XML3D 0.4.3 Specification

Working Draft 2 March 2013

Namespace:
`http://www.xml3d.org/2009/xml3d`
This version:
http://graphics.cs.uni-sb.de/fileadmin/cguds/projects/xml3d/specification/0.4.3/
Latest version:
http://graphics.cs.uni-sb.de/fileadmin/cguds/projects/xml3d/specification/current/
Previous version:
http://graphics.cs.uni-sb.de/fileadmin/cguds/projects/xml3d/specification/0.3/
Editors:
Boris Brönner, DFKI
Sergiy Byelozyorov, Saarland University
Stefan John, Saarland University
Felix Klein, IVCI
Dmitri Rubinstein, DFKI
Philipp Slusallek, Saarland University, DFKI & IVCI
Kristian Sons, DFKI

Abstract

This specification defines the features and syntax for XML3D, an extension to HTML5 to allow interactive three-dimensional graphics graphics.

The aim is to create a web standard that is easy to learn by web designers and web programmers without deep knowledge about 3D programming. This is the main difference to approaches like WebGL and O3D. In contrast to these imperative techniques, XML3D attempts to achieve maximum compatibility with both HTML5 and XHTML. Many of XML3D's facilities are modeled directly after HTML and SVG, including its use of CSS, its approach to event handling, and its approach to the Document Object Model.

XML3D allows several graphical objects: triangle meshes, lines, point etc. Graphical objects can be grouped, styled and transformed. The feature set also includes a simple and unified shading system that is independent of source language, target architecture and rendering engine without sacrificing runtime performance (not yet available in this version).

Sophisticated applications of XML3D are possible through the use of a supplemental scripting language which accesses the Document Object Model (DOM), providing complete access to all elements, attributes and properties. A rich set of event handlers such as 'onmouseover' and 'onclick' can be assigned to any 3D object or to a group of 3D objects. Because of its compatibility and leveraging of other Web standards, features like scripting can be done on XHTML and XML3D elements simultaneously within the same web page.

Status of This document

The work on XML3D and its specification is still work in progress. XML3D is a project of the Computer Graphics Lab of the Saarland University, the DFKI and the Intel Visual Computing Institute. For more information about XML3D and our integration into Mozilla Browser and WebKit please visit the XML3D website and see our XML3D paper.

Table of Contents

Elements

- int
- lightshader
- light
- mesh
- proto
- shader
- texture
- transform
- view
- xml3d

Data Types

- XML3DBox
- XML3DDataEntry
- XML3DDataResult
- XML3DMatrix
- XML3DRay
- XML3DRotation
- XML3DVec3

Elements

Element <xml3d>

Description

The xml3d element is the root of a XML3D scene and also describes the rendering area this scene is displayed in.

In a stand-alone XML3D document, the xml3d element is the root element of the XML file. It also can be embedded inline as a fragment within a parent XML document. In this case, the standard XML namespace rules apply to indicate which elements belong to which namespace.

Example for a stand-alone XML3D document:

```
<xml3d width="300" height="200" xmlns="http://www.xml3d.org/2009/xml3d">
  <group>
    <mesh>
    ...
    </mesh>
  </group>
</xml3d>
```

Example for a XML3D embedded into XHTML:

```
<?xml version="1.0" encoding="utf-8"?>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xml3d="http://www.xml3d.org/2009/xml3d">
  <head>
    <title>A simple Mesh embedded into XHTML</title>
  </head>
  <body>
    <div style="color: red">
      <xml3d:xml3d id="myXml3d">
        <xml3d:defs>
          <xml3d:data id="mySimpleMesh">
            <xml3d:int name="index">0 1 2 1 2 3</xml3d:int>
            <xml3d:float3 name="position">-1.0 -1.0 0.0 1.0 -1.0 0.0 -1.0 1.0 0.0 1.0 1.0 0.0</xml3d:float3>
            <xml3d:float3 name="normal">0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 1.0</xml3d:float3>
            <xml3d:float2 name="texcoord">0.0 1.0 1.0 1.0 0.0 0.0 1.0 0.0</xml3d:float2>
          </xml3d:data>
        </xml3d:defs>
        <xml3d:mesh src="#mySimpleMesh"  type="triangles"/>
      </xml3d:xml3d>
    </div>
  </body>
</html>
```

Rendering area

The xml3d element defines the dimension and the background appearance. Dimensions can either be defined by the attributes height and width, or - if those attributes are not given - the element can be sized arbitrarily by a style sheet.

The background of the rendering area can be defined using the CSS2 Background properties. The initial value of the background is *transparent*. If the xml3d element is embedded in some other rendering (like HTML) the background of the parent box's shines through, otherwise the background is black.

Scene graph and transformation

The xml3d is the root node of the scene's graph. It defines the *world coordinate system* of the scene. All transformations defined by group child nodes are *local coordinate systems* relative to the world coordinate system. XML3D uses a Cartesian, right-handed, three-dimensional coordinate system.

Views

The initial view to the scene is defined by the reference defined by the *activeView* attribute. If no *activeView* is defined, the renderer should set the *activeView* reference to the first applicable child node of the xml3d element. If the reference is not valid, or if there is no applicable view node as child node of the xml3d element, the renderer will not render the scene. In this case the rendering area should be filled with an error image. A script can set and change the *activeView* reference during runtime. If the *activeView* reference changes or gets deleted, the rules above are applied again.

Attributes

Name	Type	Default Value	Description
height	<int>	"600"	The height of the xml3d canvas in pixels.
width	<int>	"800"	The width of the xml3d canvas in pixels.
activeView	<IDREF >	""	Reference to the active view element.

