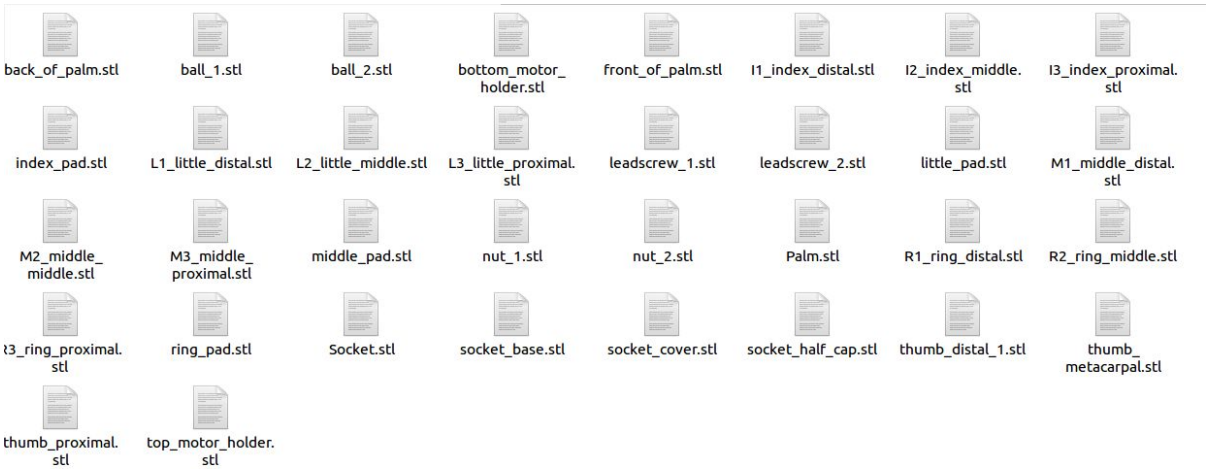
Adding Hands to OpenRave

This tutorial was made by Carlos Rubert (cescuder@uji.es), only for educational and research purposes.

In this tutorial, we'll show the procedure to create a new hand model compatible with Openrave. The Software use for add our hand model will be Openrave and Blender.

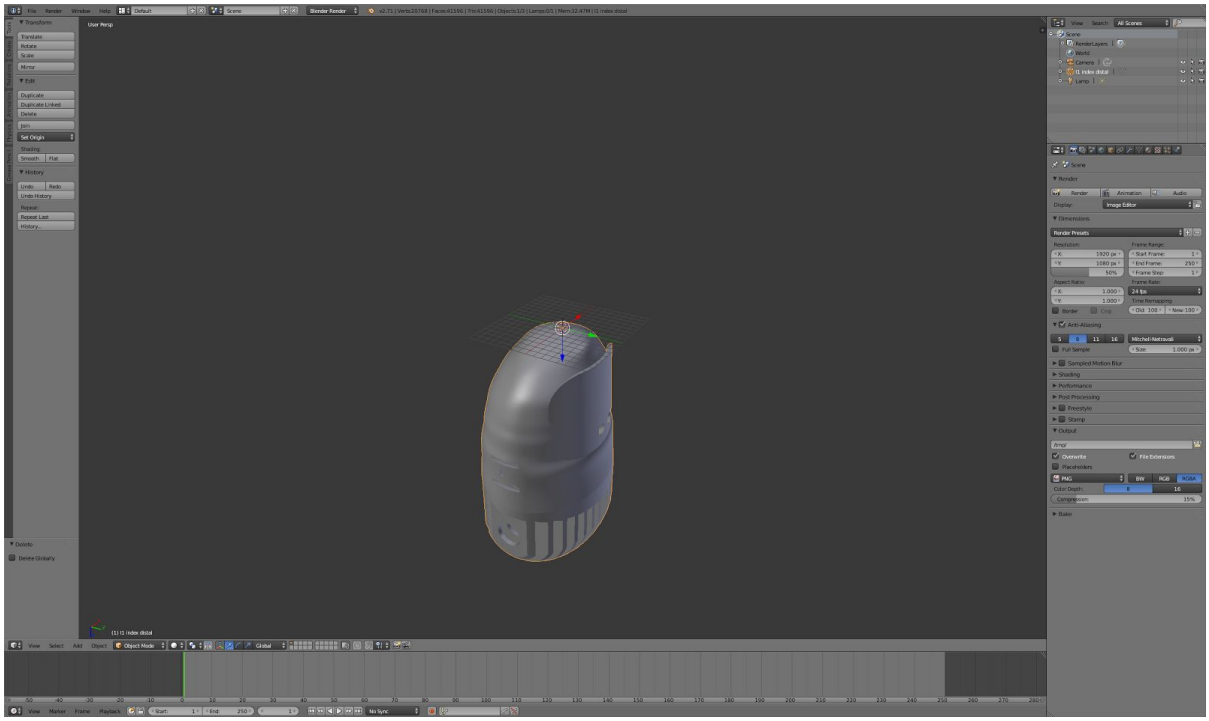
Getting the Models of the Hand

First, we need the models (.stl. iv, .wrl, .ply, etc.) of the different parts of our hand:

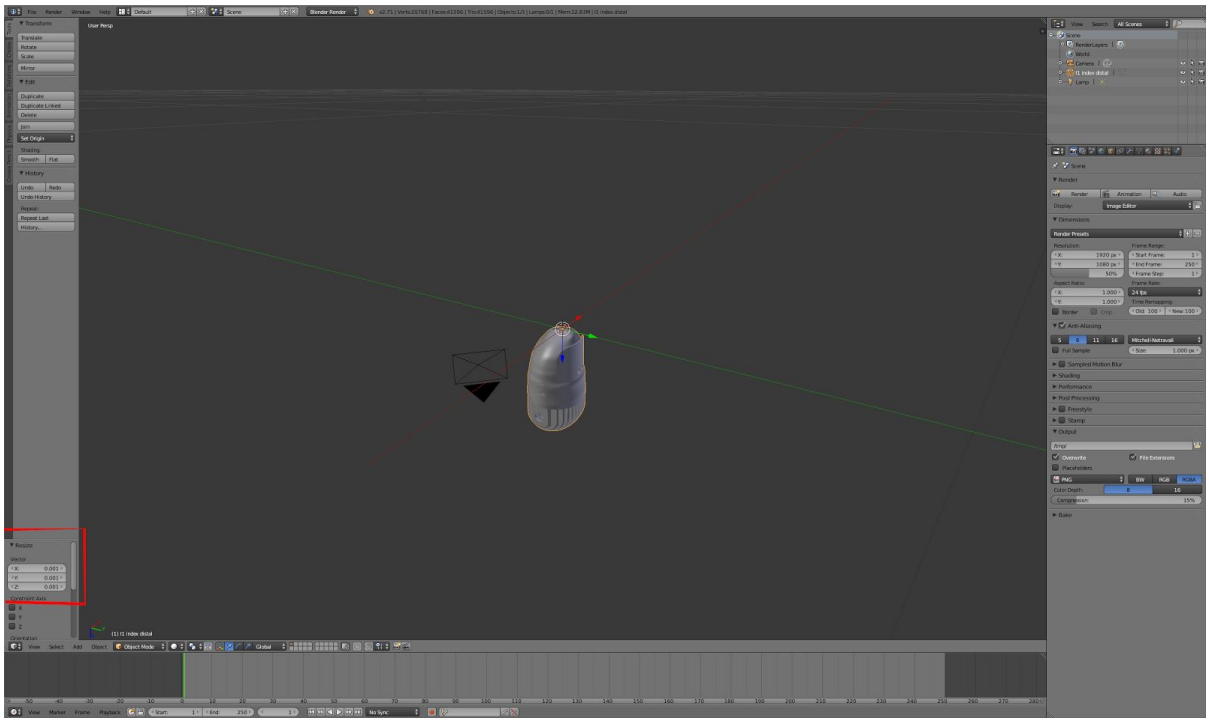


It's important to check the models are right defined, that is, the point of origin is located in the central axis of the joint, the models are properly scaled (usually software as SolidWorks export models files in meters and not in mm), sometimes there are also errors regarding visual effects of models, so important to avoid this kind of "extras to our model (as neither are not important for simulating purposes).

