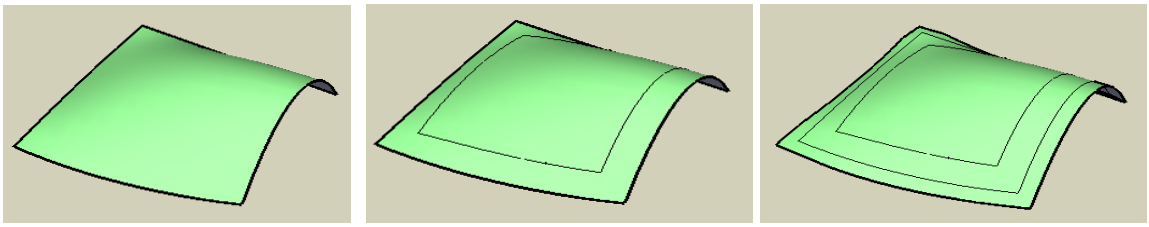
OFFSET CONTOUR ON SURFACE

OffsetOnSurface.rb

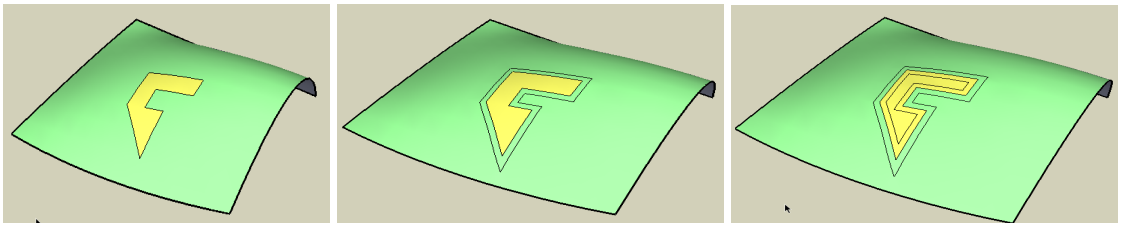
v.1.1 – 14 May 2008

Foreword

Offset on Surface tries to reproduce the behavior of the native Sketchup tool Offset on Face, but on a Surface shaped in 3D. Offset direction can be inside or outside.



It also works for inner surfaces on a larger surface:



Note that there is no mathematical solution for offsetting a contour on a surface in the general case, because there is no strict concept of distance on a surface. So the script works with some approximations and does not support all cases of topology. In practice, it is intended to work on regular and smooth surfaces.

Installation

OffsetOnSurface is now part of a bigger family of ***Tools On Surface***. Please refer to the Quick Card document for installation of all tools.

Compatibility: the macro works with **Sketchup v6 and should work in v5 as of v5.0.260**. I tested it on Windows XP and Windows Vista. It should normally work on Mac, but I did not test it myself. No idea if it would work in Sketchup version 7!!

1. Using *OffsetOnSurface*

I tried to mimic as much as possible the behavior of the native Sketchup Offset tool, with its 2 modes of selection, Explicit and Implicit, and some visual feedback. The tool also works with **generating Plain Lines or Construction Lines**. I added however a few options: Generate in a Group (F6), Isolate Surface (F7), Generate Faces when possible (F8), Generate new contours as Curves (F9), as well as Simplify Contour (F4) and Mark vertices with Construction Points (F3).

The ***OffsetOnSurface*** script acts as a classic Sketchup tool, remaining active until you invoke another tool.

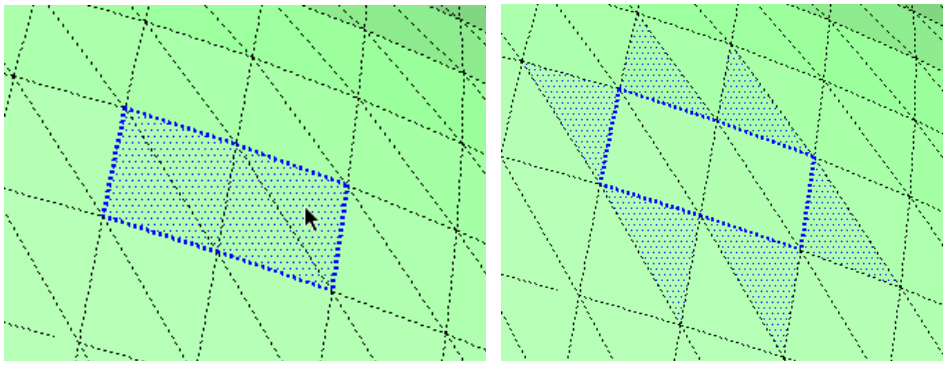
For convenience, the script supports 2 modes of selection:

1) **Implicit Selection** → you start the tool with no active selection
2) **Explicit Selection** → you make a Face / Edge selection first, and then call the tool.

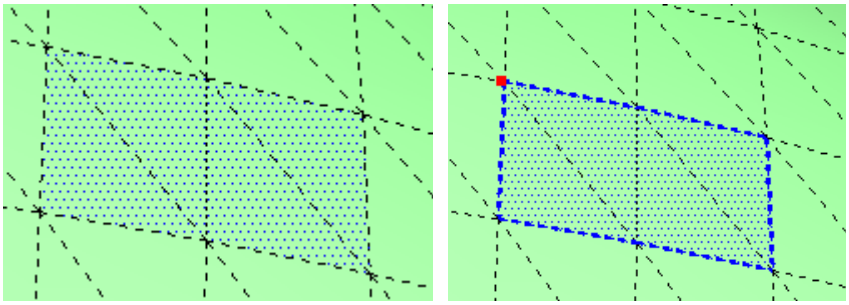
Note however that selection only applies to the active entities of the model. You cannot select a group to mean *"all faces of the group"*. You must open it first!

1) Explicit Selection (Select first, then start the tool)

The script works on a **selection of Edges and Faces** arranged so that you can determine where the inside and the outside are. This is why in practice you would rather select the faces bordering the edges to remove ambiguity. For instance, the 2 selections below define the same contour shape, but oriented reversely.



