Vertex Tools Pro

v1.1.0

---

documentation

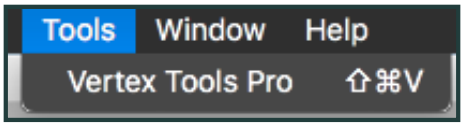
Content

1. Getting Started

Vertex Tools Pro (**VTP** from now on) is a suite of different tools to manipulate your meshes. You can paint the vertices of a mesh, deform the mesh vertices or create flowmaps. In the next few chapters we will describe the functionality in detail.
Nevertheless we suggest you check out our 6-part video tutorial at:

https://goo.gl/HAocX3

To really get started, we first have to launch VTP. To do so, click the ‚tools' menu at the topbar of Unity and select Vertex Tools Pro. You should now see the Vertex Tools Pro Editor window.



To be able to use VTP on one of your gameobjects you first have to select one. After selecting one, VTP should give you the option to initialize the gameobject and therefore to enable the use of VTP.

Click ‚Initialize everything now' and continue with any of the following chapters.

2. The Tools

2.1 Painter Interface

The interface is split up into 3 parts.

At the top you can define some General Settings:

- **Autofocus:**
  The object gets focused when starting to paint the mesh

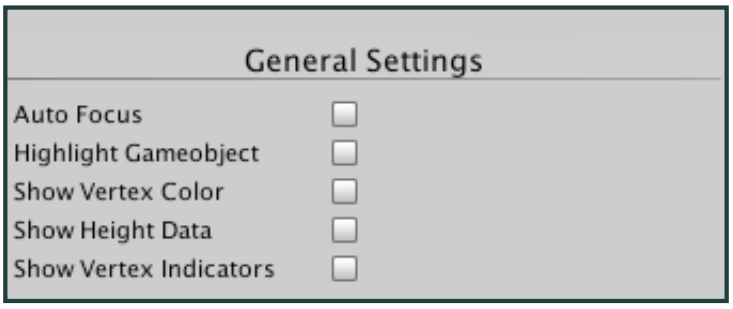
- **Highlight:**
  The object gets highlighted (non-selected objects get grayed out) when starting to paint the mesh.

- **Show Vertex Color:**
  Shows the vertex colors of the selected mesh. *(Only works when a material with a VTP Shader is assigned)*

- **Show Height Data:**
  Shows the height data of the selected mesh. *(Only works when a material with a VTP Shader is assigned)*

- **Show Vertex Indictators:**
  Shows the affected vertices while drawing.

- **Vertex Indicator Size:** *(Only visible when Show Vertex Indicators is true)*
  Adjusts the size of the vertex indicators.



Below you can find the Brush Settings:

- **Brush Size:**
  The size of the brush which is used to paint.

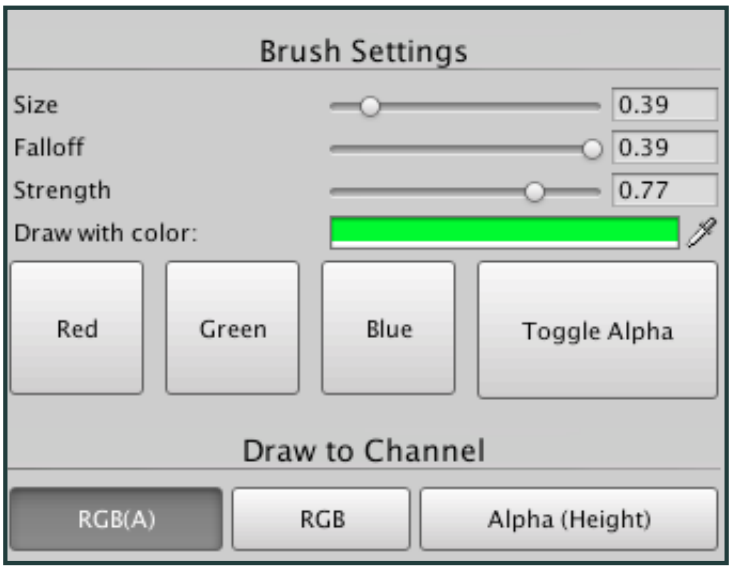
- **Brush Falloff:**
  The falloff of the brush which is used to paint.

- **Brush Strength:**
  The strength of the brush which is used to paint. A higher value means a faster replacement of the current vertex color.

