

Joint Interaction

Roland Landertshamer
0155923
University of Linz
roland.landertshamer@liwest.at

21st June 2006

1 General information

The JointInteraction module is a simple, single threaded interaction system which simulates the movement of manipulated objects in a virtual environment by using a physics engine. An interaction module allows the users of an application to manipulate the state of an object in a virtual world. The JointInteraction module was built to simulate interaction like it would be in real world, called *Natural Interaction*. The user can grab an object and then transform it by moving or rotating the virtual representation of his hand. To be able to simulate objects that are connected together via an articulation (=joint) the module uses the Open Dynamics Engine (ODE) for simulating object movement.

The JointInteraction module was built for use with the *inVRs* framework and therefore uses the object-structure of this system.

1.1 Joints:

A Joint defines a movement constraint for an object. For example a HingeJoint allows an object to rotate around an axis but not to move away from it. Usually a Joint is used to connect two objects together and allow a restricted, relative transformation of both. It can also be used for restricting the transformation of only one object relative to the static environment.

Joints have a position and/or one or more axis (see Joint types). The position and axis data are always relative to a main object. This main object can either be an EntityTransform or the static environment. If the main object is the static environment, the position and axis data has to be set in world coordinates. Otherwise the data has to be set in object coordinates of the EntityTransform.

For example we want to build a door consisting of a door frame and the door body. Therefore we need two EntityTransforms, which have to be positioned in our Environment (by setting the transformation values in the environment-configuration). Both objects have the pivot at their center (sizeX/2, sizeY/2, sizeZ/2). Let's assume that the door frame and the door body are translated to the position (10, 0, 0) and rotated by 45 degrees around the Y-axis.

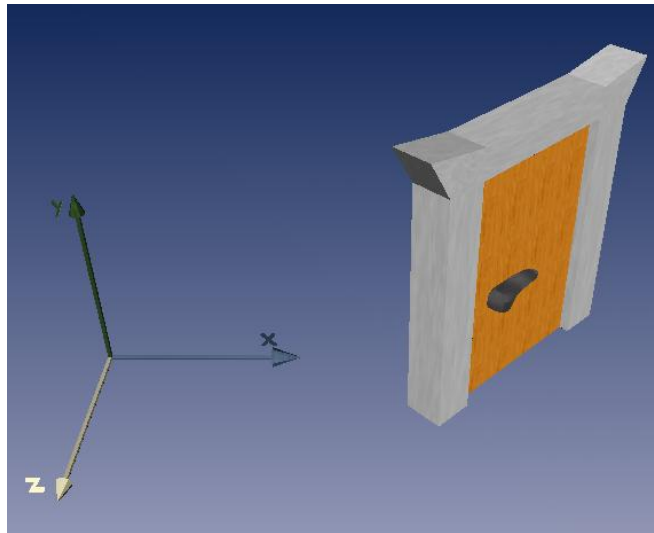


Figure 1.1: Door frame and door body translated by (10, 0, 0) and rotated by 45 degrees around Y-axis

Now we want to add a HingeJoint between these two objects. The first step is to choose the main object, where the joint should be attached to. In our case, the door frame would be a good choice. Next we have to find the position of the Hinge axis. Since all Joint data is always relative to the main object (in object coordinates of the main object) the position of the axis is a simple offset to the main object's pivot. In our case this would be a vector in the object's X-direction, like (1, 0, 0). At last we have to set the Joint's axis, which would be (0, 1, 0) in our case.

Now we have a joint for our door example. Let's assume we want to add another door in our Environment which has another position and maybe another orientation than the first one. Finding the HingeJoint for

this second door is very easy, as long as you are familiar with copy and paste. Since the transformation data of the Joint is relative to its main object you can insert as many objects as you want in your Environment, the Joint configuration is identical for all.