But if you select just the 4 inside faces, the script will automatically find the corresponding edges:



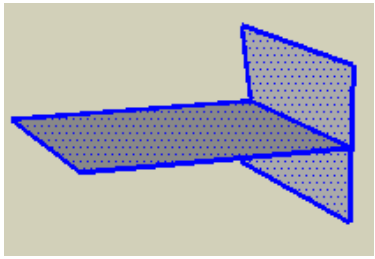
Note that the script will simply ignore all other entities which are not face or edges, that is construction lines and points, groups and components when not open, images, etc…

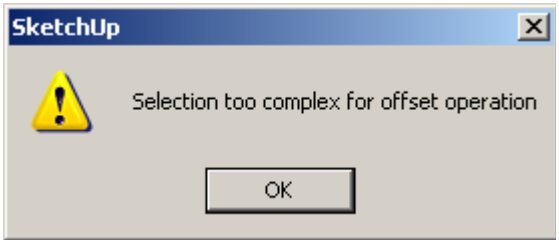
A selected edge is kept in the selection if and only if it is bordered by one or two faces (i.e. not zero, and not 3 or more)

- **If it is bordered by one face, then it is not mandatory to select the face**
- **If it is bordered by 2 faces, then you must select one of the two faces only**

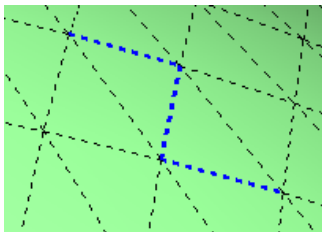
If you don't respect the rules relating edges and faces to waive any ambiguity, the script will unselect the edge.

It will also apology by a message when the topology is to complex to calculate. This happens in particular when edges are common to more than 2 faces.

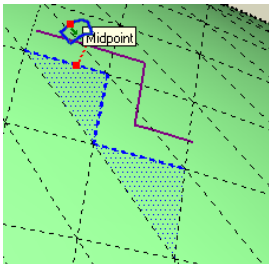




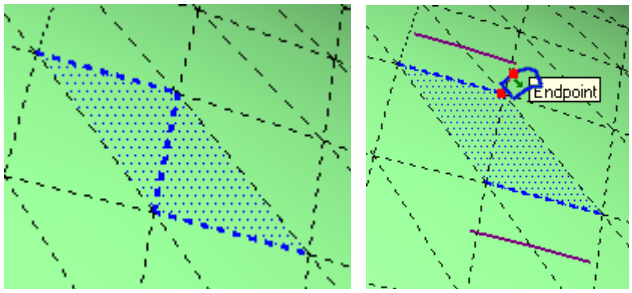
In the Explicit selection Mode you are not obliged to select closed contours. The script should normally works on a few edges. But keep in mind the rule about the ambiguity of edge selection. If you only select edges as below, this will not work:



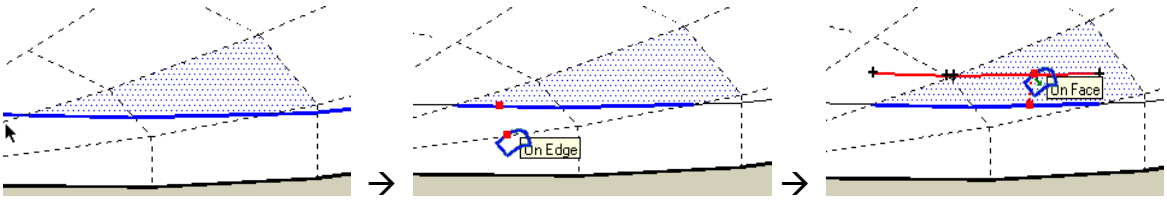
You must select faces to remove ambiguity:



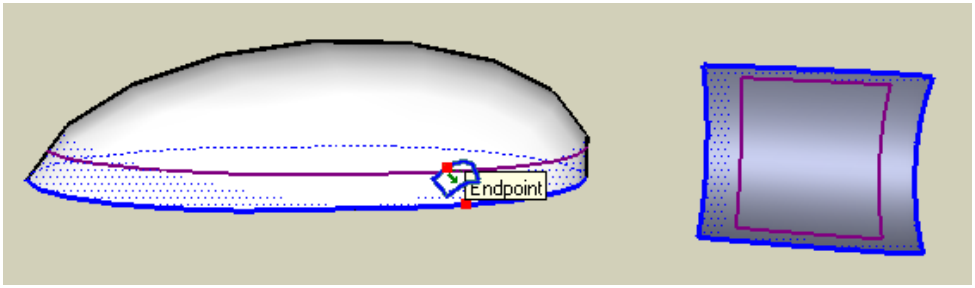
In the case below however, the edge bordered by 2 selected faces will be unselected:



Note that the selection of edges coupled with selection of surface will work even if the edges are part of a Sketchup curve. For instance, even if you select a curve and 2 faces, you will actually retain the 2 edges bordering the faces:



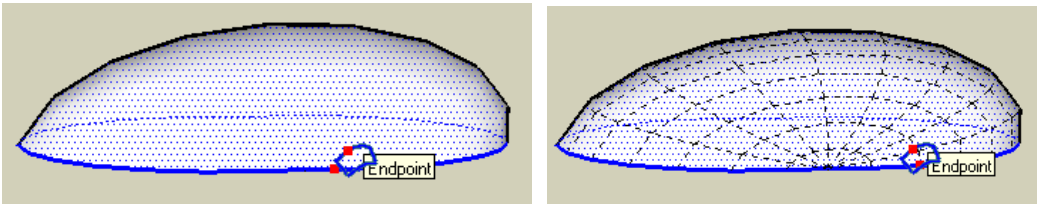
Finally, the selection does not need to contain *contiguous* contours. If it makes sense for you to select surfaces in different parts of the model, the script will handle them in parallel.



2) Implicit Selection (Start the tool with NO selection)

After starting the tool, you move the cursor over the model to select the surfaces, which will be highlighted.

