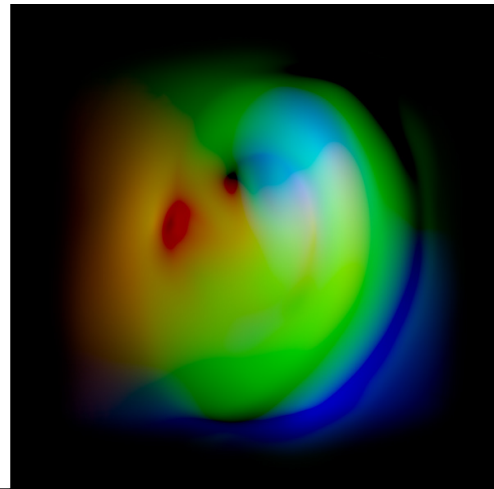
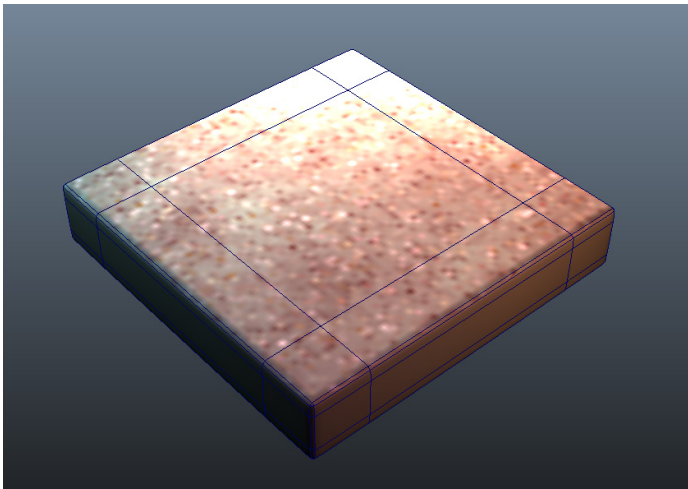
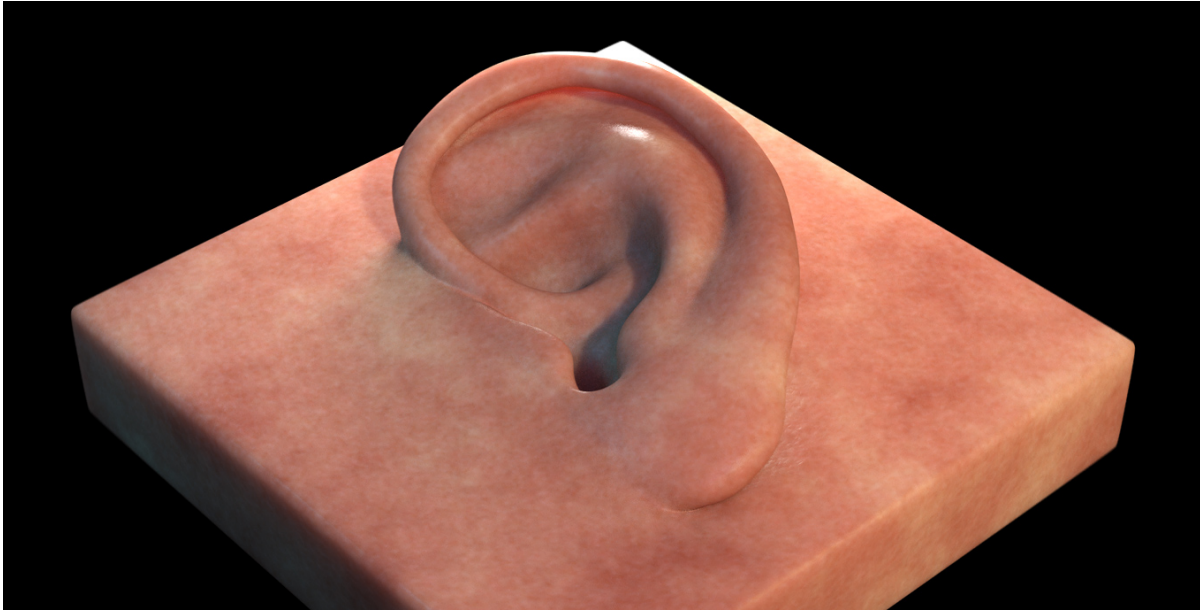


