COLLADA – Digital Asset Schema Release 1.4.1

Patch Release Notes

June 2006

Editors: Mark Barnes and Ellen Levy Finch, Sony Computer Entertainment Inc.

Publication date: June 2006

Khronos Group
P.O. Box 1019
Clearlake Park, CA 95424, U.S.A.

Sony Computer Entertainment Inc.
2-6-21 Minami-Aoyama, Minato-ku,
Tokyo 107-0062 Japan

Sony Computer Entertainment America
919 E. Hillsdale Blvd.
Foster City, CA 94404, U.S.A.

Sony Computer Entertainment Europe
30 Golden Square
London W1F 9LD, U.K.

Table of Contents

This page intentionally left blank.

COLLADA 1.4.1 Patch Release Notes

About This Document

This Release Note provides an overview of changes for the 1.4.1 COLLADA Digital Asset schema release. The 1.4.1 version of the schema and the *COLLADA – Digital Asset Schema Release 1.4.1 – Specification* are available for download from:

`http://www.khronos.org/collada/`

Schema Changes Since Version 1.4.0

Changes in this schema release are compatible with existing COLLADA 1.4.0 documents unless otherwise specified.

For details about schema features, including those that have changed for this release, refer to the updated *COLLADA – Digital Asset Schema Release 1.4.1 – Specification.*

<color_target>, <depth_target>, and <stencil_target> now have optional `face` and `mip` attributes

All <*_target> elements now have the attributes:

```
<xs:attribute name="index" type="xs:nonNegativeInteger"
    use="optional" default="0"/>
<xs:attribute name="mip" type="xs:nonNegativeInteger"
    use="optional" default="0"/>
<xs:attribute name="slice" type="xs:nonNegativeInteger"
    use="optional" default="0"/>
<xs:attribute name="face" type="fx_surface_face_enum"
    use="optional" default="POSITIVE_X"/>
```

Resolves bug 125.

<COLLADA>'s `version` attribute now supports 1.4.1

Valid values for `version` attribute are now 1.4.0 and 1.4.1.

Resolves bug 171.

<COLLADA> element now has optional `xml:base` attribute

```
<xs:element name="COLLADA">
...
    <xs:attribute ref="xml:base"/>
```

Resolves bug 207.

<connect_params>'s ref attribute type has changed

The ref attribute for <connect_param> is now xs:token.

Resolves bug 203.

<convex_mesh> child elements are now all optional

Child elements <source> and <vertices>, which were previously required, are now optional. This allows correct, unambiguous syntax when the attribute convex_hull_of is used, which indicates that the application should compute the convex hull of the specified mesh. In this case, the presence of child elements is potentially misleading. In 1.4.0, the only way to unambiguously use convex_hull_of was by defining empty sources and vertices, such as the following:

```
<convex_mesh id="cm" convex_hull_of="#someMesh">
    <source id="empty"/>
    <vertices id="verts">
        <input semantic="POSITION" source="#empty"/>
    </vertices>
</convex_mesh>
```

This is no longer necessary.

NOTE: If convex_hull_of is *not* used, child elements <source> and <vertices> should still be specified to define a valid <convex_mesh>.

Resolves bug 380.

<extra> child element added to several elements

Optional, unbounded <extra> child element added to:

- <bind_material>
- <sampler1D>
- <sampler2D>
- <sampler3D>
- <samplerCUBE>
- <samplerDEPTH>
- <samplerRECT>
- <surface>
- <profile_CG>
- <profile_CG> / <technique>
- <profile_CG> / <technique> / <pass>
- <profile_GLES>
- <profile_GLES> / <technique>
- <profile_GLES> / <technique> / <pass>
- <profile_GLSL>
- <profile_GLSL> / <technique>
- <profile_GLSL> / <technique> / <pass>

- <texture_unit>

Resolves bugs 174, 175.

`fx_setparam_common` unused type removed

The type `fx_setparam_common` was defined but not used and has been removed.

Resolves bug 140.

<input> element's semantic attribute value list has changed

For <input>'s `semantic` attribute:

- TEXTURE value is no longer valid. (It should have been removed in 1.4.0.)
- The following are now valid values.
  - CONTINUITY
  - LINEAR_STEPS
  - MORPH_TARGET
  - MORPH_WEIGHT
  - TEXBINORMAL
  - TEXTANGENT

Resolves bug 138, 305, 316, 344. (COLLADA documents that use a TEXTURE semantic value must be changed.)

