



Creation of Collada files

15.09.2015

Ender Yemenicioglu



What is COLLADA?

- COLLADA (COLLABorative Design Activity) is an XML-based interchange file format for the tools in content pipeline, such as Digital Content Creation(DCC) tools and conditioners
- Coupling of geometry, kinematics (and rigid-body physics) information
- Support of BREP-Geometry, NURBS
- File extension: .dae
- Hierarchical structure of the elements (parent-child relationship), instantiation and addressing concepts



Structure of COLLADA

```

<?xml version="1.0" encoding="utf-8"?>
<COLLADA version="1.5.0" xmlns="http://www.collada.org/2008/03/COLLADASchema">
  <asset>
    <contributor>
      <author>tarakos GmbH</author>
      <author_email>ender.yemenicioglu@tarakos.com</author_email>
      <author_website>www.tarakos.com</author_website>
      <authoring_tool>tarakos Vrm1 To Collada Converter Version 1.1</authoring_tool>
      <copyright>tarakos GmbH</copyright>
    </contributor>
    <created>2015-10-08T13:26:43Z</created>
    <modified>2015-10-08T13:26:44Z</modified>
    <unit meter="1" name="meter" />
    <up_axis>Z_UP</up_axis>
  </asset>
  <library_geometries id="6eefceaa-ab1f-48a6-a540-130e478703fc" name="libGeo World">
  </library_geometries>
  <library_materials id="2ec95307-3a70-4111-b97d-7ef54c0c4110" name="libMaterial World">
  </library_materials>
  <library_effects id="378ebafe-73d5-4708-b05b-f449ba598173" name="libEffect World">
  </library_effects>
  <library_visual_scenes id="c3116fe0-8780-444a-a9e0-f07d459e69c3" name="libVisualScenes World">
  </library_visual_scenes>
  <scene>
  </scene>
</COLLADA>
  
```



The description of library elements in COLLADA format

COLLADA Libraries	Description
library_animations	Contains <animation> elements. <animation> describes the transformation of an object or value over time.
library_animation_clips	Contains <animation_clip> elements. <animation_clip> separates different pieces of a set of animation curves, formulas, or both.
library_cameras	Contains <camera> elements. <camera> describes the eye point of the viewer looking at the visual scene.
library_controllers	Contains <controller> elements. <controller> is a general, generic mechanism for describing active or dynamic content.
library_geometries	Contains <geometry> elements. <geometry> categorizes the declaration of geometric information.
library_effects	Contains <effect> elements. <effect> defines the equations necessary for the visual appearance of geometry and screen-space image processing.
library_force_fields	Contains <force_field> elements. <force_field> describes a force field which affects physical objects, such as rigid bodies.
library_images	Contains <image> elements. <image> describes raster image data, but can conceivably handle other forms of imagery.
library_lights	Contains <light> elements. <light> declares a light source that illuminates a scene.
library_materials	Contains <material> elements. <material> describes the appearance of a geometric object.
library_nodes	Contains <node> elements. <node> embodies the hierarchical relationship of elements by declaring a point of interest in a scene.
library_physics_materials	Contains <physics_material> elements. <physical_material> defines the physical properties of an object.
library_physics_models	Contains <physics_model> elements. <physics_model> provides a logical grouping mechanism for a collection of rigid bodies and constraints.
library_physics_scenes	Contains <physics_scene> elements. <physics_scene> specifies an environment in which physical objects are instantiated and simulated.
library_visual_scenes	Contains <visual_scene> elements. <visual_scene> embodies the entire set of information that can be visualized from the contents.
library_joints	Contains <joint> elements. <joint> defines a single joint with one or more degree of freedom.
library_kinematics_models	Contains <kinematics_model> elements. <kinematics_model> categorizes the declaration of kinematical information.
library_articulated_systems	Contains <articulated_system> elements. <articulated_system> categorizes the declaration of generic control information for kinematics systems.
library_kinematics_scenes	Contains <kinematics_scene> elements. <kinematics_scene> embodies the entire set of information that can be articulated from the kinematic contents.
library_formulas	Contains <formula> elements. <formula> defines a formula using MathML as its common technique.