Rendering Displacements



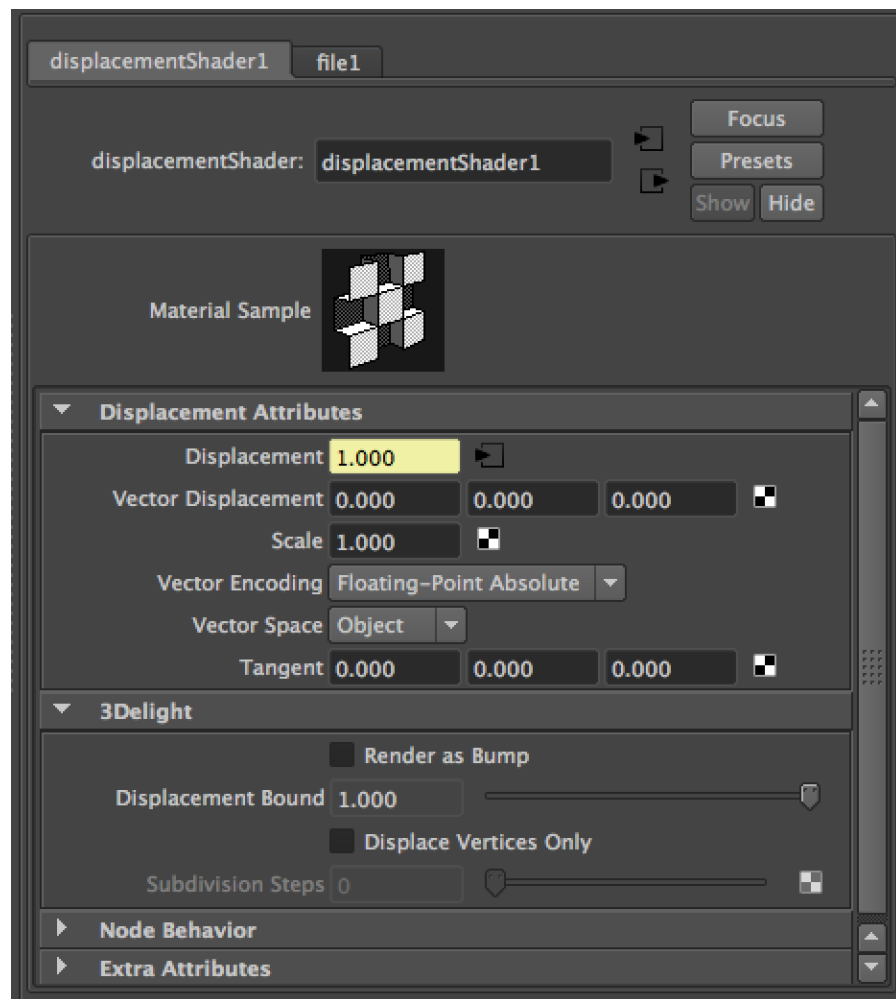
A simple vector displacement test. You can download the Maya scene [here](#) and the ear vector displacements map [here](#).

This tutorial explains how to apply displacements in *3Delight for Maya*. Both scalar and vector displacements are supported. The above is a model of an ear sculpted using vector displacements.

The usual way of rendering displacements in *Maya* is through the `displacementNode`. All parameters of the `displacementNode` shader are supported, but some parameters have an extended meaning in *3Delight for Maya*. All necessary details are explained below – including all the relevant options for generating displacement maps from Autodesk *Mudbox* and Pixologic *ZBrush*.

Content:

- [Scalar Displacements](#)
 - [Render as Bump](#)
 - [Vector Space](#)
 - [Vector Encoding](#)
- [Vector Displacement Maps](#)
- [Adjusting the Displacement Bound](#)
- [Displacement Technology](#)
- [Using Mudbox Displacements](#)
- [Using ZBrush Displacement Maps](#)
- [Using Mari Displacement Maps](#)



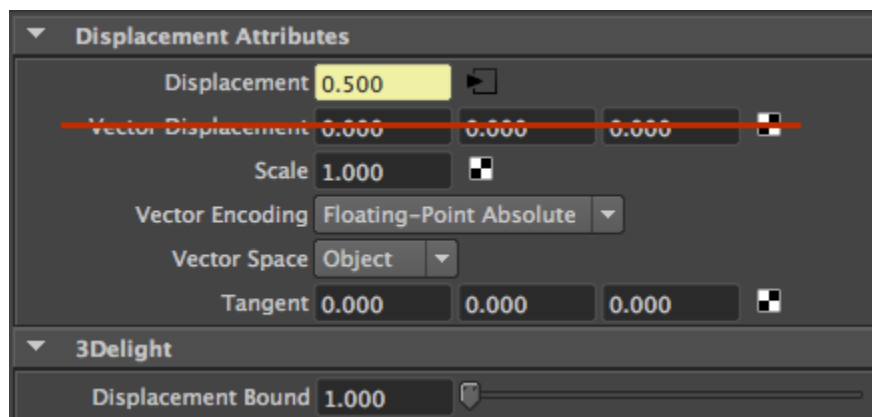
displacementNode UI

Scalar Displacements



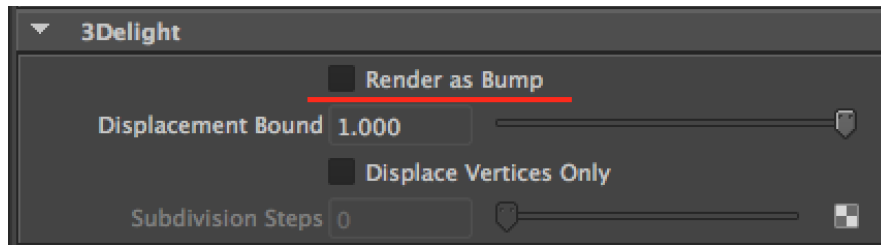
An example of scalar displacement applied on a subdivided head for skin rendering.

