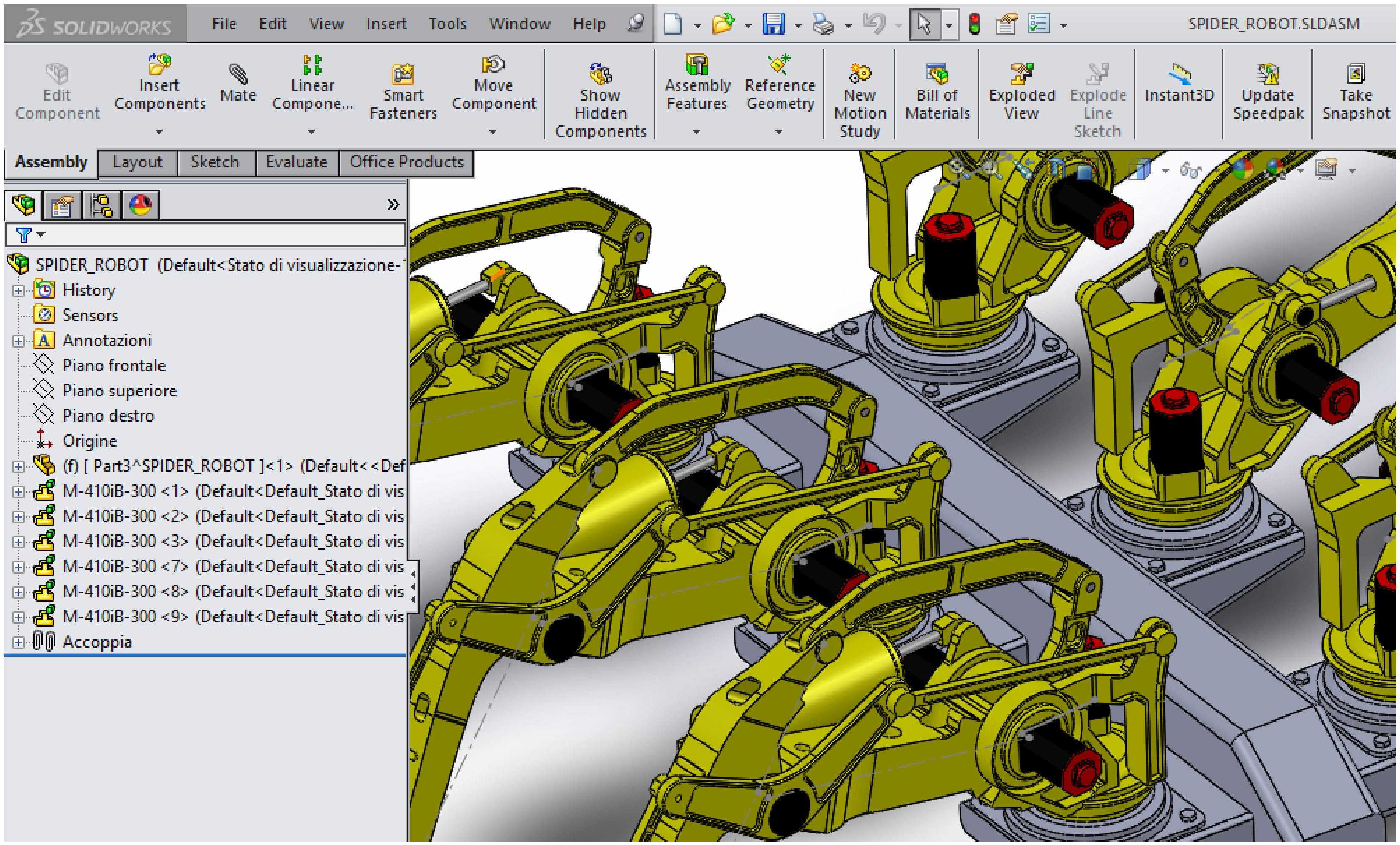
Chrono Support for Modeling Robotic Systems

A bio-inspired robot, using Chrono::Solidworks

Robotics: an example of workflow

- Use SolidWorks 3D CAD to model the robot
- Export the 3D CAD model using the Chrono::Solidworks add-in as a Python model
- Import the Python model
  - … in your C++ program (using the PYPARSER module in Chrono::Engine API)
  - … in your PYTHON program (using the Chrono::PyEngine units)
- Add actuators or additional parts/links using Chrono
- Impose motion to actuators
- Simulate the mechanism

