This lecture will be recorded

064-0026-00L: COMPAS II

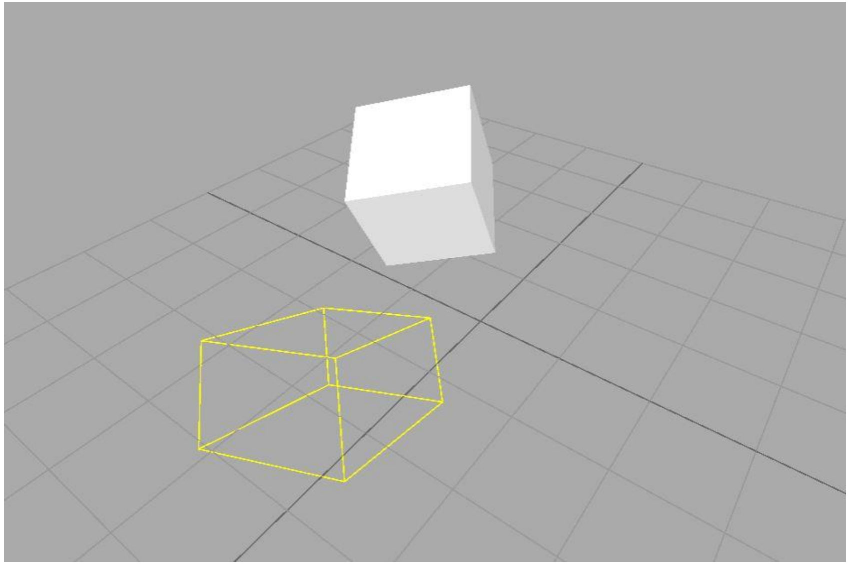
Introduction to Computational Methods for Digital Fabrication in Architecture slides + code

`https://dfab.link/fs2022`

Review of last week's assignment

Project box to xy-plane

1. Create a box at a certain location with a certain orientation.
2. Create a `Projection` (can be orthogonal, parallel or perspective)
3. Convert the box to a mesh and project the it onto the xy-plane.
4. Use artists to draw the result



TODAY

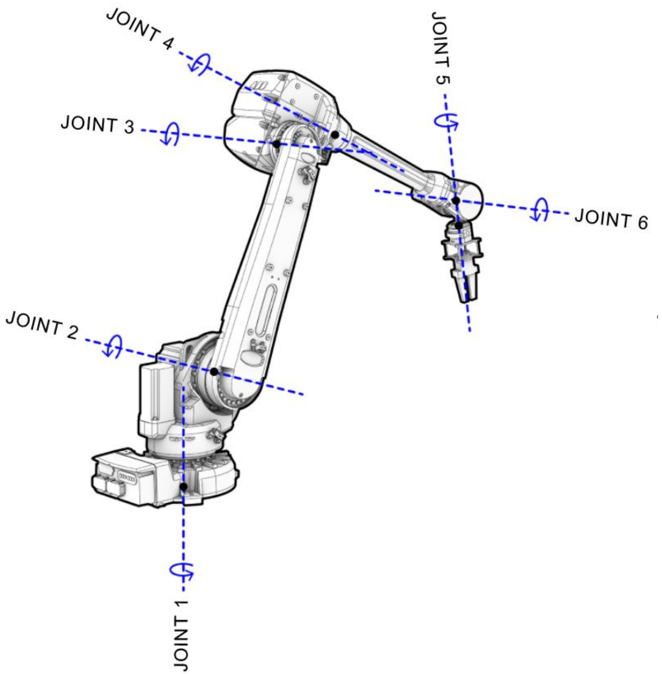
robot models

forward kinematics

inverse kinematics

Today's goal

Understand **how to represent a robot** in COMPAS robot models



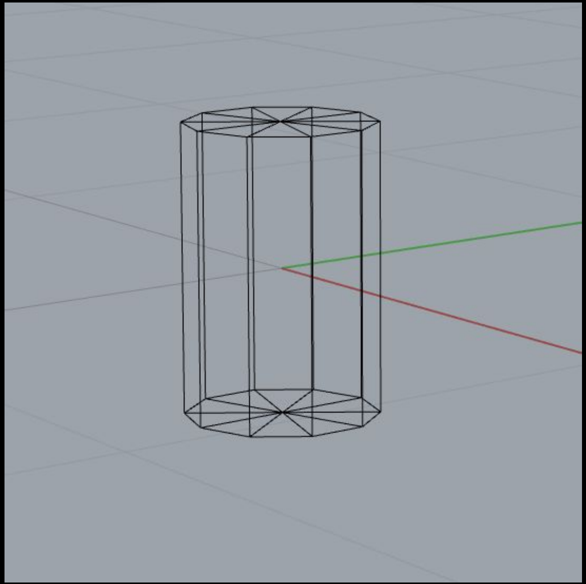
URDF format

Tree structure

Open source

```xml
<?xml version="1.0"?>
<robot name="myfirst">
 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>
</robot>
```

```xml
<?xml version="1.0"?>
<robot name="myfirst">
 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>
</robot>
```