For scalar displacements, all parameters are recognised but one: *Vector Displacement*. It is not used for obvious reason: the displacement vector is implicitly the surface normal. Despite the misleading names, both *Vector Encoding* and *Vector Space* are supported in the case of scalar displacements.



For scalar displacements, the *Vector Displacement* parameter is not used.

Render as Bump



The "render as bump" option is provided to easily switch from Displacement to Bump.

There is a number of advantages in using the displacement node rather than the bump2d/bump3d nodes to do bumps:

- unlike the bump node, the displacement node easily allows you to predict the right amount of displacement/bump in real units and according to the space.
- it is easy to switch between the two, without the need to create nodes or change connections.
- the look will be visually compatible and predictable

Vector Space

This parameter, *which also applies to scalar displacements*, defines in which 3D space the displacement is defined. The available options are:

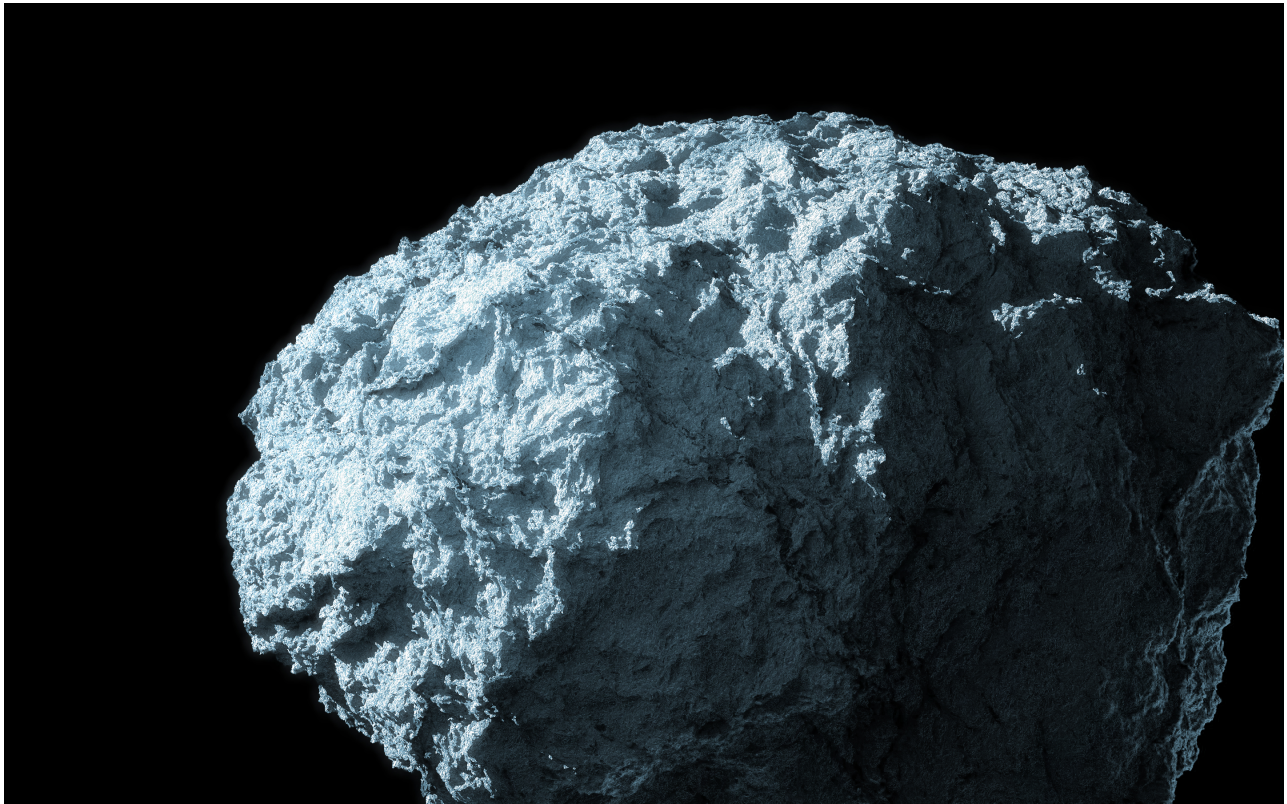
Object Space	The displacement magnitude is expressed in the same space as the model. This means that if the object is scaled by some value, the displacement will follow the same scaling.
World Space	The displacement magnitude is expressed in the world space. This means that displacement magnitude is independent of object scaling. A displacement value of 1.0 with a unit scale corresponds to a magnitude of 1 <i>Maya</i> unit.
Tangent Space	The displacement takes place in a local coordinate system defined by the tangents on the surface. This mode is useful for vector displacement.