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style
EventAttributes	onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup

Content Model

Child elements

(xml3d | data | defs | group | mesh | transform | shader | light | lightshader | script | proto | float | float2 | float3 | float4 | float4x4 | int | int4 | bool | texture | img | video | view)*

Interface XML3DXml3dElement

IDL Definition

```
interface XML3DXml3dElement :
  XML3DElement,
  EventAttributes
{
  // Attributes
  attribute int height;
  attribute int width;
  // Methods
  XML3DVec3 createXML3DVec3();
  XML3DRotation createXML3DRotation();
  XML3DMatrix createXML3DMatrix();
  XML3DRay createXML3DRay();
  XML3DElement getElementByPoint(in int x, in int y, in XML3DVec3 hitPoint, in XML3DVec3 hitNormal);
  XML3DRay generateRay(in int x, in int y);
  XML3DElement getElementByRay(in XML3DRay ray, in XML3DVec3 hitPoint, in XML3DVec3 hitNormal);
  XML3DBox getBoundingBox();
}
```

Attributes

height

The height of the xml3d canvas in pixels.

width
: The width of the xml3d canvas in pixels.

Methods

createXML3DVec3
: Creates a new instance of XML3DVec3f data type with x=y=z=0.0.

createXML3DRotation
: Creates a new instance of XML3DRotation with axis = (0, 0, 1) and angle = 0.0.

createXML3DMatrix
: Creates a new instance of XML3DMatrix. This matrix is the identity matrix.

createXML3DRay
: Creates a new instance of XML3DMatrix. This matrix is the identity matrix.

getElementByPoint
: Returns the element that is visible at point x,y or null if no element is visible. The coordinates x,y are relative to the xml3d element.

generateRay
: Returns a ray in 3D space that corresponds to point x,y. The coordinates x,y are relative to the xml3d element.

getElementByRay
: Returns the first element that is hit along the passed ray or null, if no element is hit.

getBoundingBox
: Get the Bounding Box of the element in local space.

up

Element <data>

Description

Data node which combines multiple named value elements and can be referred and contained by data containers.

The elements data, mesh, shader, and lightshader are *data containers* that combine all contained *value elements* (int, float, float2, float3, float4, float4x4, bool, and texture) into a *data table* - a map with the *name* attribute of the value element as a **unique** key and the *content* of the value element as value. Value elements can be direct children of the data container or part of another data element that is either a child of the data container or referred via the *src* attribute.

In case multiple value elements with the same name are part of a data container, only one key-value-pair is included into the resulting named data table, according to the following rules:

1. If the data container referres a data element via src, all child elements are ignored and the data table of the **referred data element is reused directly**
2. A name-value pair of a **child value** element **overrides** a name-value pair with the same name of a **child data** element
3. A name-value pair of a **later child value** element **overrides** a name-value pair with the same name of a **former child value** element
4. A name-value pair of a **later child data** element **overrides** a name-value pair with the same name of a **former child data** element

Attributes

Name	Type	Default Value	Description
compute	<string>	""	TODO: Documentation
src	<IDREF>	""	Reference to another data element or data file as URI. Can be a location (URL) to a data file or an xml3d data element. If src is defined, all child elements are ignored. Thus, the data table defined by the referred content is reused directly.
proto	<IDREF>	""	Reference to another data node that should be treated as a template. Children of this node will replaced input nodes that are marked with respective replaceby attribute.
filter	<string>	""	TODO: Documentation

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

(xml3d | data | defs | group | mesh | transform | shader | light | lightshader | script | proto | float | float2 | float3 | float4 | float4x4 | int | int4 | bool | texture | img | video | view)*

Interface XML3DDataElement

IDL Definition

```
interface XML3DDataElement :
  XML3DNestedDataContainerElement
{
  // Attributes
  attribute DOMString filter;
  // Methods
  XML3DDataResult getResult(in string filter);
  string getOutputFieldNames();
}
```

Attributes

filter

Methods

getResult
: Returns set of output fields for this node. This may start processing of the Xflow graph needed to compute them. Filter argument can be used to choose fields that need to be computed, which may reduce computation time. Repeating names in the filter or those that are not available as outputs will be ignored.

getOutputFieldNames
: Retuns a list of available output fields. Any subset of these fields may be passed to the getResult method as a filter.

up

Element <defs>

Description

Container element for elements that can be referenced from inside the scene structure.

All elements that are inside a defs section can be referenced by either elements in the scene graph or by other elements in an arbitrary defs section of the XML3D document. The defs element is similar to the SVG 'defs' element. It is recommended to define referenced elements inside a defs block to increase readability and understandability, even if some elements could also be defined directly inside the scene graph.

Elements that are descendants of a defs element are not rendered directly. defs may appear as child of the xml3d or group element.

Attributes

-

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

(data | group | mesh | transform | shader | light | lightshader | script | proto | img | video | view)*

Interface XML3DDefsElement

IDL Definition

```
interface XML3DDefsElement :
  XML3DElement
{


}
```

up

Element <group>

Description

Grouping node with transformation capabilities and surface shader assignment.

Transformation

The group element defines a coordinate system for its children that is relative to the coordinate systems of its ancestors. The elements local coordinate system is defined as a 4x4 matrix. This matrix is determined from the CSS 'transform' property as defined in CSS 3D Transforms Module Level 3 and the reference to an element that can provide a transformation via the 'transform' attribute.

The local matrix of a group node is calculated as:

$M_{css} * M_{reference}$

Where

M_{css}
: is the matrix that is the result of the elements 'transform' style property or the identity matrix if the style is not defined.

$M_{reference}$
: is the matrix that is provided from the element referenced via the 'transform' attribute or the identity matrix if the reference is not defined or not valid.