Use SolidWorks CAD to model a bio-inspired robot

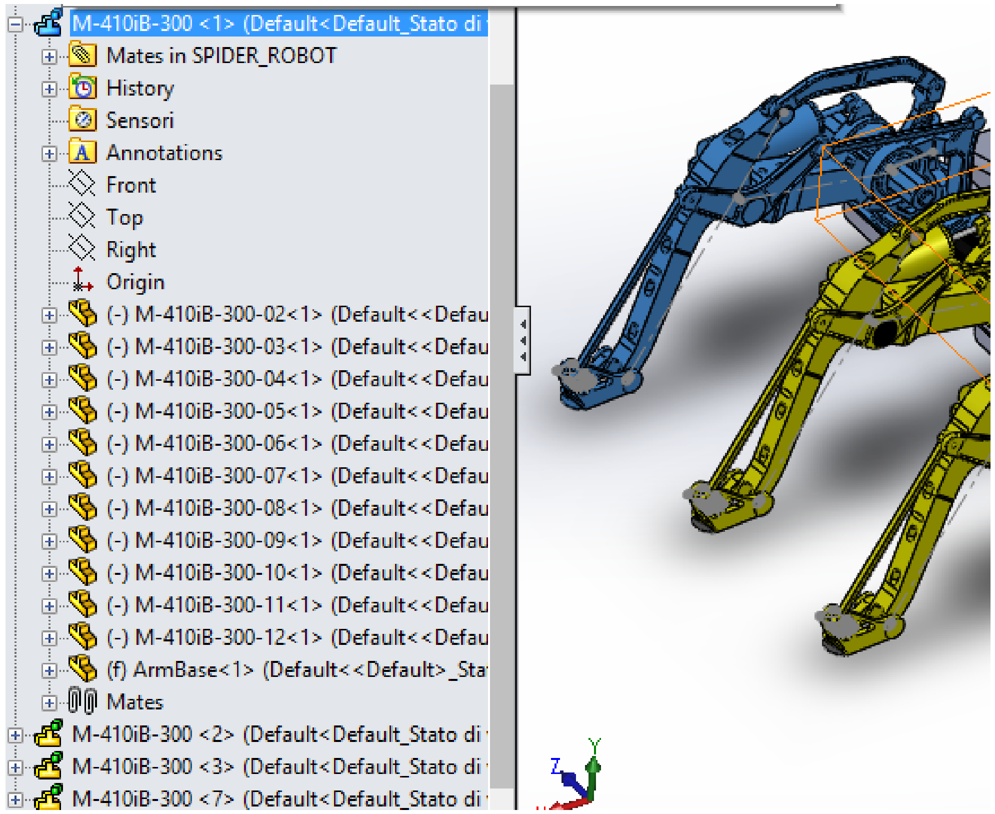


SPIDER_ROBOT (Default<Stato di visualizzazione-
History
Sensors
Annotazioni
Piano frontale
Piano superiore
Piano destro
Origine
(f) [Part3^SPIDER_ROBOT]<1> (Default<<Def
M-410iB-300 <1> (Default<Default_Stato di vis
M-410iB-300 <2> (Default<Default_Stato di vis
M-410iB-300 <3> (Default<Default_Stato di vis
M-410iB-300 <7> (Default<Default_Stato di vis
M-410iB-300 <8> (Default<Default_Stato di vis
M-410iB-300 <9> (Default<Default_Stato di vis
Accoppia

SolidWorks - rigid bodies

The Chrono::SolidWorks Add-in supports:

- Assemblies of parts
  - Ex: the robot arms
- Sub-assemblies of assembles
  - Ex: the full robot



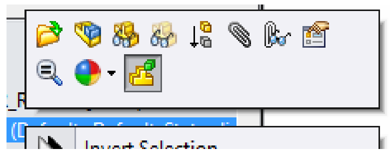
M-410iB-300 <1> (Default<Default_Stato di
Mates in SPIDER_ROBOT
History
Sensori
Annotations
Front
Top
Right
Origin
(-) M-410iB-300-02<1> (Default<<Defau
(-) M-410iB-300-03<1> (Default<<Defau
(-) M-410iB-300-04<1> (Default<<Defau
(-) M-410iB-300-05<1> (Default<<Defau
(-) M-410iB-300-06<1> (Default<<Defau
(-) M-410iB-300-07<1> (Default<<Defau
(-) M-410iB-300-08<1> (Default<<Defau
(-) M-410iB-300-09<1> (Default<<Defau
(-) M-410iB-300-10<1> (Default<<Defau
(-) M-410iB-300-11<1> (Default<<Defau
(-) M-410iB-300-12<1> (Default<<Defau
(f) ArmBase<1> (Default<<Default>_Sta
Mates
M-410iB-300 <2> (Default<Default_Stato di
M-410iB-300 <3> (Default<Default_Stato di
M-410iB-300 <7> (Default<Default_Stato di

