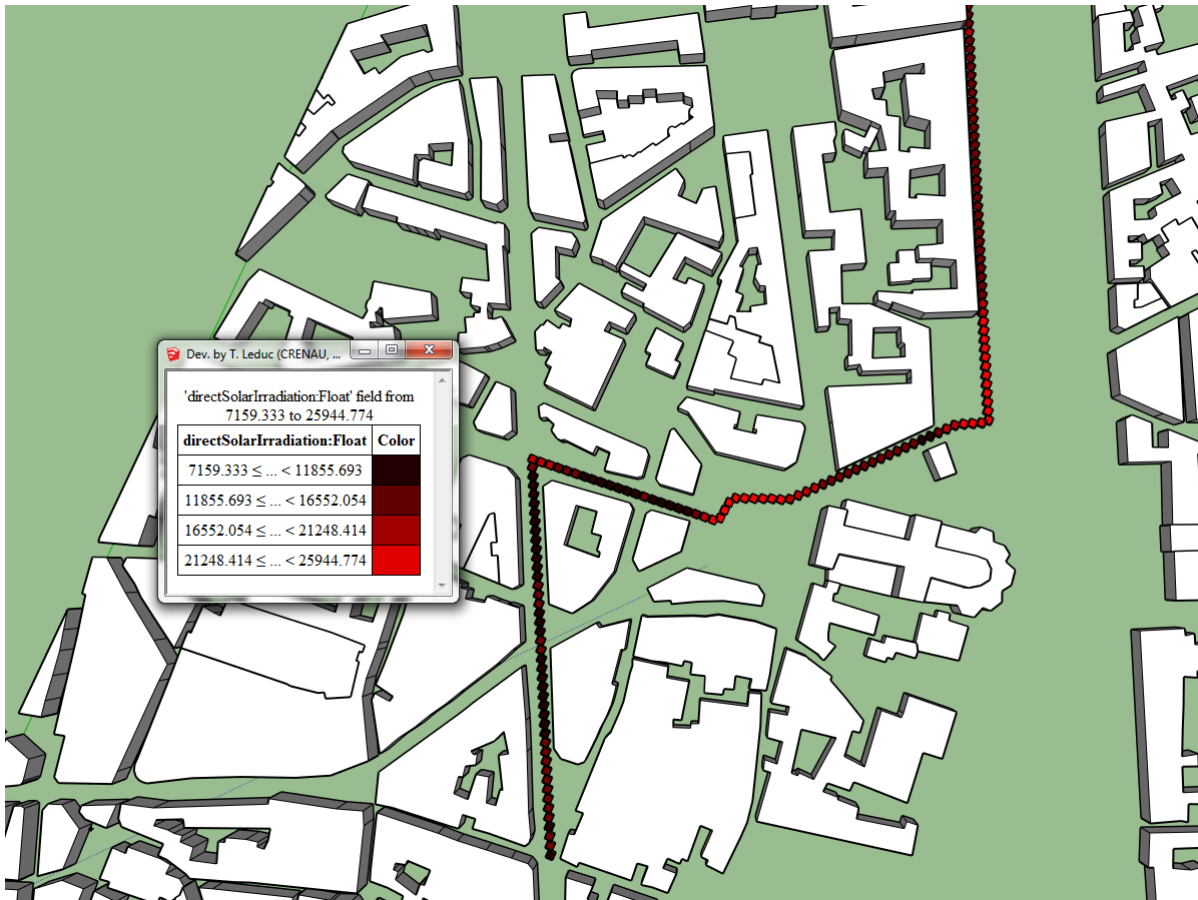


Fig. 10.5: Result of the Direct Solar Insolation along a Path, with Legend.