Vector Encoding

This parameter, *which also applies to scalar displacements*, defines in which range the displacement is defined. The available options are:

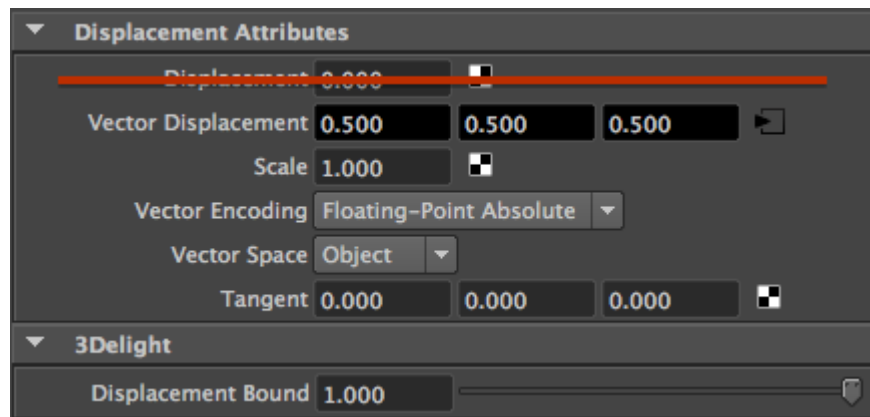
Floating Point Absolute	The displacement range is between 0 and 1.0* <i>Scale</i> . The displacement will push the surface <i>out</i> .
Signed Encoding	The displacement range is centered at 0.5 (meaning values of 0.5 would produce no displacement). Values below 0.5 would push displacement inward, values above 0.5 outward.

Vector Displacement Maps



An example of vector displacement applied on a simple spherical object.

Connecting a texture (or a procedural texture) into the *Vector Displacement* plug enables vector displacements. In this mode, the *Displacement* plug is ignored.

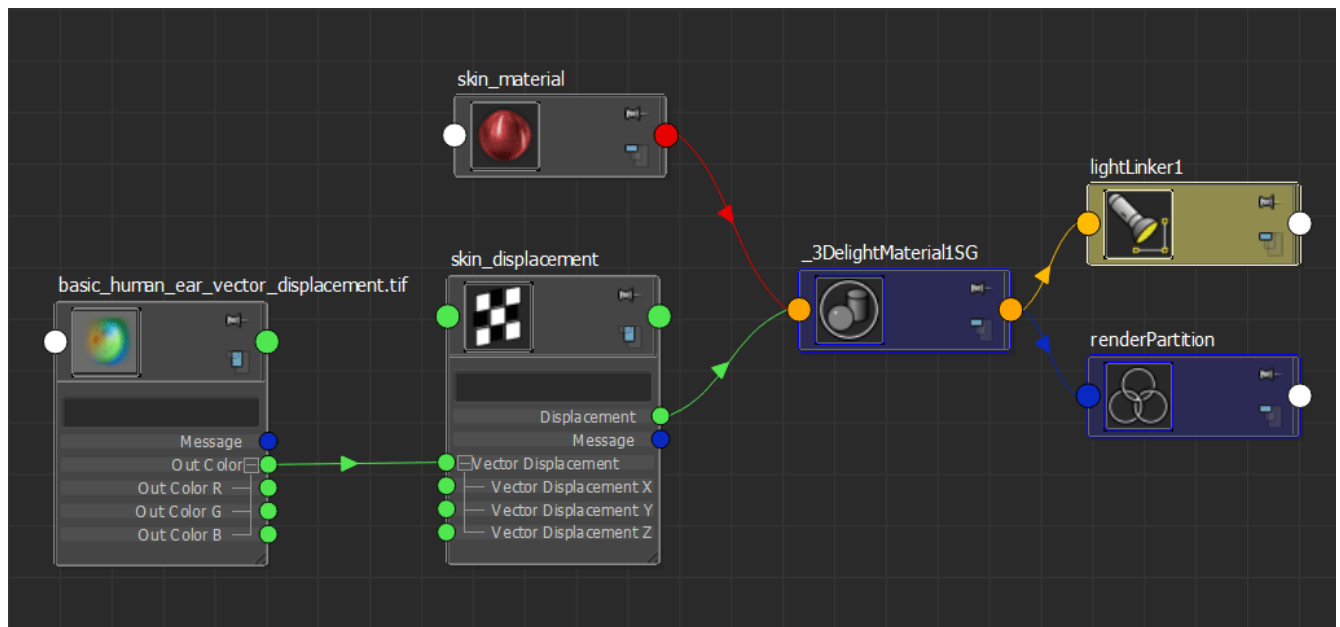


Displacement plug is ignored when Vector Displacement is connected.



By default, *Maya* will connect your *File* Node output to the *Displacement* plug of your displacement shader. This is suitable for scalar displacements, not for vector displacements. Make sure to connect the map (or network) to the proper *Vector Displacement* plug.

Here is an example of a network setup to use vector displacement map:



Displacement network to use a vector displacement map (used to render the "ear" scene).



When using vector displacements, check if your map was created in **World/Object Space** or **Tangent Space** and set the **Vector Space** parameter accordingly.