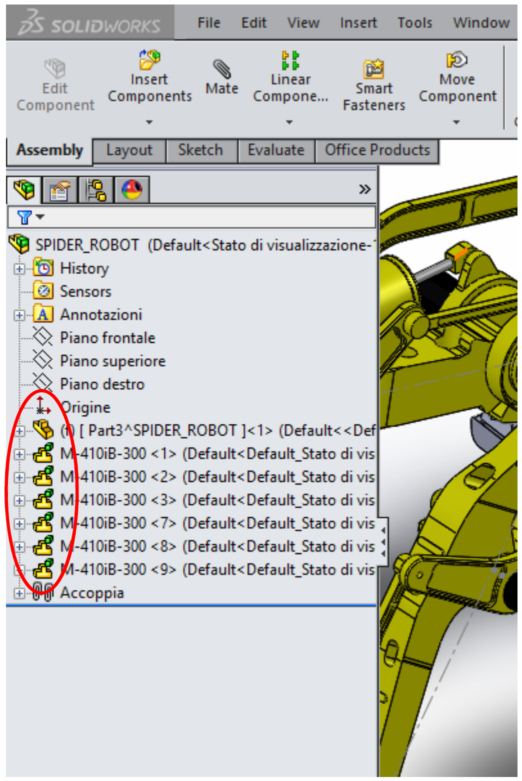
SolidWorks - rigid bodies

Hints:

- A sub-assembly by default in SW is considered a whole rigid body
  - Sub-assemblies like the robot arms would become single ChBody objects on the Chrono side…
  - Inner constraints would not be considered…

- Select the sub-assembly and use this icon of pop-up menu to switch to “flexible” SW assembly:
  - All SW parts in sub-assembly will become separate ChBody objects on the Chrono side
  - Inner constraints will be translated into ChLink constraints on the Chrono side

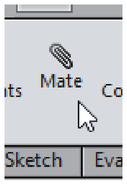


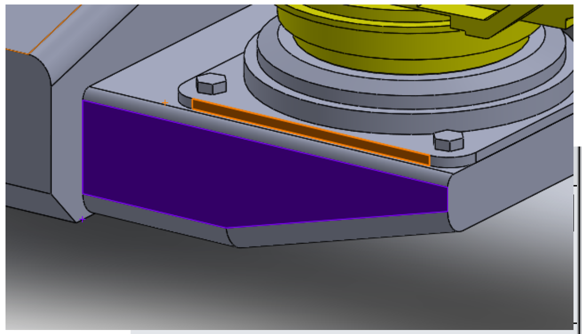

SOLIDWORKS
File
Edit
View
Insert
Tools
Window
Edit Component
Insert Components
Mate
Linear Compone...
Smart Fasteners
Move Component
Assembly
Layout
Sketch
Evaluate
Office Products
SPIDER_ROBOT (Default<Stato di visualizzazione-
History
Sensors
Annotazioni
Piano frontale
Piano superiore
Piano destro
Origine
(f) [Part3^SPIDER_ROBOT]<1> (Default<<Def
M-410iB-300 <1> (Default<Default_Stato di vis
M-410iB-300 <2> (Default<Default_Stato di vis
M-410iB-300 <3> (Default<Default_Stato di vis
M-410iB-300 <7> (Default<Default_Stato di vis
M-410iB-300 <8> (Default<Default_Stato di vis
M-410iB-300 <9> (Default<Default_Stato di vis
Accoppia