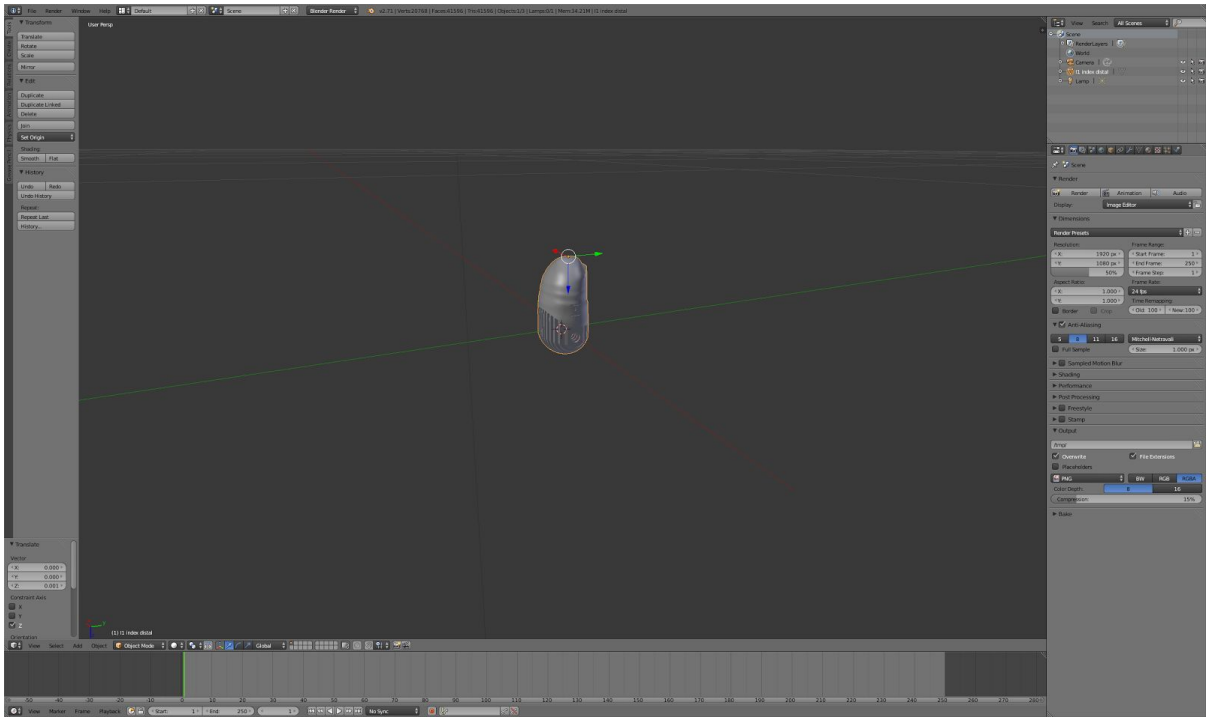
In our case, our modes are in meters, the point of origin is not properly located and there are transparency data in the models. We can solve all this problems easily using Blender:



First, we scale our models to mm (left-bottom-side):



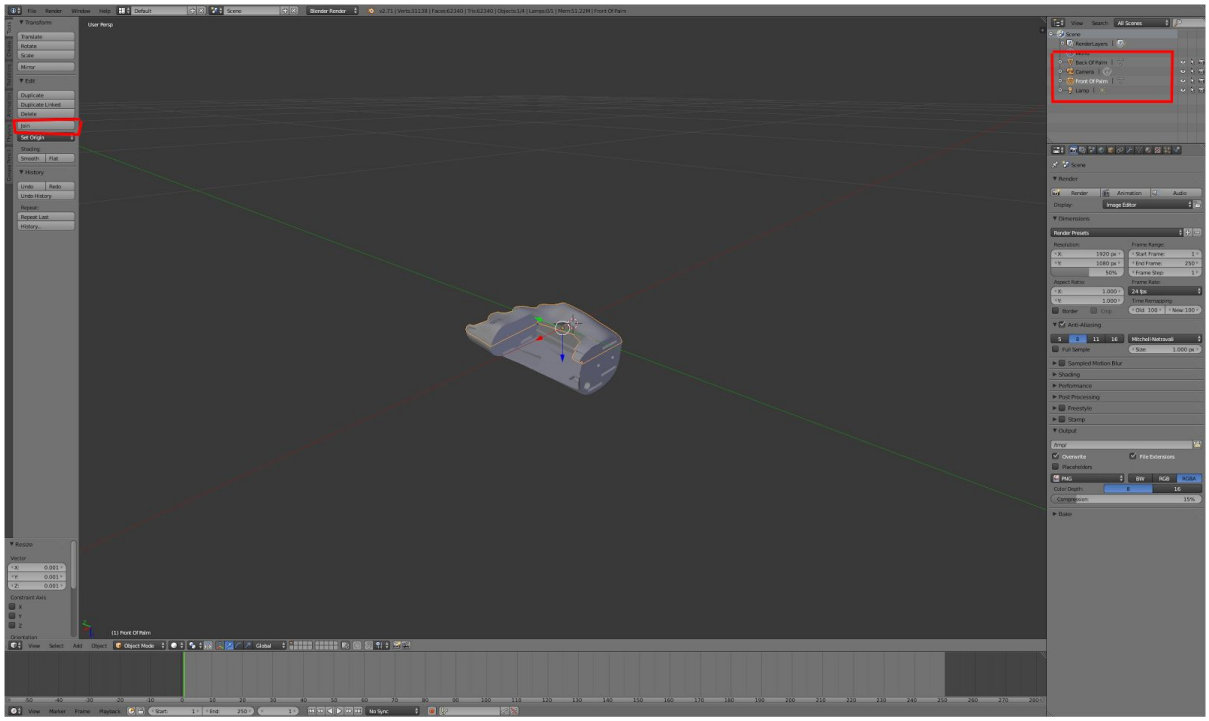
Then, we can move the model until the point of origin is in the axis of the joint:



Just exporting this model again in .stl with Blender will automatically remove the transparency data added by other software (such as SolidWorks).

For simplifying the process of adding a new model in OpenRave, is helpful to have all the models not joint coupled joined in the same model file. In our case, the palm is splitted in different models:

We can use Blender again to load these models and then join them:



Now, we can export this stl model.

When we have resized our models, relocated the point of origin and joined the models without joint, we can proceed to the next step.

Creating the kinbody file

For working with Openrave, we define two different files, one is the base model of our hand (kinbody file) and the other is the manipulator file (robot file). The definition of a manipulator and hand model can also be done in only one file, but for differentiating the function of each code, we'll split them.

As we said, the kinbody file will contain only the definition of the hand model, that is, this file with include the stl models, his the spatial relation between them (translations and rotations), and also the joint i the hand, kind of joint, degrees of movement, axis of actuation, also is possible to add closed chains of joints or mimic joints.