Surface shader

The shader attribute of a group element defines the surface shading of all its children. The shading defined by parent nodes is overridden, while following group nodes can override the shading state again.
The shader and transform attributes are depreciated. Use CSS to assign shader and transformations!

Attributes

Name	Type	Default Value	Description
visible	<boolean>	"true"	If "false", the element and all it's children are not taken into account during rendering. This flag does not affect children referenced from other parts of the scene graph.
transform	<IDREF >	""	Reference to an element that can provide a 3D transformation (i.e. transform)
shader	<IDREF >	""	Reference to an element that can provide a surface shader (i.e. shader)

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style
EventAttributes	onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup

Content Model

Child elements

(xml3d | data | defs | group | mesh | transform | shader | light | lightshader | script | proto | float | float2 | float3 | float4 | float4x4 | int | int4 | bool | texture | img | video | view)*

Interface XML3DGroupElement

IDL Definition

```
interface XML3DGroupElement :
  XML3DGraphElement
{

  // Methods
  XML3DMatrix getLocalMatrix();
  XML3DBox getBoundingBox();
}
```

Methods

getLocalMatrix
: Matrix of local coordinate system defined by transform reference and CSS transformation. Returns a new matrix. Changing this matrix will not change the group's transformation.

getBoundingBox
: Get the Bounding Box of the element in local space.

up

Element <mesh>

Description

Geometry node that describes the shape of a polyhedral object in 3D.

This is very generic description of a 3D mesh. It clusters a number of data fields and binds them to a certain name. The interpretation of these data fields is job of the currently active shader. Only connectivity information is required to build the primitives defined by the *type* attribute:

triangles
: A float3 element with name *index* is required. The data type of the bound element has to be evaluable to unsigned int. Every three entries in this field compose one triangle. The number of field entries should be an even multiple of 3. If not, the last entry or the last two entries are ignored. All other fields should have at least as many tuples as the largest value in the index field.
Note: Please be aware that some renderers can't handle indices larger than 2^{16}

Attributes

Name	Type	Default Value	Description
visible	<boolean>	"true"	If "false", the element and all it's children are not taken into account during rendering. This flag does not affect children referenced from other parts of the scene graph.
type	<string>	"triangles"	The type of geometric primitive described by this mesh element.
src	<IDREF >	""	Reference to another data element or data file as URI. Can be a location (URL) to a data file or an xml3d data element. If src is defined, all child elements are ignored. Thus, the data table defined by the referred content is reused directly.
proto	<IDREF >	""	Reference to another data node that should be treated as a template. Children of this node will replaced input nodes that are marked with respective replaceby attribute.
compute	<string>	""	TODO: Documentation

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style
EventAttributes	onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup

Content Model

Child elements

(xml3d | data | defs | group | mesh | transform | shader | light | lightshader | script | proto | float | float2 | float3 | float4 | float4x4 | int | int4 | bool | texture | img | video | view)*

Interface XML3DMeshElement

IDL Definition

```
interface XML3DMeshElement :
  XML3DGeometryElement
{
  // Attributes
  readonly attribute DOMString type;
  attribute DOMString compute;
  // Methods
  XML3DBox getBoundingBox();
}
```

Attributes

type
: The type of geometric primitive described by this mesh element.

compute

Methods

getBoundingBox
: Get the Bounding Box of the element in local space.

up

Element <transform>

Description

General geometric transformation element, that allows to define a transformation matrix using five well understandable entities.

The *center* attribute specifies a translation offset from the origin of the local coordinate system (0,0,0). The *rotation* attribute specifies a rotation of the coordinate system. The *scale* field specifies a non-uniform scale of the coordinate system. Scale values may have any value: positive, negative (indicating a reflection), or zero. A value of zero indicates that any child geometry shall not be displayed. The *scaleOrientation* specifies a rotation of the coordinate system before the scale (to specify scales in arbitrary orientations). The scaleOrientation applies only to the scale operation. The *translation* field specifies a translation to the coordinate system.

The resulting matrix *M* that represents the element's coordinate system is calculated by a series of intermediate transformations. In matrix transformation notation, where *C* (center), *SR* (scaleOrientation), *T* (translation), *R* (rotation), and *S* (scale) are the equivalent transformation matrices, the resulting matrix is calculated as:

```
M = T * C * R * SR * S * -SR * -C
```

This behavior is similar to transform descriptions in OpenInventor or VRML/X3D.

Attributes

Name	Type	Default Value	Description
translation	<XML3DVec3>	"0 0 0"	The translation part of the transformation.
scale	<XML3DVec3>	"1 1 1"	The scaling part of the transformation.
rotation	<XML3DRotation>	"0 0 1 0"	The rotation part of the transformation.
center	<XML3DVec3>	"0 0 0"	Origin for scale and rotation.
scaleOrientation	<XML3DRotation>	"0 0 1 0"	Rotational orientation for scale.

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

None.

Interface XML3DTransformElement

IDL Definition

```
interface XML3DTransformElement :
  XML3DTransformProviderElement
{
  // Attributes
  readonly attribute XML3DVec3 translation;
  readonly attribute XML3DVec3 scale;
  readonly attribute XML3DRotation rotation;
  readonly attribute XML3DVec3 center;
  readonly attribute XML3DRotation scaleOrientation;

}
```

Attributes

translation
: The translation part of the transformation.

scale
: The scaling part of the transformation.

rotation
: The rotation part of the transformation.

center
: Origin for scale and rotation.

scaleOrientation
: Rotational orientation for scale.

up

Element <shader>

Description

The shader element describes a surface shader for a geometry.