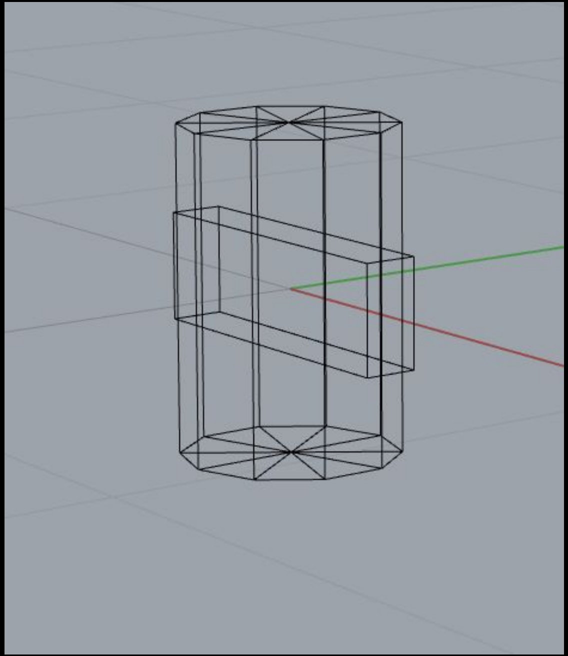


Visualize model

```python
from compas.artists import Artist
from compas.robots import RobotModel

model = RobotModel.from_urdf_file('models/01_myfirst.urdf')

artist = Artist(model, layer='Robot')
artist.clear_layer()
artist.draw_visual()
artist.redraw()
```

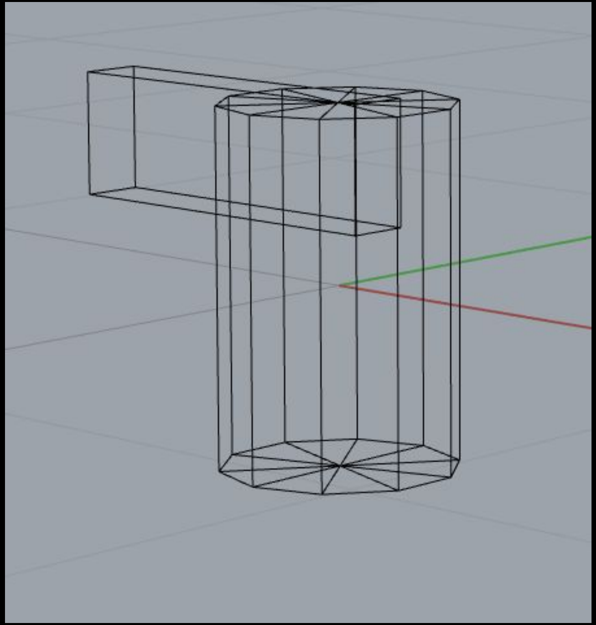
```xml
<?xml version="1.0"?>
<robot name="multipleshapes">

 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <box size="0.6 0.1 0.2"/>
     </geometry>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
 </joint>

</robot>
```

```xml
<?xml version="1.0"?>
<robot name="multipleshapes">

 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <box size="0.6 0.1 0.2"/>
     </geometry>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
 </joint>

</robot>
```



