Displacement

Displacement maps can be an excellent tool for adding surface detail that would take far too long using regular modeling methods. Displacement mapping differs from bump mapping in that it alters the geometry, and therefore will have a correct silhouette, and self-shadowing effects. Depending on the type of input, the displacement can occur in two ways: Float, RGB & RGBA inputs will displace along the normal while a vector input will displace along the vector.

The example above shows how a simple plane, with the addition of a displacement map, can produce an interesting looking simple scene.

You should ensure that your base mesh geometry has a sufficient number of polygons otherwise subtle differences can occur between the displaced low-resolution geometry and the high-resolution mesh from which it was generated.

Make sure that you use a 32-bit or 16-bit floating-point format to store your image, and not an integer format. An integer format will not work correctly. This is because integer formats do not support negative pixel values, which are used by floating-point displacement maps.

Always ensure that you use the highest quality texture maps for displacement mapping. Arnold works well with very high-resolution maps, as long as the maps have been preprocessed with the maketx utility. It will convert them into .tx files (which are tiled, mipmapped files). See the pages about the maketx utility and .tx files.
Displacement Parameters (Arnold Property Modifier)

MAXtoA has displacement options on a per-object basis. This may be useful in a scene that has two objects with the same texture but requires different shape displacement values. Another example would be an object that has more than one shader but requires two different Height values such as in the example below:

![Same displacement shader assigned to two meshes, however, the mesh on the right has a per-object Height of 2](image)

The per-object MAXtoA displacement parameters are divided into the following groups:

- **Enable**
  - Height: 1.0
  - Zero: 0.0
  - Bounds Pad: 0.0
- **Autobump**
  - Enable
  - **Camera (Primary Rays)**
    - Diffuse Reflections
    - Specular Reflections
    - Diffuse Transmission
    - Specular Transmission
    - Volume Scattering
- **Displacement Map**
  - Use Map
  - No Map
**Height**

Controls the amount of displacement. Displacement height can have either positive or negative values. This attribute only applies with normal displacement. You can use this value to compensate for any inconsistencies between the exported displacement map and the low-resolution geometry.

![Height Examples](image)

**Zero**

This is a floating point value which is applied as a shift to the displacement amount. It defines the value of the displacement map that is considered to be zero displacement. This value can vary depending on how the displacement map has been generated.

![Zero Examples](image)

**Bounds Padding**

Padding defines how much to extend the bounding box of the object so that it can include any additional displacement coming from the displacement shader. When the bounding box is hit first by a ray, the displacement will be computed, so an unnecessarily high value will decrease the rendering efficiency. On the other hand, a low value could result in a clipping of the displaced mesh.

![Bounds Padding Examples](image)
The proper workflow for displacement in Arnold is to have the shader give the final displacement value and then to offset the bounding box with the bounds padding attribute.

The mesh on the left has a bounds padding value of 0.5 and the mesh on the right has a bounds padding value of 1 (per object settings).

Below is another example of a situation where bounds padding is required. A checker texture is connected to a displacement shader that is assigned to a sphere. In the image on the left, parts of the render return black. This is because the Bounds padding needs to be increased for the displaced mesh. Increasing the Bounds padding to 3 fixes the problem. This effect may increase or decrease depending on the scale of the object that is being displaced and the amount of displacement used.
**Autobump**

*Autobump* puts the high frequencies of a displacement map into the bump attribute so that you do not need as many Subdivision Iteration values. *Autobump* is visible to camera rays only by default. The visibility parameters let you make *Autobump* visible to other rays (e.g., Specular and Transmission) however that can increase render times.

The *Autobump* algorithm needs UV coordinates to compute surface tangents. Make sure your polymesh has a UV set applied.

**Technical information:**

When *Autobump* is enabled, Arnold makes a copy of all of the vertices of a mesh before displacement (let's call that the "reference" mesh, or Pref). Prior to shading at some surface point on the displaced surface P, the equivalent Pref for that point is found on the non-displaced surface, and the displacement shader is evaluated there (at Pref) to estimate what would be the equivalent normal at P if we had subdivided the polymesh at an insanely high tessellation rate.

The main difference between Arnold's *Autobump* and using the displacement shader for bump mapping is that *Autobump* has access to Pref whereas bump2d does not and would be executing the displacement shader on already-displaced points which could "compound" the displacement amounts.

The only extra storage is for copying P prior to displacement. There is no analysis of the displacement map; Arnold displaces vertices purely based on where they "land" in the displacement map (or procedural)
regardless of whether it happens to "hit" a high-frequency spike or not.

Disabled (subdivision iterations 2)  Disabled (subdivision iterations 5)  Enabled (subdivision iterations 2)

Subdivisions

Subdivision Iterations set to 8. Subdivision Type set to Cat Clark (Arnold Properties Modifier).

Subdivision Iterations: 2  Subdivision Iterations: 4  Subdivision Iterations: 8

Changing the Subdivision Type to either CatClark or Linear subdivision rules and increasing the iterations will improve the displacement quality. In this example, the Subdivision Iterations have been increased to 8.

You must be careful when increasing the number of subdivision iterations (each iteration quadruples the geometry). This subdivision happens at render time, whenever a ray hits the bound box of the object. This
is a better choice compared to increasing the subdivisions of the mesh within the DCC software (which will send the tessellated geometry to the renderer).