- **Draw Color:**
  The color of the brush which is used to paint. Red, Green and Blue represent the individual texture channels of VTP shader.

- **Red, Green, Blue and Toggle Alpha button:**
  Shortcuts to set Draw Color to Red, Green or Blue or to toggle the Alpha value.

- **Draw To Channel:**
  You can choose which channel to draw to. When selecting e.g. the Alpha channel only Alpha is affected during painting.



At the bottom you can find the actual painting commands:

- **Start Painting:**
  As it says, it starts the painting process. You can now draw the vertex colors in your scene view.

- **Save Painting:** *(Only visible when in paint mode)*
  Saves the current state of the mesh.

- **Cancel without save:** *(Only visible when in paint mode)*
  Cancels the painting process and reverts back to the old state.

- **Undo and Redo:** *(Only visible when in paint mode)*
  Undo or Redo painting steps.

- **Flood Fill with Color**: *(Only visible when in paint mode)*
  Fills the entire mesh with the selected Draw Color.

2.2 The Painting process

2.2.1 Painting

After initializing the gameobject the next step is to assign a material with one of our VTP shaders. You can find these under VTP in the shader menu. After you assigned the material you can immediately start painting. Select the gameobject and paint in the scene view. The actual painting is done by clicking and holding the left mouse button and draw over the vertices. Do not forget to save your mesh after you are done.

2.2.2 The Shader

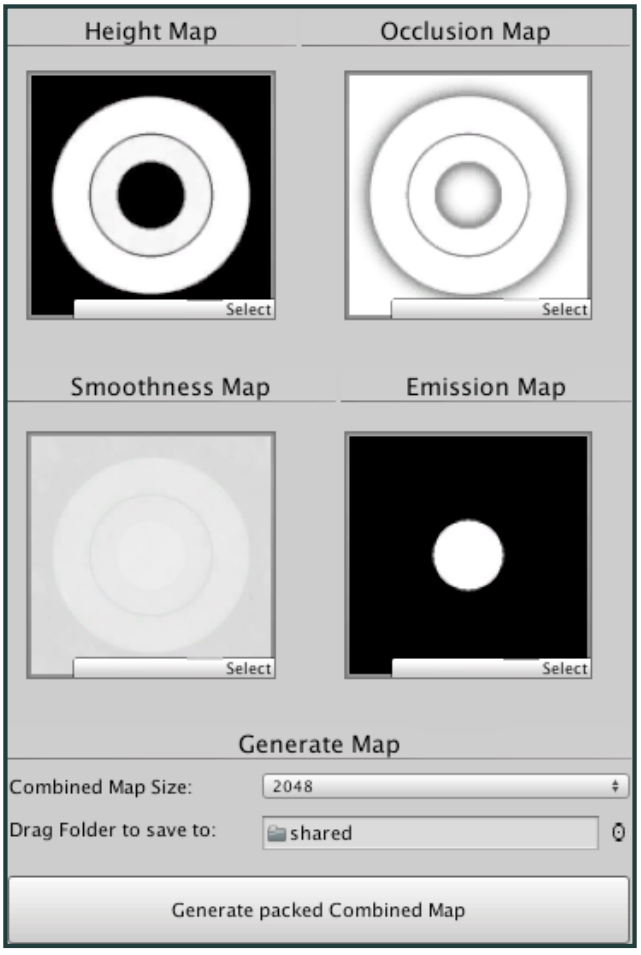
Our Shaders support both metallic and specular workflow. Furthermore we use packed textures to increase performance as much as possible. We will therefore start with the creation of the packed textures.

2.2.2.1 Packing Textures

In order to make the creation of the packed textures as easy as possible we provide you with a texture assistant. You can find it in the VTP window at the top under ‚Texture Assistant'.

First, if you are using metallic workflow you have to combine your grayscale metallic map and your albedo map. Just drag the albedo and metallic map into the desired slots, choose the output size and the output directory and click „Generate". Your packed texture will be generated and can afterwards be dragged into the shader slot.

Secondly, you have to pack a „Combined Map". This includes a heightmap, an occlusion map, a smoothness map and an emission map. As you did when packing the albedo and metallic map, just drag the maps into the respective slots, choose the output size and the output directory and hit „Generate". The packed texture will be generated and can afterwards be dragged into the shader slot.