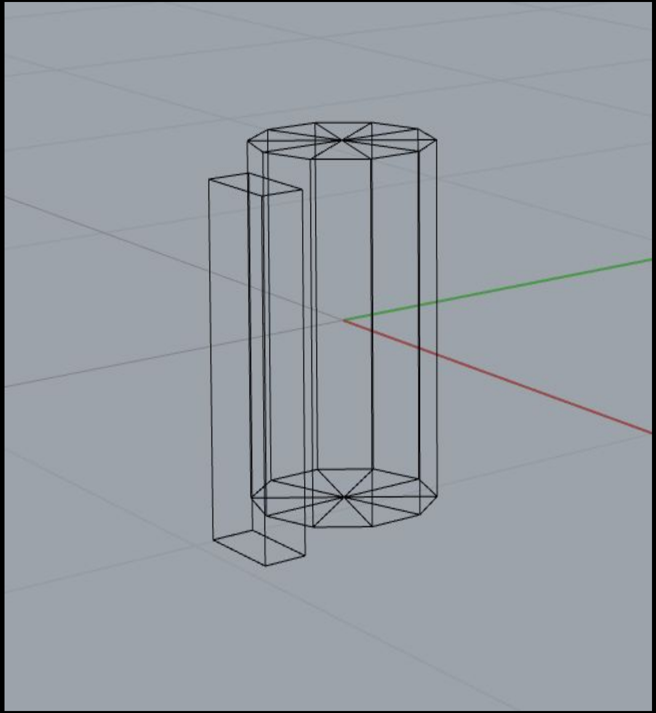
```xml
<?xml version="1.0"?>
<robot name="origins">

 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <box size="0.6 0.1 0.2"/>
     </geometry>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
   <origin xyz="0 -0.22 0.25"/>
 </joint>

</robot>
```

```xml
<?xml version="1.0"?>
<robot name="origins">

 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <box size="0.6 0.1 0.2"/>
     </geometry>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
   <origin xyz="0 -0.22 0.25"/>
 </joint>

</robot>
```

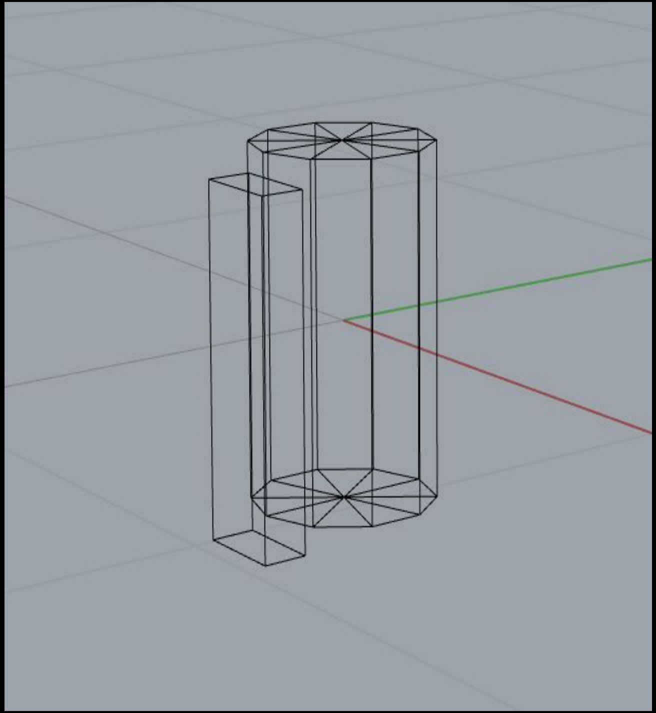


```xml
<?xml version="1.0"?>
<robot name="origins">

 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <box size="0.6 0.1 0.2"/>
     </geometry>
     <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
   <origin xyz="0 -0.22 0.25"/>
 </joint>

</robot>
```

```xml
<?xml version="1.0"?>
<robot name="origins">

 <link name="base_link">
   <visual>
     <geometry>
       <cylinder length="0.6" radius="0.2"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <box size="0.6 0.1 0.2"/>
     </geometry>
     <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
   <origin xyz="0 -0.22 0.25"/>
 </joint>

</robot>
```



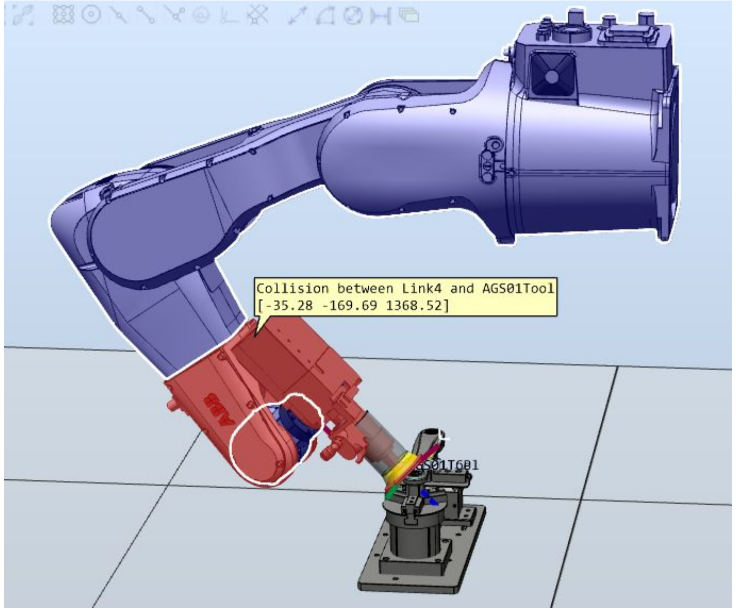
```xml
<?xml version="1.0"?>
<robot name="origins">

 <link name="base_link">
   <visual>
     <geometry>
       <mesh filename="package://basic/cylinder.obj"/>
     </geometry>
   </visual>
 </link>

 <link name="right_leg">
   <visual>
     <geometry>
       <mesh filename="package://basic/box.obj"/>
     </geometry>
     <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
   </visual>
 </link>

 <joint name="base_to_right_leg" type="fixed">
   <parent link="base_link"/>
   <child link="right_leg"/>
   <origin xyz="0 -0.22 0.25"/>
 </joint>

</robot>
```