Initially, our kinbody file we'll like this:

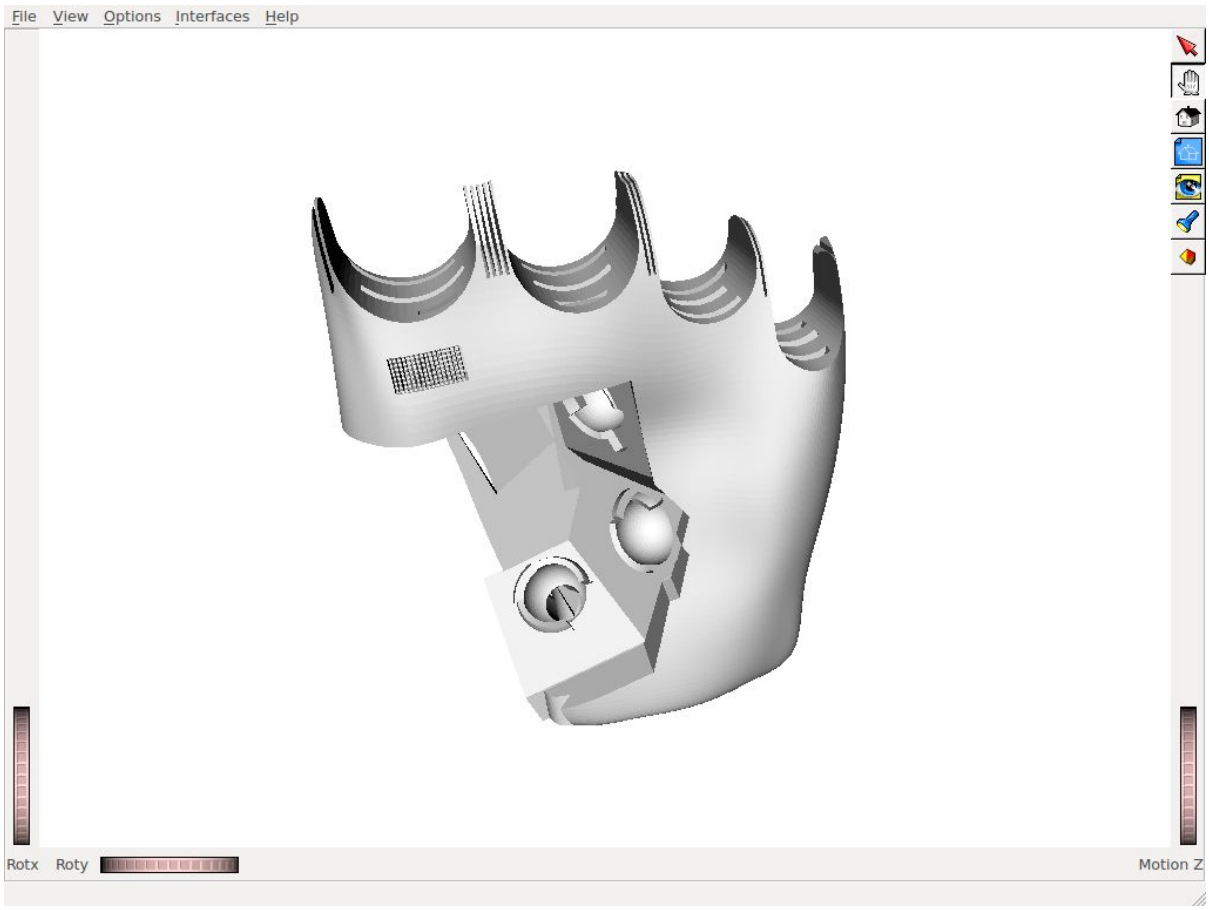
```
<KinBody name="AjitHand">
</KinBody>
```

Now, we have to define first the handbase model, (usually, the palm of the hand):

```
<KinBody name="AjitHand">
    <Body name="handbase" type="dynamic">
        <Geom type="trimesh"  modifiable="false">
            <data>ajit/Palm.stl 1</data>
            <Render>ajit/Palm.stl </Render>
        </Geom>
    </Body>
</KinBody>
```

The number “1” after the data and render files, indicate the scale of the model (so it’s possible to not rescale the models in Blender and just reescale them in Openrave)

Now, our hand in OpenRave will look this this:

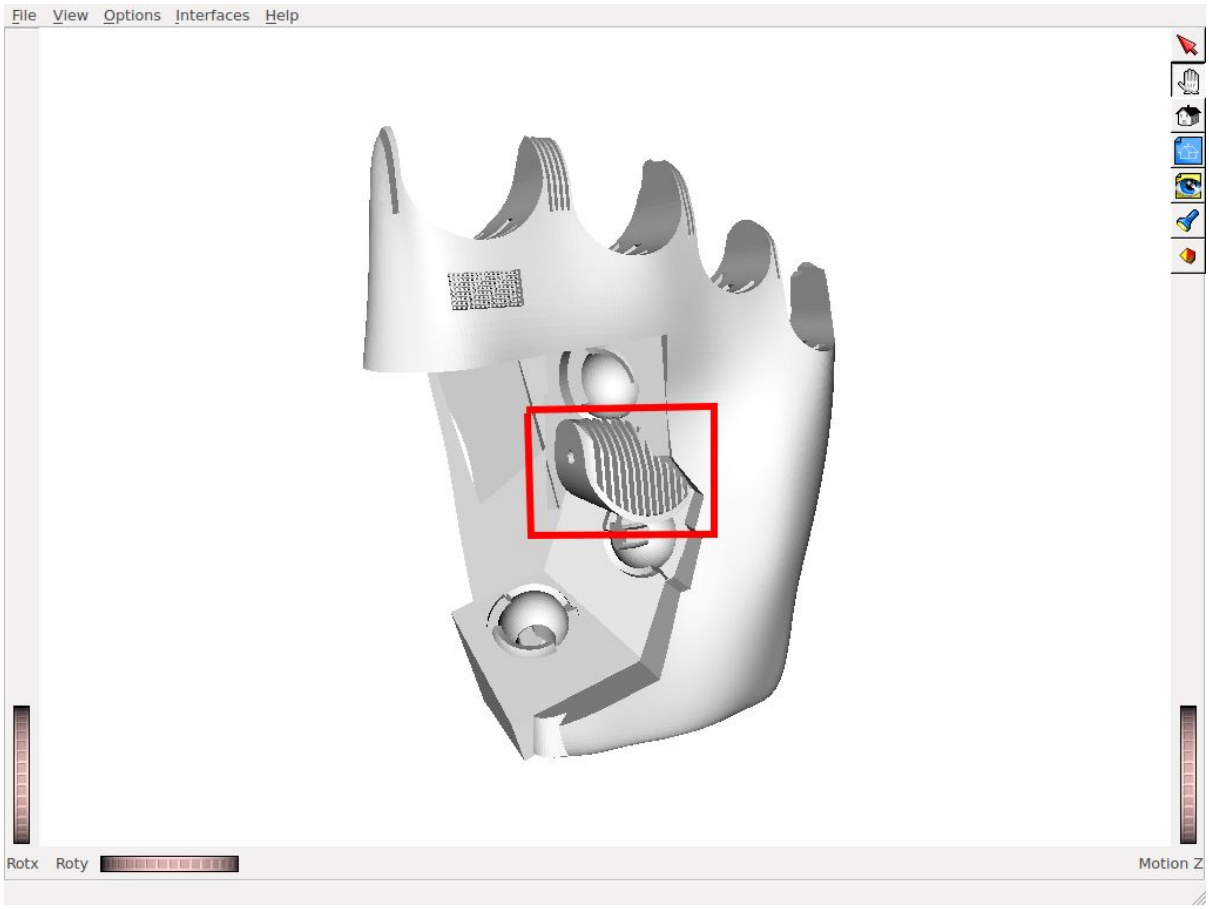


(As you can see, in the previous step of editing the models with Blender, we joined the models of the palm and the sockets of the thumb in the same model file)

Now, we we'll start adding the different models of the fingers, is important in openrave, to have all the parts of the body coupled with joints, even if these are "fake" in the real model, we'll just disable them.

```
<!-- Index -->
<Body name="Index-Pad" type="dynamic"
    <Geom type="trimesh" modifiable="false">
                <data>ajit/index_pad.stl 1</data>
                <Render>ajit/index_pad.stl 1</Render>
    </Geom>
</Body>
```

We have our model with 2 bodies but, where is the new body located?

