BEZIER AND SPLINE CURVES

beziersspline.rb v1.2

A RUBY SCRIPT FOR SKETCHUP V5 AND V6

Credits and acknowledgments to:

- ***@Last Software*** *for the original Bezier.rb macro (Aug. 2004)*
- ***Carlos António dos Santos Falé****, for the Cubic Bezier algorithm (cubicbezier.rb, March 2005), which I incorporated as an extension to my macro.*
- ***Victor Liu****, who designed the Bez-Patch script (Nov. 2006), for inspiration about the axis lock mechanism*
- ***CadFather****, for the icons of the toolbar*

Foreword

This script ***bezierspline.rb*** is an evolution of the original *Bezier.rb* script designed in 2004 by @Last Software. I must say that the capability to draw Bezier corves was certainly a great opening to designing complex models in Sketchup. I was one of those using it extensively[1].

The original macro *bezier.rb* showed however some limitations. For instance, it was not possible to change the number of segments drawn in the curve. Or to use an axis lock mechanism to edit control points.

So I decided to bring a few enhancements and came up with the following additional features in this version of the macro:

- Cubic Bezier spline curve[2] (i.e. with control points on the curve)
- Uniform B-Spline (i.e. smooth approximation of contours0
- Polylines (i.e. contours made of segments with free movements of vertices)
- Modification of Precision (i.e. number of segments for Bezier curves) in Creation and in Edition mode
- Adding / Deleting control points in Edition mode
- Drawing mode can be 'Start / End point' or 'Open-ended'
- Support of loop closing
- Plane lock in Creation mode and Axis and Plane lock in Edition mode
- Conversion of any curve to spline curves (via Polylines)
- Support of Undo for Creation and Edition modes
- Language translation support

Compatibility: the macro should work with **Sketchup v5 and v6** (Pro and free versions, English and French). I tested it on Windows XP and Windows Vista. It should normally work on Mac, although I could test it myself, as it's been a long time now that I did not touch this platform, which still has my nostalgic respect.

[1] see for instance a ship hull at Jylland Raw Hull par Fredo6 - Google Banque d'images 3D
[2] based on the macro *cubicbezier.rb* published by Carlos António dos Santos Falé

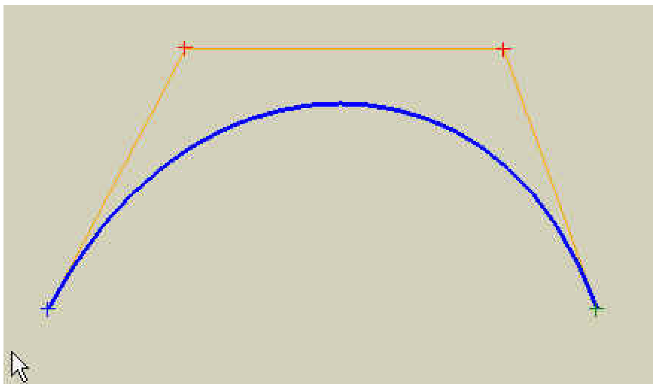
1. Overview – About Bezier Curves

This section gives some explanations about Bezier and other spline curves, and highlights the main capabilities of the macro.

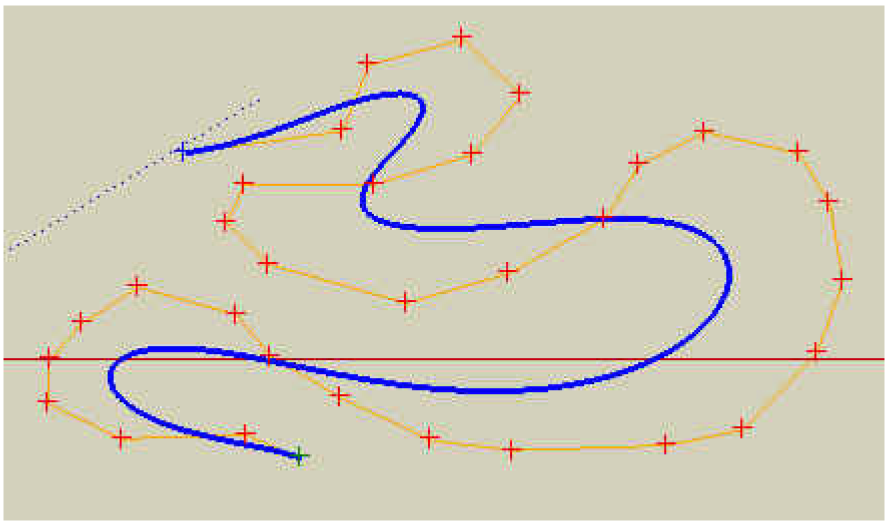
1) Control Points

Bezier curves are mathematical objects that are defined by a set of **'Control Points'**. In Sketchup, this will be the points you click when creating the curve, including the first and last points. The **'Degree'** of a Bezier curve is just the 'number of control points minus one'. This is the parameter that is used in the original *bezier.rb* macro. Personally, I found clearer to use the concept of *number of Control points*, because you can better visualize it.

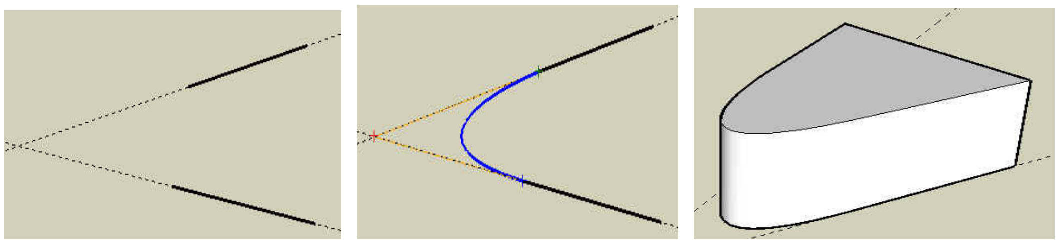
Here is a Classic Bezier curve with **4 control points** (so degree = 3). You will notice that, except for the first and last points, the control points of a **Classic Bezier curve** are not located on the curve itself.



Actually, you can have much more control points and draw rather complex shapes:

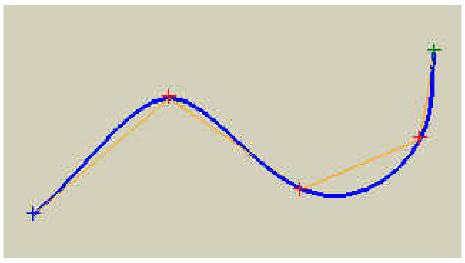


Conversely, you can have only one intermediate control point (so degree = 2). I never understood why the author(s) of the original macro *bezier.rb* did not support Bezier curves with just 3 control points (degree = 2). This simplest Bezier curve can be extremely useful. For instance, to join harmoniously two coplanar segments, you create a Bezier curve with only one intermediate point precisely located at the intersection of the 2 lines. This ensures that the curve is tangent to the 2 segments.