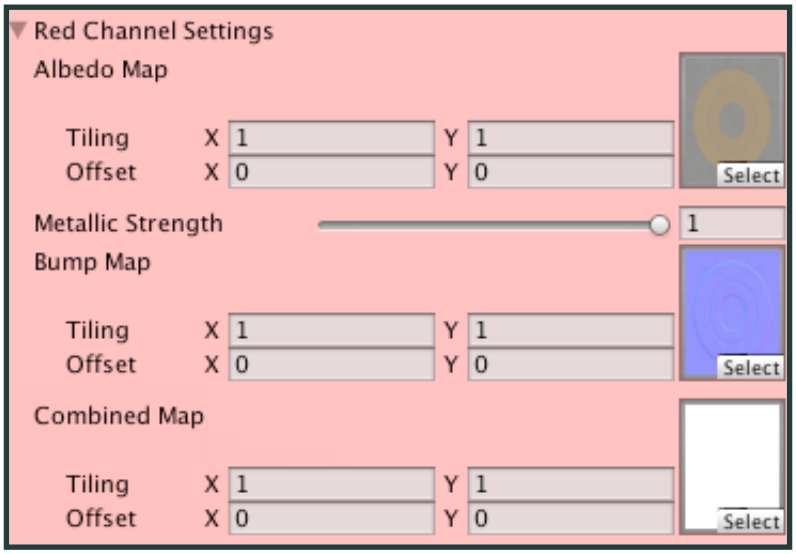
2.2.2.2 Shader Settings

Our shaders come with custom Parallax Mapping, Heightbased Blending and Flow support. Nevertheless we will begin with the basics of our shaders.

The Basics

In the inspector you will find settings for three channels (red, green, blue). Each channel has a texture slot for the packed albedo (see *2.2.2.1 Packing Textures*), a normal map and a packed combined map (see ***2.2.2.1** Packing Textures*). The specular shaders also have a slot for a specular map.

Furthermore each map has a slider to determine its strength. When you start painting vertex colors, the corresponding channel is used. So for example, if you paint your mesh with pure red, the textures and settings of the red channel will be used.

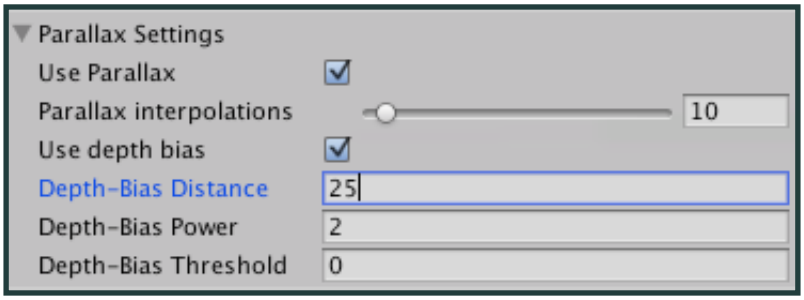


shader slots of the red channel of our metallic shaders

Parallax Occlusion Mapping

To further enhance visual quality our shaders support **Parallax Occlusion Mapping**. This has to be enabled in the material setting's **Parallax Setting** foldout.

You will notice, that once you enable it more options will be visible. The first one lets you set the **interpolations** used. We suggest values between 7 and 20. Keep in mind that this setting has a huge impact on performance. The more interpolations you use, the less FPS you'll get. To compensate this, you have the option to use a **depth bias**. The depth bias defines a distance in which the parallax will be calculated and rendered. The **Power** determines the falloff. If you have a power of 1 you will get a linear falloff of Parallax. The **Threshold** enables you to disable the calculation of the parallax effect even before the distance is fully reached. Nevertheless we suggest 0 as the default value.



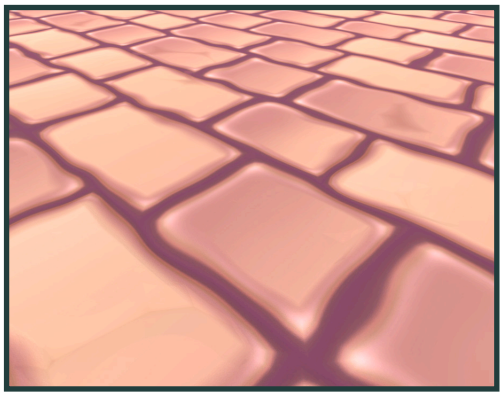
Once you enabled Parallax every Channel (red, green, blue) shows more options – the **Parallax Strength**. This is where you can define the strength of the parallax per channel. Negative Values mean the heightmap is being intruded, positive values mean the heightmap is extruded. For visually best results we suggest negative values or slightly positive values for more contrast. Note that setting the Parallax Strength to 0 also disables Parallax calculation for that particular channel, which again, results in a better performance.