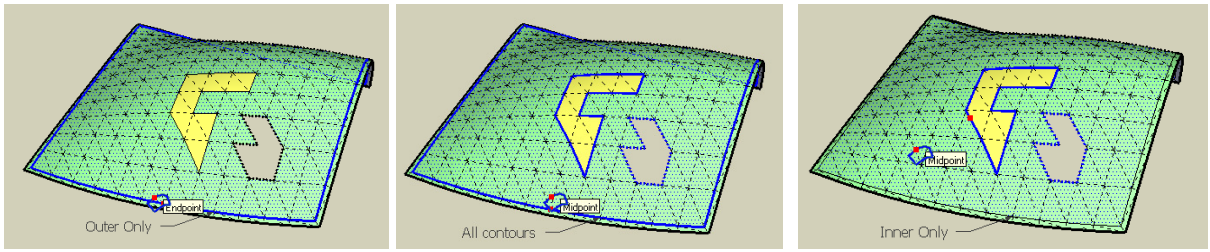
> **A surface is defined as a set of contiguous faces where Edges are SOFT or HIDDEN, as in Sketchup. Edges which are just smoothened will count as borders.**
>
> **This Implicit selection mode will work the same whether you are or not in the mode *"View Hidden Geometry"*.**



In the Implicit Selection mode too, there are ambiguities because a surface can have actually one outer contour and one or several inner contours (usually because of holes). As there is no way to decide what you wish to do, the macro provides 3 modes of automatic selection, accessible via the **contextual menu** or the **toggle key F5**.

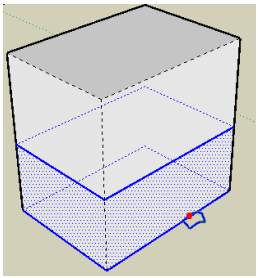
The current mode is shown is the status bar:

1) **Outer contour** → this is the default
2) **All contours**
3) **Inner contour(s) only**



Of course, you can always switch to the Explicit selection mode to select the exact contour you want.

Note that in some very specific cases, it is not possible to decide whether a contour is 'inner' or 'outer'. If this happens, then the script selects both contours.

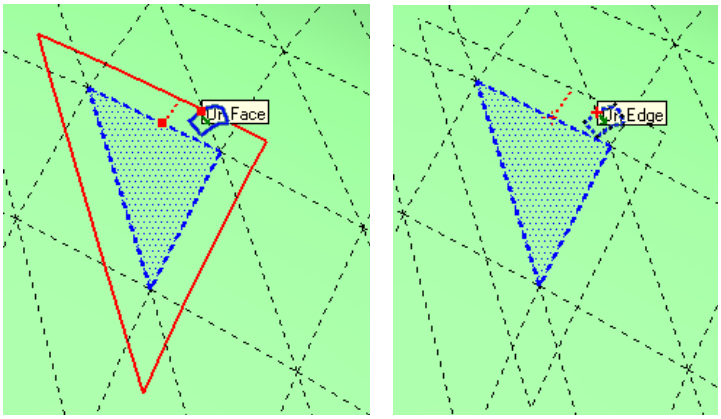


Finally, by keeping SHIFT depressed, you can select several adjacent surfaces by mouse-over. Be aware however that selection will be done for any surfaces the mouse would pass over, so that, in practice, this is more applicable to adjacent surfaces.

3) Plain Lines and Construction Lines

The tool supports both modes.

You can toggle between both by pressing **F2** or **Ctrl Alone**. The cursor will reflect the current mode.



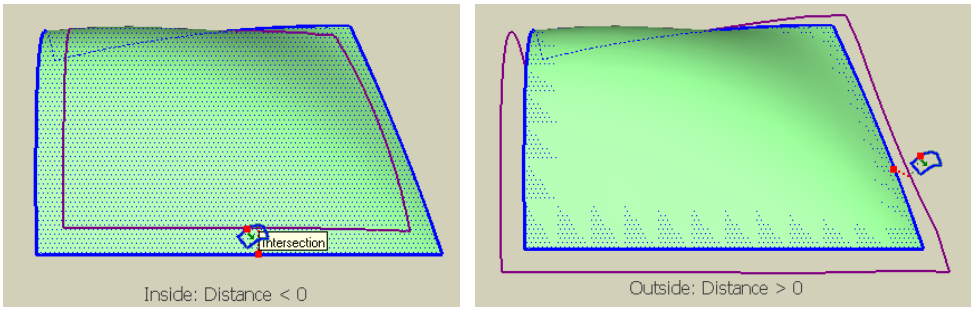
4) Distance of Offset

When you have a surface selected, resulting either from an implicit or explicit selection, you will see a **Red Dot** (or cross) on the contour. By **clicking and dragging** (or clicking, releasing and moving), you can offset the contour with visual feedback. Note that the path of the Red Dot (in red dashed) will follow the surface. By pressing **Escape**, you would go back to the surface selection. By keeping **Shift** depressed while dragging, you will skip inferences.

The main parameter of the script is the **Distance of Offset**, which by convention[1] is:

- **Negative** when you go toward the **inside** of the contour
- **Positive** when you go toward the **outside** of the contour

Note that the convention Inside / Outside is irrespective of the orientation of faces.



At any time, you can type the value of the distance in the VCB. The convention is the following:

- **If you did not move the Red Dot yet**, then the sign of the value will tell whether the offset is performed toward the Inside (negative distance) or toward the Outside (positive value).
- **If you moved the Red Dot in a particular direction**, then a positive value will change the distance but keep the orientation, whereas a negative value will also reverse the direction.

In all cases, the offset operation will execute after dragging the contour or typing a value in the VCB (as in Sketchup) or **double-click** on the surface (when the red point is mobile) to do an offset with the same distance as the last operation.

As of version 1.1, you can change the distance right **after** an offset operation, by typing a new value in the VCB. Note however that, due to limitations in the Sketchup Ruby API, the operations can take sometimes[2] (there is a progress bar), and depends actually on the total number of entities in your active model, either the whole model if you work at the highest level or the current opened group or component (this is the only performance dependency however). So, you may or not exercise this capability.

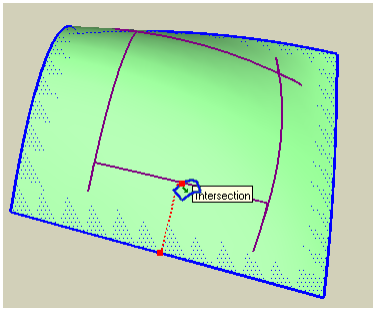
---

1 The native Sketchup Offset has rather strange convention, where the sign of the distance entered in the VCB is relative to the offset already entered.