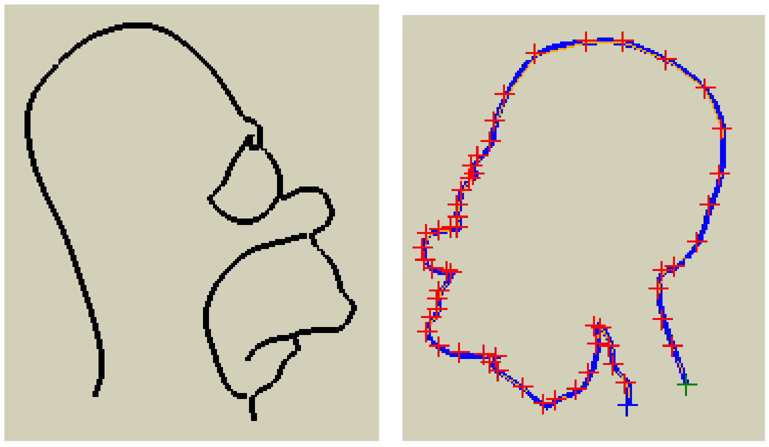


The main issue with Bezier curves is that it is not so easy to reproduce a given shape, unless you become very familiar with how to relate control points to the actual curve. So in practice, Bezier curves are often used in architectural construction to smoothen profiles when arcs cannot make it.

Cubic Bezier curves are different mathematical objects. They are defined by a set of control points too, but the underlying mathematical algorithm has the property to keep the control points on the curve itself. This is why they are sometimes called 'splines'[3]. Here is an example, with few control points.



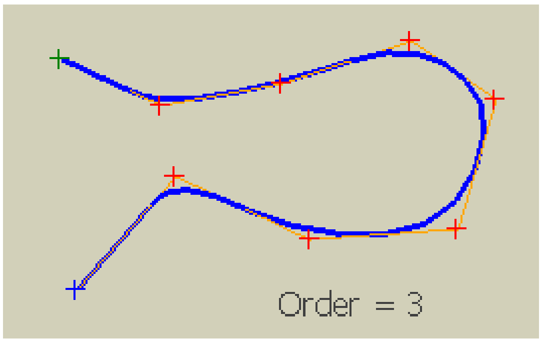
Splines make it easier to shape contours with more accuracy than traditional Bezier curves, because the control points just have to be on the contour. Indeed, you may need a decent number of control points to obtain the desired result (in the *bezierspline.rb* macro, you can draw curves with tens or hundreds of control points; I put the upper limit at 500)!



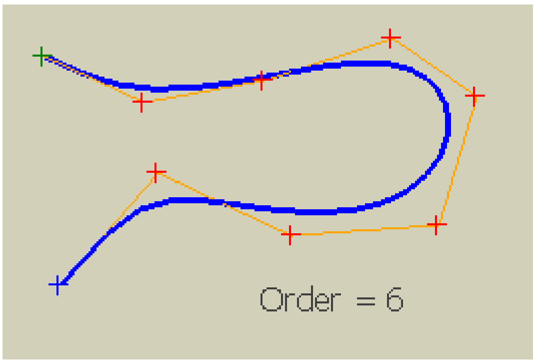
[3] I am not familiar enough with the mathematical details, but there are 'true' spline curves which are based on different algorithms (in particular B-Splines and β-Splines).

Uniform B-Spline curves are somehow intermediary between Bezier curves and Cubic Bezier splines. The control points are not located on the curve. The shape of the curve can be adjusted by the value of an extra parameter, called ***order***, that will globally control how 'smooth' the curve is.

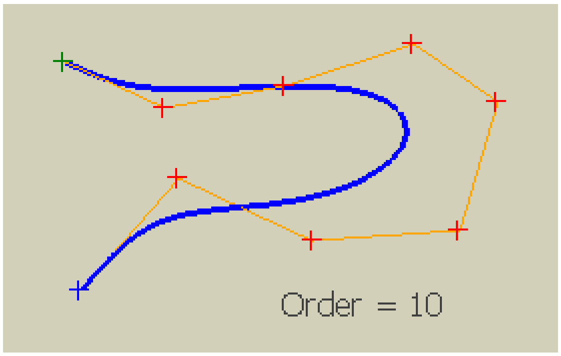
For instance, the curve hereunder is drawn with Order = 3, which makes it follow relatively closely the polygon of control points.



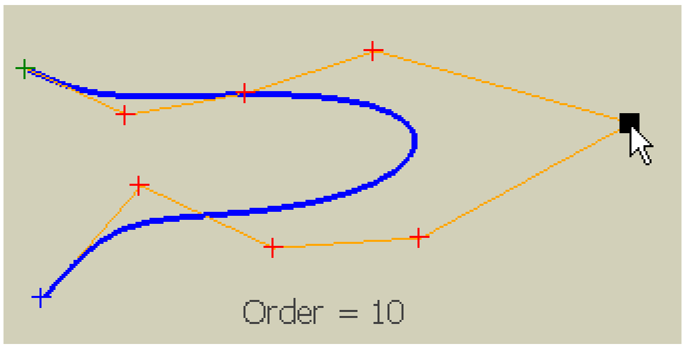
If you increase the Order, you can smooth the curve. For instance, with Order = 6



By increasing the Order, you will increase the smoothing and simplify the shape.



One key characteristic of B-Spline curves is that changes to the position of control points would generate a rather local change on the curve, while keeping the shape 'smooth'.

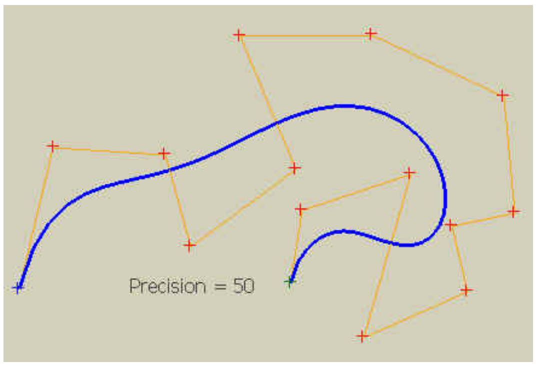


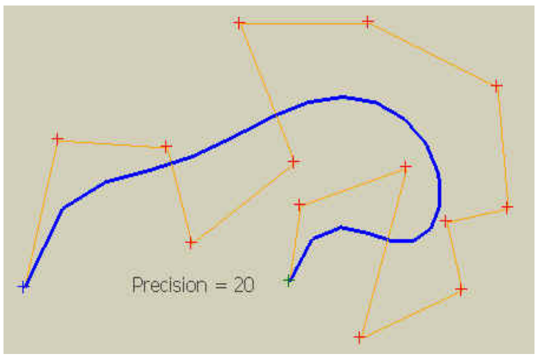
2) Precision

Because in Sketchup, everything is built out of line segments, we have to specify a second parameter, let's call it the **'Precision'**, which deals with how precise we wish to draw the actual representation of the mathematical curve in the model.