The shader element connects arbitrary shader attributes with some shader code. The shader code is referenced with the *script* reference. The shader attributes are bound to the shader using the bind mechanism.

The URI syntax is used to define the shader script. This can be either a URL pointing to a script location in- or outside the current resource or a URN pointing to a XML3D standard shader. Following XML3D fixed-function shaders are defined:

Matte:
`urn:xml3d:shader:matte`
Diffuse:
`urn:xml3d:shader:diffuse`
Phong:
`urn:xml3d:shader:phong`
... more to come ...

Example:

```
<shader id="red" script="urn:xml3d:shader:phong">
    <float3 name="diffuseColor">1 0 0</float3>
</shader>
```

Attributes

Name	Type	Default Value	Description
script	<IDREF>	""	Reference to the shader script as URI. Can be a location (URL) or one of the pre-defined shaders as URN. TODO: Should be of type "AnyURI"
src	<IDREF>	""	Reference to another data element or data file as URI. Can be a location (URL) to a data file or an xml3d data element. If src is defined, all child elements are ignored. Thus, the data table defined by the referred content is reused directly.
proto	<IDREF>	""	Reference to another data node that should be treated as a template. Children of this node will replaced input nodes that are marked with respective replaceby attribute.
compute	<string>	""	TODO: Documentation

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

(xml3d | data | defs | group | mesh | transform | shader | light | lightshader | script | proto | float | float2 | float3 | float4 | float4x4 | int | int4 | bool | texture | img | video | view)*

Interface XML3DShaderElement

IDL Definition

```
interface XML3DShaderElement :
  XML3DSurfaceShaderProviderElement
{


}
```

up

Element <light>

Description

The light element defines a light in the scene graph.

The light source location and orientation is influenced by the scene graph transformation hierarchy. The radiation characteristics of the light source is defined by the referenced lightshader (s. shader attribute). The light can be dimmed using the *intensity* attribute and can be switched on/off using the *visible* attribute. If *global* is set to 'false', the light source will only light the objects that is contained in its parent group or xml3d element. Otherwise it will illuminate all the objects in its scene graph.

Intensity and visibility (switch) should be controllable via CSS property.

Attributes

Name	Type	Default Value	Description
visible	<boolean>	"true"	If "false", the element and all it's children are not taken into account during rendering. This flag does not affect children referenced from other parts of the scene graph.
shader	<IDREF>	""	Reference to a lightshader element.
global	<boolean>	"false"	If 'false', the light source will only light the children of it's parent node.
			Will be multiplied with the light source contribution. It is possible to 'dim' the light using values below 1 or to

intensity <float>	"1"	brighten it up using values above 1.

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style
EventAttributes	onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup

Content Model

Child elements

None.

Interface XML3DLightElement

IDL Definition

```
interface XML3DLightElement :
  XML3DGraphElement
{
  // Attributes
  attribute boolean global;
  attribute float intensity;

}
```

Attributes

global
: If 'false', the light source will only light the children of it's parent node.

intensity
: Will be multiplied with the light source contribution. It is possible to 'dim' the light using values below 1 or to brighten it up using values above 1.

up

Element <lightshader>

Description

The light shader element describes a light source.

The lightshader element connects arbitrary light shader attributes with a light shader code. The light shader code is referenced via the *script* reference. The shader attributes are bound to the shader using the data mechanism.

The URI syntax is used to define the light shader script. This can be either a URL pointing to a script location in- or outside the current resource or a URN pointing to a XML3D standard light shader. Following XML3D fixed-function light shaders are defined:

Pointlight:
`urn:xml3d:lightshader:point`
Spotlight:
`urn:xml3d:lightshader:spot`
Directional Light:
`urn:xml3d:lightshader:directional`
... more to come ...

Example:

```
<lightshader id="myLight" script="urn:xml3d:lightshader:point">
    <float3 name="color">1 1 0.8</float3>
    <float3 name="attenuation">1 0 0</float3>
</lightshader>
```

Attributes

Name	Type	Default Value	Description
script	<IDREF>	""	Reference to the shader script as URI. Can be a location (URL) or one of the pre-defined shaders as URN. TODO: Should be of type "AnyURI"
src	<IDREF>	""	Reference to another data element or data file as URI. Can be a location (URL) to a data file or an xml3d data element. If src is defined, all child elements are ignored. Thus, the data table defined by the referred content is reused directly.
proto	<IDREF>	""	Reference to another data node that should be treated as a template. Children of this node will replaced input nodes that are marked with respective replaceby attribute.
compute	<string>	""	TODO: Documentation

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

(xml3d | data | defs | group | mesh | transform | shader | light | lightshader | script | proto | float | float2 | float3 | float4 | float4x4 | int | int4 | bool | texture | img | video | view)*

Interface XML3DLightShaderElement

IDL Definition

```
interface XML3DLightShaderElement :
  XML3DLightShaderProviderElement
{


}
```

up

Element <proto>

Description

Same as data node, but can be used as a prototype via proto attribute of data

Attributes

Name	Type	Default Value	Description
compute	<string>	""	TODO: Documentation
src	<IDREF>	""	Reference to another data element or data file as URI. Can be a location (URL) to a data file or an xml3d data element. If src is defined, all child elements are ignored. Thus, the data table defined by the referred content is reused directly.
proto	<IDREF>	""	Reference to another data node that should be treated as a template. Children of this node will replaced input nodes that are marked with respective replaceby attribute.
filter	<string>	""	TODO: Documentation

Common Attributes

Group Name	Attributes

CoreAttributes id, class
StyleAttributes style

Content Model

Child elements

(xml3d | data | defs | group | mesh | transform | shader | light | lightshader | script | proto | float | float2 | float3 | float4 | float4x4 | int | int4 | bool | texture | img | video | view)*

Interface XML3DProtoElement

IDL Definition

```
interface XML3DProtoElement :
  XML3DNestedDataContainerElement
{
  // Attributes
  attribute DOMString filter;
  // Methods
  string getOutputFieldNames();
}
```

Attributes

filter

Methods

getOutputFieldNames
: Retuns a list of available output fields. Any subset of these fields may be passed to the getResult method as a filter.

up

Element <float>

Description

The text nodes of the float element specify an array of float values.

