COLLADA 1.4 Quick Reference Card - Page 1

COLLADA™ defines an XML-based schema to allow transport of 3D assets between applications, enabling diverse 3D authoring and content processing tools to be combined into a production pipeline.

All elements on this card apply to the COMMON profile unless otherwise noted.

- [n] refers to chapters in COLLADA 1.4 Specification: www.khronos.org/collada
- Attributes are green. *Optional Attributes* are italic.
- Elements are blue. [Placeholder elements] are in brackets.
- ⊞ element expanded elsewhere on card.
- ⊡ element expanded in specification.
- ⋯ indicates sequence.
- ⊟ indicates choice.
- xs:* types are defined in the XML Schema language specification.
- The default cardinality is 1.
- <any> may contain any well-formed XML data.
- Type TargetableFloat is a floating point value that has a sid attribute.
- Type TargetableFloat3 is a floating point vector value that has an sid attribute.
  - Color model is RGB for float3, and RGBA for float4 values.
  - Spatial coordinates are Cartesian for float (X), float2 (XY), and float3 (XYZ) values.
  - Texture coordinates are Cartesian for float (S), float2 (ST), and float3 (STP) values; and homogenous for float4 (STPQ) values.

Metadata Elements [5]

Declares the root of the document that contains some of the content in the COLLADA schema.

COLLADA

	version ‡		
⋯	asset		⊞
⋯ ⊟	library_animations	[0..*]	⊞
	library_animation_clips	[0..*]	⊡
	library_cameras	[0..*]	⊞
	library_controllers	[0..*]	⊞
	library_effects	[0..*]	⊞
	library_force_fields	[0..*]	⊡
	library_geometries	[0..*]	⊞
	library_images	[0..*]	⊞
	library_lights	[0..*]	⊞
	library_materials	[0..*]	⊞
	library_nodes	[0..*]	⊞
	library_physics_materials	[0..*]	⊡
	library_physics_models	[0..*]	⊡
	library_physics_scenes	[0..*]	⊡
	library_visual_scenes	[0..*]	⊡
	scene	[0..1]	⊞
	extra	[0..*]	⊞

Parent: none

‡ *version*: 1.4.0, 1.4.1

Defines asset-management information.

asset

⋯	contributor	[0..*]	
⋯	— author	[0..1]	xs:string
	— authoring_tool	[0..1]	xs:string
	— comments	[0..1]	xs:string
	— copyright	[0..1]	xs:string
	— source_data	[0..1]	xs:anyURI
	created		dateTime
	keywords	[0..1]	xs:string
	modified		dateTime
	revision	[0..1]	xs:string
	subject	[0..1]	xs:string
	title	[0..1]	xs:string
	unit	[0..1]	⊞
	up_axis ‡	[0..1]	⊡

Parents: camera, COLLADA, light, material, source, geometry, image, animation, animation_clip, controller, extra, node, visual_scene, library_*, effect, force_field, physics_{material, scene, model}, profile_*, profile_{CG, COMMON, GLES}/technique (FX)

‡ *up_axis*: X_UP, Y_UP, Z_UP. Default = Y_UP

The parent of all library_* elements is COLLADA

Declares a module of <animation> elements.

library_animations

	id		xs:ID
	name		xs:NCName
⋯	asset	[0..1]	⊞
	animation	[1..*]	⊞
	extra	[0..*]	⊞

Declares a module of <camera> elements.

library_cameras

	id		xs:ID
	name		xs:NCName
⋯	asset	[0..1]	⊞
	camera	[1..*]	⊞
	extra	[0..*]	⊞

Declares a module of <controller> elements.

library_controllers

	id		xs:ID
	name		xs:NCName
⋯	asset	[0..1]	⊞
	controller	[1..*]	⊞
	extra	[0..*]	⊞

Declares a module of <effect> elements.

library_effects

	id		xs:ID
	name		xs:NCName
⋯	asset	[0..1]	⊞
	effect	[1..*]	⊞
	extra	[0..*]	⊞

Declares a module of <geometry> elements.

library_geometries

	id		xs:ID
	name		xs:NCName
⋯	asset	[0..1]	⊞
	geometry	[1..*]	⊞
	extra	[0..*]	⊞

Declares a module of <image> elements.

library_images

	id		xs:ID
	name		xs:NCName
⋯	asset	[0..1]	⊞
	image	[1..*]	⊞
	extra	[0..*]	⊞

Declares a module of <light> elements.

library_lights

	id		xs:ID
	name		xs:NCName
⋯	asset	[0..1]	⊞
	light	[1..*]	⊞
	extra	[0..*]	⊞

Declares a module of <material> elements.

library_materials

	id		xs:ID
	name		xs:NCName
⋯	asset	[0..1]	⊞
	material	[1..*]	⊞
	extra	[0..*]	⊞

Declares a module of <node> elements.

library_nodes

	id		xs:ID
	name		xs:NCName
⋯	asset	[0..1]	⊞
	node	[1..*]	⊞
	extra	[0..*]	⊞

Defines unit of distance for COLLADA elements and objects.

unit

	meter	float
	name	xs:NMTOKEN

Parent: asset

Instantiates a COLLADA resource.

instance_animation, instance_{camera, light, node}, instance_{visual, physics}_scene, instance_physics_material, instance_force_field — InstanceWithExtra

	url		xs:anyURI
	sid		xs:NCName
	name		xs:NCName
⋯	extra	[0..*]	⊞

Parents:
instance_animation: animation_clip;
instance_{camera, light, node}: node;
instance_{visual, physics}_scene: scene;
instance_physics_material: {instance}_rigid_body, shape;
instance_force_field: physics_scene, instance_physics_model

Instantiates <rigid_body> within an <instance_physics_model>.

instance_rigid_body

	body		xs:NCName
	sid		xs:NCName
	target		xs:anyURI
⋯	technique_common		
⋯	— angular_velocity ‡	[0..1]	float3
	— velocity ‡	[0..1]	float3
	— dynamic	[0..1]	⊡
	— mass	[0..1]	TargetableFloat
	— mass_frame	[0..1]	⊞
	— inertia	[0..1]	TargetableFloat3
⊟ [0..1]	— — instance_physics_material		⊞InstanceWithExtra
	— — physics_material		⊞
	— shape	[0..*]	⊞
	technique (core)	[0..*]	⊞
	extra	[0..*]	⊞