The Green and Blue channel also has an option to **use the baselayer's (Red Channel) heightmap**. This means the parallax for that particular channel is not calculated from its own heightmap, but instead the shader uses the heightmap of the Red channel. This is great for thin layers, which are only supposed to cover the baselayer and which should not fill cavities of the baselayer.

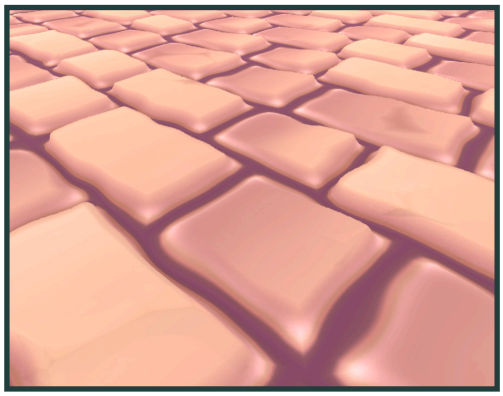
Short version: Parallax greatly increases the visual result, but it has its downside on the performance.

Therefore be sure to:

- Only use Parallax on channels which really need it
  (Otherwise set the Parallax Strength to 0)
- Adjust the interpolations.
  When working with low strengths, 5 interpolations can be enough
- Definitely make use of the depth bias.
  It does not make sense to render Parallax far away.



texture without the use of parallax



same texture with parallax

Heightbased Blending

Another way to greatly increase visual quality is the **Heightbased Blending**. When Heightbased Blending is used, vertex colors do not just get blended. For the calculation of the blending the height maps of the channels are taken into account. We use the red channel as our baselayer. So the green and blue channels are laying on top of the red channel and the blending is calculated with reference to the corresponding height maps. You can enable Heightbased Blending at the shader's General Settings foldout.

After enabling it, you will see additional Heightbased Blending settings in the green and blue channels.

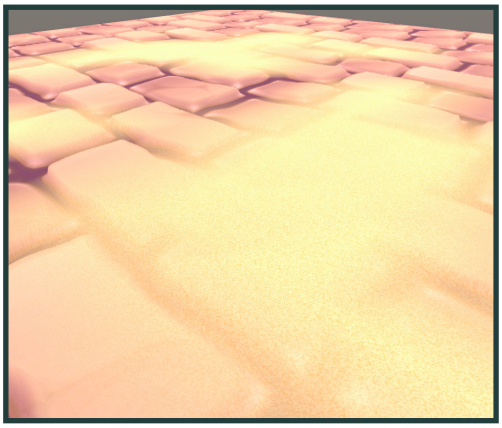
- **Baseheight of Layer:**
  This defines the base height of a layer. The higher the value, the more the layer will be drawn on top.

- **HeightMap Multiplier:**
  This defines how much the height map of the current channel shall be taken into account when calculation the blending. A value of 0 means, that the channel will be seen as a flat layer, while a value of 1 means, that the values of the channels height map will be added to the Base Height. This is a great option to further increase visual variety.

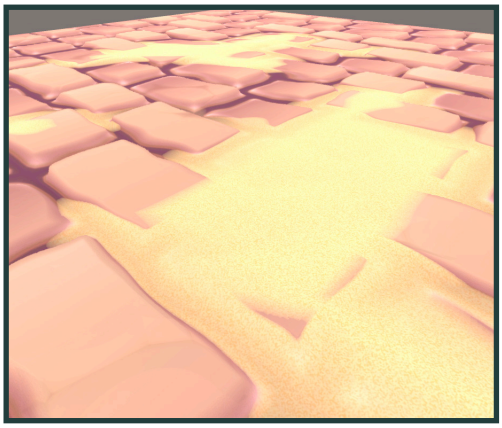
- **Edge Blend Strength:**
  This defines the amount of blending at the edges. Water for example tends to blend at the edges, while a concrete material would look way more realistic with an Edge Blend Strength near 0.