User agents must interpret the contents of the element as the values. The values of the array are separated with whitespace characters only.

Example:

```
<float>1.0 2 -3.1415 20090.0098</float>
```

Attributes

Name	Type	Default Value	Description
name	<string>	""	The name to bind the data field to. This could be for example a shader field.
param	<boolean>	"false"	Name to be used when this input is to be replaced with data from the template.
key	<float>	"0.0"	Defines sequence number of the data input.

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

None.

Interface XML3DFloatElement

IDL Definition

```
interface XML3DFloatElement :
  XML3DDataSourceElement
{
  // Attributes
  attribute FloatArray value;

}
```

Attributes

value

up

Element <float2>

Description

The text nodes of the float2 element specify an array of 2-tuples of floats.

User agents must interpret the contents of the element as the values. The values of the array are separated with whitespace characters only. The number of elements must be a multiple of 2.

Example:

```
<float2>1.0 0.0 0.5 0.5 0.25 0</float2>
```

Attributes

Name	Type	Default Value	Description
name	<string>	""	The name to bind the data field to. This could be for example a shader field.
param	<boolean>	"false"	Name to be used when this input is to be replaced with data from the template.
key	<float>	"0.0"	Defines sequence number of the data input.

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

None.

Interface XML3DFloat2Element

IDL Definition

```
interface XML3DFloat2Element :
  XML3DDataSourceElement
{
  // Attributes
  attribute Float2Array value;

}
```

Attributes

value

up

Element <float3>

Description

The text nodes of the float3 element specify an array of 3-tuples of floats.

User agents must interpret the contents of the element as the values. The values of the array are separated with whitespace characters only. The number of elements must be a multiple of 3.

Example:

```
<float3>1.0 0.0 -0.25 0.5 0.3 -5.23</float3>
```

Attributes

Name	Type	Default Value	Description
name	<string>	""	The name to bind the data field to. This could be for example a shader field.
param	<boolean>	"false"	Name to be used when this input is to be replaced with data from the template.
key	<float>	"0.0"	Defines sequence number of the data input.

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

None.

Interface XML3DFloat3Element

IDL Definition

```
interface XML3DFloat3Element :
  XML3DDataSourceElement
{
  // Attributes
  attribute Float3Array value;

}
```

Attributes

value

up

Element <float4>

Description

The text nodes of the float4 element specify an array of 4-tuples of floats.

User agents must interpret the contents of the element as the values. The values of the array are separated with whitespace characters only. The number of elements must be a multiple of 4.

Example:

```
<float4>2.21 -5.0 -0.25 0.5 0.3 -5.23 200 20.20</float4>
```

Attributes

Name	Type	Default Value	Description
name	<string>	""	The name to bind the data field to. This could be for example a shader field.
param	<boolean>	"false"	Name to be used when this input is to be replaced with data from the template.
key	<float>	"0.0"	Defines sequence number of the data input.

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

None.

Interface XML3DFloat4Element

IDL Definition

```
interface XML3DFloat4Element :
  XML3DDataSourceElement
{
  // Attributes
  attribute Float4Array value;

}
```

Attributes

value

up

Element <float4x4>

Description

The text nodes of the float4 element specify an array of 4-tuples of floats.

Attributes

Name	Type	Default Value	Description
name	<string>	""	The name to bind the data field to. This could be for example a shader field.
param	<boolean>	"false"	Name to be used when this input is to be replaced with data from the template.
key	<float>	"0.0"	Defines sequence number of the data input.

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

None.

Interface XML3DFloat4x4Element

IDL Definition

```
interface XML3DFloat4x4Element :
  XML3DDataSourceElement
{
  // Attributes
  attribute Float4x4Array value;

}
```

Attributes

value

up

Element <int>

Description

The text nodes of the int element specify an array of integer values.

User agents must interpret the contents of the element as the values. The values of the array are separated with whitespace characters only.

Example:

```
<int>0 1 2 0 3 4 1 3 5</int>
```

Attributes

Name	Type	Default Value	Description
name	<string>	""	The name to bind the data field to. This could be for example a shader field.
param	<boolean>	"false"	Name to be used when this input is to be replaced with data from the template.
key	<float>	"0.0"	Defines sequence number of the data input.

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

None.

Interface XML3DIntElement

IDL Definition

```
interface XML3DIntElement :
  XML3DDataSourceElement
{
  // Attributes
  attribute IntArray value;

}
```

Attributes

value

up

Element <int4>

Description

The text nodes of the int element specify an array of 4-tuples of integers.

Attributes

Name	Type	Default Value	Description
name	<string>	""	The name to bind the data field to. This could be for example a shader field.
param	<boolean>	"false"	Name to be used when this input is to be replaced with data from the template.
key	<float>	"0.0"	Defines sequence number of the data input.

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

None.

Interface XML3DInt4Element

IDL Definition

```
interface XML3DInt4Element :
  XML3DDataSourceElement
{
  // Attributes
  attribute IntArray value;

}
```

Attributes

value

up

Element <bool>

Description

The text nodes of the bool element specify an array of boolean values.