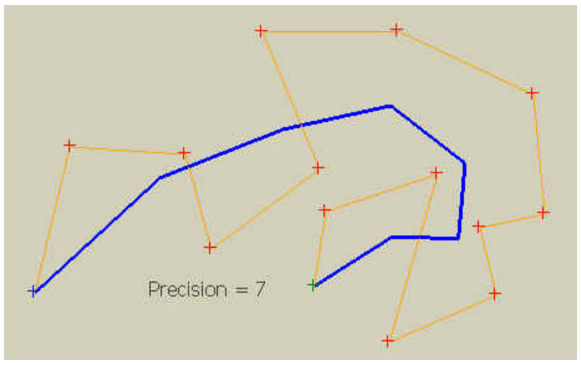
For **Classic Bezier curves**, the Precision is actually the **total number of segments** for the displayed curve. This parameter is independent from the number of control points.

Below is a Classic Bezier curve with 15 control points, shown with a precision of **50**, and the same curve with a precision of **15**. You will notice that the shape is the same, and only the fineness of the representation differs. Playing with the Precision parameter allows adjusting the appropriate visual representation of the model without creating too much polygons[4].





The precision can however go even below the number of control points. Here is the same curve with a Precision of 7.



So low-poly aficionados can play with the Precision parameter, while keeping the general shape of the curve as accurate as possible.

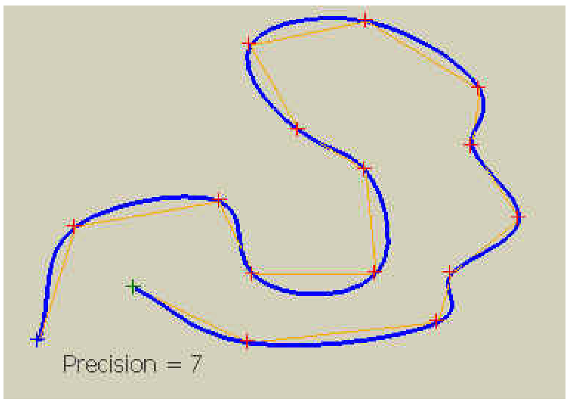
By the way, the modification of Precision can be applied to existing Bezier curves created with the original *bezier.rb* macro, as I arranged to keep compatibility.

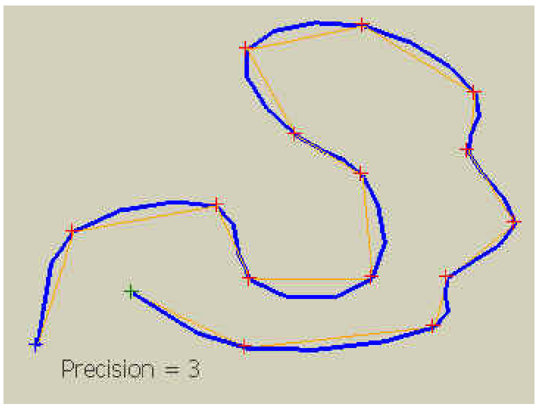
For **Uniform B-spline curves**, the Precision is also the **total number of segments** for the displayed curve.

4 In practice, it is more efficient to change the precision of a Bezier curve than to apply the SimplifyContour macro, which may alter the shape.

For Cubic Bezier spline curves, the Precision is the **number of segments between 2 control points**. So if you have 10 control points and a precision of 5, the total number of segments of the curve object in your Sketchup model will be (10-1) * 5 = 45. By default, the Precision is set to 5, which gives a rather nice visual effect, but may end up with a rather high number of segments for a complex curve.

Here too, playing with the Precision parameters allows you to fine-control the resulting curve as a compromise between visual effect and number of segments. Below two versions of the same curve constructed with 16 control points with *Precision = 7* (so 15 * 7 = 105 segments) and a *Precision = 3* (so 15 * 3 = 45 segments).



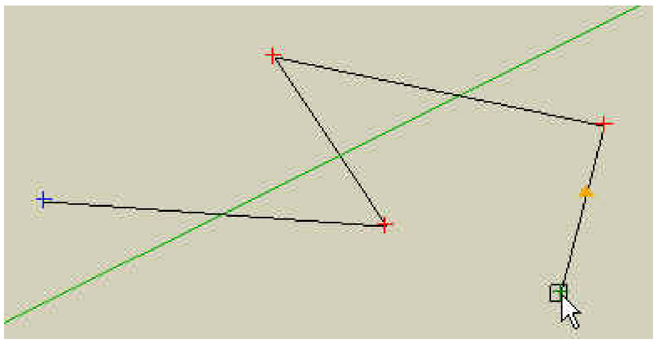


3) A word about Polylines

When I redesigned the *bezierspline.rb* ruby macro, I realized that it was possible to separate the mathematical algorithm from the graphical management of the curve edition in Sketchup. So I designed an extension environment to allow other contributors developing new mathematical curves and plug them to the macro[5].

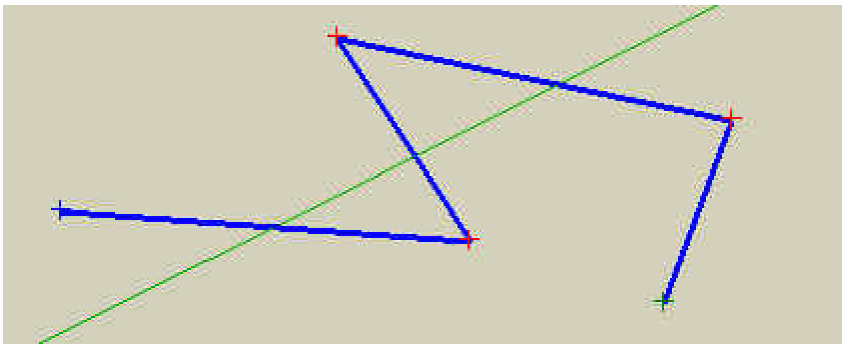
Looking for an example, I found out that the simplest version of a mathematical curve would be interesting and came up with the Polyline. Actually, a Polyline curve is strictly the curve defined by its control points.

The first interesting property is that you can move around any vertex o the Polyline, add or remove vertex. For instance, you draw a polyline with 4 control points….