Visualize model with meshes

```python
from compas.artists import Artist
from compas.robots import LocalPackageMeshLoader
from compas.robots import RobotModel

model = RobotModel.from_urdf_file('models/05_origins_meshes.urdf')

loader = LocalPackageMeshLoader('models', 'basic')
model.load_geometry(loader)

artist = Artist(model, layer='Robot')
artist.clear_layer()
artist.draw_visual()
artist.redraw()
```

Collision checking



Source: https://forums.robotstudio.com/discussion/10611/how-to-generate-collision-free-path-with-powerpacs

Use different visual/collision geometry

Use bounding volumes

Use primitives

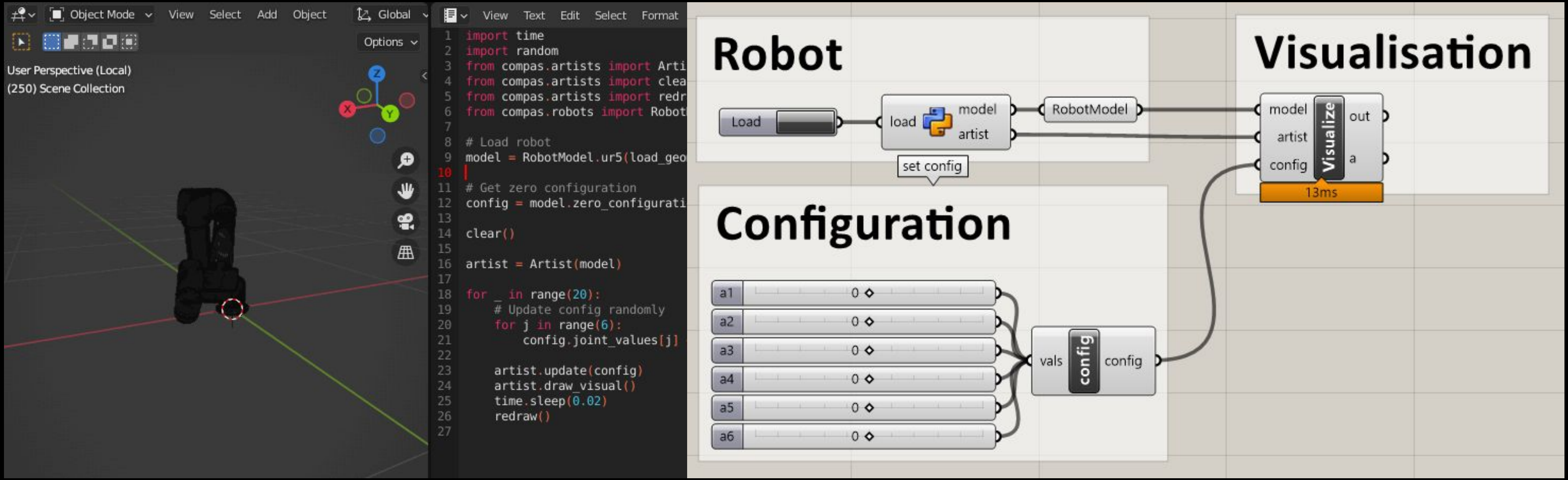
Load local model

```
from compas.artists import Artist
from compas.robots import RobotModel

model = RobotModel.ur5(load_geometry=True)

artist = Artist(model, layer='Robot')
artist.clear_layer()
artist.draw_visual()
artist.redraw()
```

Load Github model

```python
from compas.robots import GithubPackageMeshLoader
from compas.robots import RobotModel

# Select Github repository, package and branch where the model is stored
r = 'ros-industrial/abb'
p = 'abb_irb6600_support'
b = 'kinetic-devel'

github = GithubPackageMeshLoader(r, p, b)
urdf = github.load_urdf('irb6640.urdf')

# Create robot model from URDF
model = RobotModel.from_urdf_file(urdf)
print(model)
```