Parent: instance_physics_model

‡ *angular_velocity, velocity*: Default = 0 0 0

Scene Elements [5]

Describes the entire set of information that can be visualized from the contents of a COLLADA resource.

scene

⋯	instance_physics_scene	[0..*]	⊞ InstanceWithExtra
	instance_visual_scene	[0..1]	⊞ InstanceWithExtra
	extra	[0..*]	⊞

Parent: COLLADA

Declares an environment in which physical objects are instantiated and simulated.

physics_scene

	id		xs:ID
	name		xs:NCName
⋯	asset	[0..1]	⊞
	instance_force_field	[0..*]	⊞ InstanceWithExtra
	instance_physics_model	[0..*]	⊞
	technique_common		
⋯	— gravity	[0..1]	TargetableFloat3
	— time_step	[0..1]	TargetableFloat
	technique (core)	[0..*]	⊞
	extra	[0..*]	⊞

Parent: library_physics_scenes

Describes the entire set of information that can be visualized from the contents of a COLLADA resource.

visual_scene

	id		xs:ID
	name		xs:NCName
⋯	asset	[0..1]	⊞
	node	[1..*]	⊞
	evaluate_scene	[0..*]	⊞
	extra	[0..*]	⊞

Parent: library_visual_scenes

Allows the instantiation of a physics model within another physics model, or in a physics scene.

instance_physics_model

	url		xs:anyURI
	sid		xs:NCName
	parent		xs:anyURI
⋯	instance_force_field	[0..*]	⊞ InstanceWithExtra
	instance_rigid_body	[0..*]	⊞
	instance_rigid_constraint	[0..*]	⊡
	extra	[0..*]	⊞

Parents: physics_scene, physics_model

Scene Elements Continued >

©2010 Khronos Group - Rev. 1110

Scene Elements (continued)

Declares instantiation of a COLLADA <geometry> resource.

instance_geometry

bind_material	[0..1]	⊞
extra	[0..*]	⊞

Parents: node, shape

Binds a specific material to a piece of geometry, binding varying and uniform parameters at the same time.

bind_material

param (core)	[0..*]	
name		xs:NCName
sid		xs:NCName
semantic		xs:NMTOKEN
type		xs:NMTOKEN
technique_common		
instance_material	[1..*]	
symbol		xs:NCName
target		xs:anyURI
sid		xs:NCName
name		xs:NCName
bind (material)	[0..*]	⊡
bind_vertex_input	[0..*]	⊡
extra	[0..*]	⊞
technique (core)	[0..*]	⊞
extra	[0..*]	⊞

Parents: instance_geometry, instance_controller

Declares instantiation of a COLLADA <controller> resource.

instance_controller

skeleton	[0..*]	xs:anyURI
bind_material	[0..1]	⊞
extra	[0..*]	⊞

Parent: node

Describes an alternative way to evaluate a <visual_scene>.

evaluate_scene

name		xs:NCName
render	[1..*]	
camera_node		xs:anyURI
layer	[0..*]	xs:NCName
instance_effect	[0..1]	⊞

Parent: visual_scene

Describes hierarchical relationship of elements in a scene.

node

id		xs:ID
name		xs:NCName
sid		xs:NCName
type ‡		NodeType
layer		ListOfNames
asset	[0..1]	⊞
lookat	[0..*]	⊞
matrix	[0..*]	⊞
rotate	[0..*]	⊞
scale	[0..*]	⊞
skew	[0..*]	⊡
translate	[0..*]	⊞
instance_camera	[0..*]	⊞ InstanceWithExtra
instance_controller	[0..*]	⊞
instance_geometry	[0..*]	⊞
instance_light	[0..*]	⊞ InstanceWithExtra
instance_node	[0..*]	⊞ InstanceWithExtra
node	[0..*]	⊞
extra	[0..*]	⊞

Parents: library_nodes, node, visual_scene
‡ *type*: JOINT, NODE. Default = NODE

Animation Elements [5]

Declares interpolation sampling function for an animation.

sampler

id		xs:ID
input (unshared)	[1..*]	InputLocal
semantic ‡		xs:NMTOKEN
source		xs:URIFragmentType

Parent: animation
‡ *semantic*: see note for input (shared) on page 3

Declares an output channel of an animation.

channel

source		xs:URIFragmentType
target		xs:token

Parent: animation

Geometry Elements [5]

Describes visual shape and appearance of object in scene.

geometry

id		xs:ID
name		xs:NCName
asset	[0..1]	⊞
convex_mesh		⊞
mesh		⊞
spline		⊡
extra	[0..*]	⊞

Parent: library_geometries

Describes basic geometric meshes using vertex and primitive information.

mesh

source (core)	[1..*]	⊞
vertices		⊞
lines	[0..*]	⊞
linestrips	[0..*]	⊞
polygons	[0..*]	⊞
polylist	[0..*]	⊞
triangles	[0..*]	⊞
trifans	[0..*]	⊞
tristrips	[0..*]	⊞
extra	[0..*]	⊞

Parent: geometry

Declares the attributes and identity of mesh vertices.

vertices

id		xs:ID
name		xs:NCName
input (unshared)	[1..*]	InputLocal
semantic ‡		xs:NMTOKEN
source		xs:URIFragmentType
extra	[0..*]	⊞

Parents: mesh, convex_mesh
‡ *semantic*: see note for input (shared) on page 3

Declares the binding of geometric primitives and vertex attributes for a mesh element to produce individual triangles.

triangles

name		xs:NCName
count		uint
material		xs:NCName
input (shared)	[0..*]	⊞ InputLocalOffset
p	[0..1]	ListOfUInts
extra	[0..*]	⊞

Parents: mesh, convex_mesh

Declares animation information.

animation

id		xs:ID
name		xs:NCName
asset	[0..1]	⊞
source (core)	[1..*]	⊞
sampler	[1..*]	
id		xs:ID
input	[1..*]	⊡ InputLocal
channel	[1..*]	⊞
animation	[0..*]	⊞
animation	[1..*]	⊞
sampler	[1..*]	⊞
channel	[1..*]	⊞
animation	[0..*]	⊞
animation	[1..*]	⊞
extra	[0..*]	⊞