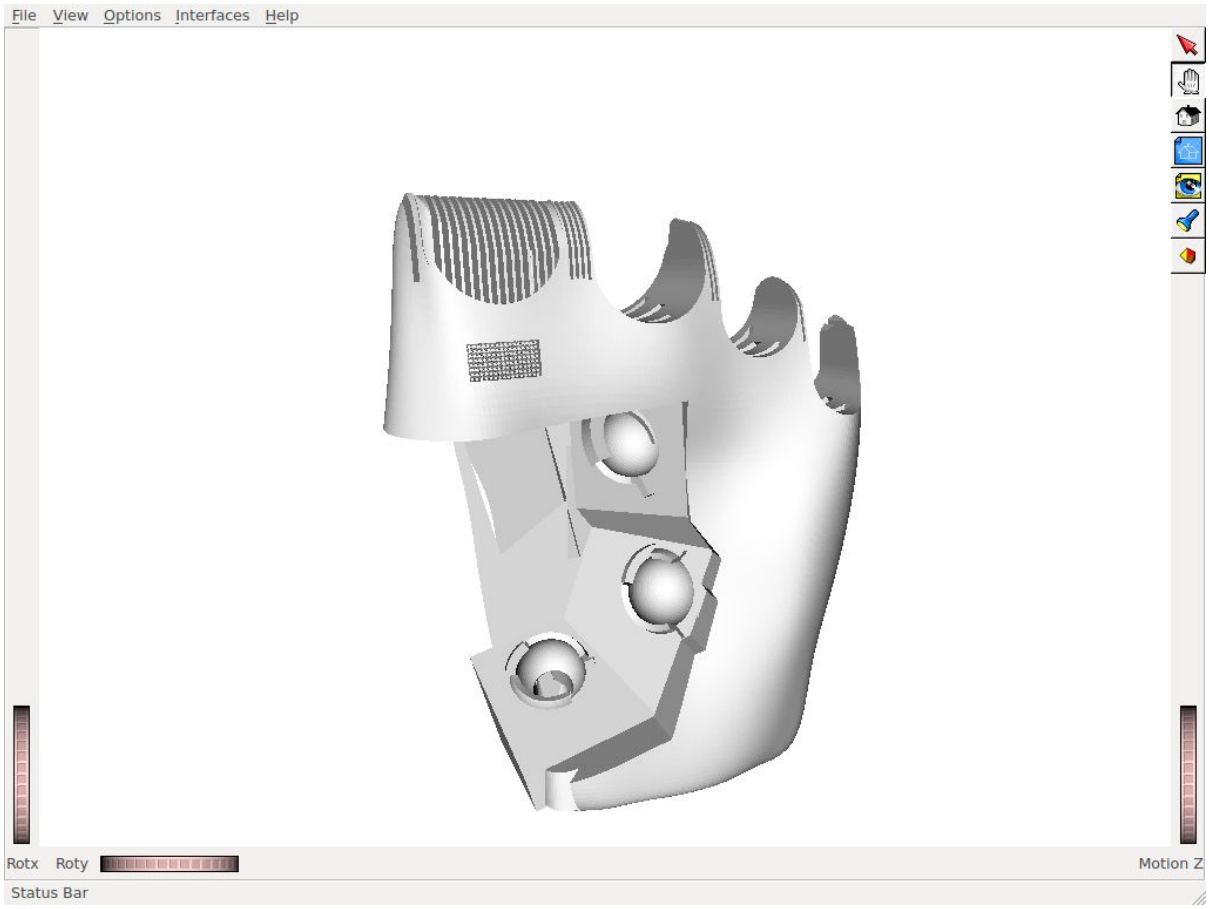


As you can see, the new bodies added to the hand model always are located at the point of origin(0,0,0). Now, we can modify the position of this body using, translations and rotations:

```
<Body name="Index-Pad" type="dynamic">
    <offsetfrom>handbase</offsetfrom>
    <Translation>0.0335 0.057 -0.0055</Translation>
    <rotationaxis>0 0 1 0</rotationaxis>
    <Geom type="trimesh" modifiable="false">
        <data>ajit/index_pad.stl 1</data>
        <Render>ajit/index_pad.stl 1</Render>
    </Geom>
</Body>
```

Let’s analyze this code, first we put and “offset=handbase”, that’s used as point of reference for the new body of the model, if no offset is defined, then point of origin (0,0,0) will be used. The translation parameter is used for moving the body of the index pad, using always the offset as point of reference, the values there are for translation in the axis X, Y and Z, and the values are: m dm cm mm, etc. So in this case, we moved the index pad 3 cm and 3,5 mm in the X axis, 5 cm and 7 mm in the Y axis and 5,5 mm in the Z Axis.
In this case the rotationaxis parameters has not been used, as the model is already in the right orientation

Here is how looks our model now:



There is no easy way to relocate the bodies of the model (as in other 3D software such SolidWorks), so it has to be done manually, changing the parameters of the translation and rotation until it seems to be okey.

We have our hand model with two bodies but, we need to establish the joint relationship of this bodies. In this case, this joint is “fake, so will disable it in the code:

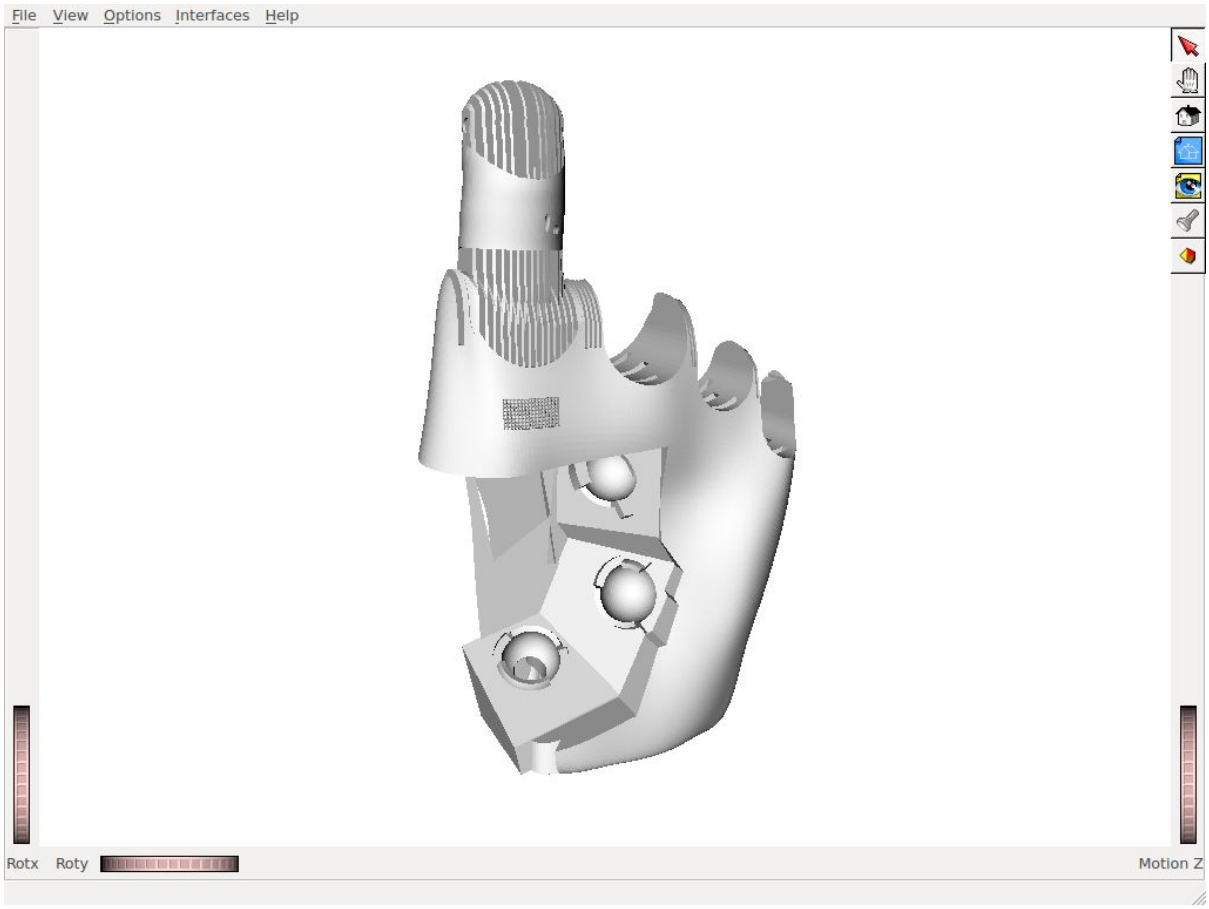
```
<Joint type="hinge" name="JIndex-Pad" enable="false">
    <Body>handbase</Body>
    <Body>Index-Pad</Body>
    <offsetfrom>Index-Pad</offsetfrom>
</Joint>
```

For defining the joints, first we need to set which kind of joint is, OpenRave allows different types of joints, but now all of them work well, so we're going to try to use only the hinge and slider type of joints. Also, we have to define which bodies are coupled by the joint, and which one of this bodies will affected by the DoF

It's time now to add the proximal phalanx of the finger, first we add this body to our model and we place it in the right position modifying the translation and rotation parameters:

```
<Body name="Index-P" type="dynamic">
                <offsetfrom>Index-Pad</offsetfrom>
                <Translation>-0.001 -0.0012 -0.00035</Translation>
                <rotationaxis>0 0 1 180</rotationaxis>
                <rotationaxis>0 1 0 -90</rotationaxis>
                <Geom type="trimesh" modifiable="false">
                        <data>ajit/l3_index_proximal.stl 1</data>
                        <Render>ajit/l3_index_proximal.stl 1</Render>
                </Geom>
        </Body>
```

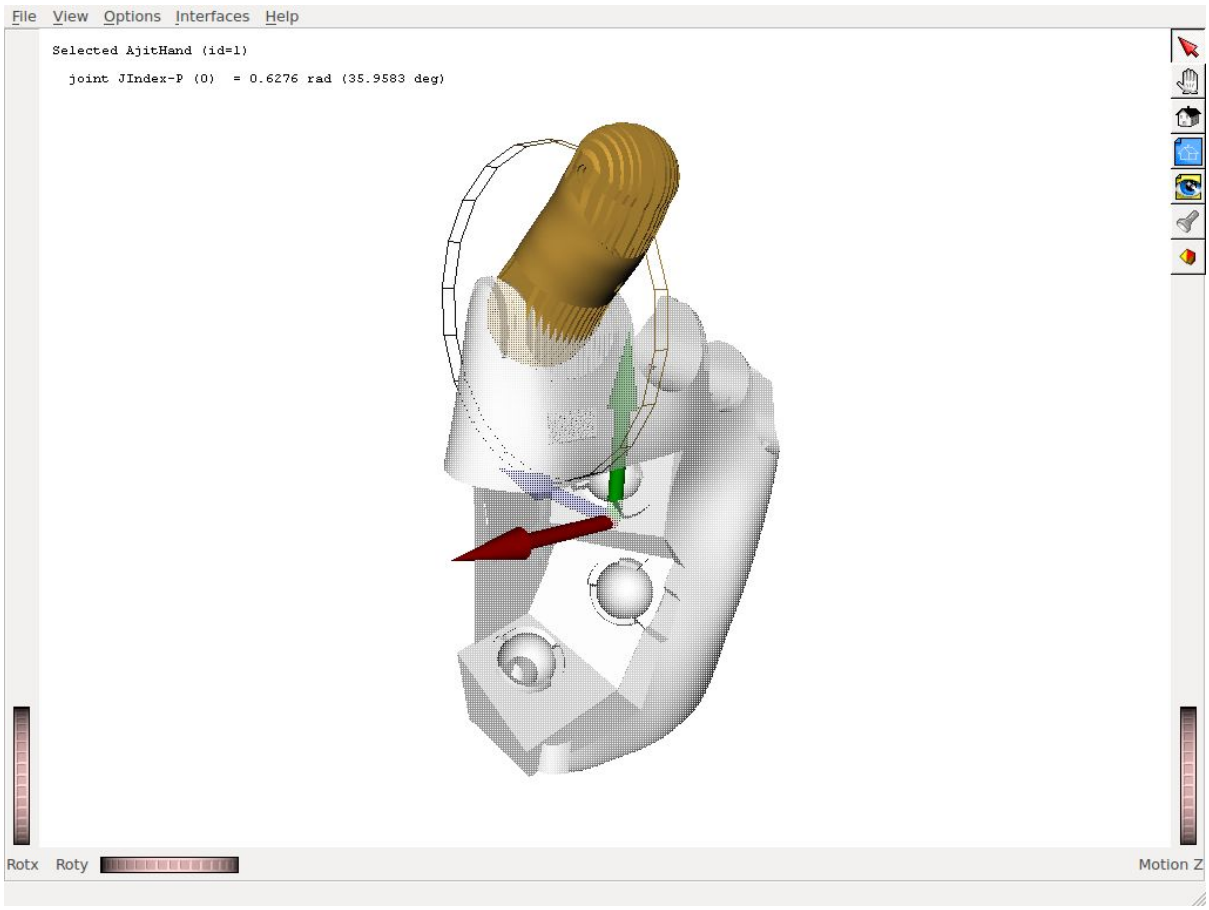
In this case, we also have modified the rotation of the body, 180° around the Z axis and -90° around the Y axis.