Instantiation and addressing

```

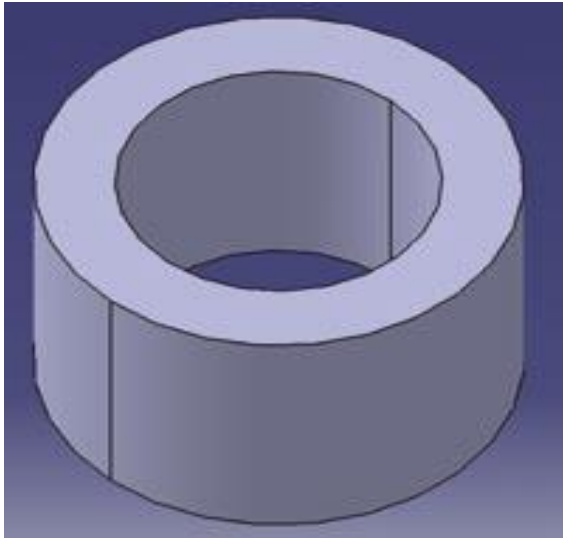
<library_materials id="libmat">
  <material id="material4278190335">
    <instance effect url="#effect4278190335"/>
  </material>
  <material id="material4278255360">
  </material>
  <material id="material4294901760">
  </material>
</library_materials>

<library_effects id="libeffect">
  <effect id="effect4278190335">
    <profile_COMMON>
      <technique sid="common">
        <phong>
          <ambient>
            <color>1 0 0 1</color>
          </ambient>
          <diffuse>
            <color>1 0 0 1</color>
          </diffuse>
        </phong>
      </technique>
    </profile_COMMON>
  </effect>
  <effect id="effect4278255360">
  </effect>
  <effect id="effect4294901760">
  </effect>
</library_effects>
  
```

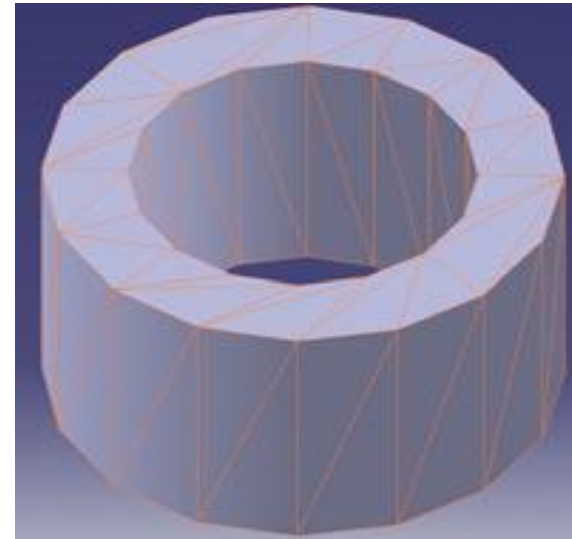
- The actual data representation of an object can be defined only once. However, the object can appear in a scene more than once. Each appearance in the scene is called an instance of the object.
- COLLADA uses two mechanisms to address elements and values:
 - 1) URI addressing: Referencing the id attribute of an element. In “url” and “source” attributes.
 - 2) Scoped-Identifier (SID) addressing: Referencing the “sid” attribute of an element. Mainly in “target” and “ref” attributes, **<SIDREF>** and **<SIDREF_array>** elements.



COLLADA Geometry



BREP Model



Tessellated
Representation

- BREP represents the data structure of an object by giving information about each of the object's faces, edges and vertices and how they are joined together.
- To visualize the geometries, they must first be "tessellated". That means they should be converted into a format which is suitable for graphics cards.
- Tessellated geometries are introduced by the element <mesh>.