Updating model visualization



Loading external models

```
with RosClient("localhost") as ros:
    robot = ros.load_robot(load_geometry=True)
    robot.info()

    artist = Artist(robot.model)
    artist.draw_visual()
```

Building your own robot

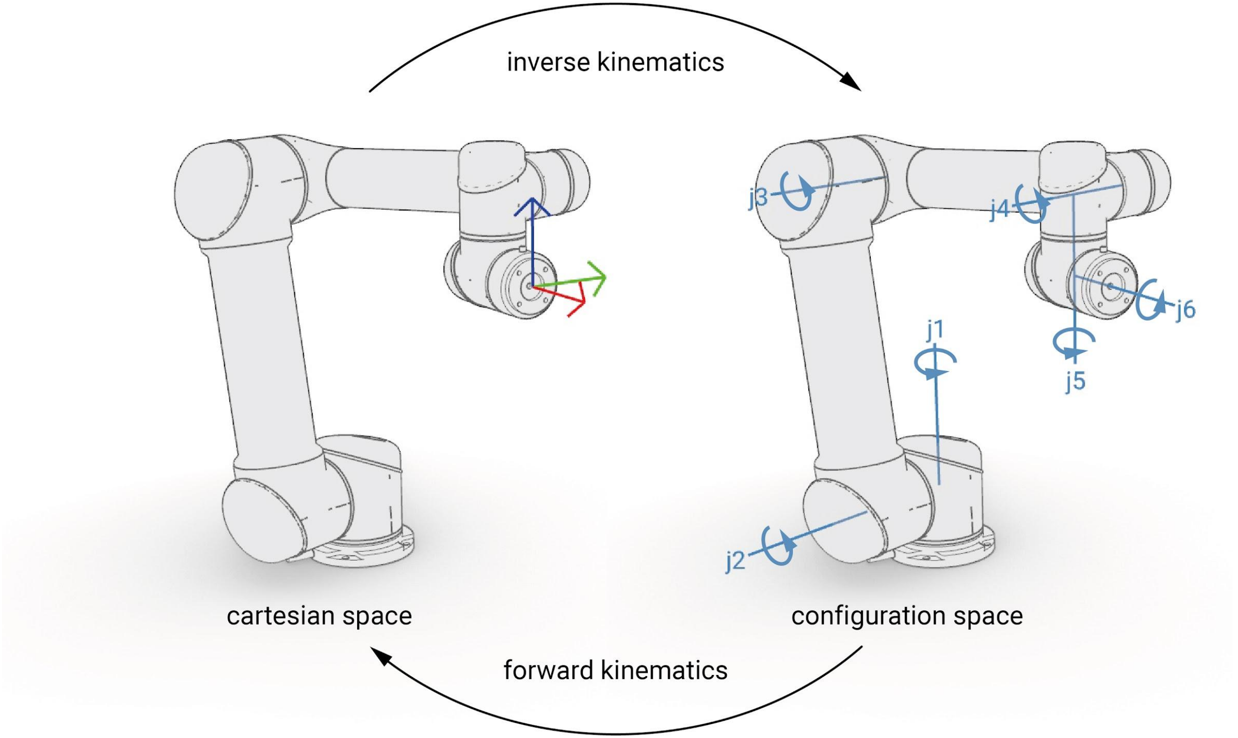
```
model = RobotModel('ur10e',
              joints=[
                  Joint('shoulder_pan_joint', 'revolute', parent='base_link', child='shoulder_link'),
                  Joint('shoulder_lift_joint', 'revolute', parent='shoulder_link', child='upper_arm_link'),
                  Joint('elbow_joint', 'revolute', parent='upper_arm_link', child='forearm_link'),
                  Joint('wrist_1_joint', 'revolute', parent='forearm_link', child='wrist_1_link'),
                  Joint('wrist_2_joint', 'revolute', parent='wrist_1_link', child='wrist_2_link'),
                  Joint('wrist_3_joint', 'revolute', parent='wrist_2_link', child='wrist_3_link'),
              ], links=[
                  Link('base_link'), Link('shoulder_link'), Link('upper_arm_link'), Link('forearm_link'),
                  Link('wrist_1_link'), Link('wrist_2_link'), Link('wrist_3_link'),
              ])
print(model)
```