User agents must interpret the contents of the element as the values. The boolean *true* is encoded as text value 'true', the boolean *false* is encoded as text value 'false'. There are only whitespace characters between the boolean values of an array.

Example:

```
<bool>true true false true</bool>
```

Attributes

Name	Type	Default Value	Description
name	<string>	""	The name to bind the data field to. This could be for example a shader field.
param	<boolean>	"false"	Name to be used when this input is to be replaced with data from the template.

key	<float>	"0.0"	Defines sequence number of the data input.

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

None.

Interface XML3DBoolElement

IDL Definition

```
interface XML3DBoolElement :
  XML3DDataSourceElement
{
  // Attributes
  attribute BoolArray value;

}
```

Attributes

value

up

Element <texture>

Description

Set states on how to sample a texture from an image and to apply to a shape.

The texture source and its dimensions are defined by the texture element's children. The states how to apply the texture is set via the texture element's attributes. Use the attributes to influence

- the dimensions of the texture (type)
- how the texture is applied, if texture coordinates fall outside 0.0 and 1.0 (wrapS, wrapT, wrapU)
- how to apply the texture if the area to be textured has more or fewer pixels than the texture (filterMin, filterMag)
- how to create minified versions of the texture (filterMip)
- what border color to use, if one of the wrapping states is set to 'border'

See the attribute documentation for more details.
Note: As per the OpenGL ES spec, a texture will be rendered black if:

- The width and height of the texture are not power-of-two **and**
- The texture wrap mode is not CLAMP_TO_EDGE **or**
- filterMin is neither NEAREST nor LINEAR

Attributes

Name	Type	Default Value	Description
name	<string>	""	The name to bind the data field to. This could be for example a shader field.
param	<boolean>	"false"	Name to be used when this input is to be replaced with data from the template.
key	<float>	"0.0"	Defines sequence number of the data input.
type	<string>	"2d"	Set the dimensions of the texture.
filterMin	<string>	"linear"	Specifies how to apply the texture if the area to be textured has fewer pixels than the texture.

filterMag	<string>	"linear"	Specifies how to apply the texture if the area to be textured contains more pixels than the texture.
filterMip	<string>	"nearest"	Specifies what mipmap level to use, when a minification filter is required for the texture.
wrapS	<string>	"clamp"	Specifies what happens when the texture coordinates fall outside of the 0.0 to 1.0 range in s direction.
wrapT	<string>	"clamp"	Specifies what happens when the texture coordinates fall outside of the 0.0 to 1.0 range in t direction.
wrapU	<string>	"clamp"	Specifies what happens when the texture coordinates fall outside of the 0.0 to 1.0 range in u direction.
borderColor	<string>	""	Specifies what color to use, if wrapping mode is 'border' and texture coordinates fall outside of the 0.0 to 1.0 range.

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style

Content Model

Child elements

(img | video)

Interface XML3DTextureElement

IDL Definition

```
interface XML3DTextureElement :
  XML3DDataSourceElement
{
  // Attributes
  attribute DOMString type;
  attribute DOMString filterMin;
  attribute DOMString filterMag;
  attribute DOMString filterMip;
  attribute DOMString wrapS;
  attribute DOMString wrapT;
  attribute DOMString wrapU;
  attribute DOMString borderColor;

}
```

Attributes

type
: Set the dimensions of the texture.

filterMin
: Specifies how to apply the texture if the area to be textured has fewer pixels than the texture.

filterMag
: Specifies how to apply the texture if the area to be textured contains more pixels than the texture.

filterMip
: Specifies what mipmap level to use, when a minification filter is required for the texture.

wrapS
: Specifies what happens when the texture coordinates fall outside of the 0.0 to 1.0 range in s direction.

wrapT
: Specifies what happens when the texture coordinates fall outside of the 0.0 to 1.0 range in t direction.

wrapU
: Specifies what happens when the texture coordinates fall outside of the 0.0 to 1.0 range in u direction.

borderColor
: Specifies what color to use, if wrapping mode is 'border' and texture coordinates fall outside of the 0.0 to 1.0 range.

up

Element <view>

Description

The view node interface represents a camera in 3D world coordinates.

Attributes

Name	Type	Default Value	Description
visible	<boolean>	"true"	If "false", the element and all it's children are not taken into account during rendering. This flag does not affect children referenced from other parts of the scene graph.
position	<XML3DVec3>	"0 0 0"	The position of the camera in local coordinates.
orientation	<XML3DRotation>	"0 0 1 0"	The orientation of the camera relative to the default orientation. In the default position and orientation, the viewer is on the Z-axis looking down the -Z-axis toward the origin with +X to the right and +Y straight up.
fieldOfView	<float>	"0.785398"	The fieldOfView field specifies a preferred minimum viewing angle from this viewpoint in radians. A small field of view roughly corresponds to a telephoto lens; a large field of view roughly corresponds to a wide-angle lens. The field of view shall be greater than zero and smaller than pi. The value of fieldOfView represents the minimum viewing angle in any direction axis perpendicular to the view.

Common Attributes

Group Name	Attributes
CoreAttributes	id, class
StyleAttributes	style
EventAttributes	onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup

Content Model

Child elements

None.