* Drath, Rainer. (2010) Datenaustausch in der Anlagenplanung mit AutomationML. Berlin Heidelberg: Springer-Verlag, 2010



COLLADA mesh

- To describe geometric primitives that are formed from the vertex data, the **<mesh>** element may contain zero or more of the primitive elements **<lines>**, **<linestrips>**, **<polygons>**, **<polylist>**, **<triangles>**, **<trifans>**, and **<tristrips>**.
- The **<vertices>** element under **<mesh>** is used to describe mesh-vertices. Polygons, triangles, and so forth index mesh-vertices, not positions directly.
- Mesh-vertices must have at least one **<input>** (unshared) element with a semantic attribute whose value is **POSITION**.
- **<source>** provides the bulk of the mesh's vertex data.
- **<source>** contains a **<float_array>** element to define coordinates for the positions or normals, etc.
- The **<accessor>** element under **<source>****<technique_common>** declares an access pattern into the float array element.
- The indices in a **<p>** element refer to different inputs depending on their order. The first index in a **<p>** element refers to all inputs with an **offset** of 0. The second index refers to all inputs with an **offset** of 1.



COLLADA mesh

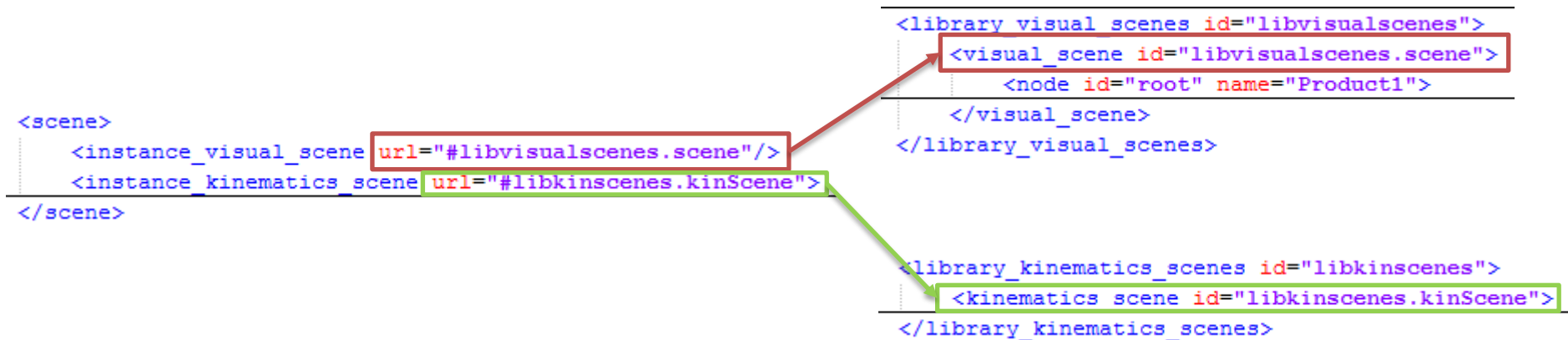
```

<geometry id="geo0" name="GEOM_0">
  <mesh>
    <source id="geo0.positions">
      <float_array id="geo0.positions-array" count="60">-3000 -3000 0 -3000 -3000 0 -3000 -3000 0 -3000 3000 0 -3000
      15000 1500 0 15000 3000 -3000 0 3000 -3000 0 3000 -3000 0 3000 3000 0 3000 3000 0 3000 3000 0 3000 3000 0</float_array>
    <technique_common>
      <accessor count="20" source="#geo0.positions-array" stride="3">
        <param name="X" type="float"/>
        <param name="Y" type="float"/>
        <param name="Z" type="float"/>
      </accessor>
    </technique_common>
  </source>
  <source id="geo0.normals">
  <vertices id="geo0.vertices">
    <input semantic="POSITION" source="#geo0.positions"/>
  </vertices>
  <triangles count="2" material="mat0">
    <input offset="0" semantic="VERTEX" source="#geo0.vertices"/>
    <input offset="1" semantic="NORMAL" source="#geo0.normals"/>
    <p>17 17 15 15 2 2 2 2 4 4 17 17</p>
  </triangles>
  <triangles count="3" material="mat1">
  <triangles count="3" material="mat2">
  <triangles count="3" material="mat3">
  <triangles count="3" material="mat4">
  
```




COLLADA scene

- **<scene>** embodies the entire set of information that can be visualized from the contents of a COLLADA resource.
- Each COLLADA document can contain, at most, one **<scene>** element.
- The scene graph is built from the **<visual_scene>** elements instantiated under **<scene>**.
- In the **<scene>** one or more kinematics scenes can be instantiated. For this instance, new parameters can be defined. Because the kinematics models that are defined in the kinematics scene are completely separate from the geometric appearance, the instance can be bound to elements of an instantiated visual scene.
- The instantiated **<physics_scene>** elements describe any physics being applied to the scene.