2 Configuration

Two types of configuration files exist:

- simulation configuration
- joint configuration

2.1 Simulation configuration

The simulation configuration file stores the global module-settings. The possible elements are:

- **stepsPerFrame**: This value sets the amount of ODE-simulation-steps per frame and kind of force. That means, that a value of e.g. 5 stepsPerFrame causes 5 simulation steps for calculating the impact of force and another 5 simulation steps for the impact of torque. The higher this value gets, the lower is the error produced by ODE. This error can vary from system to system, so the stepsPerFrame have to be found by testing (with an object with at least 2 joints).
- **maxDist**: This value sets the maximum distance of the user object to the grabbed object. If the distance gets above this value the object will be released automatically.
- **maxRotDist**: This value sets the maximum angular distance of the user object to the grabbed object. If the orientation of the user object and the orientation of the grabbed object differ by an angle above this value, the object will be released automatically.

2.2 Joint configuration

This configuration file stores the joint information. Since a joint always connects objects of type EntityTransform and every EntityTransform has a unique ID in one Environment, the joint configuration is also only valid in one Environment. A joint configuration file contains a list of joint-elements:

```
<?xml version="1.0"?>
<joints>
  <joint type="..." id="...">
    ...
  </joint>
  <joint type="..." id="..." active="...">
    ...
  </joint>
  ...
</joints>
```

Every joint-element has two required and one optional attribute:

- **type**: defines the joint-type (see Joint types)
- **id**: sets a unique joint-ID (valuetype = integer)
- **active** (optional): sets the initial state of the joint (see Activation Conditions)

Joint types

Since the joint-simulation is done with ODE (v0.5), the available joint types are equal to the ones in the physics engine. The supported types are:

- **hinge**: like a door hinge (see Hinge Joint)
- **ball**: fixed position, free orientation (see Ball and Socket Joint)
- **slider**: fixed orientation, movable along an axis (see Slider Joint)
- **universal**: a joint with 2 perpendicular axis (see Universal Joint)
- **hinge2**: a joint with 2 axis (must not be perpendicular) (see 2-axis Hinge Joint)

Hinge Joint:

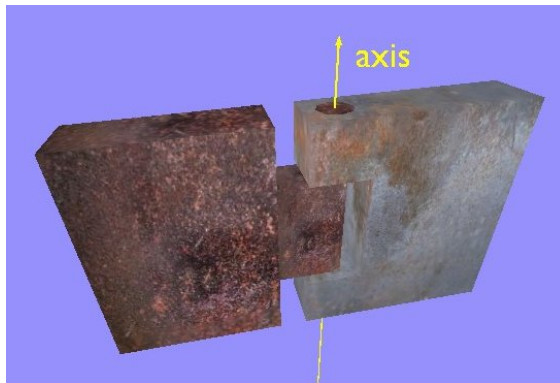


Figure 2.1: Hinge Joint

A Hinge Joint behaves like a door hinge. It has a fixed axis where the objects can rotate around. The configuration file entry for a Hinge Joint consists of the following elements:

- **entities**: Entry for setting the EntityTransforms which are connected by the Joint
- **anchor**: Setting the anchor point of the Hinge axis
- **axis**: Entry for the direction vector of the Hinge axis
- **angles** (optional): Entry for restricting the Joint angle
- **activeIF**, **activateIF**, **deactivateIF** (optional): activation conditions (see Activation Conditions)

The element **entities** sets the objects that should be connected. Objects are represented by EntityTransforms, so the entry has to contain the IDs of the connected EntityTransforms.

The element has three attributes: *id1*, *id2* and *anchorEntity*.

The attribute *id1* sets the ID of the first EntityTransform to which the joint should be connected.

The attribute *id2* sets the ID of the second EntityTransform. If the Joint should not connect two objects but one object to the static environment, then *id2* can be set to zero.

The attribute *anchorEntity* sets the main object, where the Joint should be connected to (like the door frame in our Example). The value of *anchorEntity* has to be 1 or 2 for the EntityTransform with *id1* or *id2*.

The element **anchor** sets the offset of the Hinge axis to the main object's (=anchorEntity) pivot point. This offset has to be set in object coordinates of the main object (see Example).

The element has three attributes: *xPos*, *yPos* and *zPos*, where each attribute sets the offset in the corresponding axis.

The element **axis** sets the direction of the Hinge axis. Like the anchor point, the direction vector also has to be set in object coordinates of the main object. For better usability, the direction vector doesn't have to be normalized.

The element has three attributes: *xDir*, *yDir* and *zDir*, where each attribute sets the part of the direction vector in the corresponding axis.