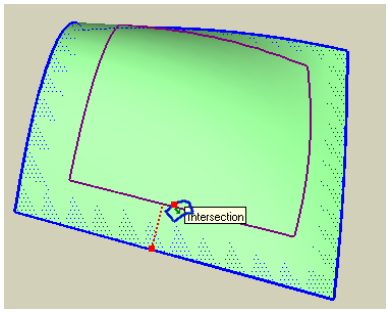
2 The reason is that the script needs to perform an Undo and retrieve the original selection, which in Ruby is not supported by native routines.

5) Simplifying Contours

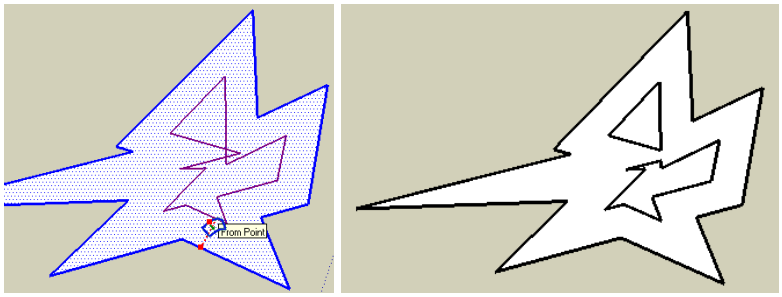
One of the real problem of an offset operation is that, if you want to be 'geometrically correct', then in most cases you will not obtain the desired result, as in the example below.



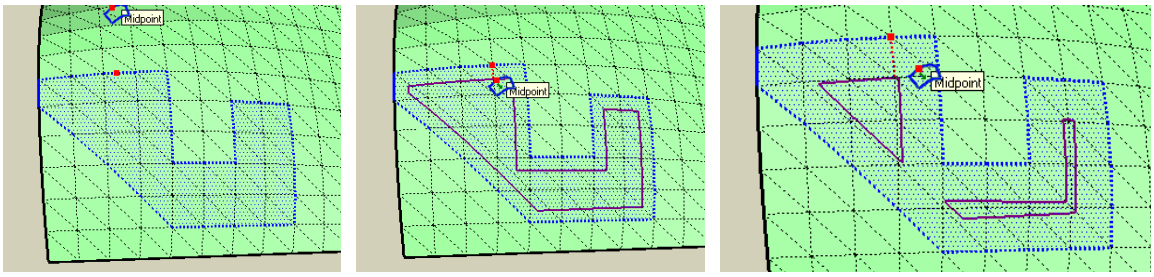
This actually also happens on plane surface with the native Sketchup Offset tool. The script applies a simplification algorithm to me sure that vertices are eliminated when appropriate. The same offset would then give the following result:



Since version 1.1 (based on a suggestion from **a4architect**), the script does handle situations where the generated contour crosses itself, as in the following case:



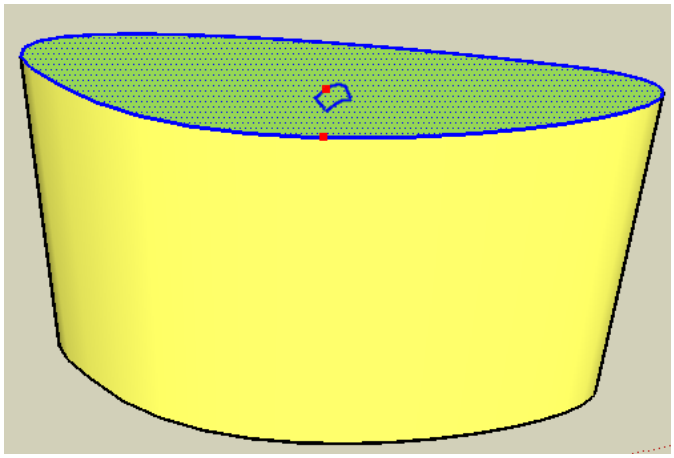
This also works on curved surfaces:



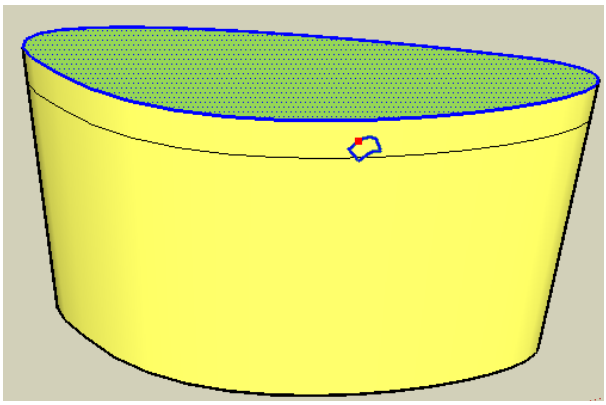
For convenience, I left the option to simplify or not the generated contour, using the toggle key **F4** (or item in the contextual menu).

6) Isolating Contours on Surface

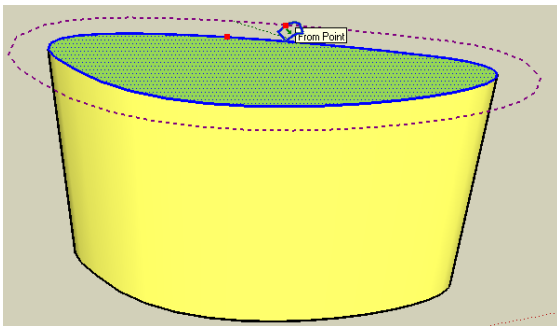
When you offset a contour which is partially or totally embedded within a larger surface, you have the option to ignore the rest of the model. This option may not be obvious to you in the first place, but consider the following case, where you select the top face:

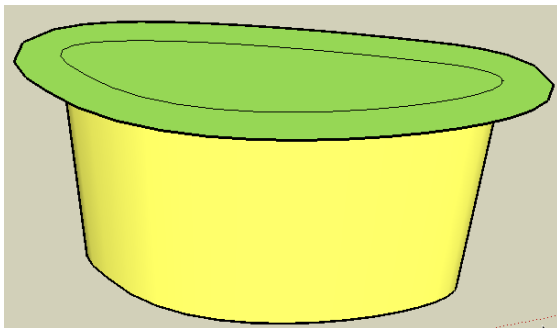


If you offset the contour toward the Outside, it will actually spread on the lateral faces (which may be what you want by the way!):



To avoid this behavior, you can instruct the script to ignore the rest of the model and do as if the face was 'standalone'.