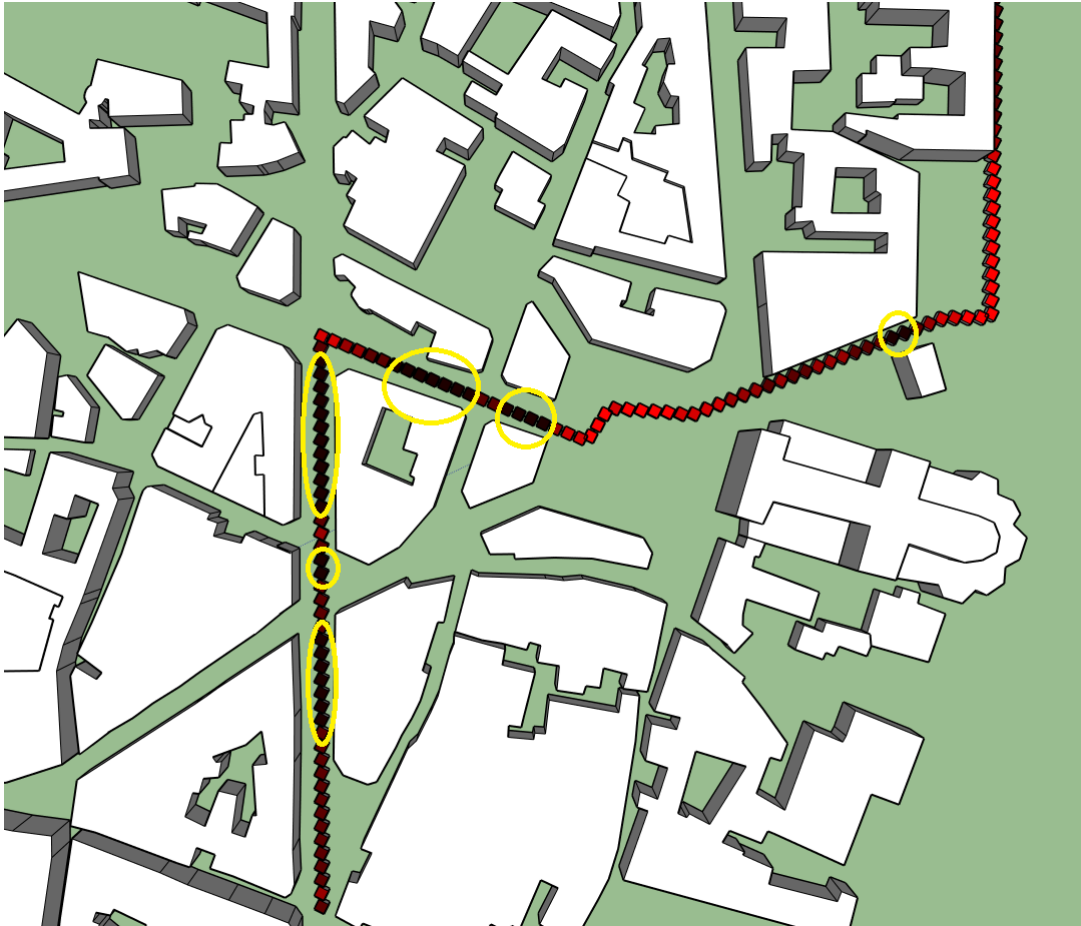


Fig. 10.6: Pinpointed Areas with Least Direct Sunlight, 2m above Street Level.

CHAPTER 11

T4SU Ruby API

Everything explained in this manual can be reproduced through the Ruby Console.

Warning: This section is still under construction, and only presents a few limited examples. Nonetheless, we encourage you to understand the logic behind these small scripts and adapt them to your own needs.

Readers:

```
AscReader.new.readAndPlot("d:/tleduc/t4gs/papers/scan-2016/dev/data/dem_ecn_5m.asc  
↪", 1.m, true, 1)
```

```
ShpDbfReader.new.readAndPlot('d:/tleduc/t4gs/papers/scan-2016/dev/data/vegetation.  
↪shp')
```

Sampling:

```
SampleFacesAreas.new("vegetation", 15.m, true, -10.cm).run
```

Listing Samples :

```
EntitiesListing.allConstructionPointsInLayer("SampleFacesAreas_15.00.m_15.00.m").  
↪size
```

Returns the number of entites.

ProjectAlong Z Axis :

```
ProjectAlongZAxis.new("SampleFacesAreas_15.00.m_15.00.m", true).exec
```

Select “true” for “upwards” and “false” for “downwards”.

```
EntitiesListing.allConstructionPointsInLayer("ProjectAlongZAxis_upwards_  
↪SampleFacesAreas_15.00.m_15.00.m").size
```

Make Layer Invisible :

```
Sketchup.active_model.layers['vegetation'].visible = false
```

Remove Layer :

```
Sketchup.active_model.layers.remove('SampleFacesAreas_15.00.m_15.00.m', true)
```

Load Trees :

```
treeCompDef = Sketchup.active_model.definitions.load("~/tree.skp")  
transforms = []  
i = 0  
EntitiesListing.allConstructionPointsInLayer('ProjectAlongZAxis_upwards_  
↪SampleFacesAreas_15.00.m_15.00.m').each {  
  |p| transforms.push(Geom::Transformation.new(p.position))  
  Sketchup.active_model.active_entities.add_instance(treeCompDef, transforms[i])  
  i += 1  
}  
  
transforms = nil
```

Create a Partial 3D Isovist :

```
pisov = PIsovist3D.new(aperture = Angle.toRadians(30), z0 = 1.6.m, nbRays = 256, ↪  
↪transparency = 0.5, spotColorName = 'Red', tetrahedraColorName = 'Yellow', ↪  
↪sketchOption = 1)
```

Iterating over a Point :

```
EntitiesListing.allConstructionPointsInLayer('layerName').each { |p|p.set_  
↪attribute("sln_dictionary", 'motion_direction:Array', [0,0,-1]) isov.  
↪execWithArgs(pickedPoint = p.position, pickedFace = p) }
```


Create a line at a point, pointing downwards:

```
EntitiesListing.allConstructionPointsInLayer('...').each { |p| e = Sketchup.active_  
  ↪model.entities.add_edges(p.position, Geom::Point3d.new (p.position.x, p.position.  
  ↪y, p.position.z - 1))
```

Create a 3D isovist:

```
Iso3D = Isovist3D.new(0.m, 20.m, 64, 0.5, 'Red', 'Yellow', 1)
```

Partial 3D Isovist

```
Piso3D = PIsovist3D.new(Angle.toRadians(30), 0.M, 20.m, 64, 0.5, 'Red', 'Yellow', ↪  
  ↪1)
```


I
Introduction
Listing, [85](#)