Interface XML3DViewElement

IDL Definition

```
interface XML3DViewElement :
  XML3DGraphElement
{
  // Attributes
  readonly attribute XML3DVec3 position;
  readonly attribute XML3DRotation orientation;
  attribute float fieldOfView;
  // Methods
  void setDirection(in XML3DVec3 direction);
  void setUpVector(in XML3DVec3 up);
  void lookAt(in XML3DVec3 target);
  XML3DVec3 getDirection();
  XML3DVec3 getUpVector();
  XML3DMatrix getViewMatrix();
}
```

Attributes

position
: The position of the camera in local coordinates.

orientation
: The orientation of the camera relative to the default orientation. In the default position and orientation, the viewer is on the Z-axis looking down the -Z-axis toward the origin with +X to the right and +Y straight up.

fieldOfView
: The fieldOfView field specifies a preferred minimum viewing angle from this viewpoint in radians. A small field of view roughly corresponds to a telephoto lens; a large field of view roughly corresponds to a wide-angle lens. The field of view shall be greater than zero and smaller than pi. The value of fieldOfView represents the minimum viewing angle in any direction axis perpendicular to the view.

Methods

setDirection
: Rotates the view so that it's orientation points towards direction. Position of the view is not modified. The view is rotated so that the upVector is preserved.

setUpVector
: Modifies the orientation, so that the default up vector (0, 1, 0) rotated by the orientation becomes the view's new up vector. Position of the view is not modified.

lookAt
: Modifies the orientation, so that the direction of the view points to the point specified by the target parameter. The position of the view is not modified.

getDirection
: Get the direction of the view defined by the orientation attribute. The result is the orientation multiplied with the default direction (0, 0, -1). Creates a new vector instance, the element is not changed.

getUpVector
: Get the up vector of the view defined by the orientation attribute. The result is the orientation multiplied with the default up vector (0, 1, 0). Creates a new vector instance, the element is not changed.

getViewMatrix
: Matrix of view coordinate system defined by orientation and position attribute of the element.

up

Data Types

This section describes the data types that are provided via the XML3D elements' DOM interfaces.

Interface XML3DBox

Description

TODO: Documentation

IDL Definition

```
interface XML3DBox :
{
  // Attributes
  attribute XML3DVec3 min;
  attribute XML3DVec3 max;
  // Methods
  XML3DVec3 size();
  XML3DVec3 center();
  void makeEmpty();
  boolean isEmpty();
  XML3DBox set(in XML3DBox second);
}
```

Attributes

min
max

Methods

size
center
makeEmpty
isEmpty
set
: Sets the values of this box from the second box passed as parameter. The method returns this.

up

Interface XML3DDataEntry

Description

The XML3DDataEntry datatype represents an immutable Xflow output field.

IDL Definition

```
interface XML3DDataEntry :
{
  // Attributes
  readonly attribute DataFieldType type;
  readonly attribute anySimpleType value;

}
```

Attributes

type
: Specifies type of the field.

value
: Contains the value of the field.

up

Interface XML3DDataResult

Description

The XML3DDataResult datatype represents a map from the names to the output fields, which result from the computation of the Xflow graph.

IDL Definition

```
interface XML3DDataResult :
{
  // Attributes
  readonly attribute unsignedInt length;
  // Methods
  string key(in unsignedInt index);
  XML3DDataEntry get(in string name);
}
```

Attributes

length
: Specifies number of output fields in this result.

Methods

key
: Returns key corresponding to index or NULL if index is out of [0, length-1] bounds. This method should be used to implement foreach functionality that iterates over fields in this object.

get
: Returns data entry corresponding to the key or NULL if key is not present in the map.

up

Interface XML3DMatrix

Description

The Xml3dMatrix interface represents a 4x4 homogeneous matrix.

IDL Definition

```
interface XML3DMatrix :
```

```
{
  // Attributes
  attribute float m11;
  attribute float m12;
  attribute float m13;
  attribute float m14;
  attribute float m21;
  attribute float m22;
  attribute float m23;
  attribute float m24;
  attribute float m31;
  attribute float m32;
  attribute float m33;
  attribute float m34;
  attribute float m41;
  attribute float m42;
  attribute float m43;
  attribute float m44;
  // Methods
  void setMatrixValue(in string str) raises(DOMException);
  XML3DMatrix multiply(in XML3DMatrix secondMatrix);
  XML3DMatrix inverse() raises(DOMException);
  XML3DMatrix translate(in float x, in float y, in float z);
  XML3DMatrix scale(in float scaleX, in float scaleY, in float scaleZ);
  XML3DMatrix rotate(in float rotX, in float rotY, in float rotZ);
  XML3DMatrix rotateAxisAngle(in float x, in float y, in float z, in float angle);
}
```

Attributes

m11
: Represents the value in the 1st column of the 1st row.

m12
: Represents the value in the 2nd column of the 1st row.

m13
: Represents the value in the 3rd column of the 1st row.

m14
: Represents the value in the 4th column of the 1st row.

m21
: Represents the value in the 1st column of the 2nd row.

m22
: Represents the value in the 2nd column of the 2nd row.

m23
: Represents the value in the 3rd column of the 2nd row.

m24
: Represents the value in the 4th column of the 2nd row.

m31
: Represents the value in the 1st column of the 3rd row.

m32
: Represents the value in the 2nd column of the 3rd row.

m33
: Represents the value in the 3rd column of the 3rd row.

m34
: Represents the value in the 4th column of the 3rd row.

m41
: Represents the value in the 1st column of the 4th row.

m42
: Represents the value in the 2nd column of the 4th row.

m43
: Represents the value in the 3rd column of the 4th row.

m44
: Represents the value in the 4th column of the 4th row.