<instance_material> has a new <bind_vertex_input> child element

The <bind_vertex_input> element binds geometry vertex streams (identified as <input> elements within <geometry> elements) to material effect vertex stream semantics. Although applications commonly perform automatic binding of vertex streams with identical semantic identifiers, there are frequently mismatches in a semantic identifier's meaning. Use <bind_vertex_input> to remove these ambiguities, which are most commonly caused by:

- Generalizations; for example, TEXCOORD0 vs. DIFFUSE-TEXCOORD
- Spelling differences; for example, COLOR vs. COLOUR
- Abbreviations
- Verbosity
- Synonyms

Child elements of <instance_material> must appear in the following order if present:

Name/example	Description	Default	Occurrences
<bind>			0 or more
<bind_vertex_input>	See following table.		0 or more
<extra>			0 or more

<bind_vertex_input> has the following attributes:

semantic	**xs:NCName**	Which effect parameter to bind. Required.
input_semantic	**xs:NCName**	Which input semantic to bind. Required.
input_set	**uint**	Which input set to bind. Optional.

For an example and a discussion, see "Texture Mapping in <profile_COMMON>" in the "Specification Errata and Addenda" section.

Resolves bug 348, 360.

<instance_rigid_body>'s <*velocity> elements now have defaults

Added default values in <instance_rigid_body> for <velocity> (0, 0, 0) and <angular_velocity> (0, 0, 0).

Resolves bug 317.

<instance_*> elements now have optional `name` and `sid` attributes

Added optional `name` and `sid` attributes to all <instance_*> elements that did not already have them, which are the following:

- <instance_animation>
- <instance_camera>
- <instance_controller>
- <instance_effect>
- <instance_force_field>
- <instance_geometry>
- <instance_light>
- <instance_material>
- <instance_node>
- <instance_physics_material>
- <instance_physics_scene>
- <instance_visual_scene>

Added optional `name` attribute to the following:

- <instance_physics_model>
- <instance_rigid_material>

Resolves bug 189.

<library_physics_material> now has `id` and `name` attributes

Resolves bug 371.

<newparam>'s `sid` attribute type has changed

The `sid` attribute for <newparam> is now `xs:token` under the following parent elements of <newparam>:

- <profile_CG>
- <profile_CG> / <technique>
- <profile_GLSL>
- <profile_GLSL> / <technique>

Resolves bug 203.

<pass> no longer has extra <sequence> layer

Removed redundant <xs:sequence> from <pass> elements.

Resolves bug 177.

<pass> render state <color_material_enable> now applies to all profiles

A gl_pipeline_setting. missing from COLLADA FX 1.4.0, has been added to allow authors to indicate when runtimes should perform glEnable(GL_COLOR_MATERIAL) and glDisable(GL_COLOR_MATERIAL) or feature equivalents in profiles such as GLES, GLSL, and Cg.

```
<xs:element name="color_material_enable">
  <xs:complexType>
    <xs:attribute name="value" type="bool" use="optional" default="true"/>
    <xs:attribute name="param" type="xs:NCName" use="optional"/>
  </xs:complexType>
</xs:element>
```

Resolves bug 390.

<profile_*> elements now have optional <asset> child element

An optional <asset> child element is now available in:

- <profile_COMMON>
- <profile_GLES>
- <profile_GLSL>
- <profile_CG>

Resolves bug 176.

<profile_*> elements now have optional `id` attribute

Added an optional `id` attribute to <profile_COMMON>, <profile_CG>, <profile_GLES>, and <profile_GLSL>.

Resolves bug 322.

<profile_GLES> now has optional `platform` attribute

Added an optional `platform` attribute to <profile_GLES> to match other <profile_*>.

Resolves bug 323.

<profile_GLES> / <technique> / <annotate> child element is now unbounded

Resolves bug 206.

<profile_GLSL> / <technique> now has optional <annotate> child element

<annotate> element is now the optional first child element of <profile_GLSL>'s <technique>:

```
<xs:element name="annotate" type="fx_annotate_common"
    minOccurs="0" maxOccurs="unbounded"/>
```

Resolves bug 206.