COLLADA scene

- The **<visual_scene>** element forms the root of the scene graph topology.
- **<visual_scene>** contains one or more **<node>** elements.
- The **<node>** element embodies the hierarchical relationship of elements in a scene.
- **<node>** can have a wide range of child elements, including **<node>** elements themselves.
- The transformation elements transform the coordinate system of the **<node>** element.
- Transformation elements are: **<lookat>**, **<matrix>**, **<rotate>**, **<scale>**, **<skew>**, **<translate>**
- **<instance_geometry>** allows the node to instantiate a geometry object.



COLLADA scene

```

<visual_scene id="libvisualscenes.scene">
  <node id="root" name="Product1">
    <node id="Product.1_0" name="Product">
      <instance_geometry url="#geo0">
        <bind_material>
          <technique common>
            <instance_material symbol="mat0" target="#material4278190335"/>
            <instance_material symbol="mat1" target="#material4278190335"/>
            <instance_material symbol="mat2" target="#material4278190335"/>
            <instance_material symbol="mat3" target="#material4278190335"/>
            <instance_material symbol="mat4" target="#material4278190335"/>
          </technique_common>
        </bind_material>
      </instance_geometry>
    </node>
    <node id="Product2.1_1" name="Product2">
    <node id="Product3.1_2" name="Product3">
  </node>
</visual_scene>

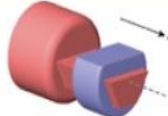

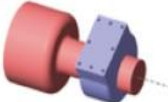
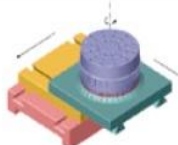


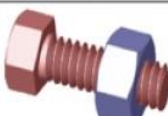
<library_geometries id="libgeom">
  <geometry id="geo0" name="GEOM_0">
    <mesh>
      <source id="geo0.positions">
      <source id="geo0.normals">
      <vertices id="geo0.vertices">
        <triangles count="2" material="mat0">
        <triangles count="3" material="mat1">
        <triangles count="3" material="mat2">
        <triangles count="3" material="mat3">
        <triangles count="3" material="mat4">
      </mesh>
      <extra>
    </geometry>
    <geometry id="geo1" name="GEOM_1">
    <geometry id="geo2" name="GEOM_2">
  </library_geometries>

  <library_materials id="libmat">
    <material id="material4278190335">
    <material id="material4278255360">
    <material id="material4294901760">
  </library_materials>

```



COLLADA kinematics: Definition of joints

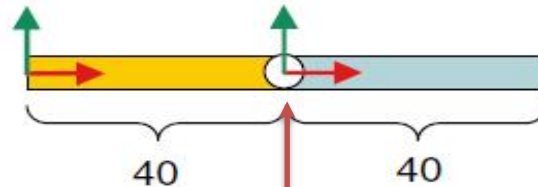
Type of joint	Degree of Freedom	Example
Prismatic	1x trans.	
Revolute	1x rot.	
Cylindrical	1x rot., 1x trans.	
Planar	1x rot., 2x trans.	
Ball and socket	3x rot.	
Universal	2x rot.	
Screw	1x rot., 1x trans.	

```

<library_joints>
  <joint name="prismatic">
    <prismatic>
      <axis sid="axis">1 0 0</axis>
      <limits>
        <max>50</max>
        <min>-30</min>
      </limits>
    </prismatic>
  </joint>
  <joint name="universal">
    <revolute sid="revolute1">
      <axis sid="axis">0 1 0</axis>
      <limits>
        <max sid="max">180</max>
        <min sid="min">-180</min>
      </limits>
    </revolute>
    <revolute sid="revolute2">
      <axis sid="axis">0 0 1</axis>
    </revolute>
  </joint>
</library_joints>
  