Parent: library_animation, animation

Describes a section of the animation curves to be used together as an animation clip.

animation_clip

id		xs:ID
name		xs:NCName
start ‡		xs:double
end		xs:double
asset	[0..1]	⊞
instance_animation	[1..*]	⊞ InstanceWithExtra
url		xs:anyURI
extra	[0..*]	⊞
extra	[0..*]	⊞

Parent: library_animation_clips ‡ *start*: Default = 0.0

Declares the binding of geometric primitives and vertex attributes for a mesh element to produce polylists.

polylist

name		xs:NCName
count		uint
material		xs:NCName
input (shared)	[0..*]	⊞ InputLocalOffset
vcount	[0..1]	ListOfUInts
p	[0..1]	ListOfUInts
extra	[0..*]	⊞

Parent: mesh

Declares the binding of geometric primitives and vertex attributes for a mesh element to produce polygons.

polygons

name		xs:NCName
count		uint
material		xs:NCName
input (shared)	[0..*]	⊞
p	[0..*]	ListOfUInts
ph	[0..*]	
p		ListOfUInts
h	[1..*]	ListOfUInts
extra	[0..*]	⊞

Parents: mesh, convex_mesh

Declares the binding of geometric primitives and vertex attributes for a mesh element to produce lines.

lines

name		xs:NCName
count		uint
material		xs:NCName
input (shared)	[0..*]	⊞ InputLocalOffset
p	[0..1]	ListOfUInts
extra	[0..*]	⊞

Parents: mesh, convex_mesh

Declares the binding of geometric primitives and vertex attributes for a mesh element to produce connected triangles.

trifans, tristrips

name		xs:NCName
count		uint
material		xs:NCName
input (shared)	[0..*]	⊞ InputLocalOffset
p	[0..*]	ListOfUInts
extra	[0..*]	⊞

Parents: mesh, convex_mesh

Declares the binding of geometric primitives and vertex attributes for a mesh element to produce linestrips.

linestrips

name		xs:NCName
count		uint
material		xs:NCName
input (shared)	[0..*]	⊞ InputLocalOffset
p	[0..*]	ListOfUInts
extra	[0..*]	⊞

Parents: mesh, convex_mesh

Using <p> to represent assembly of mesh primitive

The first index in a <p> element refers to all inputs with an offset attribute value of 0. The second index refers to all inputs with an offset of 1. There is an index value for each unique input offset attribute value. Each vertex of the primitive is assembled using the value(s) read from indexed inputs. After each input is sampled, producing a primitive vertex, the next index in the <p> element again refers to the inputs with offset of 0.

Controller Elements [5]

Declares generic control information.

controller

id		xs:ID
name		xs:NCName
asset	[0..1]	⊞
skin		⊞
morph		⊞
extra	[0..*]	⊞

Parent: library_controllers

Controller Elements Continued >

Controller Elements (continued)

Describes the data required to blend between sets of static meshes.

morph

method ‡		MorphMethodType
source		xs:anyURI
source (core)	[2..*] ⊞	
targets		
input (unshared)	[2..*]	InputLocal
semantic ‡		xs:NMTOKEN
source		xs:URIFragmentType
extra	[0..*] ⊞	
extra	[0..*] ⊞	

Parent: controller

‡ *method*: NORMALIZED, RELATIVE. Default = NORMALIZED
semantic: see note for input (shared)

Declares vertex and primitive information sufficient to describe blend-weight skinning.

skin

source		xs:anyURI
bind_shape_matrix	[0..1]	float4x4
source (core)	[3..*] ⊞	
joints		
input (unshared)	[2..*]	InputLocal
semantic ‡		xs:NMTOKEN
source		xs:URIFragmentType
extra	[0..*] ⊞	
vertex_weights (shared)		
count		uint
input (shared)	[2..*]	⊞ InputLocalOffset
vcount	[0..1]	ListOfUInts
v	[0..1]	ListOfInts
extra	[0..*] ⊞	
extra	[0..*] ⊞	

Parent: controller
‡ *semantic*: see note for input (shared)

Transform Elements [5]

Declare local coordinate system transformations.

<rotate> specifies an axis (XYZ) and rotation (Euler angle) about it as a float4.

<translate> specifies a translation (XYZ) as a float3.

rotate, translate

sid	xs:NCName

Parents: node, instance_rigid_body, {ref_}attachment, shape, technique_common/mass_frame in rigid_body

<scale> specifies a change in proportions (XYZ) of the axes as a float3.

<lookat> describes a position/orientation transformation as a float3x3, organized as three vectors in order: eye position, interest point, up-axis direction.

<matrix> describes a homogeneous transformation as a float4x4, organized in column-major order.

scale, lookat, matrix

sid	xs:NCName

Parent: node

Camera Elements [5]

Declares a view into scene hierarchy or graph. Contains elements that describe the camera's optics and imager.

camera

id		xs:ID
name		xs:NCName
asset	[0..1] ⊞	
optics	⊞	
imager	[0..1]	
technique (core)	[1..*]	
profile		xs:NMTOKEN
any	[0..*]	xs:any
extra	[0..*] ⊞	
extra	[0..*] ⊞	

Parent: library_cameras

Describes the apparatus on a camera that projects the image onto the image sensor.

optics

technique_common		
orthographic	⊞	
perspective	⊞	
technique (core)	[0..*] ⊞	
profile		xs:NMTOKEN
any	[0..*]	xs:any
extra	[0..*] ⊞	

Parent: camera

Describes the field of view of an orthographic camera.

orthographic

xmag		TargetableFloat
ymag	[0..1]	TargetableFloat
aspect_ratio		TargetableFloat
ymag		TargetableFloat
aspect_ratio	[0..1]	TargetableFloat
znear		TargetableFloat
zfar		TargetableFloat

Parents: optics / technique_common

Describes the field of view of a perspective camera. <xfov> and <yfov> values are in Euler degrees.

perspective

xfov		TargetableFloat
yfov	[0..1]	TargetableFloat
aspect_ratio		TargetableFloat
yfov		TargetableFloat
aspect_ratio	[0..1]	TargetableFloat
znear		TargetableFloat
zfar		TargetableFloat