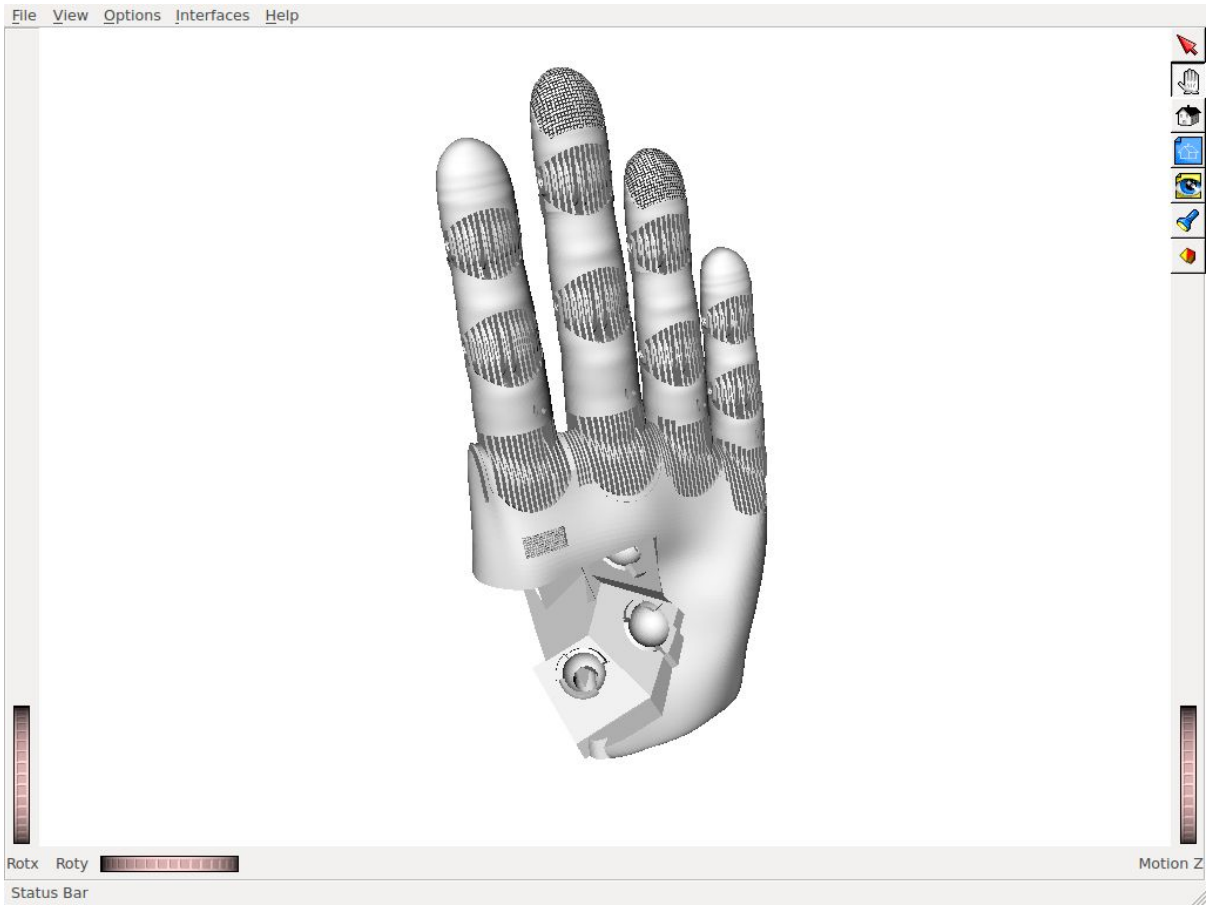
Once again, we have to define the joint coupling the pindex proximal with the index pad

```
<Joint type="hinge" name="JIndex-P" enable="true">
        <Body>Index-Pad</Body>
        <Body>Index-P</Body>
        <offsetfrom>Index-P</offsetfrom>
        <limitsdeg>0 90</limitsdeg>
        <axis>0 0 1</axis>
    </Joint>
```

As this joint is a real one, we have also added to parameters in the joint, one defines the axis uses for this DoF, and the other the range of movement. With our joint defined, we can check everything is fine in OpenRave (ctrl+click on the joint will select it, it allows the user to move the joint around his range)



This is the basic procedure for adding bodies to create a new hand model in OpenRave. After a few steps, we will have a hand model like this:



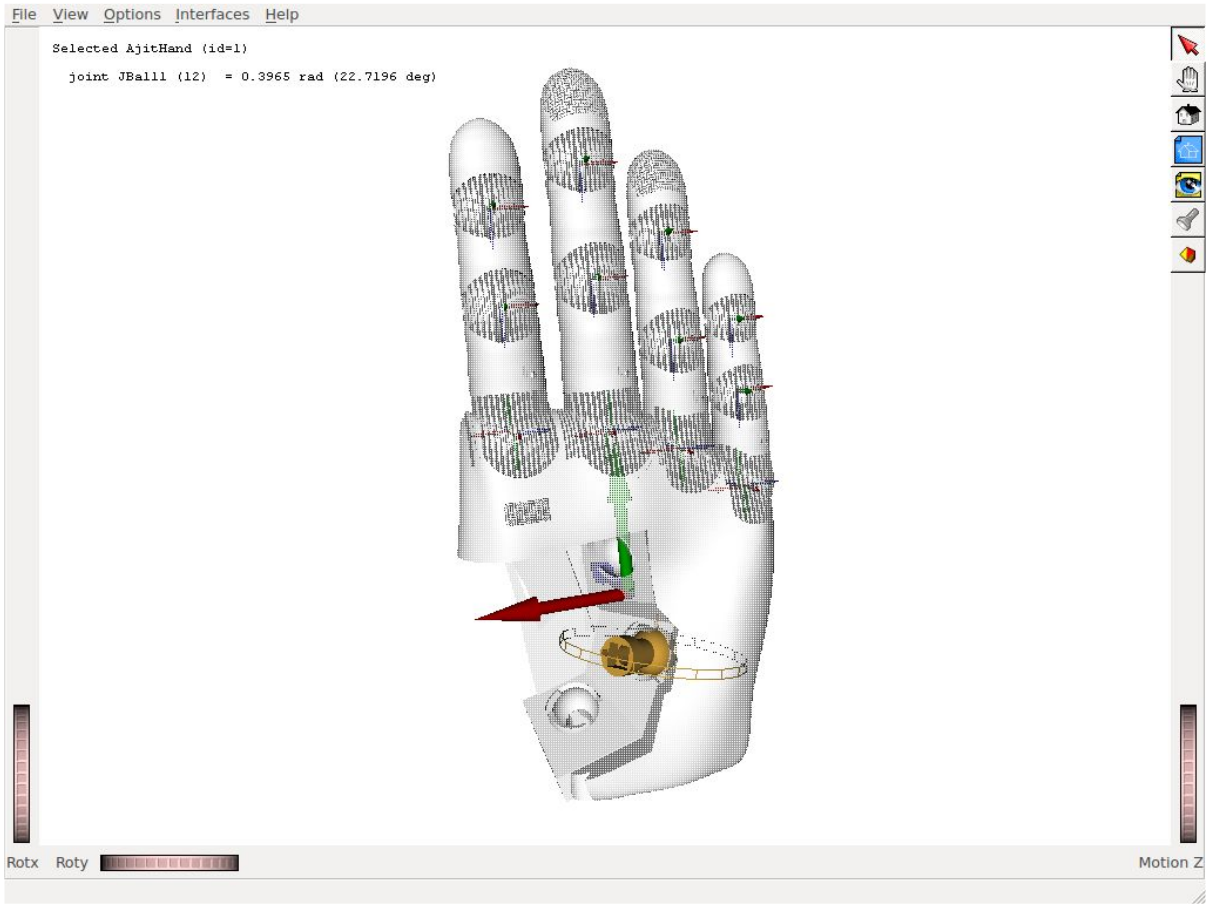
We only have to add now the Thumb. In this case, the Thumb of our hand is a complex mechanism with different joints coupled to create one DoF. However, OpenRave allows to create mimic joints so, with a bit of effort, we can define the right relations between our joints to make the thumb mechanism as accurate as possible.