Building your own robot

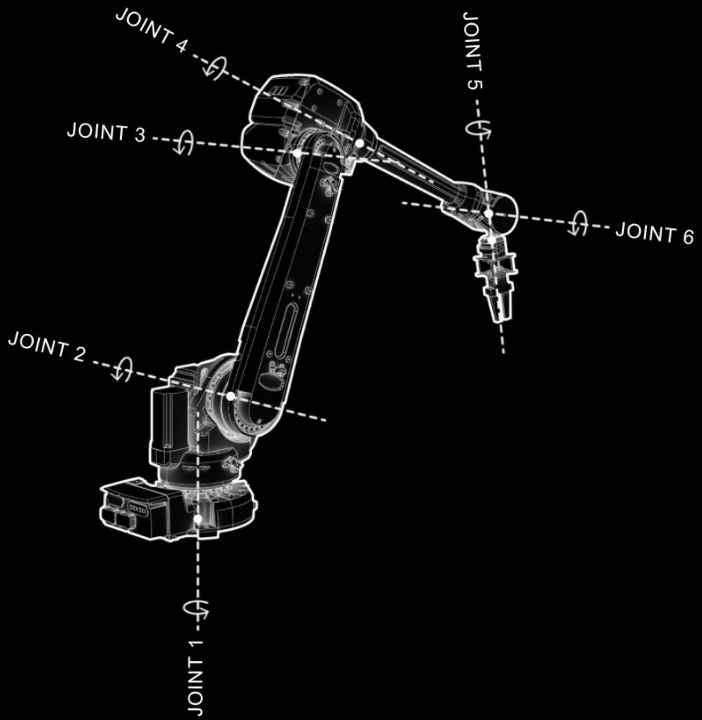
```python
# create robot model
model = RobotModel("robot", links=[], joints=[])

# add links
link0 = model.add_link("world")
link1 = model.add_link("link1", visual_mesh=mesh1)
link2 = model.add_link("link2", visual_mesh=mesh2)

# add the joints between the links
model.add_joint("joint1", Joint.CONTINUOUS, link0, link1, origin, axis)
model.add_joint("joint2", Joint.CONTINUOUS, link1, link2, origin, axis)
```