The optional element **angles** sets the minimum and maximum rotation angle of the Joint. The values have to be set in degrees where the range goes from -360 to 360 degrees. As an example for our door example there could be the limitation, that the door can only open in one direction, so the minimum angle would be 0 and the maximum angle could be 90 degrees.

Example for a Hinge Joint configuration:

```
<joint type="hinge" id="1">
  <entities id1="4" id2="5" anchorEntity="2">
    <anchor xPos="1" yPos="0" zPos="0">
      <axis xDir="0" yDir="1" zDir="0">
        <angles min="0" max="90">
      </angles>
    </axis>
  </anchor>
</entities>
</joint>
```

Ball and Socket Joint:

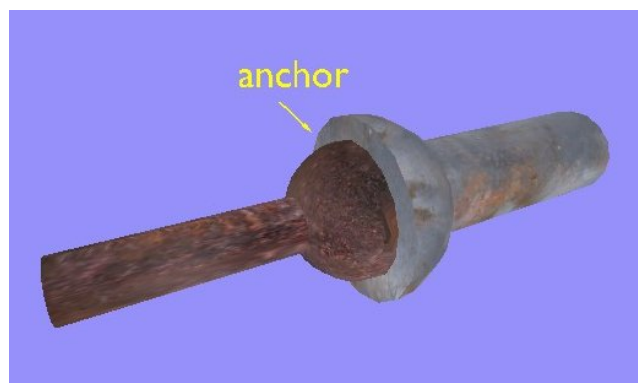


Figure 2.2: Ball and Socket Joint

The Ball and Socket Joint allows the objects to freely rotate around the Joint's anchor but not to move away from it.

The configuration file entry for this Joint consists of the following elements:

- **entities**: sets the EntityTransforms that are connected to the Joint (see Hinge Joint)
- **anchor**: sets the anchor point of the Ball and Socket Joint (see Hinge Joint)
- **activeIF**, **activateIF**, **deactivateIF** (optional): activation conditions (see Activation Conditions)

Since the Ball and Socket Joint allows the objects to rotate in any direction around the anchor point, the only data to set is the position of the anchor. Restrictions in rotation angles as in the Hinge Joint are not available here.

Example for a Ball and Socket Joint configuration:

```
<joint type="ball" id="2">
  <entities id1="3" id2="1" anchorEntity="2">
    <anchor xPos="0.93" yPos="0.21" zPos="0">
  </anchor>
</entities>
</joint>
```

Slider Joint:



Figure 2.3: Slider Joint

The Slider Joint allows the objects to move along one axis, but does not allow any rotations. The configuration file entry consists of:

- **entities**: sets the EntityTransforms that are connected to the Joint (see Hinge Joint)
- **axis**: sets the direction of the movement axis (see Hinge Joint)
- **positions** (optional): Entry for restricting the movement along the axis
- **activeIF, activateIF, deactivateIF** (optional): activation conditions (see Activation Conditions)

The Slider Joint configuration only needs the direction of the slider axis, since all other movement (not in axis direction) and rotation is avoided.

The optional element **positions** allows the user to restrict the position offset of the objects along the Joint's axis. It therefore allows to set a minimum and maximum distance from the start point. For example, take an air cylinder with the length of 50 centimeters. Let's assume that the piston of the cylinder is retracted. Now we attach a Slider Joint to the air cylinder and its piston. Since the piston is retracted, the minimum position offset for the piston is 0, because otherwise we could move the piston through the cylinder. The maximum offset of the piston could be about 50 centimeters, because otherwise we could extract the piston from the cylinder.

Example for a Slider Joint configuration:

```
<joint type="slider" id="3">  
  <entities id1="6" id2="7" anchorEntity="1">  
    <axis xDir="1" yDir="0" zDir="0">  
      <positions min="0" max="50">  
</joint>
```

Universal Joint:

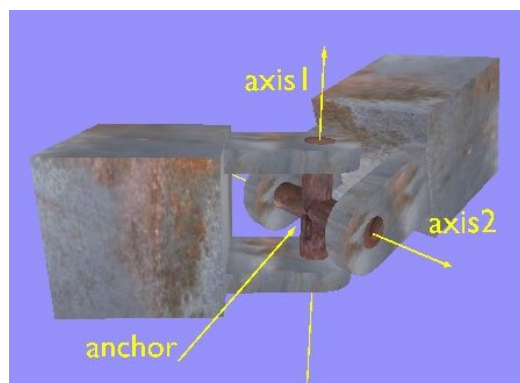


Figure 2.4: Universal Joint

The Universal Joint can be seen as a Hinge Joint with 2 perpendicular axis. Both axis intersect at the anchor point of this Joint.