In the code below, you can check the whole model of the hand with all his joints and DoF, for one, we are going to explain how the mimic joints work.

In this case, we consider the “Ball1” Body as the “main” actuator for the Thumb mechanism, and all the other joints will be moving depending on the value of this “main joint”:

```
<Body name="Ball1" type="dynamic">
        <offsetfrom>handbase</offsetfrom>
        <Translation>-0.005 -0.019 -0.004</Translation>
        <rotationaxis>0 0 1 105</rotationaxis>
        <rotationaxis>0 1 0 15</rotationaxis>
        <Geom type="trimesh" modifiable="false">
                <data>ajit/ball_1.stl 1</data>
                <Render>ajit/ball_1.stl 1</Render>
    </Geom>
</Body>
```

```
<Joint type="hinge" name="JBall1">
    <Body>handbase</Body>
    <Body>Ball1</Body>
    <axis>1 0 0</axis>
    <limitsdeg>0 45</limitsdeg>
    <offsetfrom>Ball1</offsetfrom>
</Joint>
```



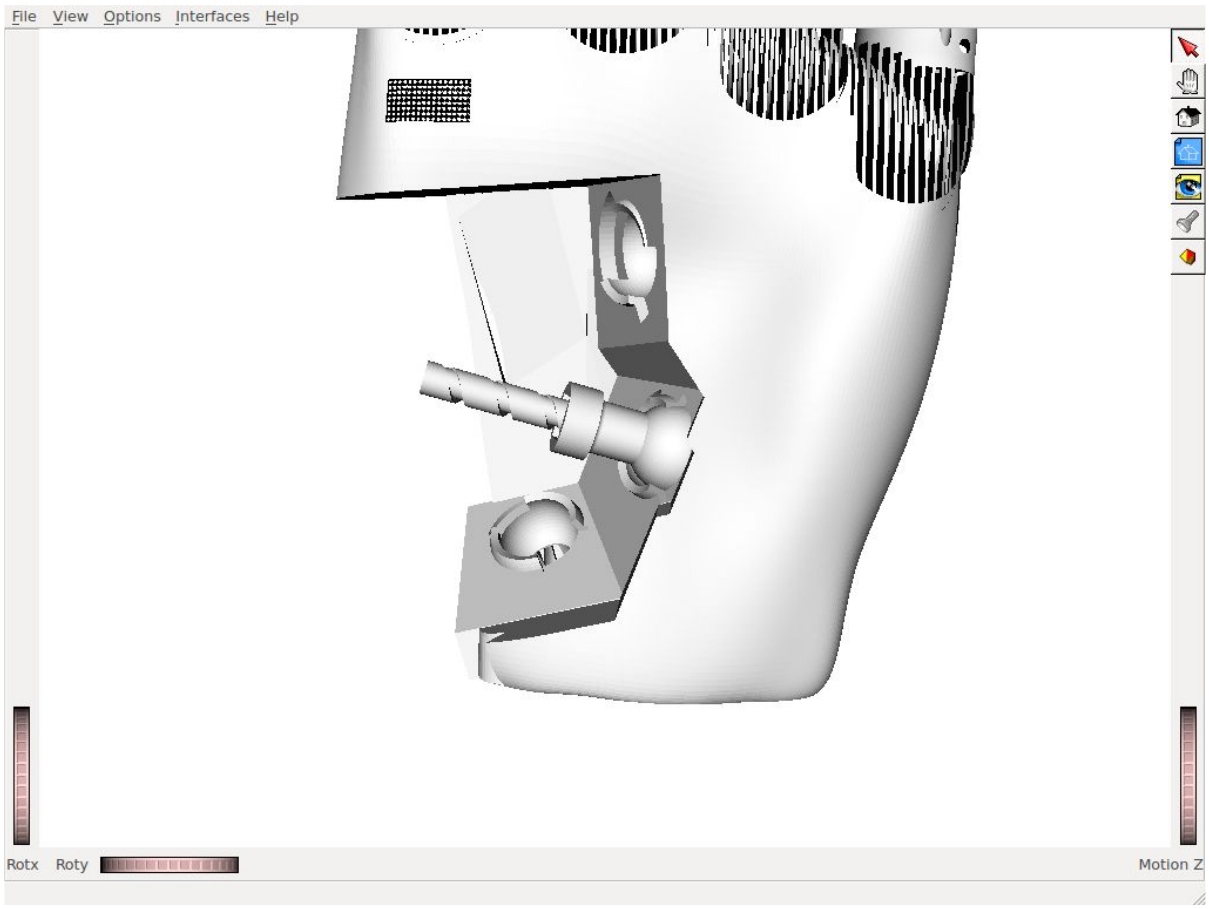
The next body will be the screw. NOTE that in this case, the joint has been defined as slider, which the most similar movement to the Screw one.

```
<Body name="LScrew1" type="dynamic">
    <offsetfrom>Ball1</offsetfrom>
    <Translation>0.0 -0.03 0.0</Translation>
    <rotationaxis>0 0 1 180</rotationaxis>
    <rotationaxis>0 1 0 0 </rotationaxis>
    <Geom type="trimesh" modifiable="false">
            <data>ajit/leadscrew_1.stl 1</data>
            <Render>ajit/leadscrew_1.stl 1</Render>
    </Geom>
</Body>
```

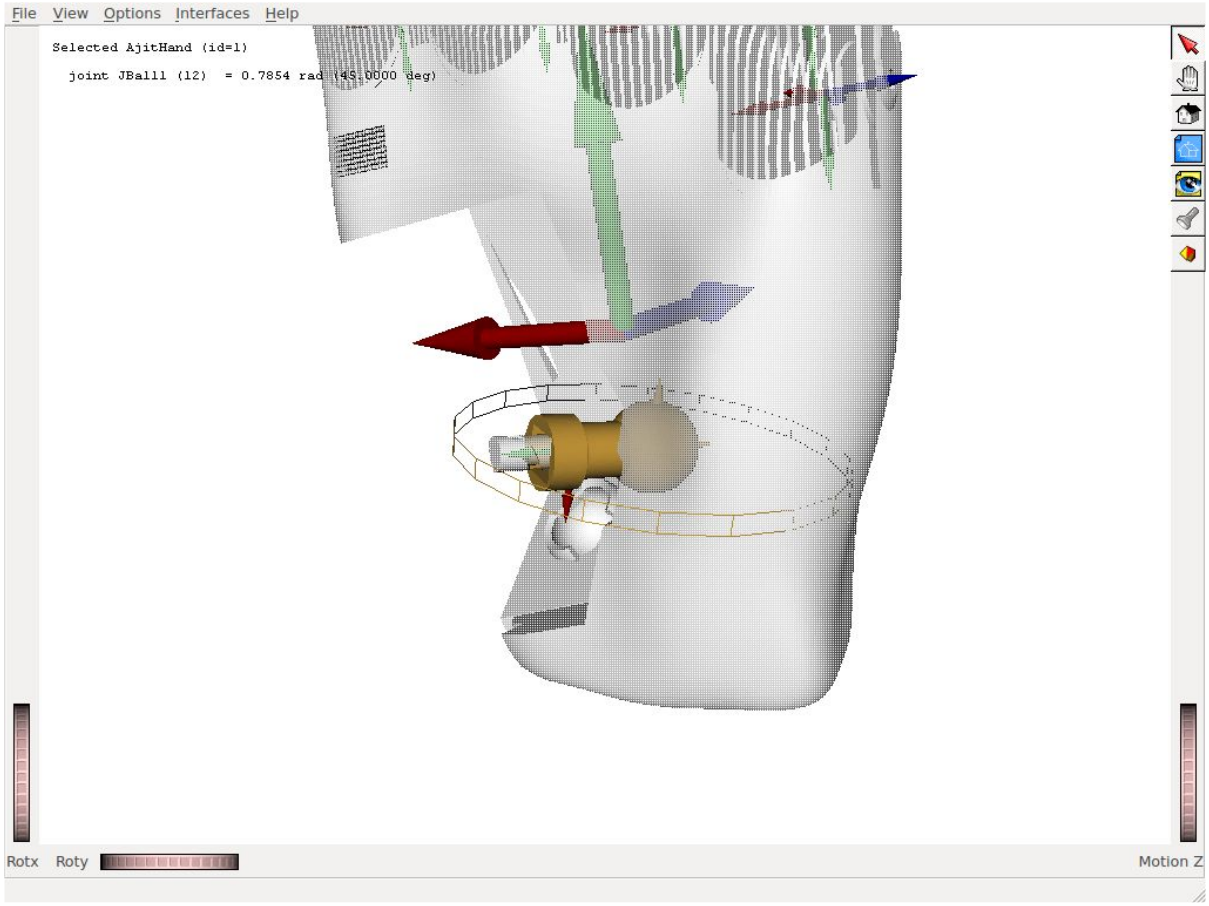
```
<Joint type="slider" name="Jlead-1" mimic_pos="JBall1*0.02" mimic_vel="|JBall1 1">
        <Body>Ball1</Body>
        <Body>LScrew1</Body>
        <limitsdeg>0 90</limitsdeg>
        <axis>0 -1 0</axis>
        <offsetfrom>LScrew1</offsetfrom>
        <maxvel>1</maxvel>
  </Joint>
```