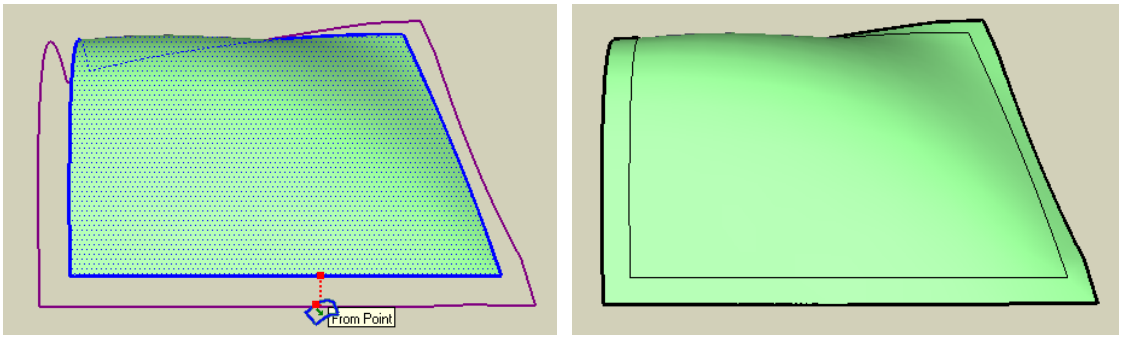


Note that the visual feedback of the offset in 'standalone' mode is in **dashed dark green** (instead of solid purple when this mode is not activated).

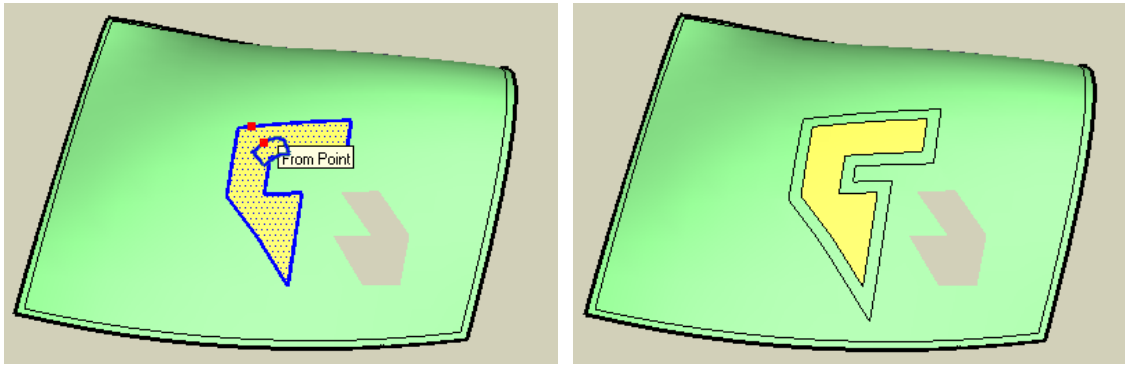
By default the script treats all selected surfaces in a non-isolated mode. You can however activate the standalone mode via the contextual menu or the **Toggle Key F7**.

7) Generate Surfaces

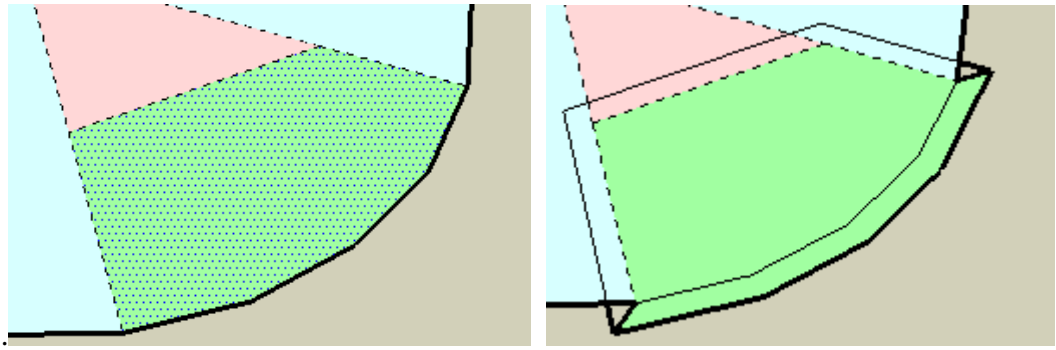
When you enlarge the contour (i.e. distance > 0), the space between the original and generated contours can be filled by a surface (as in the native Sketchup Offset tool).



This will happen only for edges that are truly at the border of the surfaces, that is, bordered by only one face. If the surface is itself within a larger surface, there will be no face generation.



As usual, you may find more complex topologies where you'll get a little bit of both!

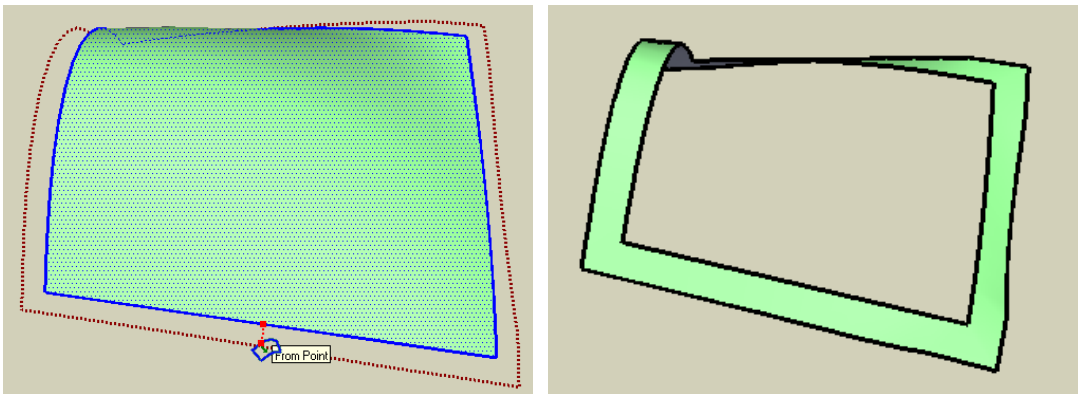


Note that the script will try to remove coplanar edges between faces (unless materials are different). This sometimes does not work correctly, as Sketchup is very sensitive to exact co-planarity.