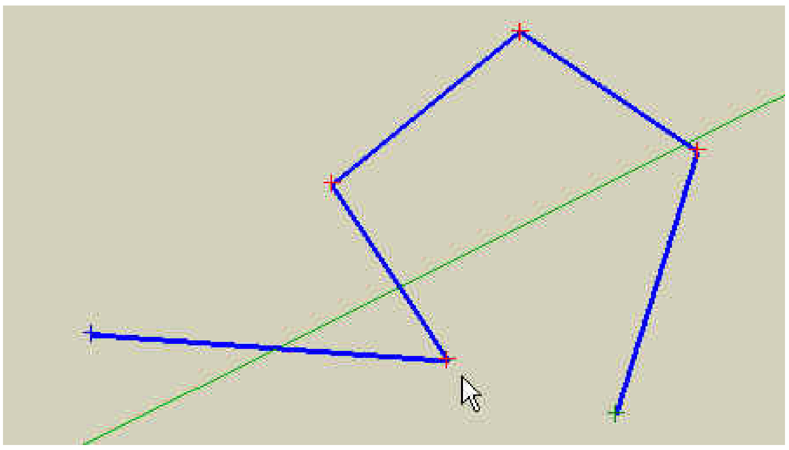


which will give you the following Sketchup object:

5 See a separate tutorial for Ruby developers on how to extend the macro.



You can then add a new vertex and move it:



The second interesting property is that you can convert any Sketchup curve to a Polyline, so that you can get the flexibility for editing the vertex. Polylines can also themselves be converted to splines.

4) Loop curves

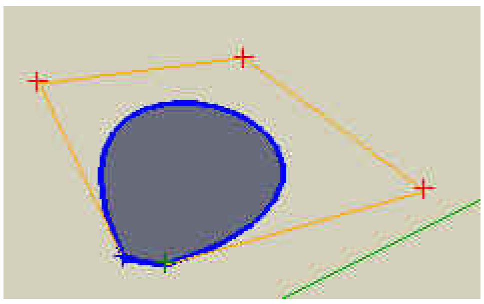
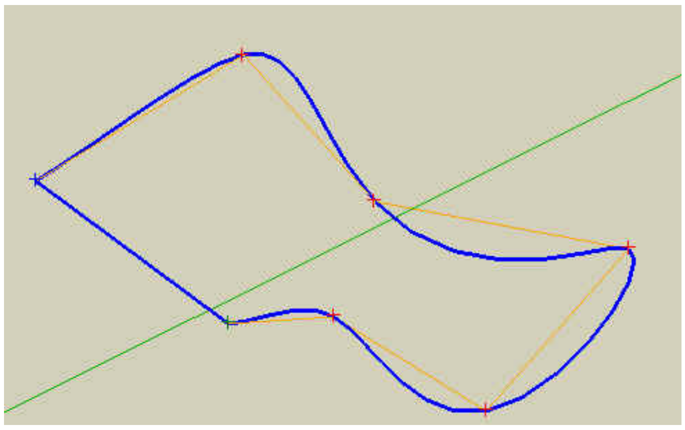
A priori, you CANNOT use the above Bezier and Cubic Bezier curves to draw closed curves, that is, loops, even if you arrange that the end point is equal to the start point. This is because nothing in the mathematical algorithm would ensure that the first and last segments are smoothly connected (i.e. tangency is respected). In addition, it is not so obvious to place the end point exactly at the same location as the start point.

There are some algorithms that can manage this constraint of 'nice' loop closing for spline and Bezier curves. Unfortunately, I did not find any.

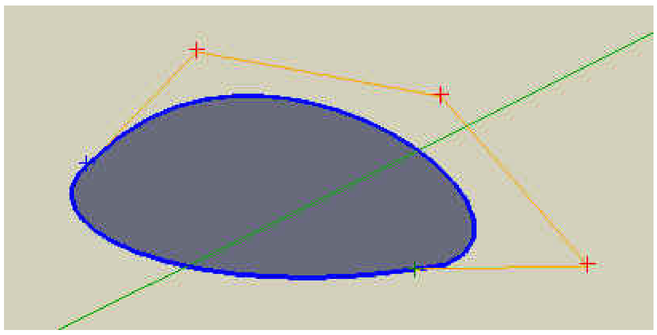
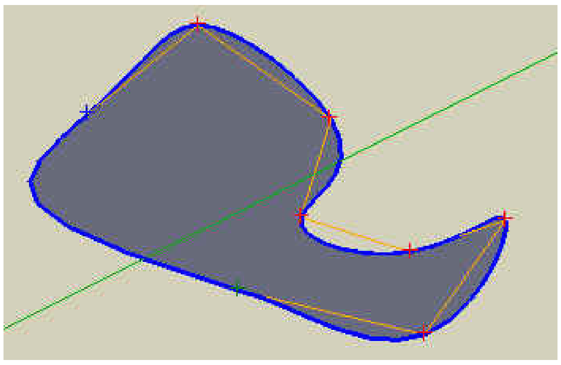
So, to give a taste of what it could be, I built a mechanism to manage loops explicitly, which ensures that there is only a single point for start and end, not two points in the same location. If your curve is on a plane, this will also ensure that your curve delimits a surface.

Closure of a curve can be of two types:

- **By a single segment**, which will make it for instance if your start- and end-points are close enough, or if it corresponds to what you want to draw.

- **By a 'nice' loop losure**. I did not find a general method to guarantee the closure is 'that nice', but just implement a quick algorithm that uses a portion of Bezier curve to close the loop to make sure the loop portion remains tangent to the start and end point.

The macro *bezierspline.rb* will allow you to optionally close your curves when drawing them. Then if you move around the control points, the 'nice' closure will be maintained. I also provide a way to adjust the number of vertices of the closure.

Be careful however that my simple algorithm is not perfect and may not give the result you expect.

5) Other mathematical curves - Extensions

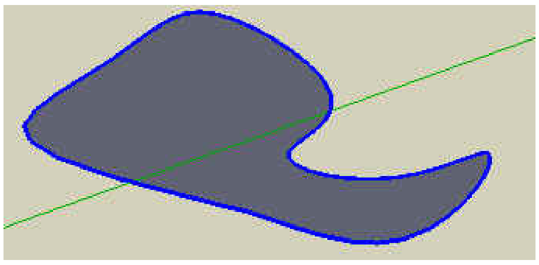
Bezierspline.rb supports an extension environment allowing anyone to incorporate a mathematical algorithm as a plugin (I mean anyone having some minimum experience in Ruby!).

I hope there will be contributors with mathematical skills who will develop additional mathematical curves. There are a lot of possibilities, such as B-spline, beta-spline, regressions, and other types of polynomial algorithms.

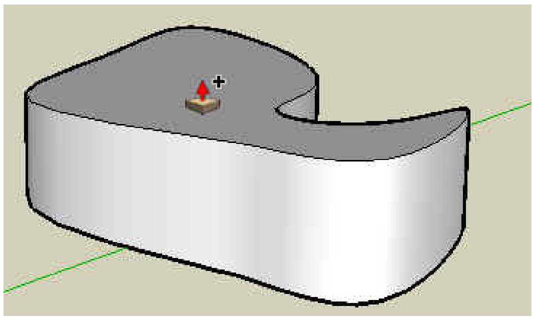
6) Curves connected to surfaces

Finally, it is very likely that your curves will delimit surfaces. When editing the curve, Sketchup will automatically reshape the surface based on your modification.