<rigid_constraint> / <technique_common> child elements now have default values

In <rigid_contraint> / <technique_common>, added default values to several child elements:

```
<xs:element name="technique_common">
 <xs:complexType>
   <xs:sequence>
      <xs:element name="enabled" default="true" minOccurs="0">
         . . .
      <xs:element name="interpenetrate" default="false" minOccurs="0">
         . . .
      <xs:element name="limits" minOccurs="0">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="swing_cone_and_twist" minOccurs="0">
                  <xs:complexType>
                     <xs:sequence>
                        <xs:element name="min" type="TargetableFloat3"
                                default="0.0 0.0 0.0" minOccurs="0">
                        </xs:element>
                        <xs:element name="max" type="TargetableFloat3"
                                default="0.0 0.0 0.0" minOccurs="0">
                        </xs:element>
                     </xs:sequence>
                  </xs:complexType>
               </xs:element>
               <xs:element name="linear" minOccurs="0">
                  <xs:complexType>
                     <xs:sequence>
                        <xs:element name="min" type="TargetableFloat3"
                                default="0.0 0.0 0.0" minOccurs="0">
                        </xs:element>
                        <xs:element name="max" type="TargetableFloat3"
                                default="0.0 0.0 0.0" minOccurs="0">
                        . . .
      <xs:element name="spring" minOccurs="0">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="angular" minOccurs="0">
                  <xs:complexType>
                     <xs:sequence>
                        <xs:element name="stiffness" type="TargetableFloat"
                                default="1.0" minOccurs="0">
                        </xs:element>
                        <xs:element name="damping" type="TargetableFloat"
                                default="0.0" minOccurs="0">
                        </xs:element>
                        <xs:element name="target_value" type="TargetableFloat"
                                default="0.0" minOccurs="0">
                        . . .
               <xs:element name="linear" minOccurs="0">
                  <xs:complexType>
                     <xs:sequence>
                        <xs:element name="stiffness" type="TargetableFloat"
                                default="1.0" minOccurs="0">
                        </xs:element>
                        <xs:element name="damping" type="TargetableFloat"
                                default="0.0" minOccurs="0">
```

```
</xs:element>
<xs:element name="target_value" type="TargetableFloat"
        default="0.0" minOccurs="0">
```

Resolves bugs 169, 210.

<setparam>'s ref attribute type has changed

The ref attribute for <setparam> is now xs:token under the following parent elements of <setparam>:

- <surface> / <generator> (in GLSL scope, was xs:string; in CG scope was xs:NCName)
- <profile_CG> / <technique> (was xs:string)
- <instance_effect> (was xs:string)
- <profile_GLSL> / <technique> (was xs:string)
- <usertype> (was xs:string)

Note: For parent <profile_GLES> / <technique>, it remains xs:NCName.

Resolves bug 203.

<surface> now has optional <format_hint> child element

If the exact format cannot be resolved using <format> then the optional <format_hint> describes the important features of the format so that the application can select a compatible or similar format. Valid child elements of <format_hint> are <channels>, <range>, <precision>, <option>, and <extra>.

Resolves bug 111.

<surface> now has optional initialization child element

Specifies how to initialize this surface. Valid initialization child elements are <init_as_null>, <init_as_target>, <init_cube>, <init_volume>, <init_planar>, and <init_from>.

Resolves bugs 108, 126, 178.

<surface> in GLSL scope now has optional <generator> child element

This results from a schema change in which <glsl_param_type> now includes:

```
<xs:element name="surface" type="glsl_surface_type"/>
```

which replaces:

```
<xs:element name="surface" type="fx_surface_common"/>
```

Resolves bug 343.

<surface> child element <init_from> is no longer an array of IDs

Changed from <xs:extension base="xs:IDREFS"> to <xs:extension base="xs:IDREF">.

Resolves bug 43. (COLLADA documents that use an array of IDs in <init_from> must be changed.)

<surface> child element <format> is no longer required

It can now occur 0 or 1 times.

Resolves bug 106.

<technique_hint> now has optional `profile` attribute

Specifies for which API profile this hint is intended. Profiles are constructed by appending this attribute's value to "`profile_`". For example, to select profile_CG, specify `profile="CG"`.

Resolves bug 136.

<technique_hint>'s `platform` attribute is now optional

Resolves bug 136.