By default the generation of surface is *On*. You can however deactivate it via the contextual menu or the **Toggle Key F8**.

8) Generate as a Group

For convenience, the offset contour and the generated faces can be generated as a Group, which you can then manipulate independently or explode. Note that the visual feedback of the offset is in **dotted dark orange** (whereas it is in solid purple when no group is generated).



Remember that the Group where the contour is generated is common to all operations on Surface, but created within the active part of the model.

By default the generation as Group is ***Off***. You can however activate it via the contextual menu or the **Toggle Key F6**.

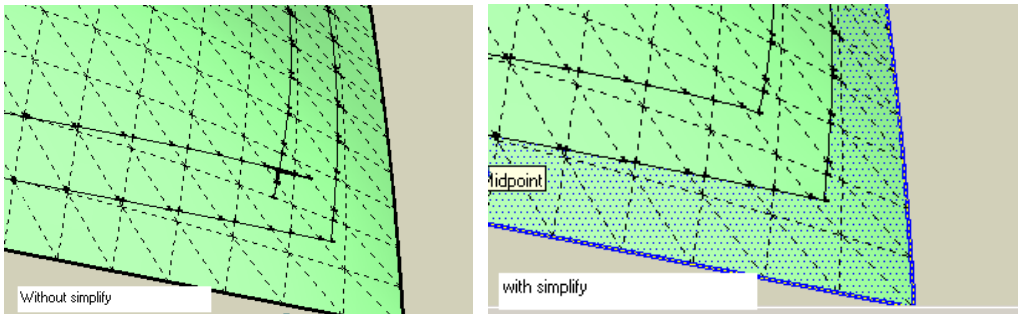
9) Generate Contour as Curves

The offset contour(s) can be generated as welded curves, which are sometimes easier to manipulate. Note that the visual feedback of the offset is in **dashed dark orange** (whereas it is in solid purple when no group is generated).

By default the generation as curves is ***On***. You can however deactivate it via the contextual menu or the **Toggle Key F9**.

10) Simplify Generated contour

The script tries to simplify the generated contour to avoid the spikes and also to treat the overlaps between different portions of the contour. However, in some cases, the simplification algorithm is a little bit messed up and it is possible to generate the contour without simplification (as the native Sketchup Offset would do), which is more geometrically accurate (…but then, you have to modify it by hand).

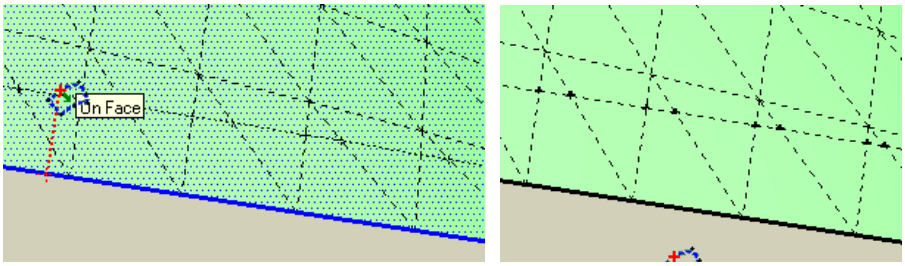


By default the simplification of the generated contour(s) is ***On***. You can however deactivate it via the contextual menu or the **Toggle Key F4**.

11) Put marks at generated vertices

I added the option to put construction points at the intersection of each vertex in the generated contour. This option is probably of little use when using the script in Plain Line mode, but can be helpful if you generate the contour as construction lines.

While you drag the contour, you'll see construction points displayed.



You can enable or disable the option with the **Toggle Key F3**.

12) Default Settings

The script comes with default settings. For convenience, I have separated the settings in a specific file, OffsetOnSurface.def, which must be located in the subfolder OFS_Dir of the Sketchup Plugin directory.

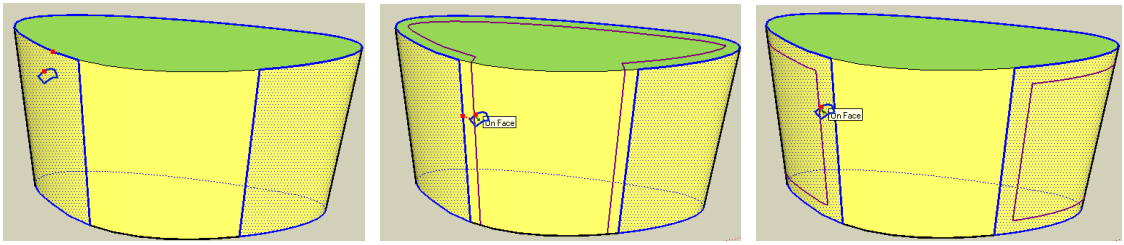
The default values are expressed as Constants, which can be possibly altered and would become the initial default when Sketchup starts up. Here are the 'factory' values:

```
#== ToolsOnSurface.def -- Default Settings ===================================
#External Constants for DEFAULT SETTINGS (could be possibly altered)
#Be careful: use <true> or <false> in lowercase. Colors must be valid in Sketchup
module SUToolsOnSurface

OFS_COLOR_PseudoSelection = "blue"       #Color of the selection
OFS_COLOR_Normal = "red"                 #color of contour in interactive mode
OFS_COLOR_Group = "darkred"              #color of contour when Group option is On
OFS_COLOR_Alone = "darkgreen"            #color of contour when Alone option is set
OFS_DEFAULT_Group = false                #Option: generate contour as a Group
OFS_DEFAULT_Alone = false                #Option: Ignore rest of model when offsetting
                                                  surface
OFS_DEFAULT_GenFaces = true              #Option: generate faces when offset is done
                                                  outside on true borders
OFS_DEFAULT_GenCurve = true              #Option: generate contours as curves (useful for
                                                  Joint Push Pull)
OFS_DEFAULT_Simplify = true              #Option: Simplify generated contour
OFS_DEFAULT_CPoint_L = false             #Option: generate marks (Plain Line mode)
OFS_DEFAULT_CPoint_C = true              #Option: generate marks (Construction Line mode)
OFS_DEFAULT_Contour_Select = 'O'         #For implicit selection -->
                                                  O = Outer only, A = All, I = Inner only


end
```