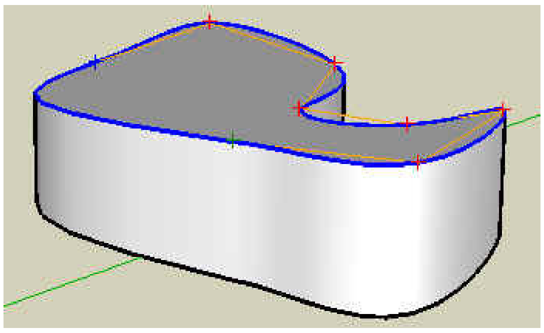
For instance, let's assume you draw the following Bezier curve.



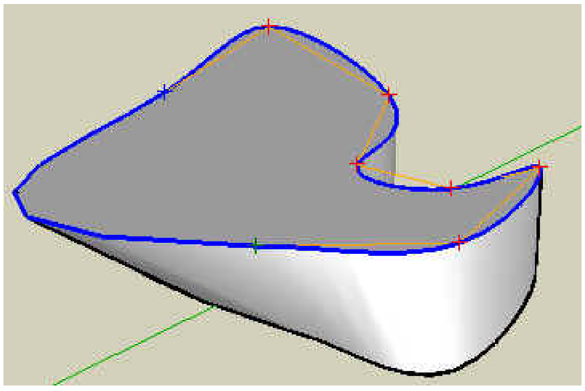
You can then push / pull this face up.



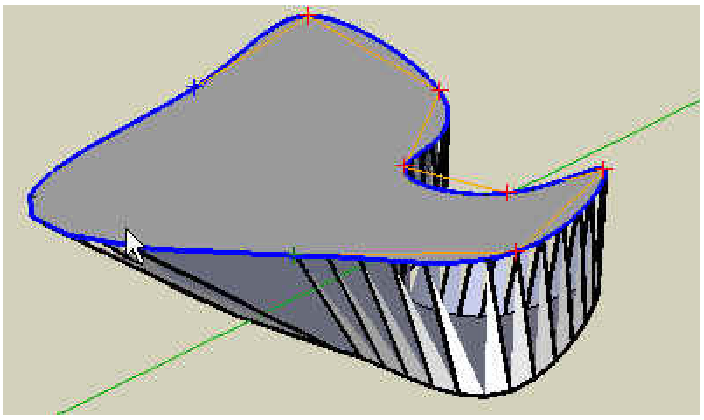
The upper face is delimited by the same Cubic Bezier spline curve, which can then be edited:



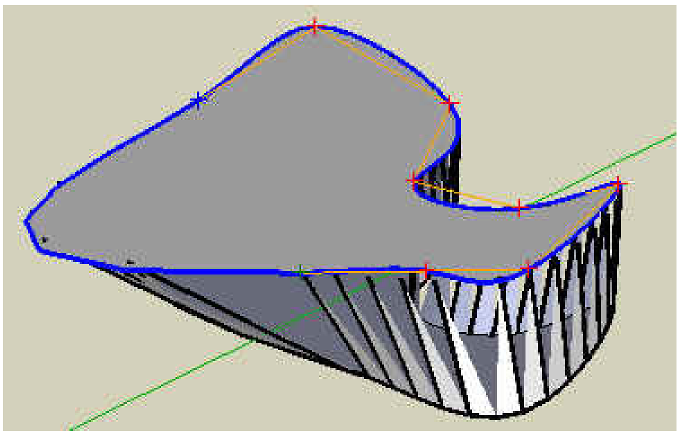
If you move the control points around, Sketchup will reshape the lateral faces accordingly:



Be careful however **that will only work if you do NOT do any operation that would change the number of vertices of the curve**. If you changed the Precision parameter, then, this is what would happen:



…same bad effect if you add or delete control points.



So, before you use spline curves to shape volumes, you should think of the Precision and Control Points you need!

2. Installation

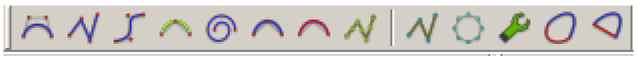
The package contains several files that should be extracted and dropped in the Sketchup Plugins folder:

1. **bezierspline.rb**: this is the main macro, containing the **master tool**, the **Classic Bezier** curves and the **Polyline** curves.
2. **LibTraductor.rb**: this is the utility macro, in charge of managing the language translation of plugins. I separated it, as it can be useful to other scripts.
3. **A subfolder BZ_Dir_12**, which contains
   - the extensions for specific curves
   - A set of icon files (extension PNG) which are used for the toolbar
   - documentation files in PDF

Normally, you would download a Zip file, that, when unzipped in the Plugins directory of Sketchup (using Select All and then extract), will automatically install the files in the right place.

Anyway, the files numbered 1 and 2 above, as well as the subfolder BZ_Dir_12 should reside in the Plugins directory of your Sketchup version (v5 or v6, Free or Pro). If you remove the files ***BZ__....rb*** from the subfolder, the corresponding tools will not appear in Sketchup. If someone publishes an extension **BZ__...**, then copying it to the BZ_Dir_12 subfolder will automatically make it available to Sketchup as an extension of the b*ezierspline* macro (that's the power of naming conventions!).

Toolbar:



It contains the curves type you can draw (left side), as well as some options available when in Edition mode Edit Curve, Show vertex, Extra Parameters, Nice Closing, Segment Closing). Due to a bug in Sketchup, I finally did not gray out icons when not applicable.

Note on language translation:

The macro will normally adapt to the language of your operating system. If it is a French version, the macro will automatically display texts in French[6].

[6] Actually this is the only language version I did myself, but I hope kind souls will provide translation to other languages (I arranged to group all strings in a declarative section of the macro file).

3. Creation Mode

When you start Sketchup, you will find the Bezier family curves in the **menu "Draw"**. By selecting a menu item, you'll see a small black square[7] and be ready to start drawing.

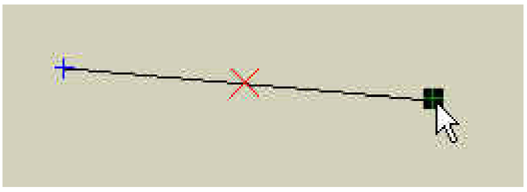
1) Drawing Mode

Curves can be drawn in 2 ways: **'Start / End points'** or **'Open-Ended'**. Each curve comes with its own default method for drawing (but you can change it).

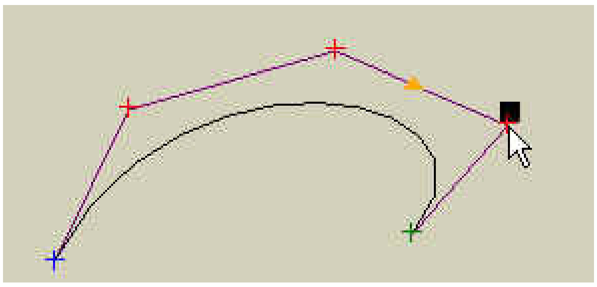
Start / End points Mode

You click on the Start point and then on the End point. Then you pick the intermediate points to shape the curve. This is the default method for Classic Bezier curves and the one used in the original *bezier.rb* macro.

