## Subdivision

<table>
<thead>
<tr>
<th>Subdivision</th>
<th>Displacement</th>
<th>Shapes</th>
<th>Attributes</th>
<th>Motion Blur</th>
<th>Visibility</th>
<th>Normals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iterations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adaptive Metric</td>
<td>Auto</td>
<td>Error</td>
<td>Smooth Derivatives</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adaptive Space</td>
<td>Raster</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UV Smoothing</td>
<td>Pin Corners</td>
<td>Smooth Derivatives</td>
<td>Merge Vertex Indices</td>
<td>Ignore Frustum Culling</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following examples have been shaded with the Utility shader in polywire mode.

### Type

This enum. Defines the subdivision rule that will be applied to the polymesh at render time. Possible values are catclark (catmull clark) and linear.

- **None**
  
  Ignores any subdivision and renders the mesh as it is.

- **Linear**
  
  Linear subdivision puts vertices in the middle of each face.

- **Catclark**
  
  The Catmull–Clark algorithm is used to create smooth surfaces by recursive subdivision surface modeling. The resulting surface will always consist of a mesh of quadrilateral faces.

Currently, for Catmull-Clark to work on a Polygon Mesh "Unique Points Per Pole" needs to be enabled at the object geometry level.
Iterations

The maximum number of subdivision rounds applied to the mesh. When subdiv_pixel_error is 0 the number of rounds will be exact instead of a maximum.

Subdivision Type: Linear

Subdivision Type: Catclark

Bear in mind that each subdivision iteration quadruples the number of polygons. If your object has 2 levels of subdivision iterations set and 4 additional iterations set in Arnold, that's 6 subdiv iterations total and therefore 426936 * 4^6 = 426936 * 4096 = 1.7 Billion polygons.

Adapative Metric

The metric used to determine the amount of error between a given subdivision level and the limit surface.

- **edge_length**: no final edge will be longer than subdiv_adaptive_error.
- **flatness**: the distance to the limit surface will be smaller that subdiv_adaptive_error.
- **auto**: in general, flatness will produce better approximation for the same number of tessellated triangles. But since subdivision tessellation does not take into account displacement, "flatness" might stop way too early and miss a lot of displacement detail (think of a displaced plane: a single quad will represent the plane exactly, but will not have enough vertices to show the displacement's detail). To avoid this issue "auto" uses the "edge_length" metric if the polymesh has a displacement shader assigned, otherwise, it will use the "flatness" metric.

Adaptive Error

The maximum allowable difference for the chosen metric (subdiv_adaptive_metric) in the chosen space (subdiv_adaptive_space). The smaller the error, the closer to the limit surface a mesh will be and the less inter-frame “popping” when the subdivision level jumps, at the expense of using more polygons. A value of 0 disables adaptive subdivision, reverting to uniform subdivision, which can be more stable in animation.

Examples:

- a value of 1.0 with subdiv_adaptive_space set to "raster" and subdiv_adaptive_metric set to "edge_length" will subdivide until all edges are shorter than 1 pixel.
- a value of 1.0 with subdiv_adaptive_space set to "object" and subdiv_adaptive_metric set to "flatness" will subdivide until the tessellation is closer than 1 unit in object space to the limit surface.
Adaptive Space

Adaptive subdivision in raster space is problematic when instancing: A tessellation that is good for one instance will not be good for another further away. Using object space subdivision will ensure that all instances will subdivide to the proper level.

Raster

Corresponds to adaptive subdivision in raster space.

Object

This method corresponds to adaptive subdivision in object space. This is useful for instances or when adaptive tessellation is desirable regardless of a specific camera. Arnold will warn when raster-space tessellation is used on more than one visible instance.

Dicing Camera

When enabled, the user provides a specific camera that will be used for all dicing (subdivision) calculations during Adaptive Subdivision (in other words, the tessellation of the object will not vary as the main camera is moved). This can be useful to fix objectionable flickering introduced by Adaptive Tessellation with certain camera moves. If you set a static dicing camera you will still get the benefits from Adaptive Subdivision (higher polygon detail closer to camera) while getting a tessellation that does not change from frame to frame.

By default, this is disabled, and should only be used when necessary, and with a carefully chosen position for the dicing camera.

UV Smoothing

The face-varying "limit" dPdu and dPdv vectors computed during subdivision are stored and used for shading instead of computing face-constant dPdu and dPdv vectors on the fly during shading. This can be useful for anisotropic shaders but has a storage cost. Choose between linear, pin borders, pin corners, or smooth.

![Linear (Pin All)](image1) ![Pin Borders](image2) ![Pin Corners](image3) ![Smooth (Pin None)](image4)

Smooth Derivatives

You may notice faceting appear in specular highlights when using anisotropy. It is possible to remove the faceted appearance by enabling smooth subdivision tangents (via Arnold subdiv_smooth_derivs parameter). Take into account this requires a subdivision iteration of at least one in the polymesh to work.

Below is an example consisting of a sphere with an anisotropic highlight. On the left image, polygon faceting is noticeable in the specular highlight. On the right image, the specular highlight is much smoother because Smooth Subdivision Tangents has been enabled for the mesh.
Smooth Tangents: Off. Highlight is very ‘faceted’.
Subdiv Iterations: 1. Highlight is slightly improved.
Subdivision Tangents: On (Subdiv Iterations: 1). Highlight is much better.

If enabled, it will compute smooth UV derivatives (dPdu and dPdv), which is the reason you need a UV map associated with the object.

When the option is disabled, these derivatives are constant per triangle, which results in a faceted appearance whenever dPdu and dPdv (like in the anisotropic computations from the Standard shader).

Enabling this option will have a memory overhead (roughly 100 extra bytes of data per vertex per keyframe).

Subdivision Face Iterations

The `subdiv_face_iterations` integer primitive attribute (uniform user data) controls the maximum amount of subdivisions per face when doing adaptive subdivision (i.e. when `pixel_error > 0`).

Integers are converted to bytes in translation to match the Arnold type.

Subdivision Soft Creases and Vertex Creases

`OpenSubdiv`-compatible edge and vertex soft creases are supported. You can control the sharpness with the Houdini Crease SOP or the `creaseweight` and `ar_creaseweight_point` vertex and point attributes. Note that the crease weight point attribute is prefixed with “ar_”
* because, contrarily to `crease` and other crease attributes, `ar_creaseweight_point` has no equivalent in Houdini and is specific to subdivisions in Arnold. A crease sharpness value bigger than `polymesh.subdiv_iterations` will de facto encode an infinitely sharp crease. If `crease_sharpness` is not present, all creases will be assumed to be infinitely sharp.

**Merge Vertex Indices**

Merges the indices and data of coincident and identical vertex attribute values (UVs, normals, vertex user data). This merging allows for the proper interpolation of vertex attributes when subdividing, and yields more compact render scenes in memory and on disk, at the cost of a slower export.

**Ignore Frustum Culling**

Disables Subdivision frustum culling of the mesh.