```



Kinematics model



```
<library_joints>
  <joint id="joint" name="revolute">
    <revolute>
      <axis>0 0 1</axis>
    </revolute>
  </joint>
</library_joints>
<library_kinematics_models>
  <kinematics_model id="kinematics_model">
    <technique_common>
      <instance_joint url="#joint" sid="joint"/>
      <link sid="upper_arm">
        <attachment full joint="joint">
          <translate>40 0 0</translate>
          <link sid="forearm"/>
        </attachment_full>
      </link>
    </technique_common>
  </kinematics_model>
</library_kinematics_models>
```



Combining geometry and kinematics

```
<scene>
  <instance_visual_scene url="#libvisualscenes.scene"/>
  <instance_kinematics_scene url="#libkinscenes.kinScene">
    <bind_kinematics_model node="Product.1 0">
      <param>libkinscenes.kinScene libkinscenes.kinScene inst_kinmodel.0</param>
    </bind_kinematics_model>
    <bind_joint_axis target="Product2.1 1/joint 1 axis0">
    <bind_joint_axis target="Product3.1 2/joint 1 axis0">
    <bind_joint_axis target="Product3.1 2/joint 0 axis0">
  </instance_kinematics_scene>
</scene>
```

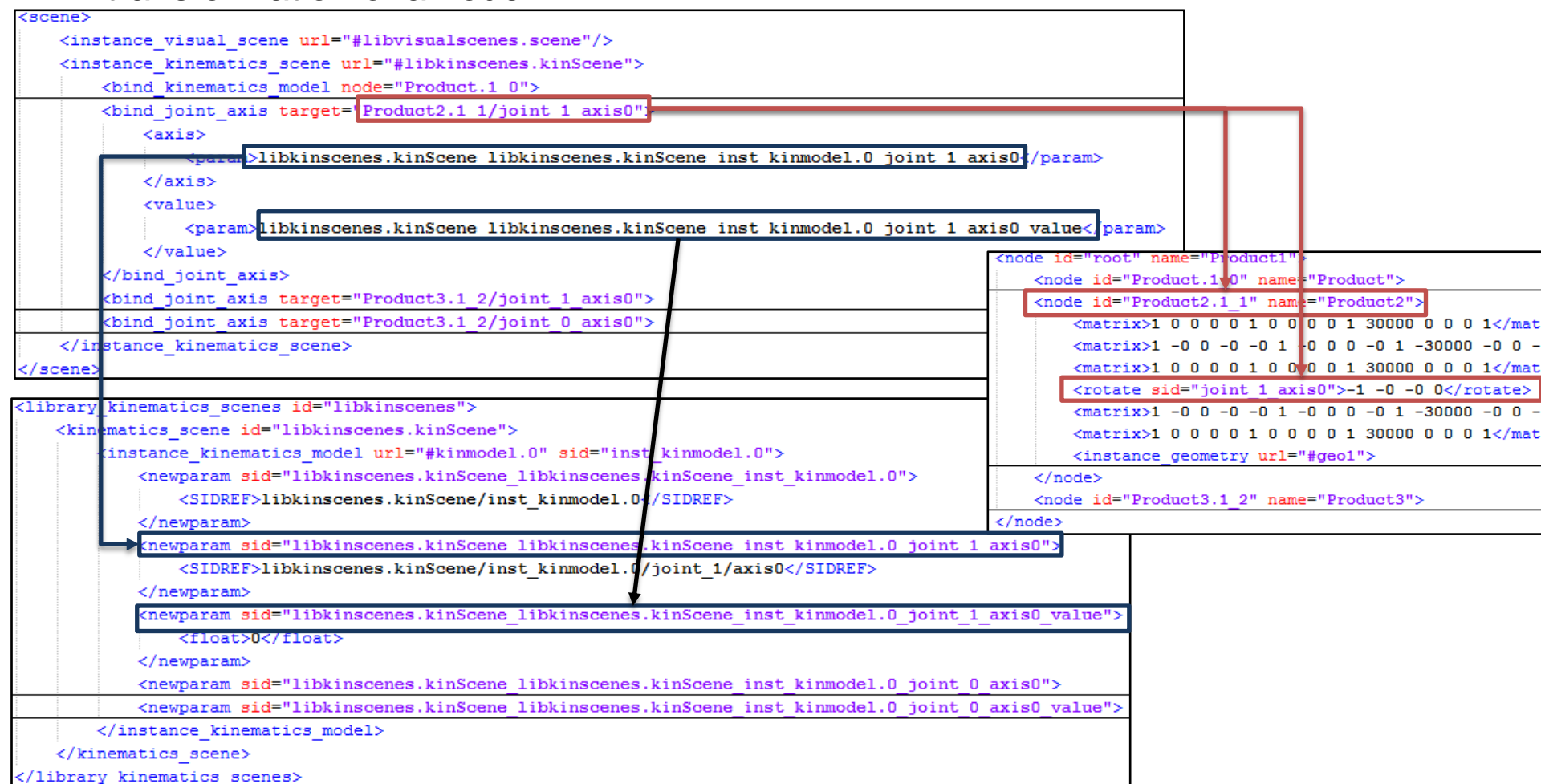
```
<library_visual_scenes id="libvisualscenes">
  <visual_scene id="libvisualscenes.scene">
    <node id="root" name="Product1">
      <node id="Product.1 0" name="Product">
      <node id="Product2.1 1" name="Product2">
      <node id="Product3.1 2" name="Product3">
    </node>
  </visual_scene>
</library_visual_scenes>
```

```
<library_kinematics_scenes id="libkinscenes">
  <kinematics_scene id="libkinscenes.kinScene">
    <instance_kinematics_model url="#kinmodel.0" sid="inst_kinmodel.0">
      <newparam sid="libkinscenes.kinScene libkinscenes.kinScene inst_kinmodel.0">
        <SIDREF>libkinscenes.kinScene/inst_kinmodel.0</SIDREF>
      </newparam>
      <newparam sid="libkinscenes.kinScene libkinscenes.kinScene inst_kinmodel.0 joi
      <newparam sid="libkinscenes.kinScene libkinscenes.kinScene inst_kinmodel.0 joi
      <newparam sid="libkinscenes.kinScene libkinscenes.kinScene inst_kinmodel.0 joi
      <newparam sid="libkinscenes.kinScene libkinscenes.kinScene inst_kinmodel.0 joi
    </instance_kinematics_model>
  </kinematics_scene>
</library_kinematics_scenes>
```