When you move to the End point, the segment will show a red cross in the middle, to indicate the mode of drawing and remind you that this segment will not actually belong to the polygon of control points.



Then you just pick the intermediate points by moving the cursor and clicking with the left button of the mouse.



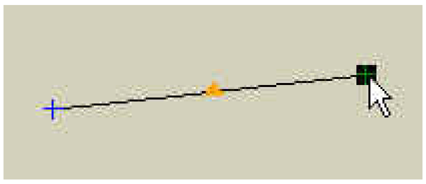
This intermediate points are displayed as red crosses, whereas the Start point is shown in blue and the End point in green. The current segment joining the current and last-entered control points is marked with a small orange triangle.

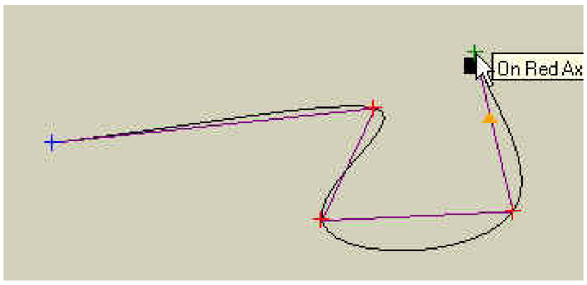
[7] See section on Axis and Plane lock or the meaning of the cursor shape.

Open-Ended Mode

I introduced this second input mode as I think it is more appropriate to draw spline curves and polylines (default mode for these curves). This is actually more natural as you just enter the control points **in sequential order**.

The first segment will be displayed with an orange triangle to indicate you draw in Open-Ended mode.





You can then continue drawing the curve 'naturally'.

Toggling between modes

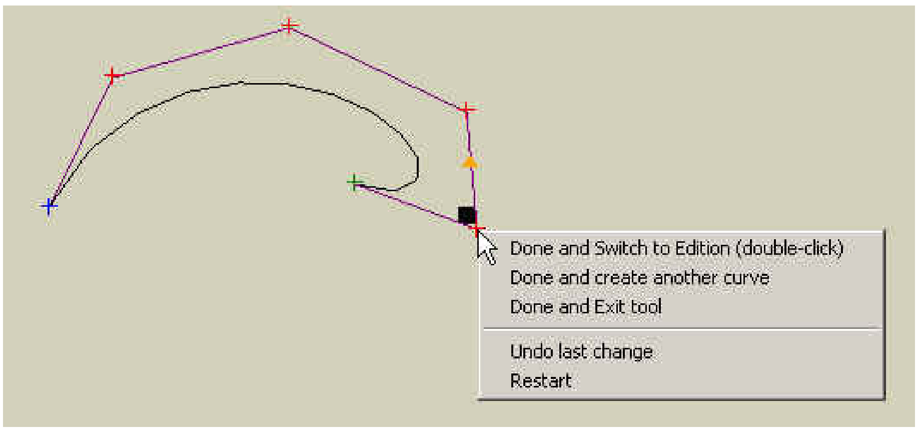
In some occasion, you may want to change the default mode. For instance, even if you draw a Cubic Bezier spline curve, you want absolutely to fix the start and end point first.

To change the drawing mode, you need to **press the Shift key twice quickly** (a kind of double-shift). The change will be indicated shortly in the status bar. For obvious reasons, you need to make your decision quickly, **before you click on the second input point**. Toggling won't work when you are more advanced in drawing the curve.

2) To finish the curve

In both drawing modes, you have several ways to finish the curve:

- **Double-click on the last control point**. This point will be counted and you will switch automatically to Edit mode.
- **Right-click to show the contextual menu**, where you have more options to finish drawing.


Done and Switch to Edition (double-click)
Done and create another curve
Done and Exit tool
Undo last change
Restart

- **In case you reach the maximum number of control points**, as indicated in the VCB and the status bar, the drawing will automatically finish and switch to Edit mode.

3) Number of control points

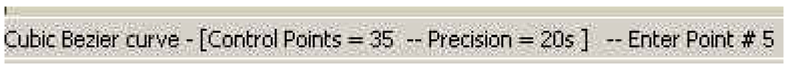
By default all curves do not really limit you on the number of control points you can enter to shape your curve. I put max values high enough that you don't feel constrained. This is also why I had to introduce the double click to finish drawing, as seen above. Remember that the original macro *bezier.rb* had a default of 4 control points (i.e. degree = 3), allowing you entering only 2 intermediate control points.

You can however specify the maximum number of control points, by typing it in the VCB (followed by Enter key). Actually, you can change it at any time provided the new value is greater than the number of points you already picked. Indeed, the macro will prevent you to enter value outside of the valid range (which is set depending on the type of curve).

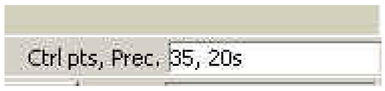
4) Specifying the precision

Each curve has its own default precision, indicated in the status bar and the VCB by a number followed by an 's'. Indeed, keep in mind that the interpretation of this Precision parameter depends on the type of curve you selected.

You can change the Precision parameter at any time during the drawing by typing the new value followed by 's'; for instance, typing '25s' when drawing a Classic Bezier curve will now create a curve with 25 segments. Of course, you can change both the precision and the maximum number of control points in the same command typed in the VCB, for instance '35, 47s'. Note that the separator has to be the comma or semi-column.



```
Cubic Bezier curve - [Control Points = 35  -- Precision = 20s ]  -- Enter Point # 5
```



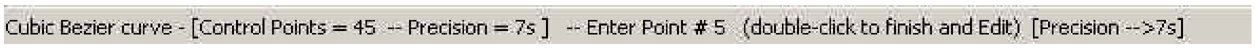
```
Ctrl pts, Prec. 35, 20s
```

Note that some curves may not have a Precision Parameter, such as the Polylines. Also, the macro will check that the new value is within the valid range.

5) The status bar

The status bar permanently display some information about your drawing:

- The type of the curve
- The maximum number of control points you can enter (you can change it, as seen)
- The current Precision parameter
- The current number of vertices for the loop closure, if applicable
- The number of the current point to be picked
- Some indication of tangency at start and end point



```
Cubic Bezier curve - [Control Points = 45  -- Precision = 7s ]  -- Enter Point # 5  (double-click to finish and Edit)  [Precision -->7s]
```

It also show transient messages when you perform some changes. These messages usually disappear after 10 seconds, or when you perform another entry.

Note that the VCB also display the current maximum number of control points and the Precision parameter (if meaningful).

6) Undo, Cancel

Let's be clear that the Sketchup Undo, via Ctrl-Z or the Edit-undo menu will simply cancel everything you have drawn (as it does in the original *bezier.rb* macro). This may be annoying when you have accurately drawn your spline along a complex contour..

The same will happen if you call another tool, usually by their short cut (the most frequent case will be with Space key to go back to selection mode).