- **Heightbased Transparency:**
  This defines the transparency based on the depth of a blended layer. It basically calculates the height difference between the channel and the Baselayer (Red channel) and determines the transparency. Water for example makes good use of Heightbased Transparency, while a concrete material would again look way more realistic with a Heightbased Transparency near 0.

To further increase the visual difference of a painted mesh, the calculated height is multiplied with the Alpha of the vertex colors. This means that one layer does not only use the same height but you can literally paint the height at each vertex. That is why we also suggest to draw the baselayer (Red Channel) with an Alpha value of 0.



heightblending disabled



heightblending enabled

FlowMapping

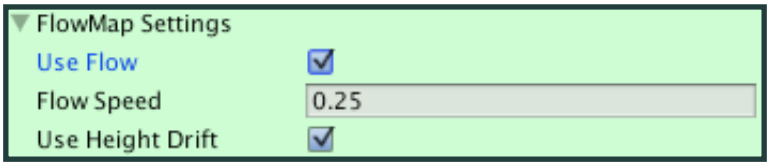
Our shaders support flow. First we will show you how to enable it in the shader. The actual process of supplying flow data to the shader will be discussed in the following chapter. FlowMap Support can be enabled in the shader's General Settings foldout. Once enabled you will see another option ‚Only Flow Normals'. When this is checked, only the normal maps of the layers (Green and Blue, Red is declared as the BaseLayer and can therefore of course not flow) flow, otherwise also Packed Albedo and Packed Combined Maps flow.

Additionally you will find new options in the green and blue channel Settings.

- **Use Flow:**
  You can choose to flow or not to flow per layer here.

- **Flow Strength:**
  This determines the speed of flow of the layer.

- **Use Drift:**
  When this is checked, also the heightmap flows and the flow is taken into account for the Heightbased Blending. Using this can give a more natural look on the flow.



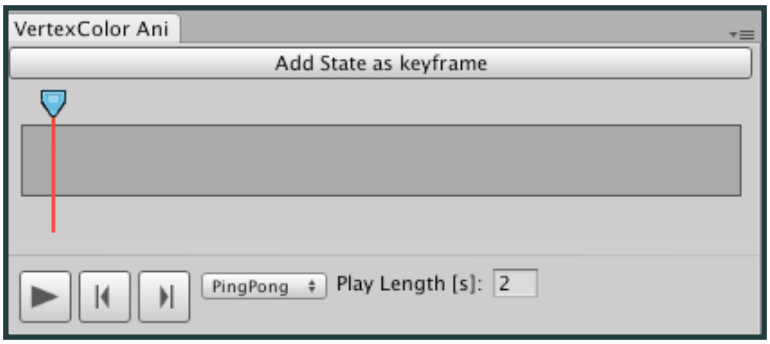
Tip:
Enable animated materials in the scene view to avoid the need to go into playmode to see the texture flowing.

2.2.2.3 Vertex Color Animator

With our Vertex Color Animator you can animate between several painted versions of your mesh.

How to use the Vertex Color Animator

You find the Vertex Color Animator in the Top Menu Bar under Tools/Vertex Color Animator. It opens a panel that you can drag into your layout.
If you select an initialized game object and press „Create Vertex Color Animator“ it looks like this:



You see there‘s a timeline with a playhead, a play button and buttons to jump to the beginning or end of the timeline. You can also choose an animation type (clamp, ping pong or loop) and set your animation lenght.
To animate between two versions of your vertex colors, simply paint the first version, save it and click „Add state as keyframe“. This creates our first keyframe:

Then, you want to go to a different point in your timeline by moving the playhead to the right and paint another version of your vertex colors.
After saving the mesh you click „Add state as keyframe“ again and you should see another keyframe in your timeline.
If you hit play you should see your animation playing!

By the way: you are not limited to just two keyframes.

Add as much as you wish to create some interesting results!