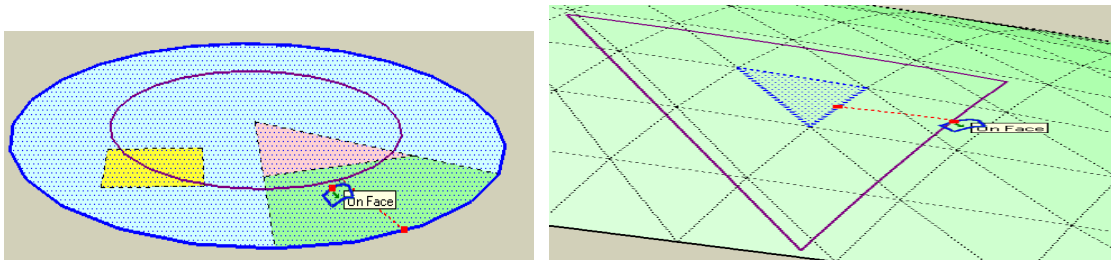
Be careful when you modify the file to respect the case and check that color values are valid! This is Ruby code actually.

2. A few examples

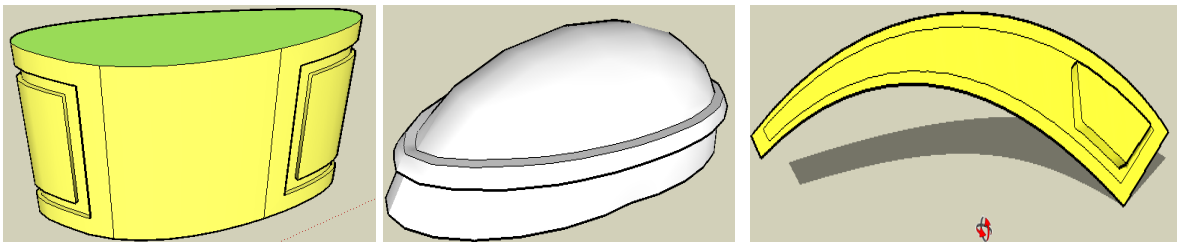
Here are a few examples of what the script can cover



Even of a plane surface, the Tool will allow you to perform Offset operations where Sketchup will refuse, because of the multiple faces.



You can of course combine the Offset On Surface tool with Joint Push Pull.

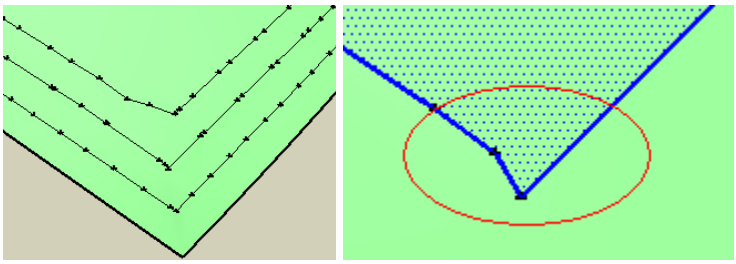


3. Known Problems, Limitations, Caveats

Now the bad news!

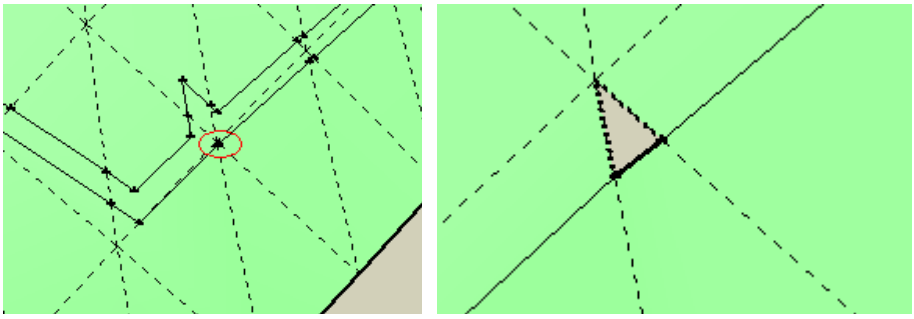
There are many limitations and known problems with this version of the script. Not to mention the unknown ones and bugs.

I would therefore recommend that if you find that the generated contour has a strange shape, for instance with unexpected triangles or sharp angles, you look closely at the original contour and fix the source of the issue, which is in general relatively easy, as lines on surface can be re-edited without harming the models. This may happen frequently when you offset contours in cascade. Look also for small triangles, too small for Sketchup to generate the face. In the example below, the disappointing transformation in the corner is due to the 'bad' shape of the corner in the original contour:



As the simplification algorithm is based on suppressing points in the generated contour, the script would have anyway a lot of difficulty to correct this type of situations.

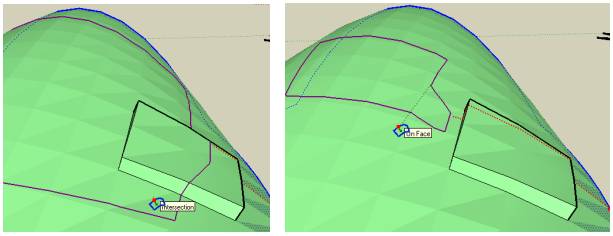
Below is a case with very small triangle. I tried to do some snapping to vertices and edges, but there is no way to avoid it in the general case. So it may simply happen, and you have to fix the problem manually in the original contour first (via the utility "Make Face").