When defining mimic joints, we have to define allways the mimic_pos and mimic_vel parameters, using always as reference one or more different joint values, this parameters allow also more complex mathematical functions such are sin, cos, etc.

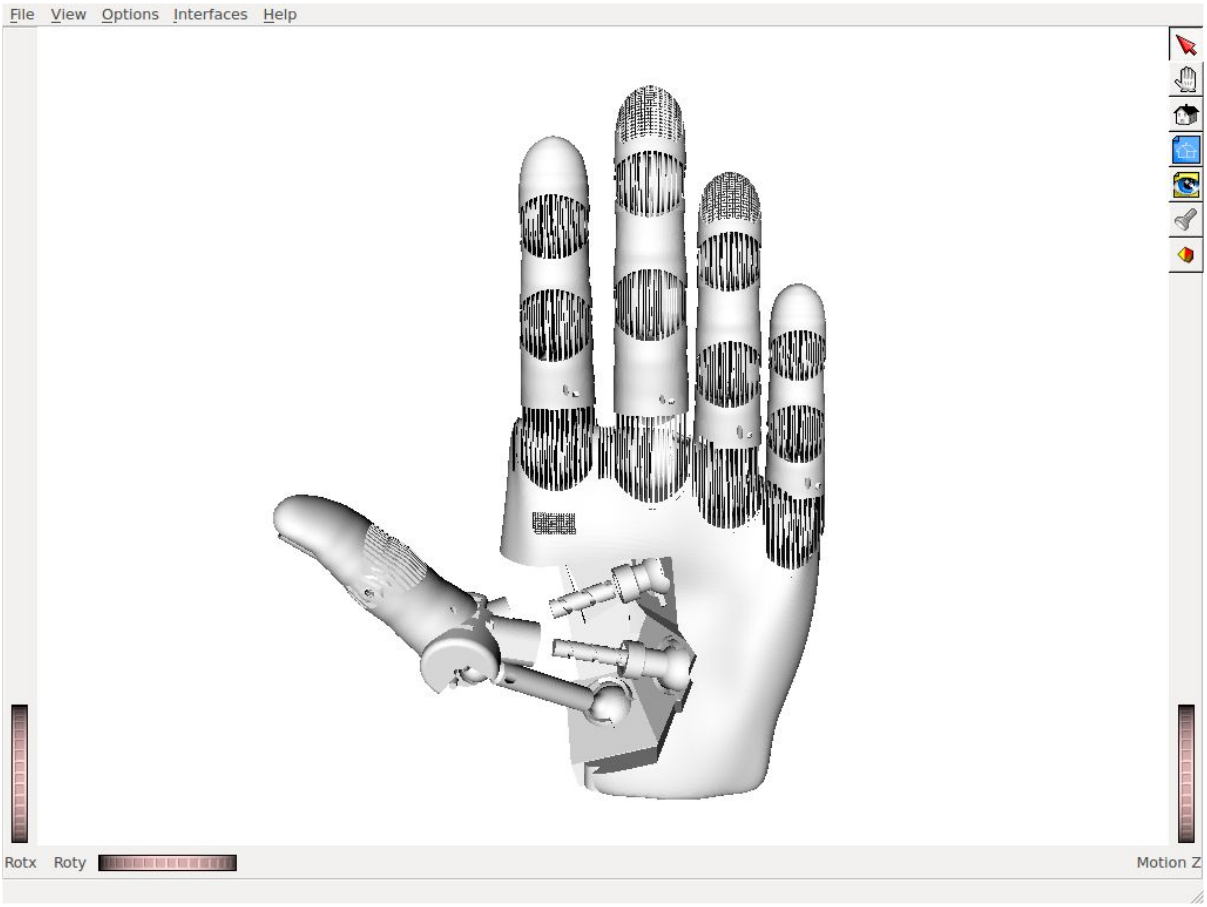
This is how look our joints at pos 0:



And if we rotate the ball to his, the screw slides inwards:



Doing this with all the remaining bodies and joints, we can finally have our hand model.



Creating the Manipulator File

Now, we have our kinbody file, with all the bodies of our hand, his joints and relationships, but still we have to define the manipulator file. This is used for teling Openrave our kinbody file will be a manipulator one, which of the joint of the manip are used for the closure procedure, where is the reference point for the manipulator and in which direction is supposed to close the hand. This information is used later when you try to generate grasps between our hand model and an object.

This is how our manipulator file should look:

```
<Robot>
  <KinBody file="../robots/ajithand.kinbody.xml"/>
  <Manipulator name="hand">
        <base>handbase</base>
        <effector>handbase</effector>
        <Translation>0.0 0.04 -0.022</Translation>
```

<joints> JIndex-P JIndex-M JIndex-D JMiddle-P JMiddle-M JMiddle-D JRing-P JRing-M JRing-D JLittle-P JLittle-M JLittle-D JBall1 JThumb-P JThumb-D</joints>
<closingdirection> 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 </closingdirection>
<direction> 0 0 -1</direction>
</Manipulator>
</Robot>

First, we define our model as a “Robot” this will tell OpenRave this is a special type of body for our environment, then we assign the kinbody file used for modeling our “robot”.

Then we define a “Manipulator” inside our robot, we assign which will be de base body of manipulator and the (if you remember, the palm body was named handbase at the start) and the endeffector (this is used as the point of reference when positioning the manipulator for generating grasps), in this case we put the end effector in the point 0 0.04 -0.022 which means the point of reference for grasping will we on the surface of the Middle-Pad.

Also, we define the joints used in the manipulator, the direction they will close (this is used for the direction of movement in the limitsdegrees parameter) and finally, the direction (axis) in which the grasp are performed. In our case is negative to the Z axis, which is the orientation of the palm.