The configuration file entry consists of:

- **entities**: sets the EntityTransforms that are connected to the Joint (see Hinge Joint)
- **anchor**: sets the anchor point of the Universal Joint (see Hinge Joint)
- **axis**: sets the direction of the two perpendicular axis.
- **activeIF, activateIF, deactivateIF** (optional): activation conditions (see Activation Conditions)

The Universal Joint has to be configured by setting an anchor point, where the two perpendicular axis intersect, and the direction of the two axis.

The element **axis** looks like at the other Joints (e.g. Hinge Joint), but has a new attribute *index*. This attribute can take the values 1 or 2 and defines, which axis is meant.

Example for a Universal Joint configuration:

```
<joint type="universal" id="7">
  <entities id1="10" id2="11" anchorEntity="1"/>
  <anchor xPos="0" yPos="0" zPos="-8.7"/>
  <axis index="1" xDir="1" yDir="0" zDir="0"/>
  <axis index="2" xDir="0" yDir="1" zDir="0"/>
</joint>
```

2-axis Hinge Joint:

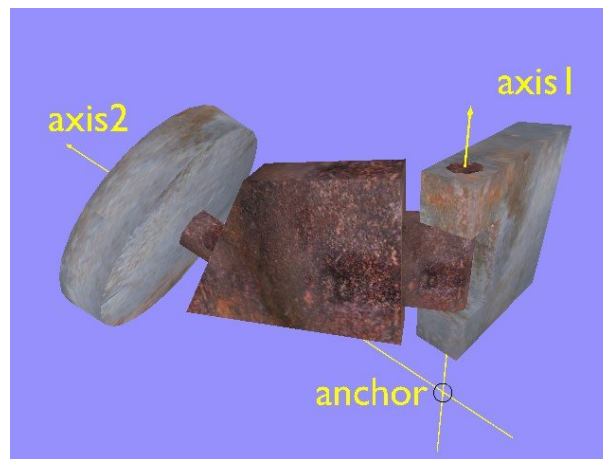


Figure 2.5: 2-axis Hinge Joint

The 2-axis Hinge Joint is equal to the Universal Joint with the only difference, that the 2 axis don't have to be perpendicular. A Universal Joint can also be built with a 2-axis Hinge Joint.

The configuration file entry for a 2-axis Hinge Joint consists of:

- **entities**: sets the EntityTransforms that are connected to the Joint (see Hinge Joint)
- **anchor**: sets the anchor point of the Joint (see Hinge Joint)
- **axis**: sets the direction of the two axis (see Universal Joint)
- **activeIF, activateIF, deactivateIF** (optional): activation conditions (see Activation Conditions)

Example for a 2-axis Hinge Joint configuration:

```
<joint type="hinge2" id="8">
  <entities id1="12" id2="13" anchorEntity="1"/>
  <anchor xPos="5" yPos="5" zPos="0"/>
  <axis index="1" xDir="1" yDir="0" zDir="0"/>
  <axis index="2" xDir="0" yDir="1" zDir="1"/>
</joint>
```

Activation Conditions

Every Joint has the option to be active or not. If a Joint is active (default), it does exactly the restrictions it is built for. An inactive Joint restricts all movement and rotation, so it can be seen as if the two objects were glued together. In ODE this is called a fixed Joint. Setting a Joint inactive can be useful sometimes.

Let's look at our door-example. Let's extend our example by a doorknob. If the doorknob is not pressed, then the door should not open. So we first need a method to set the Hinge Joint between the door frame and the door body to inactive as default.

This is possible with the optional attribute *active* in the **joint**-element. The value *0* means inactive, the value *1* means active.