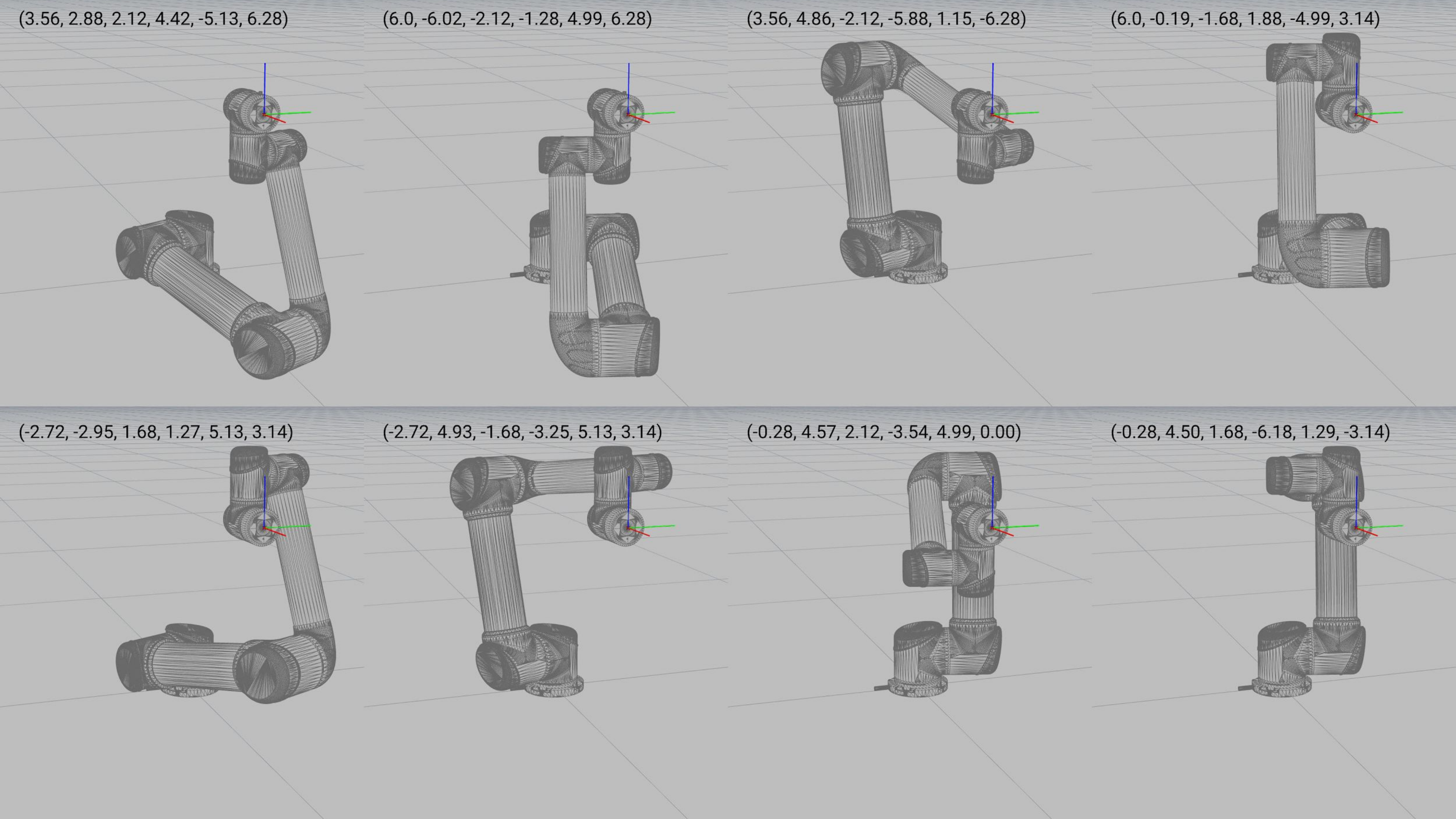
kinematics

Joint vs Cartesian space



Configuration




(3.56, 2.88, 2.12, 4.42, -5.13, 6.28)
(6.0, -6.02, -2.12, -1.28, 4.99, 6.28)
(3.56, 4.86, -2.12, -5.88, 1.15, -6.28)
(6.0, -0.19, -1.68, 1.88, -4.99, 3.14)
(-2.72, -2.95, 1.68, 1.27, 5.13, 3.14)
(-2.72, 4.93, -1.68, -3.25, 5.13, 3.14)
(-0.28, 4.57, 2.12, -3.54, 4.99, 0.00)
(-0.28, 4.50, 1.68, -6.18, 1.29, -3.14)

Forward Kinematics

```
# Create config
config = model.zero_configuration()

# Get FK for tip
print (model.forward_kinematics(config))
# Get FK for base
print (model.forward_kinematics(config, link_name=model.get_base_link_name()))
```

Inverse Kinematics

```python
from compas_fab.backends.kinematics.solvers import UR5Kinematics

f = Frame((0.417, 0.191, -0.005), (-0.000, 1.000, 0.00), (1.000, 0.000, 0.000))
solutions = UR5Kinematics().inverse(f)
```

Forward Kinematics

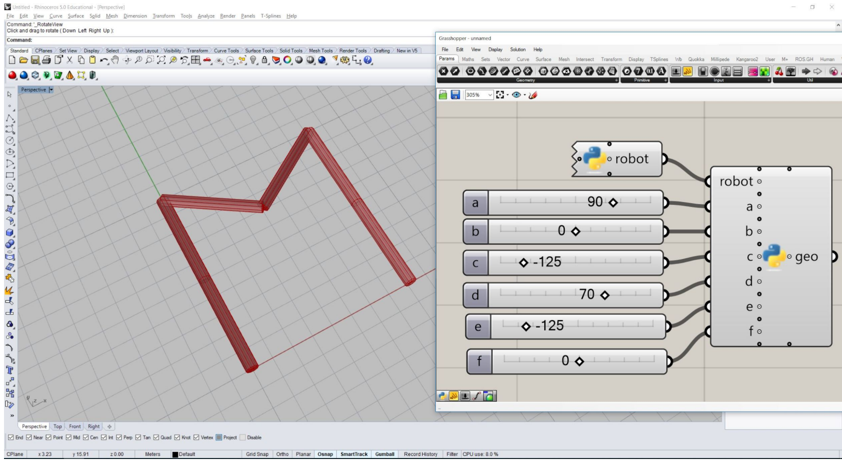
```
with RosClient('localhost') as client:
    robot = client.load_robot()
    config = model.zero_configuration()

    frame_WCF = robot.forward_kinematics(configuration)
```