Parents: optics / technique_common

Data Flow Elements [5]

Declares a data repository that provides values according to the semantics of an <input> element that refers to it.

source (core)

id		xs:ID
name		xs:NCName
asset	[0..1] ⊞	
IDREF_array	[0..1] ⊞	
Name_array	[0..1] ⊞	
bool_array	[0..1] ⊞	
float_array	[0..1] ⊞	
int_array	[0..1] ⊞	
technique_common	[0..1]	
accessor	⊞	
technique (core)	[0..*] ⊞	

Parents: morph, animation, mesh, convex_mesh, skin, spline

Declares an access pattern to one of the array elements: <float_array>, <int_array>, <Name_array>, <bool_array>, and <IDREF_array>.

accessor

count		uint
offset ‡		uint
source		xs:anyURI
stride ‡		uint
param (core)	[0..*]	
name		xs:NCName
sid		xs:NCName
semantic		xs:NMTOKEN
type		xs:NMTOKEN

Parent: source/technique_common
‡ Defaults: *offset* = 0, *stride* = 1

Declares storage for a homogenous array. <bool_array> uses type ListOfBools, an xs:list of type xs:boolean. <Name_array> uses type ListOfNames, an xs:list of type xs:Name.

bool_array, Name_array

id	xs:ID
name	xs:NCName
count	uint

Parent: source (core)

Declares the storage for a homogenous array of ID reference values of type xs:IDREFS.

IDREF_array

id	xs:ID
name	xs:NCName
count	uint

Parent: source (core)

Declares the storage for a homogenous array of type ListOfInts, which is an xs:list of type xs:long.

int_array

id	xs:ID
name	xs:NCName
count	uint
minInclusive ‡	xs:integer
maxInclusive ‡	xs:integer

Parent: source (core)
‡ Defaults: *minInclusive* = -2147483648, *maxInclusive* = 2147483647

Declares the storage for a homogenous array of type ListOfFloats, which is an xs:list of type xs:double.

float_array

id	xs:ID
name	xs:NCName
count	uint
digits ‡	xs:short
magnitude ‡	xs:short

Parent: source (core)
‡ Defaults: *digits* = 6, *magnitude* = 38

Declares the input semantics of a data source and connects a consumer to that source.

input (shared) InputLocalOffset

offset	uint
semantic ‡	xs:NMTOKEN
source	xs:URIFragmentType
set	uint

Parents: lines, linestrips, polygons, polylist, triangles, trifans, tristrips, vertex_weights

‡ *semantic*: The common semantic attribute values are: {TEX}BINORMAL, CONTINUITY, IMAGE, INPUT, WEIGHT, INTERPOLATION, INV_BIND_MATRIX, UV, VERTEX, JOINT, LINEAR_STEPS, NORMAL, OUTPUT, TEXCOORD, POSITION, MORPH_{TARGET, WEIGHT}, {TEX}TANGENT, {IN, OUT}_TANGENT

Extensibility Element [5]

Declares information used to describe some portion of the content. Each technique applies to an associated profile.

technique (core)

profile		xs:NMTOKEN
any	[0..*]	xs:any

Parents: extra, source (core), light, optics, imager, force_field, physics_material, physics_scene, rigid_body, rigid_constraint, instance_rigid_body, bind_material

Describes information about/related to its parent element.

extra

id		xs:ID
name		xs:NCName
type		xs:NMTOKEN
asset	[0..1] ⊞	
technique (core)	[1..*]	
profile		xs:NMTOKEN
any	[0..*]	xs:any

Parents: animation, animation_clip, attachment, box, camera, bind_material, capsule, COLLADA, controller, cylinder, control_vertices, convex_mesh, effect, force_field, format_hint, geometry, image, imager, instance_*, joints, library_*, light, lines, linestrips, material, mesh, morph, node, optics, pass, plane, physics_material, physics_model, physics_scene, polygons, polylist, profile_CG, profile_COMMON, profile_GLES, profile_GLSL, ref_attachment, rigid_body, rigid_constraint, sampler_*, scene, shape, skin, sphere, spline, surface, targets, tapered_capsule, tapered_cylinder, triangles, trifans, tristrips, texture_pipeline, texture_unit, vertex_weights, vertices, visual_scene, and technique (FX) (in profile_CG, profile_COMMON, profile_GLES, and profile_GLSL)

Lighting Elements [5]

Declares a light source that illuminates a scene.

light

id		xs:ID
name		xs:NCName
asset	[0..1] ⊞	
technique_common		
ambient (core)	⊞	
directional	⊞	
point	⊞	
spot	⊞	
technique (core)	[0..*] ⊞	
extra	[0..*] ⊞	

Parent: library_lights

Lighting Elements Continued >

©2010 Khronos Group - Rev. 1110

Lighting Elements (continued)

Describes an ambient light source.

ambient (core), directional

Element	Occurrence	Type
color		TargetableFloat3
sid		xs:NCName

Parent: light/technique_common

Describes a spot light source.

spot

Element	Occurrence	Type
color		TargetableFloat3
constant_attenuation	[0..1]	TargetableFloat
linear_attenuation	[0..1]	TargetableFloat
quadratic_attenuation	[0..1]	TargetableFloat
falloff_angle	[0..1]	TargetableFloat
falloff_exponent	[0..1]	TargetableFloat

Parent: light/technique_common

Describes a point light source.

point

Element	Occurrence	Type
color		TargetableFloat3
constant_attenuation ‡	[0..1]	TargetableFloat
linear_attenuation ‡	[0..1]	TargetableFloat
quadratic_attenuation ‡	[0..1]	TargetableFloat

Parent: light/technique_common

‡ Defaults: *constant_attenuation* = 1.0, *linear_attenuation* = 0.0, *quadratic_attenuation* = 0.0

Physics Material Element [6]

Describes the physical properties of an object.

physics_material

Element	Occurrence	Type
id		xs:ID
name		xs:NCName
asset	[0..1]	⊞
technique_common		
dynamic_friction ‡	[0..1]	TargetableFloat
restitution ‡	[0..1]	TargetableFloat
static_friction ‡	[0..1]	TargetableFloat
technique (core)	[0..*]	⊞
extra	[0..*]	⊞