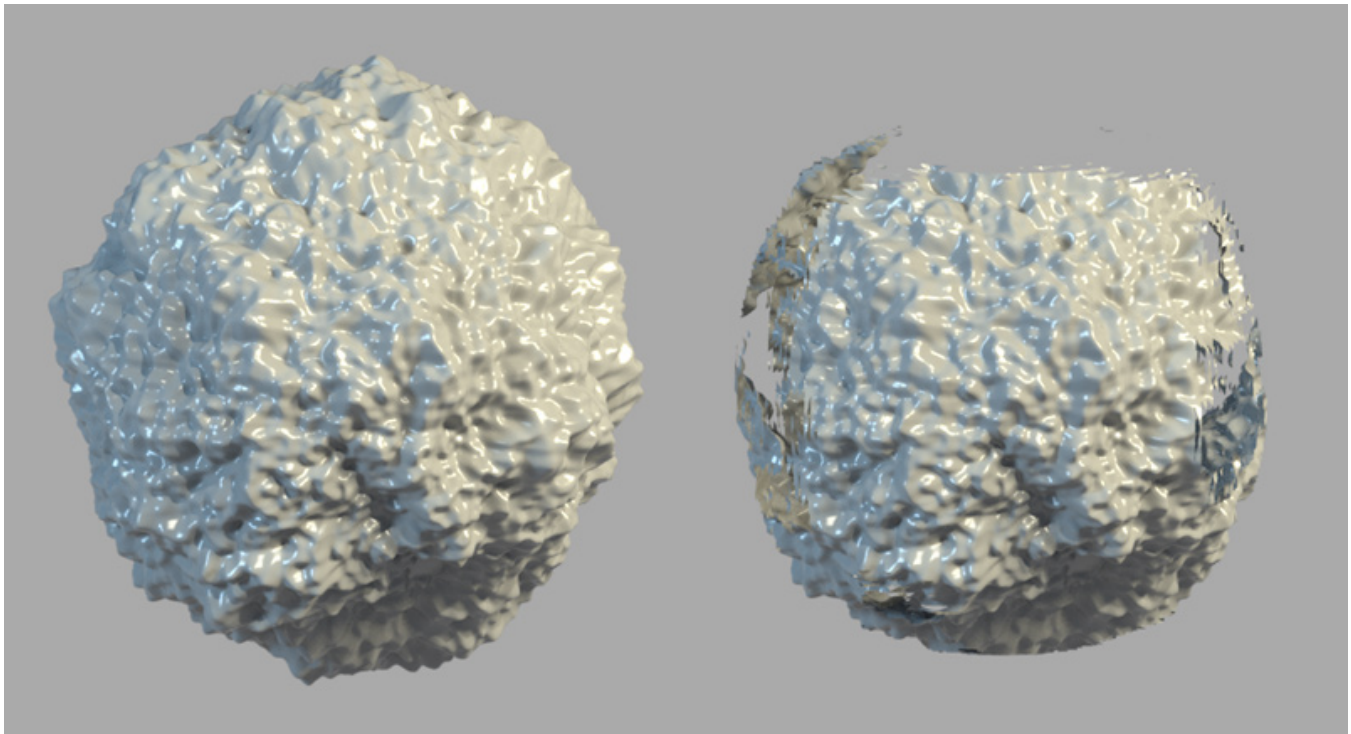
Adjusting the Displacement Bound

As explained in Displacement Technology, *3Delight* needs bounds for the displacements to render properly. *3Delight for Maya* guesses the correct displacement bound using the parameters in the displacement shader so in most cases the *Displacement Bound* is better left untouched. In some cases, however, it is necessary to adjust the bound to avoid artifacts or to optimize render times.

One instance in which the adjustment is necessary is when using *floating-point* displacement maps (or shading networks) that produce magnitudes larger than 1.0. In this case, the *Displacement Bound* should be set to the maximum value in the texture map or network. Usually renders which exceed the displacement bound will produce the following *3Delight* error :

```
3DL R2001 object 'name' (displacement 'shader', surface 'shader') exceeded its displacement bound by 25%
A displacement bounds is be specified for the object but is not big enough. This could cut off displaced
parts of the object. Maybe the bounds should be enlarged.
```

The following render shows the kind of artifacts produced by insufficient displacement bounds:



Correct Displacement Bounds (left) — Insufficient Displacement Bounds (right)

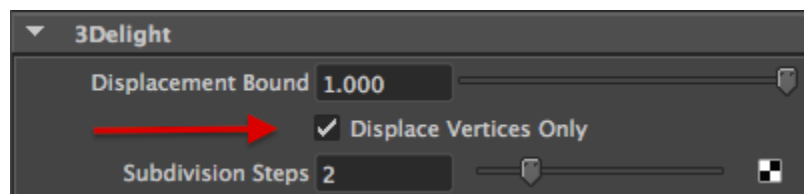
Another instance in which the *Displacement Bound* should be changed is when the range of the displacement map (or shading network) is much lower than one, in this case the bound should be lowered to reflect the actual range of the displacement map or network. Note this case won't produce artifacts, it is a simple optimization step. When *3Delight* detects displacement bounds that are too large, the following warning will be produced:

```
3DL R2093 object 'name' (displacement 'shader', surface 'shader') used only 25% of its displacement bound
The displacement bound specified for the object is much larger than needed. Performance could be increased
by reducing it.
```

Displacement Technology

3Delight supports two different displacement methods:

1. *Pixel level* displacement. Geometry is tessellated on-the-fly and adaptively to meet highest quality criteria. This is the default.
2. *Vertex level* displacement. Geometry is pre-tessellated to a given amount of subdivision prior to running the displacement shader. Vertex level displacement is enabled in the *displacementNode* inside the *3Delight* section as show in the screen shot below.