Inverse Kinematics

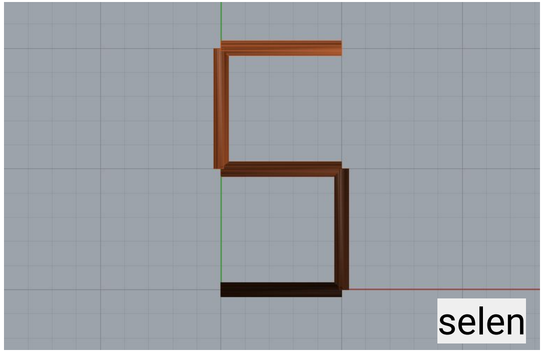
```python
from compas.geometry import Frame
from compas_fab.backends import RosClient

with RosClient('localhost') as client:
    robot = client.load_robot()

    frame_WCF = Frame([0.3, 0.1, 0.5], [1, 0, 0], [0, 1, 0])
    start_configuration = robot.zero_configuration()

    configuration = robot.inverse_kinematics(frame_WCF, start_configuration)
```

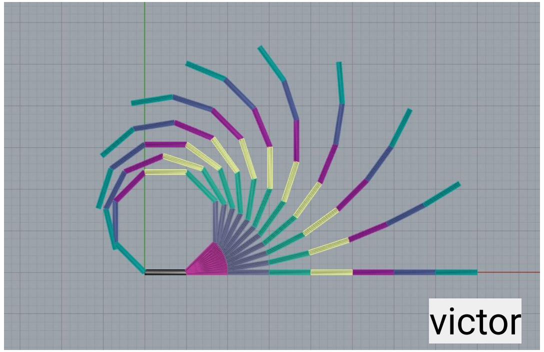
Assignment

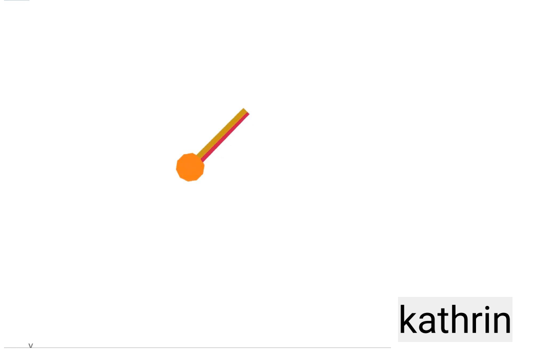
1. Build your own robot with a certain number n of links and n - 1 configurable joints.
2. Create a `Configuration` with certain values and the correct joint types.
3. Create a `Artist`.
4. Use the artist to `update` the robot with the created configuration, such that it configures into the letter of your choice (or any other identifiable figure).

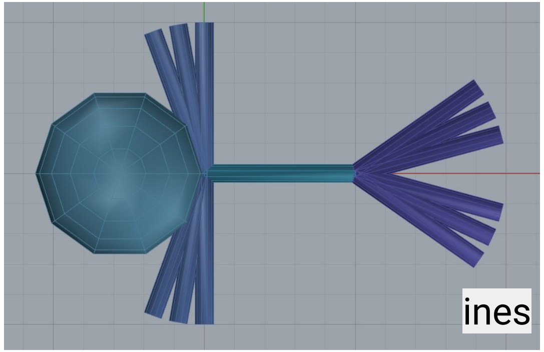


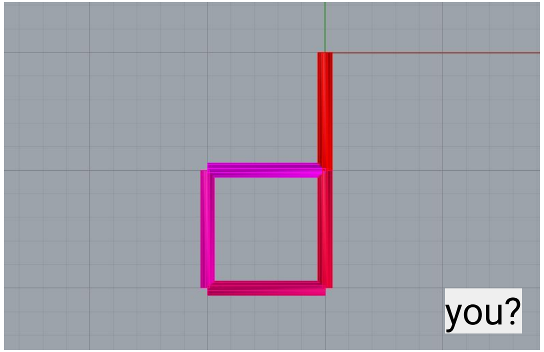
Robot gallery

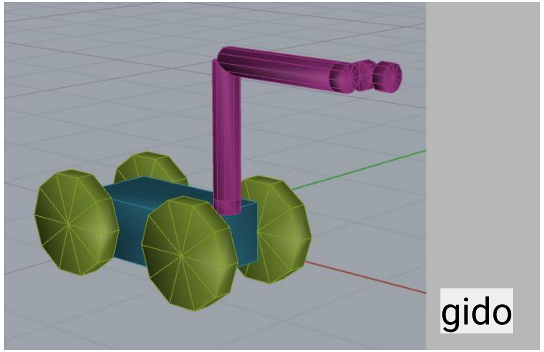












Next week

- Assignment submission due: Wed 16th March, 9AM.
- Ask for help if needed: Slack, Forum, Office Hours (Fridays, request via Slack)
- Next week:
    - Robot backends: ROS

Thanks!