<transparent> now has an optional `opaque` attribute

In <blinn>, <constant>, <lambert>, and <phong>, the child element <transparent> now has an optional `opaque` attribute whose valid values are:

- `A_ONE` (the default): Takes the transparency information from the color's alpha channel, where the value 1.0 is opaque.
- `RGB_ZERO`: Takes the transparency information from the color's red, green, and blue channels, where the value 0.0 is opaque, with each channel modulated independently.

The interaction between <transparent> and <transparency> is as follows:

- If <transparent> exists but <transparency> does not then transparency is assumed to have no effect on the percentage of opacity or transparency defined in <transparent>.
- If <transparency> exists but not <transparent> then <transparency> scales into the computation assuming the `A_ONE` opaque mode in which zero is fully transparent and one is fully opaque.
- If both <transparent> and <transparency> exist then both are honored.

Your application can get these results by assuming the following if <transparent> or <transparency> is not specified:

```
transparent = <color> 0.0 0.0 0.0 1.0 </color>
transparency = <float> 1.0 <float>
```

Use the following equations to get the correct results based on the opaque setting of <transparent>, where *fb* is the frame buffer (that is, the image behind what is being rendered) and *mat* is the material color before the transparency calculation.

- In `A_ONE` opaque mode:

```
result.r = fb.r * (1.0f - transparent.a * transparency) + mat.r *
    (transparent.a * transparency)
result.g = fb.g * (1.0f - transparent.a * transparency) + mat.g *
    (transparent.a * transparency)
result.b = fb.b * (1.0f - transparent.a * transparency) + mat.b *
    (transparent.a * transparency)
result.a = fb.a * (1.0f - transparent.a * transparency) + mat.a *
    (transparent.a * transparency)
```

- In `RGB_ZERO` opaque mode:

```
result.r = fb.r * (transparent.r * transparency) + mat.r *
    (1.0f -transparent.r * transparency)
result.g = fb.g * (transparent.g * transparency) + mat.g *
    (1.0f -transparent.g * transparency)
result.b = fb.b * (transparent.b * transparency) + mat.b *
    (1.0f -transparent.b * transparency)
result.a = fb.a * (luminance(transparent.rgb) * transparency) + mat.a *
    (1.0f - luminance(transparent.rgb) * transparency)
```

where `luminance` is the function, based on the ISO/CIE color standards (see ITU-R Recommendation BT.709-4), that averages the color channels into one value:

```
luminance = (transparent.r * transparency * 0.212671) +
            (transparent.g * transparency * 0.715160) +
            (transparent.b * transparency * 0.072169)
```

In the following example, the colors are used as specified but the RGB values are ignored for transparency calculations because `A_ONE` specifies that the transparency information comes from the alpha channel, not the RGB channels:

```
<transparent opaque=A_ONE><color>1 0 0.5 0</color></transparent>
```

Resolves bugs 397, 400.

<usertype> now has required `source` attribute

Added the following attribute:

```
<xs:attribute name="source" type="xs:NCName" use="required">
```

Resolves bug 190. (COLLADA documents that include <usertype> must add a `source` attribute.)

<usertype> now has optional <setparam> child element

Resolves bugs 191, 208.

<usertype>'s `name` attribute type has changed

The `name` attribute for <usertype> is now `xs:token`.

Resolves bug 203.

Known Schema Issues In Version 1.4.1

<surface>'s <init_*> child elements use less-flexible type of identifier

When moving FX surfaces (`fx_surface_common` or derived types) between XML databases, the associated images must be moved with them. This is because the <init_*> elements of <surface> use IDREFs, which are local, rather than URIs, which can be external.

Bug 339.

Specification Changes Since Version 1.4.0

- Added 1.4.1 schema changes.
- Verified specification content against the schema and corrected a variety of errors and omissions.
- Updated and corrected all examples.
- Reformatted "Attributes" and "Child Elements" sections for all elements to more clearly convey information.
- Reorganized some material.

Specification 1.4.1 Errata and Addenda

<bind_material> symbolic name binding is misleading

When a piece of geometry is declared, it can request that the geometry has a particular material, for example:

```
<polygons name="leftarm" count="2445" material="bluePaint">
```

This abstract symbol needs to be bound to a particular material instance. The instantiation is done by the application when processing the <instance_geometry> elements within the <bind_material> elements. The geometry is scanned for `material` attributes and actual material objects are bound to them as indicated by the <instance_material> `symbol` attributes.