Methods

setMatrixValue
: The setMatrixValue method replaces the existing matrix with one computed from parsing the passed string as though it had been assigned to the transform property in a CSS style rule.

multiply
: The multiply method returns a new matrix which is the result of this matrix multiplied by the passed matrix, with the passed matrix to the right. This matrix is not modified.

inverse
: The inverse method returns a new matrix which is the inverse of this matrix. This matrix is not modified.

translate
: The translate method returns a new matrix which is this matrix post multiplied by a translation matrix containing the passed values. If the z component is undefined, a 0 value is used in its place. This matrix is not modified.

scale
: The scale method returns a new matrix which is this matrix post multiplied by a scale matrix containing the passed values. If the z component is undefined, a 1 value is used in its place. If the y component is undefined, the x component value is used in its place. This matrix is not modified.

rotate
: The rotate method returns a new matrix which is this matrix post multiplied by each of 3 rotation matrices about the major axes, first X, then Y, then Z. If the y and z components are undefined, the x value is used to rotate the object about the z axis, as though the vector (0,0,x) were passed. All rotation values are in degrees. This matrix is not modified.

rotateAxisAngle
: The rotateAxisAngle method returns a new matrix which is this matrix post multiplied by a rotation matrix with the given axis and angle. The right-hand rule is used to determine the direction of rotation. All rotation values are in degrees. This matrix is not modified.

up

Interface XML3DRay

Description

The XML3DRay datatype represents a spatial ray with origin and direction.

IDL Definition

```
interface XML3DRay :
{
  // Attributes
  attribute XML3DVec3 origin;
  attribute XML3DVec3 direction;
  // Methods
  XML3DRay set(in XML3DRay second);
}
```

Attributes

origin
direction

Methods

set
: Sets the values of this ray from the second ray passed as parameter. The method returns this.

up

Interface XML3DRotation

Description

The Xml3dVec3f data type defines an arbitrary rotation in 3D. The rotation is represented by an axis and an angle. The right-hand rule applies.

IDL Definition

```
interface XML3DRotation :
{
  // Attributes
  readonly attribute XML3DVec3 axis;
```

```
  attribute float angle;
  // Methods
  void setRotation(in XML3DVec3 from, in XML3DVec3 to) raises(DOMException);
  void setAxisAngle(in XML3DVec3 axis, in float angle) raises(DOMException);
  void setQuaternion(in XML3DVec3 vector, in float scalar) raises(DOMException);
  void setAxisAngleValue(in string str) raises(DOMException);
  XML3DMatrix toMatrix();
  XML3DVec3 rotateVec3(in XML3DVec3 inputVector);
  XML3DRotation interpolate(in XML3DRotation rot1, in float t);
  XML3DRotation multiply(in XML3DRotation rot1);
  XML3DRotation normalize();
  Float32Array getQuaternion();
  XML3DRotation set(in XML3DRotation second);
}
```

Attributes

axis
angle

Methods

setRotation
: Replaces the existing rotation with one computed from the two vectors passed as arguments.

setAxisAngle
: Replaces the existing rotation with the axis-angle representation passed as argument

setQuaternion
: Replaces the existing rotation with the quaternion representation passed as argument

setAxisAngleValue
: Replaces the existing matrix with one computed from parsing the passed string.

toMatrix
: Returns a XML3DMatrix that descibes this 3D rotation in a 4x4 matrix representation.

rotateVec3
: Rotates the vector passed as parameter with this rotation representation. The result is returned as new vector instance. Neither this nor the inputVector are changed.

interpolate
: Linear interpolation of this rotation rot0 with the passed rotation rot1 with factor t. The result is (1-t)rot0 + t rot1. Typically realized with a spherical linear interpolation based on quaternions.

multiply
: Multiplies this rotation with the passed rotation. This rotation is not changed.

normalize
: Returns the normalized version of this rotation. Result is a newly created vector. This is not modified.

getQuaternion

set
: Sets the values of this rotation from the second rotation passed as parameter. The method returns this.

up

Interface XML3DVec3

Description

The Xml3dVec3f interface represents a three-dimensional vector as a 3-tuple of single-precision floating point values.

IDL Definition

```
interface XML3DVec3 :
{
  // Attributes
  attribute float x;
  attribute float y;
  attribute float z;
  // Methods
  void setVec3Value(in string str) raises(DOMException);
  XML3DVec3 add(in XML3DVec3 secondVec);
  XML3DVec3 subtract(in XML3DVec3 secondVec);
  XML3DVec3 multiply(in XML3DVec3 secondVec);
```

```
  XML3DVec3 scale(in float factor);
  XML3DVec3 cross(in XML3DVec3 secondVec);
  float dot(in XML3DVec3 secondVec);
  XML3DVec3 negate();
  float length();
  XML3DVec3 normalize() raises(DOMException);
  XML3DVec3 set(in XML3DVec3 secondVec);
}
```

Attributes

x
: The x component of the vector

y
: The y component of the vector

z
: The z component of the vector

Methods

setVec3Value
: The setVec3Value method replaces the existing matrix with one computed from parsing the passed string.

add
: Returns the component-wise addition of this vector with a second vector passed as parameter. Result is a newly created vector. This is not modified.

subtract
: Returns the component-wise subtraction of this vector with a second vector passed as parameter. Result is a newly created vector. This is not modified.

multiply
: Returns the component-wise multiplication of this vector with a second vector passed as parameter. Result is a newly created vector. This is not modified.

scale
: Returns the component-wise multiplication of this vector with a factor passed as parameter. Result is a newly created vector. This is not modified.

cross
: Returns the cross product of this vector with a second vector passed as parameter. Result is a newly created vector. This is not modified.

dot
: Returns the dot product of this vector with a second vector passed as parameter. This is not modified.

negate
: Returns the component wise multiplication by -1 of this vector. Result is a newly created vector. This is not modified.

length
: Returns the length of this vector.

normalize
: Returns the normalized version of this vector. Result is a newly created vector. This is not modified.

set
: Sets the values of this vector from the second vector passed as parameter. The method returns this.

up
