

# Vertex Tool

Easy way to control mask/shape  
corners in After Effects

manual for v1.0

# Vertex Tool

Round that corner with a click of a button

Vertex Tool applies rounding algorithms to a shape by placing two (in some cases more) new vertices in place of original one at selected corner, based on a specified line length. There are 12 corner rounding algorithms to choose from.

## Installation

Follow these two simple steps to install script:

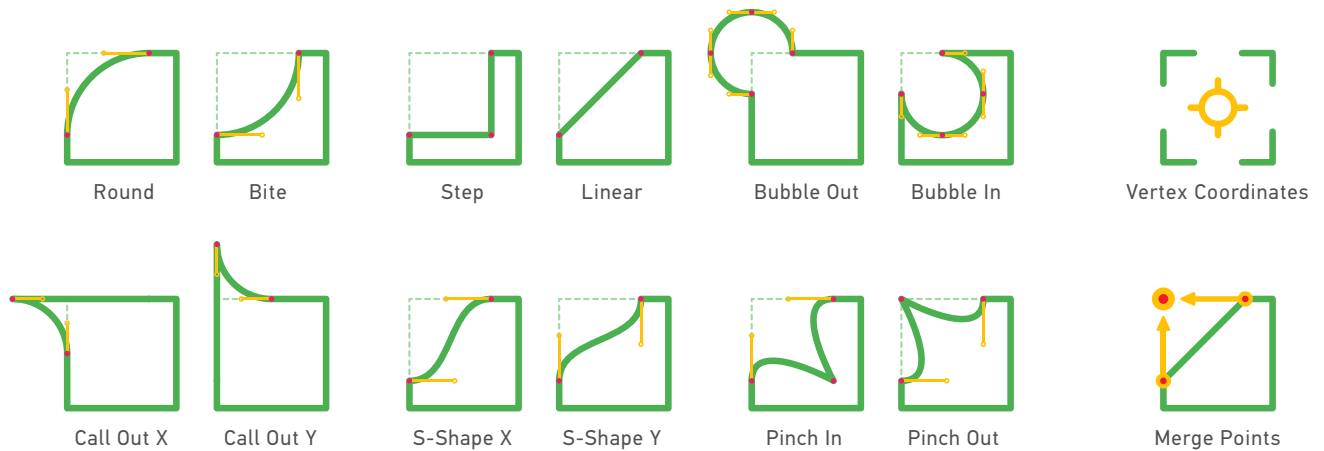
1. Unpack the archive you have downloaded and copy / paste files (both “Vertex Tool.jsx” and “Vertex Tool Help.pdf”) to “ScriptUI Panels” folder:
  - Windows: **Program Files\Adobe\Adobe After Effects <version>\Support Files\Scripts**
  - Mac OS: **Applications/Adobe After Effects <version>/Scripts**

If folder **ScriptUI Panels** does not exist, create a folder and name it “ScriptUI Panels”. Then paste the copied files into it.

2. Allow script to access network to avoid unnecessary problems while loading GUI. This option is under **General** tab of After Effects **Preference** pane:
  - Windows: **Edit > Preferences > General**
  - Mac: **After Effects > Preferences > General**

Once Installation is finished run the script in After Effects by clicking **Window > Vertex Tool**

# Interface



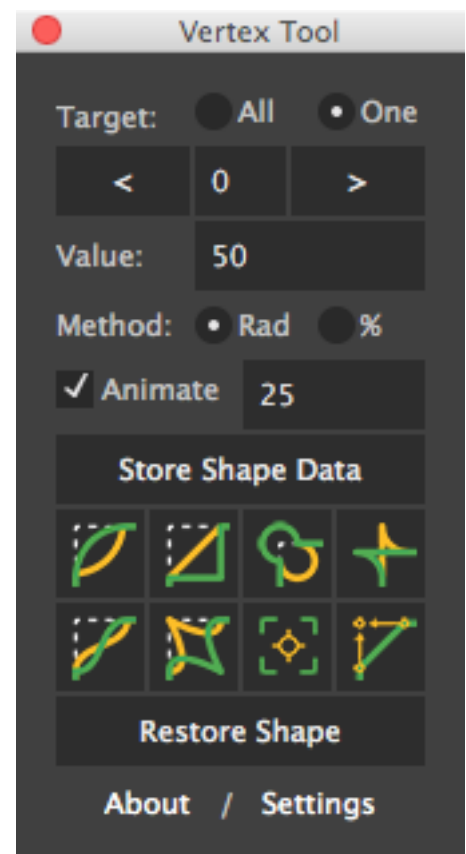
There are 12 rounding algorithms:

