

# Primitive Variables

Primitive Variables (also known as primvars) are data stored on objects. This data can be used for shading variation, much like [user attributes](#). There are two main distinctions between the two - primvars cannot vary across instances, but they allow more specific variation - constant, uniform, varying, vertex, and facevarying.

**Constant:** One value remains constant over the entire surface primitive. This is the only level that can be achieved with user attributes.

**Uniform:** One value remains constant for each uv patch segment of the surface primitive.

**Varying:** Four values are interpolated over each uv patch segment of the surface. Bilinear interpolation is used for interpolation between the four values.

**Vertex:** Values are interpolated between each vertex in the surface primitive. The basis function of the surface is used for interpolation between vertices.

**Facevarying:** For polygons and subdivision surfaces, four values are interpolated over each face of the mesh. Bilinear interpolation is used for interpolation between the four values.

## Setting arbitrary primvars

RenderMan for Katana supports primitive variables using Katana's standard [Arbitrary Attribute](#) convention.

Here is the list of supported values for `geometry.arbitrary.<group>.outputType`. Katana's documentation lists other types, but these are the only ones supported by RenderMan:

- float
- double (converted to float)
- string
- int
- color3
- normal3
- vector2
- vector3
- point2
- point3
- point4
- matrix16

Katana only has four values for arbitrary attribute "scope": primitive, face, point, and vertex. For the most part, these scope values map to the RenderMan primvar tags as follows: primitive = constant, face = uniform, point = varying, and vertex = facevarying. Subdivision meshes support different interpolation of primvars with the "point" scope using the "interpolationType" attribute. If `geometry.arbitrary.<group>.interpolationType` is set to "subdiv", the primvar will be "vertex", otherwise the primvar will be "varying".

Here is an example OpScript showing how to set an arbitrary constant color primvar:

```

Interface
{
    SetAttr
    (
        "
        geometry.
arbitrary.myColor.
scope"
        , StringA
ttribute
    (
        "
        primitive"
    ))
Interface
{
    SetAttr
    (
        "
        geometry.
arbitrary.myColor.
value"
        , FloatAt
tribute({
        0.0
        ,
        1.0
        ,
        0.0
        })
    ))
Interface
{
    SetAttr
    (
        "
        geometry.
arbitrary.myColor.
inputType"
        , StringA
ttribute
    (
        "
        color3"
    ))

```

## Using Primvars

Typically, production workflows will export these arbitrary attributes as a part of the alembic or USD asset, so no set up inside of Katana is required. Otherwise, you can use an OpScript to set the geometry.arbitrary attribute directly on a geometry location - the attribute is not inheritable.

Primvars are read by various shading plugins. [PxrPrimvar](#), [PxrSeExpr](#), [PxrVariable](#), and [PxrVary](#) are some plugins distributed with RenderMan that use arbitrary primvars.