1) Path on Surface

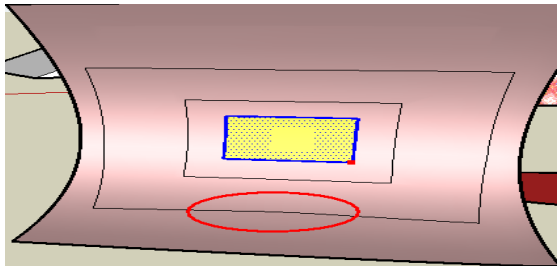
There is no real exact algorithm for offsetting edges on a surface. Actually, the script moves each vertex along the average vector of the normals to its edges with the given distance. So, the script physically reconstructs the path of each vertex along this direction on the surface, which indeed depends on which kind of 'hurdles' it will find on the path.

For instance, if your surface is not really regular, like the one below, some vertices will have their path impacted by the bumping square. It will become even worse when you pass the bumping square, as the script as not the same intuitive approach to 3D topology as we, human beings, have!



So, as a general rule, you should apply Offset On Surface on rather regular surfaces, which can be 'draped' by a plane.

The same happen for the deformation of geometric figures. In theory, it would be possible to preserve the angles from the original contour to the offset contour. In practice, it is difficult. The current script uses a simple algorithm which does not guarantee the preservation of angles. I have in stock a second algorithm that would (in most situations), but which is more complex and time-consuming. For instance, the rectangular shape is not exactly preserved when the offset distance is too large:



2) Performances

The script is actually sensitive to the number of edges in the faces of the model (i.e. not just the ones selected, but the one constituting the surface on which you offset the contour). The reason is that there is no other way than to explore the intersection of the vertex path with the edges of each face that comes up on the surface.

With surfaces made of triangles or quadrangles, the response time should be OK. However, take for instance a simple circle with 120 edges, and just offsetting its contour will slow down the visual feedback.

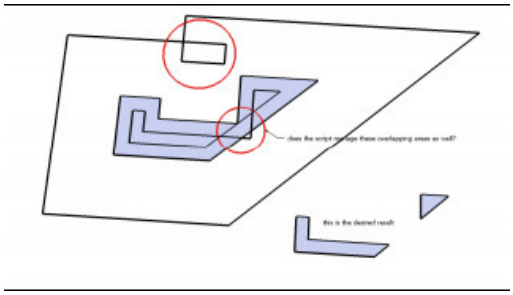
As a design goal, I tried to make performance independent on the total number of edges and faces in the whole model. Offset On Surface only explores the faces that the contour would meet. This is why in particular it cannot 'cross' holes, because it would require that the script explores the whole model to find the appropriate face beyond the hole!

Remember that if you have performance issue with the visual dragging, you can always type the distance in the VCB.

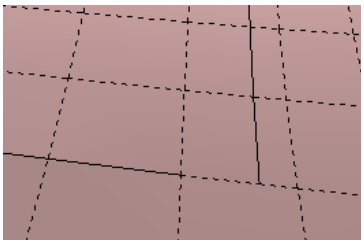
3) Remaining work

There are a number of features that could be added, but I am not quite sure this is needed.

1. **Input of distance AFTER offset operation**, as in the native Sketchup tool. In the current version, you have to Undo and start over. This is rather tricky, because when Sketchup performs an undo, the original faces are not all restored and new ones may be created. So it is not straightforward to find the selection. Maybe for another version!
2. **Improving Inference of the Red Point**, which today is only working when the surface is plane. Because the Red Point also moves onto the surface, the inference on remote point is complex to handle.
3. **Improve simplification of contours**, in particular to split generated contours in several portions as described by ***a4chitect*** in the Sketchup Community forum



   Note version 1.1: I tried to fix this problem and eliminate overlapping contours. However the algorithm may not be so waterproof in all cases.
4. **Find a workaround for a Sketchup bug in overlaying edges**. In some cases, when the generated contour is exactly superimposed to an existing edge, Sketchup does not change the Soft property. This may result in the fact that the generated contour is not closed. You have to find the edge rupture(s) and draw it by hand.



   Note version 1.1: I tried to fix this problem, but if you expect a close contour and it is not the case, you should check for areas where the generated contour follows existing edges.
5. **Eliminating empty small triangles**. Sketchup has definitely problems to generate (or actually keep) very small triangles. For the time being, you must detect them and correct the situation manually

   Note version 1.1: I tried to reduce the occurrence of these situations, but this can still happen.

   Note version 1.2: I tried to explicitly fix these situations… but only for triangles
6. **Improving the algorithm to better preserve angles**. Today I use a simplified version that does some approximations. So, in case of complex surfaces, usually not parametric, you may not obtain the expected 'natural' results.

4. Ruby API to Offset On Surface *(for developers only)*

I found useful to design an API to the Offset on Surface operation that could be used in other scripts.

First, you need to include a statement `'Require "OffsetOnSurface.rb"'` in the header of the calling script.

Here are the specifications of the **single API method**:

```
Status = SUToolsOnSurface.Api_Offset ( linemode,
                                       selection=nil, distance=0,
                                       group=false, alone=false,
                                       genfaces=true, gencurve=true,
                                       simplify=true, cpoint=false)
```

ARGUMENTS

- `linemode` indicates whether contour is generated as plain lines (true) or as construction lines (false)
- `selection` is a **list of valid drawing entities**, as an *Array*, containing edges and faces.

  Default: if `selection == nil`, then the current Sketchup selection is taken
- `distance` is the **distance of offset**, positive for outside, negative for inside, as a *Sketchup length* (so usually passed as "20.cm" or so). Must be non-zero.
- `group` allows to **generate the contour as a Group or not**, as a *Boolean*
- `alone` allows to **offset the contour as if the original contour was alone in the model**, as a *Boolean*
- `genfaces` specifies **whether faces will be generated when new contour is outside the surface**, as a *Boolean*.
- `gencurve` specifies **whether the new contour is generated as a Sketchup curve**, as a *Boolean*.
- `simplify` specifies **whether the script does simplify the generated contour**, as a *Boolean*.
- `cpoint` specifies **whether the script puts construction points at each vertex of the generated contour**, as a *Boolean*.

RETURN VALUE

The method returns the status of the operation, as an ***integer***.

`status = 0` if the operation was successful

Otherwise,

- `OFS_ERROR_NO_SELECTION = 1`
- `OFS_ERROR_INVALID_SELECTION = 2`
- `OFS_ERROR_COMPLEX_SELECTION = 3`
- `OFS_ERROR_DISTANCE_ZERO = 4`