Parents: library_physics_materials, shape, {instance_}rigid_body/technique_common

‡ *{dynamic, static}_friction, restitution*: Default = 0

FX: Rendering Elements (COMMON) [8]

Describes a specularly shaded surface where the specular reflection is shaded according to the Blinn BRDF approximation. In the diagram, * = common.

blinn, phong

Element	Occurrence	Type
emission	[0..1]	⊞ *_color_or_texture_type
ambient (FX)	[0..1]	⊞ *_color_or_texture_type
diffuse	[0..1]	⊞ *_color_or_texture_type
specular	[0..1]	⊞ *_color_or_texture_type
shininess	[0..1]	⊞ *_float_or_param_type
reflective	[0..1]	⊞ *_color_or_texture_type
reflectivity	[0..1]	⊞ *_float_or_param_type
transparent	[0..1]	*_transparent_type
opaque ‡		fx_opaque_enum
transparency	[0..1]	⊞ *_float_or_param_type
index_of_refraction	[0..1]	⊞ *_float_or_param_type

Parents: technique (FX) in profile_COMMON

‡ *opaque*: A_ONE, RGB_ZERO. Default = A_ONE

Describes a constantly shaded surface that is independent of lighting. In the diagram, * = common

constant

Element	Occurrence	Type
emission	[0..1]	⊞ *_color_or_texture_type
reflective	[0..1]	⊞ *_color_or_texture_type
reflectivity	[0..1]	⊞ *_float_or_param_type
transparent	[0..1]	*_transparent_type
opaque ‡		fx_opaque_enum
transparency	[0..1]	⊞ *_float_or_param_type
index_of_refraction	[0..1]	⊞ *_float_or_param_type

Parent: technique (FX) in profile_COMMON

‡ *opaque*: A_ONE, RGB_ZERO. Default = A_ONE

Physics Model Elements [6]

Allows for building complex combinations of rigid bodies and constraints that may be instantiated multiple times.

physics_model

Element	Occurrence	Type
id		xs:ID
name		xs:NCName
asset	[0..1]	⊞
rigid_body	[0..*]	⊞
rigid_constraint	[0..*]	⊞
instance_physics_model	[0..*]	⊞
extra	[0..*]	⊞

Parent: library_physics_models

Describes simulated bodies that do not deform.

rigid_body

Element	Occurrence	Type
sid		xs:NCName
name		xs:NCName
technique_common		
dynamic	[0..1]	⊡
mass	[0..1]	TargetableFloat
mass_frame	[0..1]	⊞
inertia	[0..1]	TargetableFloat3
	[0..1]	
instance_physics_material		⊞InstanceWithExtra
physics_material		⊞
shape	[1..*]	⊞
technique (core)	[0..*]	⊞
extra	[0..*]	⊞

Parent: physics_model

Describes components of a <rigid_body>.

shape

Element	Occurrence	Type
hollow	[0..1]	⊡
mass	[0..1]	TargetableFloat
density	[0..1]	TargetableFloat
	[0..1]	
instance_physics_material		⊞ InstanceWithExtra
physics_material		⊞
instance_geometry		⊞
plane		⊡
box		⊡
sphere		⊡
cylinder		⊡
tapered_cylinder		⊡
capsule		⊡
tapered_capsule		⊡
	[0..*]	
translate		⊞
rotate		⊞
extra	[0..*]	⊞

Parents: {instance_}rigid_body/technique_common

Defines the center and orientation of the rigid body.

mass_frame

Element	Occurrence	Type
	[1..*]	
translate		⊞
rotate		⊞

Parent: rigid_body/technique_common

Contains or refers to information that describes basic geometric meshes.

convex_mesh

Element	Occurrence	Type
convex_hull_of		xs:anyURI
	[0..1]	
source (core)	[1..*]	⊞
vertices	[0..1]	⊞
	[0..*]	
lines	[0..*]	⊞
linestrips	[0..*]	⊞
polygons	[0..*]	⊞
polylist	[0..*]	⊞
triangles	[0..*]	⊞
trifans	[0..*]	⊞
tristrips	[0..*]	⊞
extra	[0..*]	⊞

Parent: geometry

Connects components, such as <rigid_body>, into complex physics models with moveable parts.

rigid_constraint

Element	Occurrence	Type
sid		xs:NCName
name		xs:NCName
ref_attachment		
rigid_body		xs:anyURI
	[0..*]	
translate		⊞
rotate		⊞
extra	[0..*]	⊞
attachment		
rigid_body		xs:anyURI
	[0..*]	
translate		⊞
rotate		⊞
extra	[0..*]	⊞
technique_common		
enabled ‡	[0..1]	⊡
interpenetrate ‡	[0..1]	⊡
limits	[0..1]	⊡
spring	[0..1]	⊡
technique (core)	[0..*]	⊞
extra	[0..*]	⊞

Parent: physics_model

‡ Defaults: *enabled* = True, *interpenetrate* = False

Describes a diffuse shaded surface that is independent of lighting. In the diagram, * = common

lambert

Element	Occurrence	Type
emission	[0..1]	⊞ *_color_or_texture_type
ambient (FX)	[0..1]	⊞ *_color_or_texture_type
diffuse	[0..1]	⊞ *_color_or_texture_type
reflective	[0..1]	⊞ *_color_or_texture_type
reflectivity	[0..1]	⊞ *_float_or_param_type
transparent	[0..1]	*_transparent_type
opaque ‡		fx_opaque_enum
transparency	[0..1]	⊞ *_float_or_param_type
index_of_refraction	[0..1]	⊞ *_float_or_param_type

Parent: technique (FX) in profile_COMMON

‡ *opaque*: A_ONE, RGB_ZERO. Default = A_ONE

Describes scalar attributes of fixed-function shader elements inside <profile_COMMON> effects.

shininess, reflectivity, index_of_refraction, transparency — common_float_or_param_type

Element	Occurrence	Type
float	[0..1]	
sid		xs:NCName
param (FX)	[1..*]	
ref		xs:NCName