2.3 Generating Flow Data

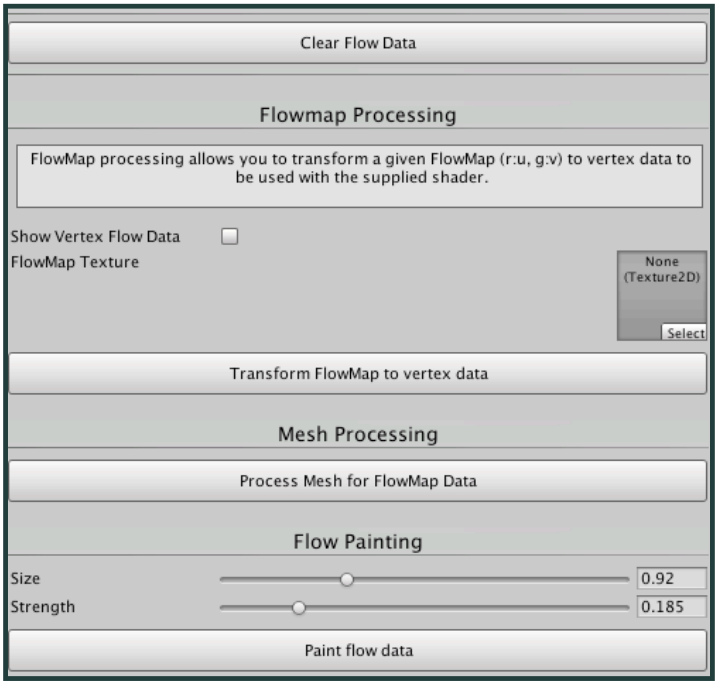
To use the Flow Support of the shaders we have to provide some flow data of course. To reduce the number of needed samplers in the shader and therefore to increase performance, we chose to not support FlowMaps as a texture, but instead use vertex-based Flow Data. You have two options to generate those. Both can be found in the VTP window under the ‚Flow Mapping' tab.

2.3.1 Process a Flow Map

The first option is to process a given FlowMap. Just drag a usual FlowMap in the respective slot in the VTP window and hit the ‚Transform FlowMap to vertex data' button. The vertex-based Flow Data is then generated by the given FlowMap. You can visualize the flow data by checking the ‚Show Vertex Flow Data' checkbox. To clear the vertex-based Flow Data just hit the ‚Clear Flow Data' button.

2.3.2 Process the Mesh

The second option is to process the mesh to generate the vertex-based Flow Data. Basically it calculates the flow at a vertex from the slopes of the mesh. To do so just click the ‚Process Mesh for FlowMap Data' button. You can visualize the flow data by checking the ‚Show Vertex Flow Data' checkbox. To clear the vertex-based Flow Data just hit the ‚Clear Flow Data' button.



2.3.3 Paint Flow Data

Painting flow data on your mesh is very easy. After you have initialized your mesh and activated Flow in the shader, you can just go to the Flow Mapping Tab of the Vertex Tools Pro panel and click on the „Paint Flow data“ button. This should give you a brush for painting flow data on the vertices. Settings for the brush (size and strength) can be changed in the Flow Mapping Tab as well.

The direction, in which the texture will flow depends on the direction you paint on the mesh, so this is a very interactive process. The speed of the texture flow also depends on the speed of your painting.

You can also alter the flow information that you generated from a mesh.

2.4 Deforming

In order to further increase the visual variety we provide you with basic mesh manipulating tools. To use them go to the ‚Deform' Tab in the VTP window. As with the painting you see options to change the brush size and the strength. Below those settings you can find the actual manipulation methods.

- **Intrude/Extrude:**
  Extrudes or intrudes (intrudes when you hold the CTRL key while deforming) the vertices in the direction of their normals.

- **Pinch:**
  Pinch will either pull all affected vertices to the brush's center or drag them away from the brush center if you press and hold CTRL while clicking.

- **Push/Pull:**
  Push or pulls (Pulls when you hold the CTRL key while deforming) the vertices in the direction of your view.

- **Smooth:**
  Smooths the vertices within the brush. This means that distance between the vertices is equalized. We suggest this after you extruded or intruded the vertices to smooth hard edges.

The actual process of deforming is the same as the painting process. Hit the ‚Deform' Button and start deforming the mesh in the scene view. Be sure not to forget to save the mesh after you are done.

Now you finally know the functionality of Vertex Tools Pro. If there are any questions left feel free to contact us at Unity's forum: http://forum.unity3d.com/threads/in-review-vertex-tools-pro-heightbased-pbr-material-blending-flow-mapping-and-mesh-deform.385577

vertextools.pro
support@vertextools.pro
twitter.com/vertextoolspro