Combining geometry and kinematics

- **<bind_joint_axis>** binds a joint axis of a kinematics model to a single transformation of a node.





COLLADA physics

- Supports basic rigid body dynamics. A rigid body is a non-deformable object that interacts with other rigid bodies according to Newton's basic laws of physics.
- A constraint specifies how one body can move in relation to the other. It can limit the various angular and linear degrees of freedom.
- Physically based systems **don't have** the same notion of hierarchy or parent-child relationships used by animated articulated models.
- The simulated physics models are visualized by having their instantiated rigid bodies directly control the placement of nodes in the visual scene. The node could display different geometry than what is defined in the physics scene.
- Convex meshes might be needed for proper collision. **<convex_mesh>** element generates the convex hull for a given mesh.
- In addition to general meshes and convex hulls, COLLADA defines analytical shapes (boxes, spheres, capsules) for collision volumes, too. -> Better representation of certain round surfaces, improved performance, and reduced memory usage.



Physics model

- The **<physics_model>** element is a logical grouping mechanism for rigid bodies and constraints. Physics models might be defined with a single rigid body or with rigid bodies which have constraints linking them.
- Rigid-bodies may contain a single shape or a collection of shapes for collision detection.
- A **<physics_material>** or **<instance_physics_material>** describes the surface properties for restitution and friction.
- **<rigid_constraint>** is specified by:
 - Two attachment frames
 - Its degrees-of-freedom (DOF).

```

<physics_model id="Bullet-PhysicsModel" name="Bullet-PhysicsModel">
  <rigid_body sid="dRigidBodyShape4-RigidBody" name="dRigidBodyShape4-RigidBody">
    <rigid_body sid="dRigidBodyShape1-RigidBody" name="dRigidBodyShape1-RigidBody">
      <technique_common>
        <dynamic>true</dynamic>
        <mass>1</mass>
        <inertia>0.141067 0.141067 0.141067</inertia>
        <instance_physics_material url="#dRigidBodyShape1-PhysicsMaterial"/>
        <shape>
          <box>
            <half_extents>0.5 0.5 0.5</half_extents>
          </box>
        </shape>
      </technique_common>
    </rigid_body>
    <rigid_body sid="dRigidBodyShape3-RigidBody" name="dRigidBodyShape3-RigidBody">
    <rigid_body sid="dRigidBodyShape2-RigidBody" name="dRigidBodyShape2-RigidBody">
    <rigid_constraint sid="BulletUnnamedConstraint-0" name="BulletUnnamedConstraint-0">
      <ref_attachment rigid_body="dRigidBodyShape2-RigidBody">
        <translate>0 -2.78861 2.56579</translate>
        <rotate>1 0 0 0</rotate>
      </ref_attachment>
      <attachment rigid_body="dRigidBodyShape1-RigidBody">
        <translate>0 2.78861 -2.56579</translate>
        <rotate>1 0 0 90</rotate>
      </attachment>
      <technique_common>
        <enabled>true</enabled>
        <interpenetrate>>false</interpenetrate>
        <limits>
          <swing_cone_and_twist>
            <min>0 0 1</min>
            <max>0 0 -1</max>
          </swing_cone_and_twist>
          <linear>
            <min>0 0 0</min>
            <max>0 0 0</max>
          </linear>
        </limits>
      </technique_common>
    </rigid_constraint>
  </physics_model>
  
```



Combining visual and physics scene

```
<scene>
  <instance physics_scene url="#Scene-Physics"/>
  <instance_visual_scene url="#Scene"/>
</scene>
```

```
<library_physics_scenes>
  <physics_scene id="Scene-Physics" name="MyPhysicsScene">
    <instance_physics_model url="#Bullet-PhysicsModel">
      <instance_rigid_body body="dRigidBodyShape4-RigidBody" target="#dRigidBodyShape4">
        <instance_rigid_body body="dRigidBodyShape1-RigidBody" target="#dRigidBodyShape1">
          <technique_common>
            <angular_velocity>0 0 0</angular_velocity>
            <velocity>0 0 0</velocity>
          </technique_common>
        </instance_rigid_body>
      <instance_rigid_body body="dRigidBodyShape3-RigidBody" target="#dRigidBodyShape3">
        <instance_rigid_body body="dRigidBodyShape2-RigidBody" target="#dRigidBodyShape2">
          <instance_rigid_constraint constraint="BulletUnnamedConstraint-0"/>
        </instance_rigid_body>
      </instance_physics_model>
    </instance_physics_model>
  </physics_scene>
</library_physics_scenes>
```

```
<library_visual_scenes>
  <visual_scene id="Scene" name="MyScene">
    <node id="dRigidBodyShape4" name="dRigidBodyShape4">
      <node id="dRigidBodyShape1" name="dRigidBodyShape1">
        <node id="dRigidBodyShape3" name="dRigidBodyShape3">
          <node id="dRigidBodyShape2" name="dRigidBodyShape2">
            <!-- content -->
          </node>
        </node>
      </node>
    </node>
  </visual_scene>
</library_visual_scenes>
```



COLLADA programming

- COLLADA DOM (Document Object Model) is an open-source (MIT-licence) application programming interface (API) for C++.
<http://sourceforge.net/projects/collada-dom/>
- Alternative: Creating your own COLLADA objects(POJOs) in Java.
- How to:
 - 1) Download COLLADA schema.
<https://www.collada.org/2008/03/COLLADASchema/>
 - 2) Xjc to generate classes from XML schema. ->Part of JDK. Under “bin” folder.
 - 3) Command “xjc collada_schema_1_5.xsd” should work, but doesn’t because of name collision error.
 - 4) Download this schemalet "simpleMode.xml":
<http://weblogs.java.net/blog/kohsuke/archive/20060315/simpleMode.xsd/simpleMode.xsd>
 - 5) Command "xjc collada_schema_1_5.xsd -extension simpleMode.xml" (make sure the file paths are correct)



Questions?
Thank you for your attention.