The following example shows <bind_material> binding a material with a geometry. The connection between the <material> `id` attribute and the `material` attribute of a <polygons> element is established by the <instance_material> element:

```
...
<material id="MyMaterial">
  ...
</material>
...
<geometry>
   ...
   <polygons name="leftarm" count="2445" material="bluePaint">
   ...
</geometry>
...
<scene>
  ...
  <instance_geometry ...>
    <bind_material>
      <technique_common>
        <instance_material symbol="bluePaint" target="MyMaterial">
          ...
        </instance_material>
      </technique_common>
    </bind_material>
  </instance_geometry>
  ...
</scene>
```

Bug 436.

<convex_mesh> child elements are now all optional

Child elements <source> and <vertices>, which were previously required, are now optional. See the "Schema Changes Since Version 1.4.0" section for usage details.

Bug 382.

<instance_material> missing description for locating a parameter in <bind> and <bind_vertex_input>

The <bind> and <bind_vertex_input> elements bind the target to a parameter in an <effect>. The search string that identifies the parameter in the <effect> is specified by the `semantic` attribute. When locating the parameter in the <effect>, search in the following order:

- Find a COLLADA FX parameter by semantic
- If the profile contains shading language code, find a parameter within the shader by semantic.
- Find a COLLADA FX parameter by `sid`.
- If the profile contains shading language code, find a parameter within the shader by name.

Bug 417.

<instance_rigid_constraint> is not documented

The **<instance_rigid_constraint>** element has the following attributes:

sid	**xs:NCName**	A text string value containing the subidentifier of this element. This value must be unique within the scope of the parent element. This allows for targeting elements of the **<rigid_constraint>** instance for animation. `Optional.`
name	**xs:NCName**	Optional.
constraint	**xs:NCName**	Which **<rigid_constraint>** to instantiate. Required.

The **<instance_rigid_constraint>** element relates to the following elements:

Occurrences	Number of elements defined in the schema
Parent elements	**instance_physics_model**
Child elements	See the following subsection

The **<instance_rigid_constraint>** element has the following child elements:

Name/example	Description	Default	Occurrences
<extra>	User-defined, multirepresentable data that adds information to the **<instance_rigid_constraint>**.	N/A	0 or more

Bug 332.

<pass> render states table is missing some information or has misspellings

- color_material_enable, a Boolean, is missing from the table. It enables and disables the use of color_material. See the “Schema Changes Since Version 1.4.0” section for usage details.
- blend_color, depth_bounds, line_stipple, and logic_op_enable are not in GLES.
- clip_plane is bool4 in GLES and float4 in other profiles
- stencil_op_separate (not in GLES), missing from the table, has child elements face, fail, zfail, and zpass; see stencil_op and stencil_mask_separate for information about the values and types.
- blend_func: correct values are DST_ALPHA, ONE_MINUS_DST_ALPHA
- blend_equation: correct value is FUNC_SUBTRACT
- stencil_op: correct value is DECR_WRAP

Bugs 391, 392, 401.

<profile_*> elements' <technique> children are incomplete or incorrect <technique> (FX) list of children is incorrect

The correct child elements for <profile_*>/<technique> are the following, and they must occur in the following order if present:

Name	profile_CG	profile_COMMON	profile_GLES	profile_GLSL	Occurrences
<asset>	yes	yes	yes	-	0 or 1
<annotate>	yes	-	yes	yes	0 or more
<include>	yes	-	-	yes	0 or more
<code>	yes	-	-	yes	0 or more
<newparam>	yes	yes	yes	yes	0 or more
<setparam>	yes	-	yes	yes	0 or more
<image>	yes	yes	yes	yes	0 or more
<blinn>	-	yes	-	-	1
<constant>	-	yes	-	-	
<lambert>	-	yes	-	-	
<phong>	-	yes	-	-	
<pass>	yes	-	yes	yes	1 or more
<extra>	yes	yes	yes	yes	0 or more

Bugs 97, 237.

<transparent> now has an optional `opaque` attribute

See the "Schema Changes Since Version 1.4.0" section for details.

Bug 406.