Example for an inactive Hinge Joint:

```
<joint type="hinge" id="1" active="0">
  ...
</joint>
```

By default, this Hinge Joint now behaves like a fixed Joint, so the door will not open any more. Now we need to define a condition, that this Hinge Joint should be activated, if the doorknob is pressed. This can be done by adding one or more conditional elements. There are four types of elements:

- **activateIF**: if the condition is true the joint will be activated
- **deactivateIF**: if the condition is true the joint will be deactivated
- **activeIF**: the joint is active, if the condition is true and inactive, if the condition is false
- **inactiveIF**: the joint is active, if the condition is false and inactive, if the condition is true

The conditional elements can contain different attributes which define the condition. The following conditions are supported:

CONDITIONS			
CONDITION	RELATED TO	POSSIBLE VALUES	EXAMPLE
isGrabbed	EntityTransform	[0 = false / 1 = true]	<activateIF entity="1" isGrabbed="1" />
isActive	Joint	[0 = false / 1 = true]	<inactiveIF joint="2" isActive="1" />
angle1LT	Joint	[-360, 360]	<deactivateIF joint="5" angle1LT="0.01" />
angle1GT	Joint	[-360, 360]	<activateIF joint="4" angle1GT="30" />
positionLT	Joint	[-∞, ∞]	<deactivateIF joint="6" positionLT="0.1" />
positionGT	Joint	[-∞, ∞]	<activateIF joint="6" positionGT="-10" />

Table 2.1: Available conditions for joint activation

The condition *isGrabbed* can be used to activate or deactivate a Joint, if a special EntityTransform is grabbed. The attribute *entity* sets the ID of the EntityTransform.

The condition *isActive* checks if another Joint is activated or not. The value of the attribute *joint* has to be the ID of the Joint to check.

The conditions *angle1LT* and *angle1GT* check if the angle of the first axis of a Joint is below (*angle1LT*) or above (*angle1GT*) the passed value. The value of the attribute *joint* has to be the ID of the desired Joint. ***At the moment these two conditions only work with Hinge Joints!!!***

The conditions *positionLT* and *positionGT* check if the position offset of a Slider Joint is below or above the passed value. As above, the value of the attribute *joint* has to be the ID of the desired Slider Joint.

Now back to our door-example. Let's assume our EntityTransform that represents the doorknob has the ID 10. If the user grabs the doorknob then the Hinge Joint should be activated, so that the door can open. For that we simply add an **activateIF**-condition to our Hinge Joint configuration:

```
<joint type="hinge" id="1" active="0">
  ...
  <activateIF entity="10" isGrabbed="1"/>
</joint>
```

That's all we have to do. Now the Hinge Joint is deactivated until the user grabs the doorknob. That looks fine at first, but what happens when the user releases the doorknob? Since we didn't define another condition, the Hinge Joint stays in active state. That means, that the user can now grab the door body and close and open the door. Closing the door by grabbing the door body is a thing that can also happen in real life, like if you slam a door. But if the door is closed, it shouldn't be able to open it again without grabbing the doorknob. So we have to add a condition, that deactivates our Hinge Joint if the door is closed. The best solution for that is to add a **deactivateIF**-condition:

```
<joint type="hinge" id="1" active="0">
  ...
  <deactivateIF joint="1" angle1LT="0.01"/>
  <activateIF entity="10" isGrabbed="1"/>
</joint>
```

The **deactivateIF**-condition causes that the Hinge Joint will be deactivated, if the angle of it's axis gets below 0.01 degrees. As we can see it is possible to define more than one condition for a Joint. The conditions are checked in the written order and every one does the action it is built for. Therefore it is important to write the conditions in the correct order. In our example the first condition deactivates the Joint if the Joint's angle is below 0.01 degrees. When the door is closed this condition deactivates the Joint. Afterwards the second condition is checked. If we grabbed the doorknob then the Joint will be activated again and we can open the door. If the two conditions were written in the reverse order the Joint would always be deactivated at last (if the door is closed), no matter if we grab the doorknob or not.

Now our door works like we wanted it to. The example demonstrated the use of the JointInteraction-module and it's functionality. For further examples see the Tutorial-file.