As Sketchup scripting environment does not let you cancel these actions, you have to be careful with your reflex.

For all these good reasons I suggest you forget about Ctrl-Z when using the *bezierspline.rb* macro. Instead, I have developed a more friendly way to 'undo' by **pressing the Escape key**, which will simply remove your last entered point and let you continue drawing from the previous point. The option is also available in the contextual menu as *"Undo last change"*.

If you wish however to restart the whole drawing, quickly **press twice the Escape key** (or select *'Restart'* in the contextual menu), and will be back to the beginning, free to choose your first point.

So get used to press Escape, not CTRL-Z, when something goes wrong in your drawing!

7) Plane Lock

In Creation mode, only the Plane lock mechanism can be activated. The objective is to help you to draw the curve along a particular plane, as it is not so easy to do it in a 2D-representation of a 3D space. This said, I will always recommend to create a plane surface beforehand, in order to benefit from the inference *'on plane'* provided by Sketchup.

There are 4 modes of Plane lock:

- ■ **Same plane as the first 3 points**. This mode is indicated by a small black square at the cursor. After you've drawn 3 points, the macro tries to make sure that the other points will be picked on this plane. This is the default mode, as I assumed that most Bezier curves will be planar.

- 🟦 🟥 🟩 **One of the 3 Axis planes**, by pressing the arrow corresponding to an axis. The plane will correspond to the axis which is <u>normal</u> to it. For instance, if you press the Up arrow (vertical 'blue' axis), the plane is the horizontal one. The mark at the cursor will appear as a square filled with the color of the axis, blue, red or green.

  You can deactivate the plane axis lock by pressing the Down arrow (as it is not assigned to any axis).

  The result of a plane lock can sometimes be misleading, as there may be a gap between the cursor and the point which is actually picked. Also, you will loose te Sketchup inference. Just a matter of habit!

  □ Finally, **you can deactivate any Plane lock mode by pressing the Ctrl key once**. The mark at the cursor will become an open square. In this mode, there is no constraint and the Sketchup inferences can be freely used. Note that the Ctrl key allows you to toggle back to Plan lock mode.

I don't pretend this solves all possible situations, but this can be helpful in some circumstances. Be careful however in Plane lock mode to never pick points above the horizon (as Sketchup have no clue there of where the axes are).

8) Loop Mode

If the curve extension supports this capability, you can close your curve at any time in 2 ways:

- **With a single segment**, by pressing the **toggle key F8**, which will actually toggle between closure and unclosure.
- **With a 'nice' closure**, by pressing the **toggle key F9**.

Pressing **F7** will remove closure.

These options are also available in the contextual menu of the curve.

Have a try to see how it works.

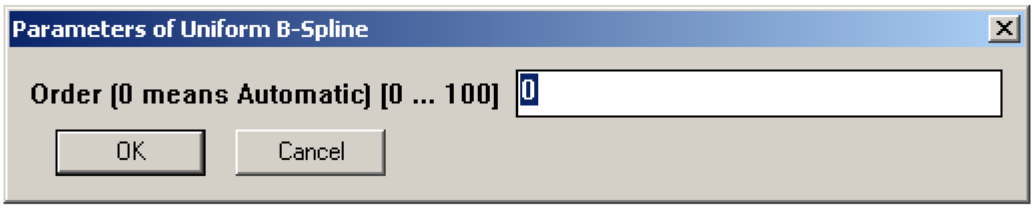
9) Marking the vertices of the curve

You show or hide the vertices of the resulting curves by toggling with the **F5** key. The vertices will be displayed with a small blue circle.

10) Extra Parameters

Uniform B-Splines give you the possibility to specify an additional parameter, called Order. This is a numeric value that will control the smoothness of the curve (the higher, the smoother). The value 0 (automatic mode) always guarantees the smoothest drawing.

At any time during Creation or Edition, you can call the dialog box to set this parameter by **pressing TAB** (or select the option in the contextual menu).


Parameters of Uniform B-Spline

Order [0 means Automatic] [0 ... 100] 0

Otherwise, the dialog box will always appear when you start a Creation of Uniform B-Spline, as well as when you convert a curve to a Uniform B-Spline.

More generally, if someone publishes an extension to bezierspline.rb, then TAB will be the way to call the extra parameter dialog box, if applicable.

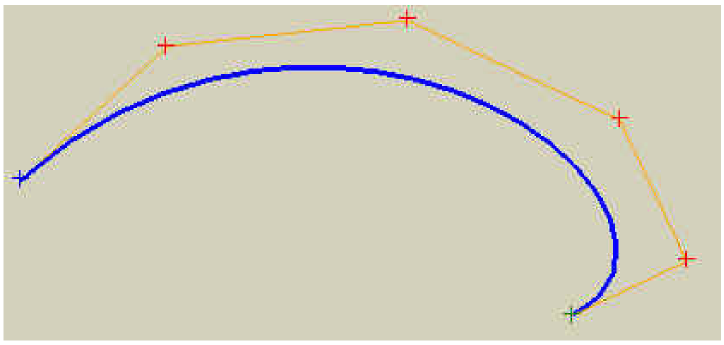
4. Edition Mode

The Edition mode allows you to modify the shape of the curve after you created it. I have added a few features compared to the original macro.

You can invoke the Edition mode by selecting the curve and choose ***"Edit … curve"*** in the contextual menu with a right mouse click.

You also switch automatically to Edition mode when you finish creating the curve.

The working environment in Edition mode displays the selected curve equipped with the polygon of control points drawn in orange. The control points are marked by a red cross.



1) Exiting the Edition Mode

You can exit the Edit mode **by double-clicking anywhere outside the curve or the polygon of control points**. The exit option is also available in the contextual menu. Indeed, you will exit the Edit mode with other standard Sketchup actions, such as choosing another tool.

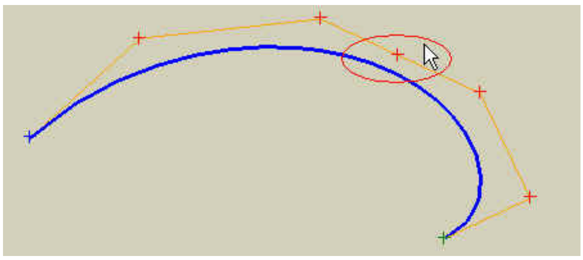
2) Moving Control Points around

You can move any control point (including start and end points) by **clicking and dragging** it to another location. The curve will reflect the change. When moving the point, the macro will take into account the current axis or plane lock if active.

Alternatively, you can **move a segment** of the polygon by clicking on it and dragging it around.

3) Adding Control Points

You can insert a new control point by simply **double-clicking on a segment** of the polygon of the control points. Adding a control point will have an impact on the shape of the curve. So, you would usually move it around after insertion.



4) Deleting Control Points

You can delete a new control point by simply **double-clicking on the point**. Removing a control point will also change the shape of the curve. Note that you can remove the Start and End points if you wish.