Parents: constant, lambert, phong, blinn

Describes color attributes of fixed-function shader elements inside <profile_COMMON> effects.

emission, reflective, ambient (FX), diffuse, specular, transparent — common_color_or_texture_type

Element	Occurrence	Type
color		⊡
param (FX)	[1..*]	
ref		xs:NCName
texture	[0..*]	
texture		xs:NCName
texcoord		xs:NCName
extra	[0..1]	⊞

Parents: constant, lambert, phong, blinn

<newparam> creates a new, named param object in the FX Runtime, and assigns it a type, an initial value, and additional attributes at declaration time.

newparam — common_newparam_type

Element	Occurrence	Type
sid		xs:NCName
semantic	[0..1]	xs:NCName
float		float
float2		float2
float3		float3
float4		float4
surface (FX)		⊞ fx_surface_common
sampler2D		⊞ fx_sampler2D_common

Parents: profile_COMMON/technique (FX)

FX: Texturing Elements (All Profiles) [8]

Declares the storage for the graphical representation of an object.

image		
id		xs:ID
name		xs:NCName
format		xs:token
height		uint
width		uint
depth		uint
asset	[0..1]	⊞
data		ListOfHexBinary
init_from		xs:anyURI
extra	[0..*]	⊞

Profiles: COMMON, CG, GLSL, GLES

Parents: library_images, effect, profile_CG, profile_GLSL, profile_COMMON, profile_GLES; technique (FX) in profile_CG, profile_COMMON, profile_GLES, profile_GLSL

Declares a two-dimensional texture sampler.

sampler2D		fx_sampler2D_common gl_sampler_2d
source		xs:NCName
wrap_s ‡	[0..1]	fx_sampler_wrap_common
wrap_t ‡	[0..1]	fx_sampler_wrap_common
minfilter ‡	[0..1]	fx_sampler_filter_common
magfilter ‡	[0..1]	fx_sampler_filter_common
mipfilter ‡	[0..1]	fx_sampler_filter_common
border_color	[0..1]	⊡ fx_color_common
mipmap_maxlevel ‡	[0..1]	xs:unsignedByte
mipmap_bias ‡	[0..1]	float

Profiles: COMMON, CG, GLSL, External, Effect

Parents: newparam, setparam, usertype, array, shader/bind

‡ *wrap_s, wrap_t*: NONE, WRAP, MIRROR, CLAMP, BORDER. Default = WRAP

minfilter, magfilter, mipfilter: NONE, NEAREST, LINEAR, {NEAREST, LINEAR}_MIPMAP_NEAREST, {NEAREST, LINEAR}_MIPMAP_LINEAR, Default = NONE

Defaults: *mipmap_maxlevel* = 255, *mipmap_bias* = 0

Declares a resource that can be used both as the source for texture samples and as the target of a rendering pass. Child elements differ depending on the profile used. In the diagram, * = common

surface		fx_surface_common
type ‡		fx_surface_type_enum
init_as_null	[0..1]	xs:anyType
init_as_target		xs:anyType
init_cube		⊡ fx_surface_init_cube_*
init_volume		⊡ fx_surface_init_volume_*
init_planar		⊡ fx_surface_init_planar_*
init_from	[1..*]	fx_surface_init_from_*
mip ‡		xs:unsignedInt
slice ‡		xs:unsignedInt
face ‡		fx_surface_face_enum
format	[0..1]	xs:token
format_hint	[0..1]	⊡ fx_surface_format_hint_*
size ‡	[0..1]	int3
viewport_ratio ‡		float2
mip_levels ‡	[0..1]	xs:unsignedInt
mipmap_generate ‡	[0..1]	xs:boolean
extra	[0..*]	⊞

Profiles: COMMON, CG, GLES, GLSL, External, Effect

Parents: COMMON - newparam, setparam;
CG - newparam, setparam, array, shader/bind, usertype;
GLES - newparam, setparam, texture_unit;
GLSL - newparam, setparam, array, shader/bind

‡ *type*: UNTYPED, 1D, 2D, 3D, RECT, CUBE, DEPTH
init_from/face: POSITIVE_{X,Y,Z}, NEGATIVE_{X,Y,Z}. Default = POSITIVE_X

Defaults: *size* = 0 0 0, *viewport_ratio* = 1 1, *mip_levels* = 0, *mipmap_generate* = False, *init_from/mip, init_from/slice* = 0

FX: Shader Elements (Other Profiles) [8]

Declares and prepares a shader for execution in the rendering pipeline of a <pass> element.

shader		
stage ‡		{cg, glsl}_pipeline_stage
annotate	[0..*]	⊡ fx_annotate_common
compiler_target	[0..1]	⊡
compiler_options	[0..1]	xs:string
name		
source		xs:NCName
bind (shader)	[0..*]	⊡

Profiles: CG, GLSL

Parent: profile_{CG,GLSL}/technique/pass,

‡ *stage:* CG: VERTEX, FRAGMENT
GLSL: VERTEXPROGRAM, FRAGMENTPROGRAM

Declares all the render states, shaders, and settings for one rendering pipeline.

pass		
sid		xs:NCName
annotate	[0..*]	⊡ fx_annotate_common
color_target	[0..1]	⊡ fx_colortarget_common
depth_target	[0..1]	⊡ fx_depthtarget_common
stencil_target	[0..1]	⊡ fx_stenciltarget_common
color_clear	[0..1]	⊡ fx_clearcolor_common
depth_clear	[0..1]	⊡ fx_cleardepth_common
stencil_clear	[0..1]	⊡ fx_clearstencil_common
draw	[0..1]	⊡ fx_draw_common
[render_states] †	[1..*]	⊡
shader		⊞
extra	[0..*]	⊞

Profiles: CG, GLES, GLSL

Parents: profile_CG/technique (FX) and profile_GLSL/technique (FX). <pass> is also a child of profile_GLES/technique (FX), in which case it excludes the child element <shader>.

† *[render_states]*: Refer to the Render States subsection in the description of <pass> in the specification. The schema indicates use of group gl_pipeline_settings for profiles GLSL or CG, and gles_pipeline_settings for GLES.