Enabling Vertex Displacement

Both approaches have their *pros* and *cons* as shown in the table below.

	Pixel Level Displacement	Vertex Level Displacement
Start up time	Fast startup time because no pre-processing is required.	<i>Potentially</i> slower startup time because of pre-tessellation requirement. This depends on the number of <i>Subdivision Steps</i> . The higher the steps the slower is the pre-processing time.

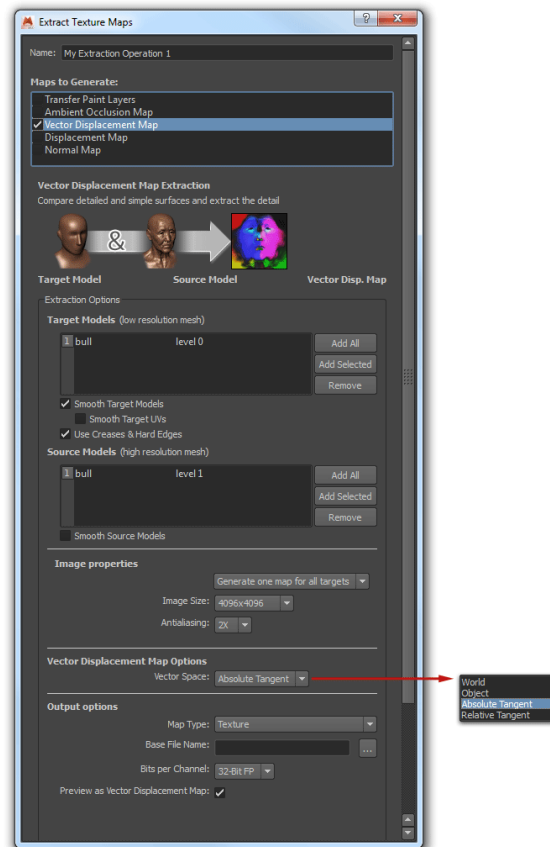
Overall Render Time	Slower render times.	Usually faster render times.
Memory Usage	Lower memory usage thanks to a smart caching system. This is not always true though: in this case, <i>3Delight</i> needs to keep original geometry in memory and this could be expensive in case of subdivision surfaces.	<i>Potentially</i> higher memory usage depending on the <i>Subdivision Steps</i> . Although in the case of subdivision surfaces, this method might use less memory overall because final tessellated and displaced geometry is stored as a polygonal mesh (more compact than the subdivision mesh counterpart).
Quality	Provides very high quality displacement but might be subject to cracking, especially for vector displacement and large displacements.	Displacement quality depends on the <i>Subdivision Steps</i> . <div>ⓘ Note that <i>3Delight</i> will use bump mapping on the final geometry faces to achieve a fine grained look. So even with low <i>Subdivision Steps</i>, high frequency details are preserved by varying the normal over the surface.</div>

Using *Mudbox* Displacements

3Delight for Maya can render *Mudbox* displacements (both vector and scalar) using the standard `displacementNode` shader. The following table explains how to export both scalar and vector displacement maps.

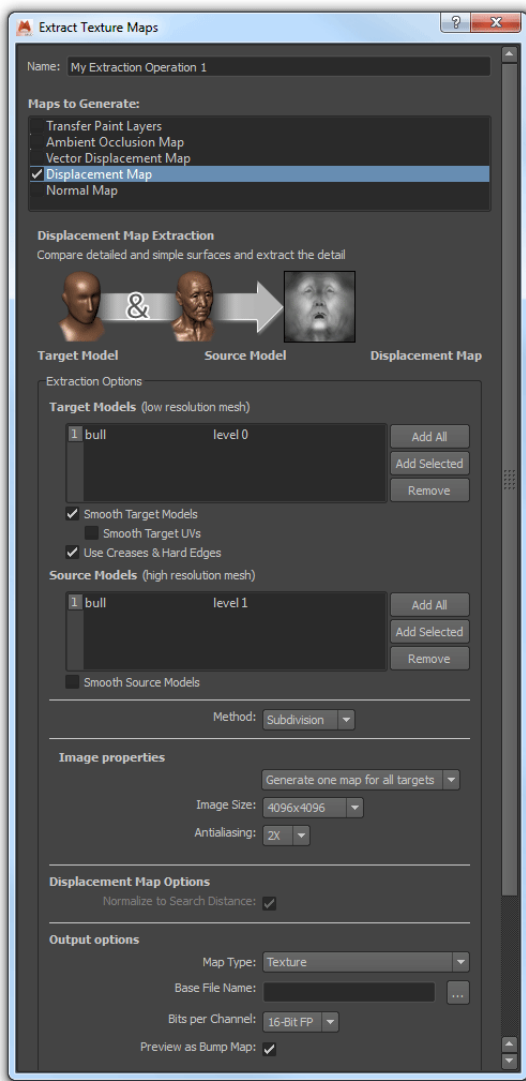
Scalar Displacement Export

Vector Displacement Export



For vector displacement:

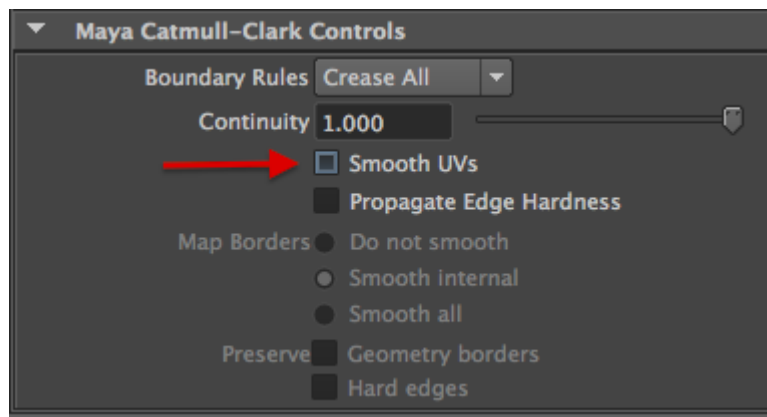
1. Select *Vector Displacement* at the top of the panel.
2. Make sure that *Smooth Target UVs* option is OFF.
3. Regarding *Vector Space*:



- Non-deforming models with animated transformations, *Object* space is recommended.
- Completely static non-deforming models can use *World* space (but *Object* space will do just fine).
- For deforming models, *Absolute Tangent* is needed.

For scalar displacements:

1. Select the *Displacement Map* option at the top.
2. Make sure that *Smooth Target UVs* option is OFF.



Disable *Smooth UVs*.

Using ZBrush Displacement Maps

3Delight for Maya can render ZBrush displacements (both vector and scalar) using the standard `displacementNode` shader. The following table explains how to export both scalar and vector displacement maps.

In ZBrush you can export displacement maps via:

- Tool Displacement
- Tool Vector Displacement
- Zplugin [Multi Map Exporter](#) (which allows to export other types of maps in one go). **This is the recommended choice.**

There are two kind of displacement output:

1. `Scalar` Displacement: this displaces the surface along the normal direction by a scalar value (a single float), pushing the surface up, or down or both. In scalar displacement the direction cannot vary *while* displacing the surface.
2. `Vector` Displacement: here the surface is displaced through a vector (three floats), and its encoding allows for "multi-directional" displacement, so the surface is pushed up/down/both in a direction that may vary *while* displacing.

In both cases the workflow to export displacements is the same:

- Make sure your mesh in ZB has "good", non-overlapping, UVs.
- Go the lowest subdivision level of your mesh (typically level 1).
- Export your displacement map with the recommended settings.
- While at the lowest subdivision level, always export your base mesh model as an OBJ file (or use GoZ).

The relevant interfaces for both scalar displacement (labelled "Displacement" on the left) and vector displacement (on the right) are:

ZBrush → zPlugin

Multi Map Exporter

Displacement

Vector Displacement

Normal

Texture from Polypaint

Ambient Occlusion

Cavity

Export Mesh

Create All Maps

SubTools Merge Maps

Map Size 4096

512 1024 2048 4096

Map Border 4 **FlipV** ReUV

Export Options ▶

File names Switch MT

Estimate Time Reset

Displacement Map ▶

SubDiv level 1

Adaptive DpSubPix 4

SmoothUV Mid 0

3 Channels **32Bit** **exr**

Scale 1

Intensity 0

16Bit Scale Get Scale

Vector Disp Map ▶

Normal Map ▶

Ambient Occlusion Map ▶

Cavity Map ▶

Mesh Export ▶

Load/Save Presets

Multi Map Exporter

Displacement

Vector Displacement

Normal

Texture from Polypaint

Ambient Occlusion

Cavity

Export Mesh

Create All Maps

SubTools Merge Maps

Map Size 4096

512 1024 2048 4096

Map Border 4 **FlipV** ReUV

Export Options ▶

File names Switch MT

Estimate Time Reset

Displacement Map ▶

Vector Disp Map ▶

SubDiv level 1

vd Tangent **32Bit** **exr**

vd SUV **vd SNormals**

Normal Map ▶

Ambient Occlusion Map ▶

Cavity Map ▶

Mesh Export ▶

Load/Save Presets

(Scalar) Displacement and Vector Displacement map creation options

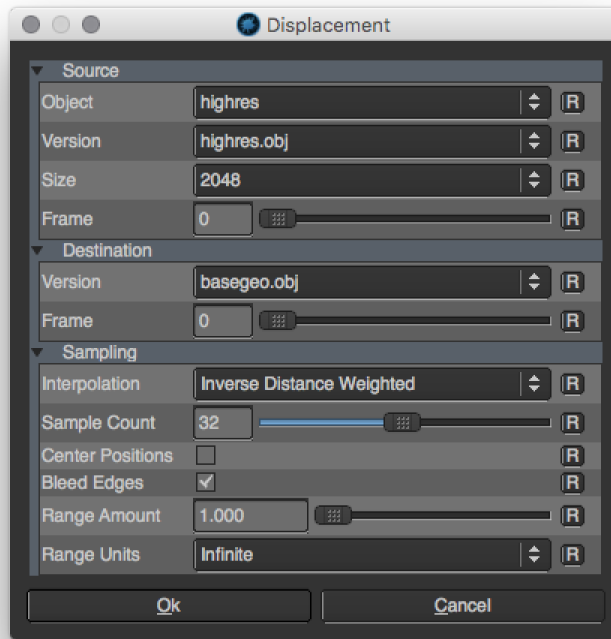
Option	Displacement Type	Description
Map Size	Scalar/vector	Usually for heroes and closeups, pick twice the resolution of your rendering output (if you are rendering 720p use 2K, if you are rendering 1080p use 4K).
Adaptive	Scalar	ON (this produces higher quality maps). Always use it.
DpSub Pix	Scalar	This is a legacy ZBrush option and should <i>never</i> be used: leave it to 0.
Smooth UV / vdSUV	Scalar/Vector	ON (smooths the UVs are before the map is generated). This is needed because when you enable subdivision surfaces in Maya, the UVs by default are also smoothed.
Mid	Scalar	This is the mid range for the displacement, to use only for 16 bit maps, but they are not recommended (see below). When using 32 bit maps, you want your mid range to be set to 0. By default the mid range will be 0.5, meaning 50% grey, which makes no sense when using 32 bit images.
FlipV	Scalar	ON (flips the UV's V direction, this is necessary for 3Delight).
Scale	Scalar	1.0 (when using 32 bit maps, recommended below, the value should be 1.0).
3Channels	Scalar	OFF (this outputs 1 channel only, this is enough for 3Delight). 3 Channels will give the displacement information in all channels instead of just the red one.
32Bit / vd 32bit	Scalar/Vector	ON (this export a 32 bit float map in TIF format). <u>We recommend to use 32bit maps</u> when exporting displacement maps from Brush.
vdTangent	Vector	There is no recommended default here, simply for static models use OFF while for animated/deforming model use ON.
vdNormals	Vector	ON (this smooths the low resolution model's surface normals and results in a smoother map).
SubDiv level	Scalar	This dictates which subD level the displacement map will be generated from. You normally want this set to 1 as the displacement will then be generated from the lowest level, meaning that the map will have the difference between the lowest and highest level.
EXR	Scalar/Vector	Whether to use TIF or EXR format. Both are good.
Intensity	Scalar	It is safe to change this only when rendering 32 bit maps. We recommend not to change and leave it to default settings.

Using Mari Displacement Maps

Mari allows you to generate a displacement map using a high polygon source model and a low polygon destination model to calculate the displacement. The workflow consists of selecting the source object, its version, size, and the frame (if animated) from the options provided for the high polygon model. Then specify what destination version and frame to which the displacement should correlate on the low polygon model. The relevant UI can be accessed in two ways:

1. Menubar | Objects > Generate > Displacement
2. Right-click | an object in the Objects Palette > Generate > Displacement

You will need to adjust the sampling options, such as Interpolation and Sample Count to specify the quality on how the displacement map is generated. Other parameters are available and self descriptive: Center Positions, Bleed Edges, Range Amount, Range Units. As an example refer to the screenshot below:



Displacement Options in Mari

i When you want to generate a displacement map in Mari, the channel you have selected needs to be 16-bit (Half) or 32-bit (Float).

i In Mari displacement channels default to a 50% gray background – this corresponds to the default surface. Mari treats darker colors as negative displacement (moving into the surface) and lighter colors as positive displacement (moving out from the surface).

What Bit Depth to Use?

Short answer: use 32-bit floating point displacements.

This is a very common question and the answer depends on both the type of the map used (scalar or vector) and the type of the displacements used. Disk space consumption can also be a consideration. *3Delight for Maya* work fine with all bit depths. The table below provides some hints on what bit depth should be used depending on the situation.

Type	When to use	Observations
8-bit integer	Best suited for small scalar displacements. For example, simulation of small surface bumps on: skin, leather, some fruits (orange/strawberry), fabrics, rusted metal, aged wood, etc.. Making sure close to the full range of data is used over the min /max displacement scale is important.	For practical reasons, integer formats are less used than their floating point counter-parts. This is unfortunate. Integer displacement maps are excellent for almost all scalar displacements and have <u>better precision than floating point data for the same bit depth</u> . Unpracticality comes from the fact that the range is implicitly between 0 and 1 and scaling must be specified in the displacement shader to bring the magnitude to the desired range. Integers are not practical for vector displacements.
16-bit integer	Best suited for medium to large scalar displacements. It provides enough precision and uses less disk space than floating point data.	
32-bit integer	Enormous precision suited for special cases. A camera positioned to reveal an extreme enlargement of a displaced surface is an example.	

16-bit floating point	A versatile data format that has both good range and good precision for displacement maps. This is the recommended choice for scalar displacement maps.	Floating point is very good at representing data at different scale, <i>which is generally not the case for most scalar displacement maps</i> . Vector displacement maps, as produced by <i>Mudbox and ZBrush</i> , generally need such precision and floating-point data is advised. One practical advantage for floating point data is that the range can be correctly and directly stored in the data. For example, using a range of 0 to 0.1 in the data is perfectly valid and can correspond directly to the displacement scale as seen on the object. When using integer data, this range would have to be encoded between 0 and 1 and then scaled back to between 0 and 0.1 in the displacement shader.
32-bit floating point	This is the recommended choice for vector displacement maps.	