5) Changing the Precision parameter

When applicable you can change the Precision parameter by just typing the new value, followed by a 's' in the VCB. The change will be immediately reflected.

6) Axis Lock and Plane Lock

In addition to the Plane lock mechanism seen previously in Creation mode, the Edition mode support a more conventional Axis lock, forcing the move of control points along one of the 3 axis.

To do this you need first to deactivate the plane lock, if active, (via Ctrl Key, to be back to the open square), and then press the **Arrow key** corresponding to the axis. A triangle mark filled with the axis color will show at the cursor. As you can expect, pressing the Down Arrow will cancel the Axis lock.

You can activate the Axis lock even when you move the point with the mouse button down, but note that the reference for the forced move is always the initial position of the point.

Again, if you want accuracy, I suggest you first build your own guides with construction lines and points. Axis lock won't always make it.

7) Undo and Cancel modifications

Unlike what happened in Creation mode, the Sketchup Undo (CTRL-Z) and Redo (CTRL-Y) are supported. However **I recommend you use the Escape key instead**, because it will beep and stop undoing changes when you reach the initial state of the curve when you entered the Edition mode (Ctrl-Z will continue undoing).

By pressing quickly the **Escape key twice**, you will cancel all changes.

The Undo options are also available in the contextual menu.

To Redo the last cancelled change, just press Ctrl-Y.

8) Loop Mode

It works the same as in Creation mode, with the keys F7, F8 and F9.

9) Marking the vertices of the curve

You show or hide the vertices of the resulting curves by toggling with the **F5** key.

10) Extra Parameters

It works the same with the **TAB** key, as seen previously in the section about Creation Mode.

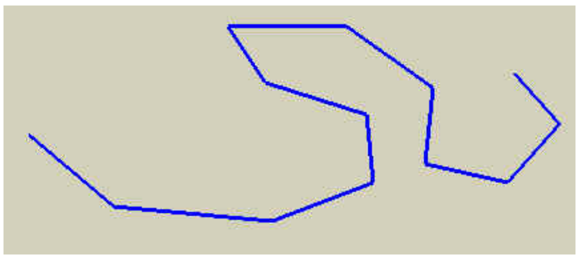
5. Converting Curves

When applicable, it is possible to convert a Sketchup curve into one of the curve of the BZ_... family. This option is normally available in the contextual menu of the selected Sketchup curve.

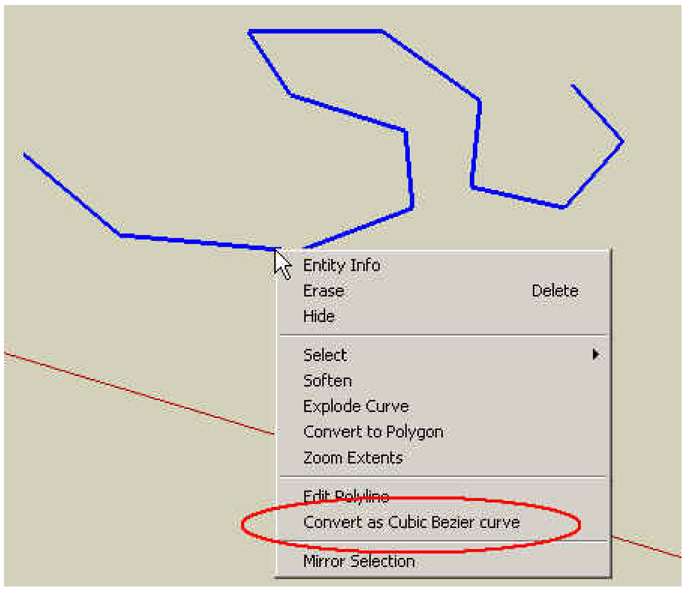
In the version I provided:

- **You can convert any Sketchup curve to a Polyline**. When you have some segments in continuity, you may have first to **'weld'** them (using the macro *weld.rb*).Then you'll see the option to convert them to a Polyline in the contextual menu.
- **Any Polyline can be converted to a Cubic Bezier spline curve** or to a **Uniform B-Spline curve**
- However, I did not find a way to convert a curve to a Classic Bezier curve (as the control points are not on the curve).

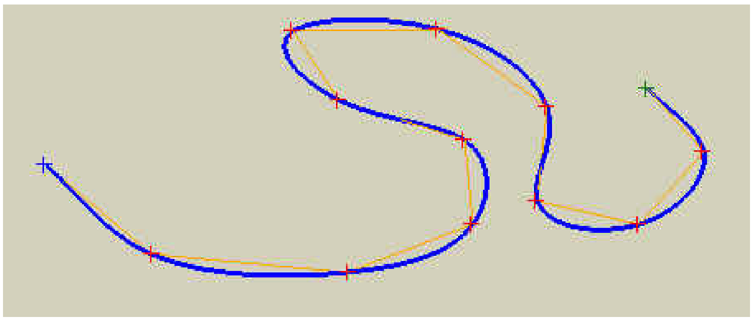
For instance here is a Polyline curve:



When you select the curve and show the contextual menu, you will have a menu item allowing you to convert the Polyline into a Cubic Bezier curve.

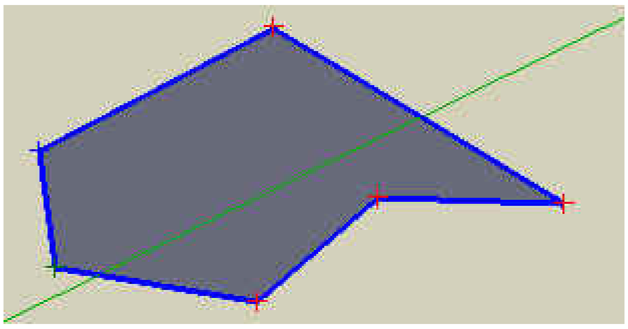


Here is the result, which you can further edit to move, add, delete the control points, or change the Precision parameter.

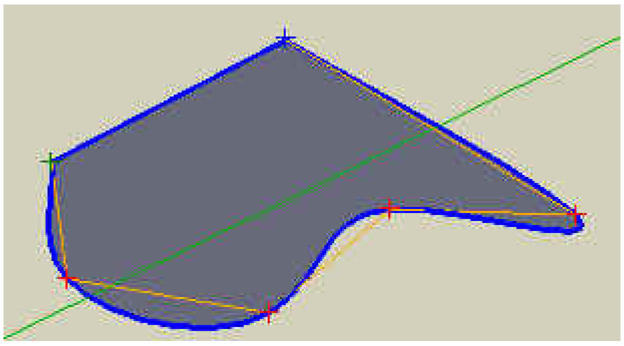


Note that if the curve you want to convert is a loop, then the macro *bezierspline.rb* will normally detect it and activate the loop mode.

For instance, if you start from the following curve…



Then the conversion to a Cubic Bezier curve will give …..



You will then need to manually specify a 'nice' closure by pressing **F9**.