<newparam> creates a new, named param object in the FX Runtime, and assigns it a type, an initial value, and additional attributes at declaration time.

newparam		fx_newparam_common gles_newparam
sid		xs:NCName
annotate	[0..*]	⊡ fx_annotate_common
semantic	[0..1]	xs:NCName
modifier ‡	[0..1]	fx_modifier_enum_common
[values] †		⊡ fx_basic_type_common

Profile: Effect, GLES

Parent: For fx_newparam_common: effect ;
For gles_newparam: profile_GLES, profile_GLES/technique (FX)

‡ *modifier*: CONST, UNIFORM, VARYING, STATIC, VOLATILE, EXTERN, SHARED

† *[values]*: Includes elements from the following list, where *n* is 1, 2, 3, or 4: bool, bool*n*, int, int*n*, float, float*n*, float*n*x*m*, surface (FX), and enum.

For **fx_newparam_common** the list includes sampler{1D, 2D, 3D, CUBE, RECT, DEPTH}.

For **gles_newparam** the list includes sampler_state and texture_{pipeline, unit}.

newparam		{glsl, cg}_newparam
sid		{glsl, cg}_identifier
annotate	[0..*]	⊡ fx_annotate_common
semantic	[0..1]	xs:NCName
modifier ‡	[0..1]	fx_modifier_enum_common
[values] †		⊡ {glsl, cg}_param_type
array		⊡ {glsl, cg}_newarray_type
usertype		⊡ cg_setuser_type

Profile: CG, GLSL

Parents: profile_{GLSL, CG}, profile_{GLSL, CG}/technique (FX)

Child <usertype> excluded from glsl_newparam

‡ *modifier*: CONST, UNIFORM, VARYING, STATIC, VOLATILE, EXTERN, SHARED

† *[values]:* Includes elements from the following list, where *n* is 1, 2, 3, or 4: bool, bool*n*, int, int*n*, float, float*n*, string, sampler{1D, 2D, 3D, CUBE, RECT, DEPTH}, and enum.

For **glsl_newparam** the list includes float2x2, float3x3, float4x4, and surface (GLSL), .

For **cg_newparam** the list includes bool*n*x*m*, int*n*x*m*, half, half*n*, half*n*x*m*, fixed, fixed*n*, fixed*n*x*m*, float*n*x*m*, and surface.

FX: Texturing Elements (Other Profiles) [8]

Declares a two-dimensional texture sampler state for element <profile_GLES>.

sampler_state		gles_sampler_state
sid		xs:NCName
wrap_s ‡	[0..1]	gles_sampler_wrap
wrap_t ‡	[0..1]	gles_sampler_wrap
minfilter ‡	[0..1]	fx_sampler_filter_common
magfilter ‡	[0..1]	fx_sampler_filter_common
mipfilter ‡	[0..1]	fx_sampler_filter_common
mipmap_maxlevel ‡	[0..1]	xs:unsignedByte
mipmap_bias ‡	[0..1]	float
extra	[0..*]	⊞

Profile: GLES

Parents: newparam, setparam

‡ *wrap_s, wrap_t*: REPEAT, CLAMP, CLAMP_TO_EDGE, MIRRORED_REPEAT. Default = REPEAT

minfilter, magfilter, mipfilter: NONE, NEAREST, LINEAR, {NEAREST, LINEAR}_MIPMAP_NEAREST, {NEAREST, LINEAR}_MIPMAP_LINEAR, Default = NONE

Default: *mipmap_maxlevel* = 255, *mipmap_bias* = 0

Defines a texture unit that will be mapped to hardware texture units based on its usage in <texture_pipeline> commands.

texture_unit		gles_texture_unit
sid		xs:NCName
surface	[0..1]	xs:NCName
sampler_state	[0..1]	xs:NCName
texcoord	[0..1]	
semantic		xs:NCName

Profile: GLES

Parents: setparam, newparam

Defines a <texture_pipeline> command for combiner-mode texturing. In the diagram, * = gles_texcombiner.

texcombiner		gles_texcombiner_command_type
constant	[0..1]	⊡ gles_texture_constant_type
RGB ‡	[0..1]	*_commandRGB_type
operator ‡		*_operatorRGB_enums
scale		float
argument	[1..3]	⊡ *_argumentRGB_type
alpha ‡	[0..1]	*_commandAlpha_type
operator ‡		*_operatorAlpha_enums
scale		float
argument	[1..3]	*_argumentAlpha_type
source ‡		*_source_enums
operand ‡		*_operandAlpha_enums
unit		⊞ xs:NCName

Profile: GLES

Parents: newparam/texture_pipeline, setparam/texture_pipeline, pass/texture_pipeline/value

‡ *RGB, RGB/operator*: REPLACE, MODULATE, ADD, ADD_SIGNED, INTERPOLATE, SUBTRACT, DOT3_{RGB, RGBA}

alpha, alpha/operator: REPLACE, MODULATE, ADD, ADD_SIGNED, INTERPOLATE, SUBTRACT

alpha/argument/source: TEXTURE, CONSTANT, PRIMARY, PREVIOUS.

alpha/argument/operand: ONE_MINUS_SRC_ALPHA, SRC_ALPHA. Default = SRC_ALPHA

Texturing Elements (Other Profiles) Continued >

Texturing Elements (Other Profiles) (cont'd)

Declares a resource that can be used both as the source for texture samples and as the target of a rendering pass. This element inherits the elements from <surface> (FX) and adds the following:

	surface		cg_surface_type glsl_surface_type
	type ‡		fx_surface_type_enum
	generator	[0..1]	
	annotate	[0..*]	⊡ fx_annotate_common
	code		⊡ fx_code_profile
[1..*]	include		⊡ fx_include_common
	name		⊞
	setparam ‡	[0..*]	

Profile: CG, GLSL, GLES

Parents: COMMON - newparam, setparam;
CG - newparam, setparam, array, shader/bind, usertype;
GLES - newparam, setparam, texture_unit;
GLSL - newparam, setparam, array, shader/bind

‡ *type*: UNTYPED, 1D, 2D, 3D, CUBE, DEPTH, RECT

setparam: for surface (CG), type is cg_setparam_simple, for surface (GLSL), type is glsl_setparam_simple

Defines a set of texturing commands that will be converted into multitexturing operations using glTexEnv in regular and combiner mode.