- Round
- Bite
- Step
- Linear
- Bubble Out
- Bubble In
- Call Out X
- Call Out Y
- S-Shape X
- S-Shape Y
- Pinch In
- Pinch Out

and 2 action buttons:

- Vertex Coordinates
- Merge Points

Main rounding shapes (green stroke) are accessed by mouse click, while alternate shapes (yellow stroke) by **Cmd + click** (Mac) or **CTRL + click** (Win).



# How it Works

Select a path on Shape layer or Mask on Solid layer and launch the script.

Select **Target: All** when applying rounding method to all vertices of your shape or use **Target: One** to act on one specified vertex. If second method is used, a target object will appear in the composition, that represents vertex in focus. Use buttons **<** and **>** to navigate to your vertex of choice (**Shift + Click** and **Cmd + Click** (Mac) or **CTRL + Click** (Win) skips predefined number of vertices) or enter vertex number manually.

Enter rounding value and choose **Radius** or **Percentage** method. If rounding value is greater than the distance between selected and two neighbouring vertices, the minimum distance between neighbouring vertices will be used as rounding value.

If you are going to transition from original shape to rounded one, then enable **Animate** option and set morphing duration in frames. This will ensure that transition goes as smooth as possible. Failing to enable this checkbox will result in weird, wonky and undesired transformation.

Do not forget to make a copy of your shape when dealing with complex shapes. **Store Shape Data** options does similar thing - it reads shape data (position of all vertices and in/out tangents) and stores it in expression field. Later, if you need to restore original shape, you can do so with **Restore Shape** button. Original shape and keyframe will be created at time, at witch it was recorded.

Click on an icon of your choice to apply default rounding method (green stroke). Use **Cmd + Click** (Mac) or **CTRL + Click** (Win) to apply alternative shape (yellow stroke).

**Set Custom Vertex Coordinates** will open up a dialog where you can specify coordinates for selected vertex. You can also define in/out tangents and choose either **Layer Space** or **Screen Space** coordinates system.

Interpolation:		
Vertex Coordinates	300	0
inTangent	-10	10
outTangent	10	-10

At the bottom are 'Cancel' and 'OK' buttons.

**Find Intersection Point** will calculate intersection of two lines:

**First line:** Current Vertex - 1 to Current Vertex;

**Second line:** Current Vertex + 2 to Current Vertex + 1.

If these two lines intersect at any point in space, then two vertices (“Current Vertex” and “Current Vertex + 1”) will be repositioned there. This option is useful when you need to transition from rounded to a sharp corner.

**Set Custom Vertex Coordinates** and **Find Intersection Point** options work if **Target: One** is selected.

A = Current Vertex index-1;

B = Current Vertex;

C = Current Vertex index +2;

D = Current Vertex index+1;

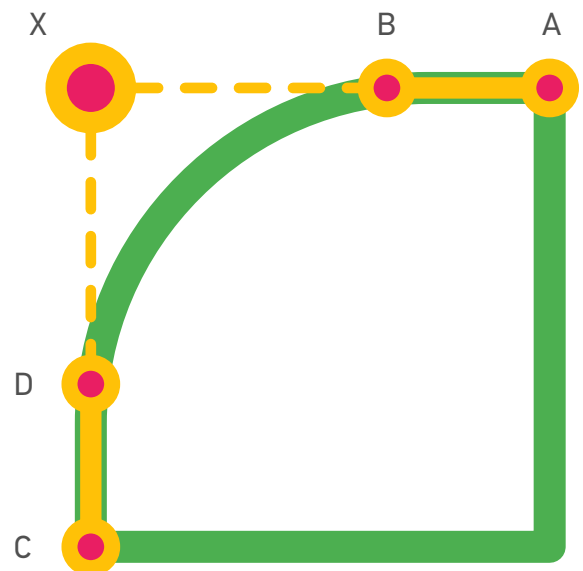
First Line = AB;

Second Line = CD;

BX = AB lines extension;

DX = CD lines extension;