Skinning a Skeleton in COLLADA

Definitions related to skinning in COLLADA:

- Bind shape (or base mesh): The vertices of the mesh referred to by the `source` attribute of the <skin> element.
- Joints: Nodes specified by `sid` in the <source> referred to by the <input> element with `semantic="JOINT"`. The `sid` s are typically stored in a <Name_array> where one name represents one `sid` (node). Upon instantiation of a skin controller, the <skeleton> elements define where to start the `sid` lookup. The joint matrices can be obtained at runtime from these nodes.
- Weights: Values in the <source> referred to by the <input> element with `semantic="WEIGHT"`. Typically stored in a <float_array> and taken one floating-point number at a time. The <vertex_weights> element describes the combination of joints and weights used by the skin.
- Inverse bind matrix: Values in the <source> element referred to by the <input> element with `semantic="INV_BIND_MATRIX"`. Typically stored in a `<float_array>` taken 16 floating-point numbers at a time. The <joints> element associates the joints to their inverse bind matrices.
- Bind shape matrix: A single matrix that represents the transform of the bind shape before skinning.

The skinning calculation for each vertex *v* in a bind shape is

$$outv = \sum_{i=0}^{n} \{ ((v * BSM) * IBMi * JMi) * JW \}$$

where:

- *n* = number of joints that influence vertex *v*
- BSM = bind shape matrix
- IBM*i* = inverse bind matrix of joint *i*
- JM*i* = joint matrix of joint *i*
- JW = joint weight/influence of joint *i* on vertex *v*

Common optimizations include:

- (*v* * BSM) is calculated and stored at load time.

Bug 419.

Texture Mapping in <profile_COMMON>

This section provides an introduction to textures, samplers, surfaces, and images.

To use an image as a texture, use the element relationships as follows:

```
texture->sampler->surface->image
```

From the smallest part to the largest:

- An <image> is embedded or referenced file data. It might be a format of traditional 2D planes, such as BMP, or it might be a complicated 3D format, such as DDS or OpenEXR, consisting of multiple image planes. An image is not, by itself, a surface although it might contain the complete set of information to form a surface.
- A <surface> collects one or more images to form a single cohesive structure designed for 3D hardware concepts, such as MIP mapping, cubes, and volumes. Refer to the *COLLADA Specification* for details about <surface>.
- A <sampler*> contains instructions on how to read data at a specific 1D, 2D, or 3D coordinate from a <surface>. It references the <surface> and specifies what operations to perform to sample the data at a given coordinate. A sampler's instructions include information on how to map the coordinate onto the image, such as wrap or mirror. The instructions also include filtering modes to instruct how one or more texels near by coordinate are combined to produce the final output color.
- A profile_COMMON's <texture>'s responsibility is to bind geometry's texture coordinate set (array) to a <sampler*> so that the sampler can fetch the correct colors. The `texcoord` attribute on the <texture> is actually a semantic name. It is expected that the <instance_geometry>'s <instance_material><bind_vertex_input> makes the connection between the <texture>'s `texcoord` attribute and the mesh's texture coordinate array. See <instance_material> for details about <bind_vertex_input>.

Some DCC applications also specify <extra> information that modifies the `TEXCOORD`s before they are plugged into the sampler, such as offsetU, offsetV, rotateUV, or noise.

The following is an example of texturing using <instance_material> and related elements to instantiate a material with an <image> supplied through a <sampler2D> parameter:

```
...
<image id="image_id">
  <init_from>image_file.dds</init_from>
</image>
...
<effect id="effect_id">
 ...
  <profile_COMMON>
    <technique sid="technique_sid">
      <newparam sid="surface_param_id">
        <surface>
```

```
            <init_from>image_id</init_from>
            ...
          </surface>
        </newparam>
        <newparam sid="sampler2D_param_id">
          <sampler2D>
            <source>surface_param_id</source>
            ...
          </sampler2D>
        </newparam>
        <lambert>
          <diffuse>
            <texture texture="sampler2D_param_id" texcoord="myUVs"/>
          </diffuse>
        </lambert>
        ...
</effect>
...
<material id="material_id">
  <instance_effect url="#effect_id" />
</material>
...
<geometry id="geometry_id">
  ...
  <input semantic="TEXCOORD" source="#..." offset=".." />
  <triangles material="material_symbol" count=”...”>
  ...
</geometry>
...
<scene>
  ...
  <instance_geometry url="#geometry_id">
    <bind_material>
        <technique_common>
          <instance_material symbol="material_symbol" target="#material_id">
              <bind_vertex_input semantic="myUVs" input_semantic="TEXCOORD" />
          </instance_material>
        </technique_common>
    </bind_material>
  </instance_geometry>
  ...
</scene>
```

Bugs 360, 449.