	texture_pipeline, texture_pipeline/value	gles_texture_pipeline
	sid	xs:NCName
	texcombiner	⊞ gles_texcombiner_command_type
	texenv	⊞ gles_texenv_command_type
[1..*]	extra	⊞

Profile: GLES

Parents: newparam, setparam, pass/render_state

Defines a texture_pipeline command for simple, noncombiner-mode texturing.

texenv		gles_texenv_command_type
operator ‡		gles_texenv_mode_enums
unit		⊞ xs:NCName
constant (combiner)	[0..1]	⊡ gles_texture_constant_type

Profile: GLES

Parents: newparam/texture_pipeline, setparam/texture_pipeline, pass/texture_pipeline/value

‡ *operator*: REPLACE, MODULATE, DECAL, BLEND, ADD

FX: Materials Elements [8]

Describes the visual appearance of a geometric object.

material		
id		xs:ID
name		xs:NCName
asset	[0..1]	⊞
instance_effect		⊞
extra	[0..*]	⊞

Parent: library_material

Instantiates a COLLADA effect.

instance_effect		
url		xs:anyURI
sid		xs:NCName
name		xs:NCName
technique_hint	[0..1]	
platform		xs:NCName
profile		xs:NCName
ref		xs:NCName
setparam	[0..*]	⊡
extra	[0..*]	⊞

Parents: material, render

FX: Effects Elements [8]

Declares a self-contained description of a COLLADA effect.

effect		
id		xs:ID
name		xs:NCName
asset	[0..1]	⊞
annotate	[0..*]	⊡ fx_annotate_common
image	[0..*]	⊞
newparam	[0..*]	⊞ fx_newparam_common
[fx_profile_abstract] ‡	[1..*]	xs:anyType
extra	[0..*]	⊞

Profile: Effect

Parent: library_effects

‡ *[fx_profile_abstract]*: Exactly one of profile_{CG, GLES, GLSL,COMMON}

Opens a block of platform-independent declarations for the common, fixed-function shader. * = common

	profile_COMMON		
	image		⊞
[0..*]	newparam		⊞ *_newparam_type
	technique (FX)		⊞
	id		xs:ID
	sid		xs:NCName
	asset	[0..1]	⊞
	image		⊞
[0..*]	newparam		⊞ *_newparam_type
	constant		⊞
	lambert		⊞
	phong		⊞
	blinn		⊞
	extra	[0..*]	⊞
	extra	[0..*]	⊞

Profile: COMMON

Parent: effect

Declares platform-specific data types and techniques for the GLES language.

	profile_GLES		
	id		xs:ID
	platform		xs:NCName
	asset	[0..1]	⊞
	image		⊞
[0..*]	newparam		⊞ gles_newparam
	technique (FX)	[1..*]	⊞
	extra	[0..*]	⊞

Profile: GLES

Parent: effect

Declares platform-specific data types and techniques for the Cg language.

	profile_CG		
	id		xs:ID
	platform ‡		xs:NCName
	code		fx_code_profile
	sid		xs:NCName
[0..*]	include		fx_include_common
	sid		xs:NCName
	url		xs:anyURI
	image		⊞
[0..*]	newparam		⊞ cg_newparam
	technique (FX)	[1..*]	⊞

Profile: CG

Parent: effect

‡ *platform*: Default = “PC”

Declares platform-specific data types and techniques for the GLSL language.

	profile_GLSL		
	id		xs:ID
	asset	[0..1]	⊞
	code		⊡ fx_code_profile
[0..*]	include		⊡ fx_include_common
	image		⊞
[0..*]	newparam		⊞ glsl_newparam
	technique (FX)	[1..*]	⊞
	extra	[0..*]	⊞

Profile: GLSL

Parent: effect

Declares information to process content. Each technique applies to an associated profile. Child elements differ depending on parent. Refer to parent descriptions for list of children.

	technique (FX)		
	id		xs:ID
	sid		xs:NCName
	annotate	[0..*]	⊡ fx_annotate_common
	code		⊡ fx_code_profile
[0..*]	include		⊡ fx_include_common
	image		⊞
	newparam ‡		⊞ glsl_newparam
[0..*]	setparam ‡		⊡ glsl_setparam
	pass	[1..*]	⊞
	extra	[0..*]	⊞

Profiles: GLSL, CG, GLES

Parents: profile_GLSL, profile_CG, profile_GLES

‡ The type for child elements <newparam> and <setparam> differ depending on parent of <technique> (FX), as follows:

- profile_GLSL/technique (FX): types are glsl_*
- profile_CG/technique (FX): types are cg_*
- profile_GLES/technique (FX): types are gles_*

Visit www.collada.org for more on COLLADA, including a forum, a model bank, directories of extensions and conditioners, and more.

Get your copy of *COLLADA: Sailing the Gulf of 3d Digital Content Creation* from your technical bookstore or www.amazon.com.

Extending COLLADA

COLLADA allows you to extend its data model and add functionality to your documents. These extensions take the form of alternative <technique>, additive <extra>, and scalable <input> elements. For more information and a list of published extensions, see https://collada.org/mediawiki/index.php/Portal:Extensions_directory.

<technique> profiles
Declares alternative techniques to <technique_common> that provide a better description for a specific profile.

<extra> types
Declares new techniques that add descriptions to existing ones. This extra information can represent additional real data or semantic (meta) data to the application.

<input> semantics
Declares new streams that add to data flows.

Example:

```
<extra type="MY_TYPE">
    <technique profile="PROFILE-A">
        < ...
    </technique>
    <technique profile="PROFILE-B">
        < ...
    </technique>
</extra>
```

COLLADA Certification Logos: Baseline, Superior, and Exemplary

Certification logos are earned by a COLLADA product that has passed conformance testing and indicate the degree of feature support you can expect from the software. For more information about conformance, please visit http://www.khronos.org/collada/adopters/

The Khronos Group is an industry consortium creating open standards for the authoring and acceleration of parallel computing, graphics and dynamic media on a wide variety of platforms and devices. See www.khronos.org to learn more about the Khronos Group.

COLLADA is a trademark of Khronos Group Inc.

Reference card production by Miller & Mattson www.millermattson.com