X = intersection point of AB and CD lines



## Settings

**Target size** - target gizmo size in pixels;

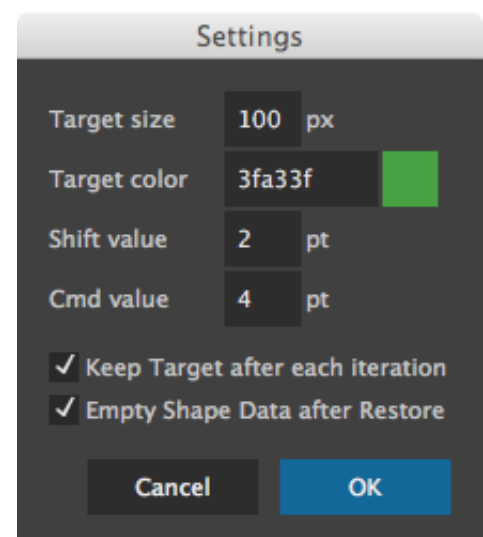
**Target color** - the color of that get gizmo;

**Shift value** - Shift button to skips predefined number of vertices;

**Cmd value** - Cmd/CTRL button skips predefined number of vertices;

**Keep Target after each iteration** - after rounding is finished, in case of **Target:One**, target gizmo will not be removed from composition;

**Empty Shape Data after Restore** - empties expression after **Restore Shape** is finished;



## Under the Hood (simplified version)

For instance, you have a shape with 4 vertices:

A, first vertex (index 0),	C, third vertex (index 2),
B, second vertex (index 1),	D, forth vertex (index 3).

Now, when creating a new vertex between first and second vertices (between A and B (index 0 and index 1)), After Effects automatically assigns Index 1 to this new vertex and shifts all later vertices indexes by one. You get result that looks like this:

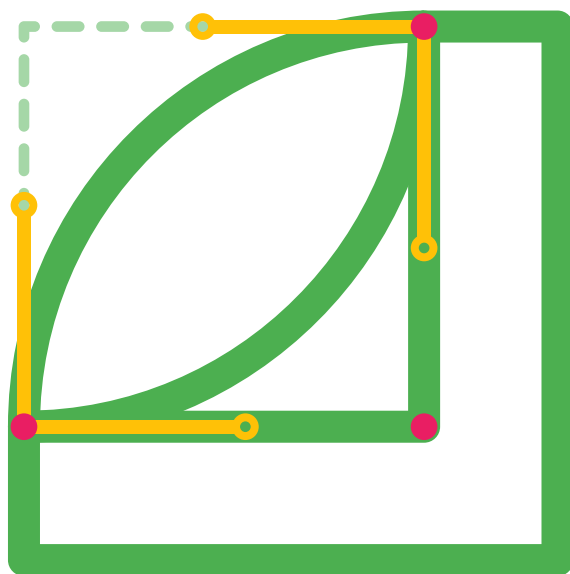
A, first vertex (index 0),  
NEW VERTEX, second vertex (index 1),  
B, third vertex (index 2) that used to be “second vertex with index 1”,  
C, forth vertex (index 3) that used to be “third vertex with index 2”,  
D, fifth vertex (index 4) that used to be “forth vertex win index 3”.

Because of this way After Effects sets indexes to new vertices it is crucial to understand, that transitions from one shape to another is not that simple. However, if you need to morph between original shape and rounded one, you should use **Ani-mate** function - it is executed in similar way I described below.

For the script to compensate indexing behaviour, each time you apply rounding method that is defined by 2 vertices (all except **step**, **bubble**, **pinch**), it:

1. Creates new vertex point (NVP) at index position “Current Vertex - 1” and sorts the index array;
2. Sets NVP coordinates position to be equal to “Current Vertex”. That is:  
$$\text{NVP.X} = \text{CurrentVertex.X} \quad \text{NVP.Y} = \text{CurrentVertex.Y}$$
3. Assings NVP.inTangent = CurrentVertex.inTangent and resets CurrentVertex.inTangent to [0,0];
4. Removes old shape and creates a new one with these new values;
5. “Slides” NVP and CurrentVertex points on Bezier curve by distance, calculated from Rounding Value;
6. Shortens tangent handles by value calculated from Rounding Value with De Casteljau algorithm.
7. Removes shape created at stage 4 and creates a new one, based on latest values.

This workflow is not necessarily the exact way the code works, but it mimics the logic of the code.



# Vertex Tool

by Tomas Šinkūnas  
[www.rendertom.com](http://www.rendertom.com)