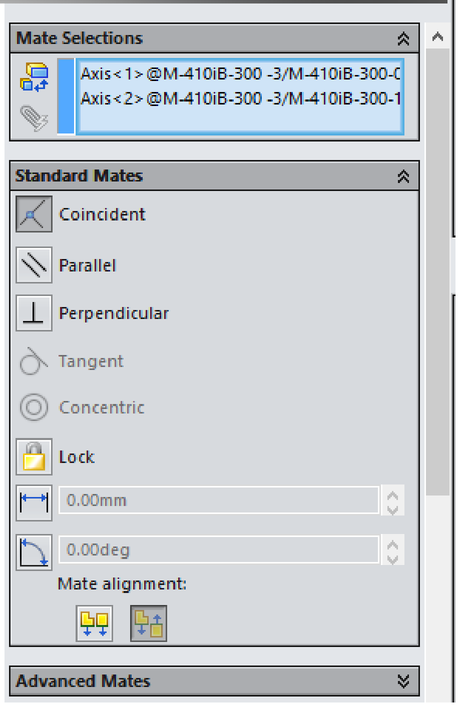
SolidWorks - constraints

- Use the “mates” tool to create constraints in SW
- Most SW “mates” constraints will be translated into ChLink constraints on the Chrono side:
  - Coincident (plane vs plane, plane vs point, edge vs plane, …)
  - Parallel (plane vs plane, plane vs edge, edge vs edge, …)
  - Parpendicular …
  - Distance …
  - Concentric …

- Some SW mates are not yet supported, sorry
  - Advanced mates
  - Tangent etc.







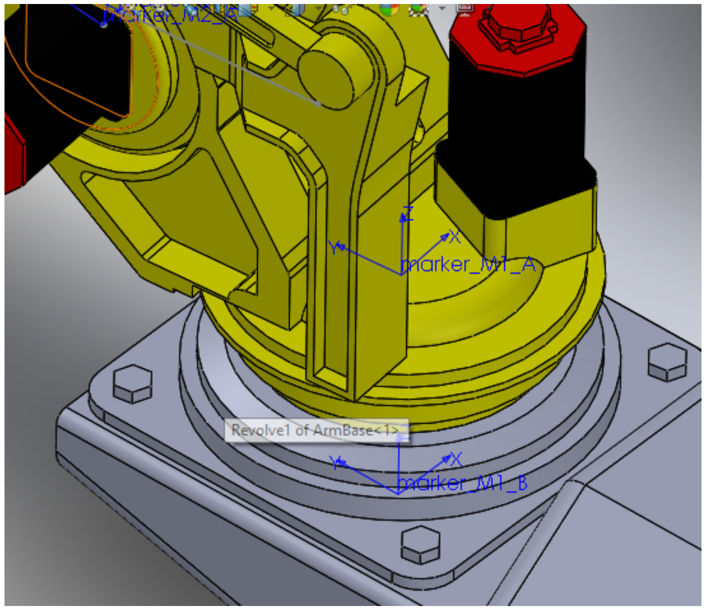
Axis<1>@M-410iB-300 -3/M-410iB-300-C
Axis<2>@M-410iB-300 -3/M-410iB-300-1

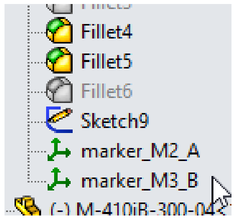
0.00mm

0.00deg

SolidWorks - markers

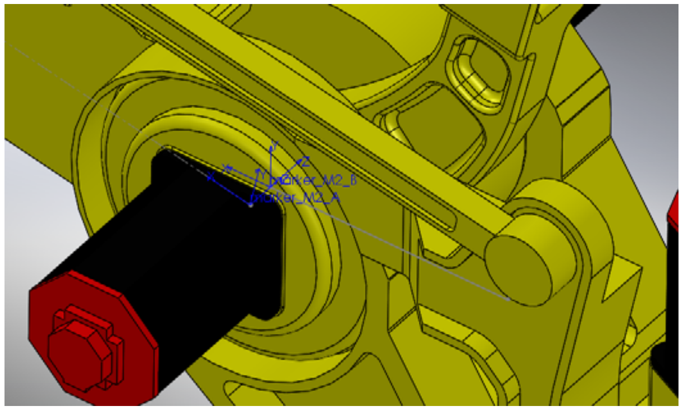
- SolidWorks “coordinate systems” are translated in ChMarker objects on the Chrono side
- This is a useful way to add custom Chrono constraints from the C++ side, later, by referencing pairs of markers
- In our robot example: add pair of markers where you need to create ChLinkEngine objects (with Z axis aligned to the engine shaft):
- Give them a mnemonic name in SW, so it is easier to retrieve them from the C++ side:

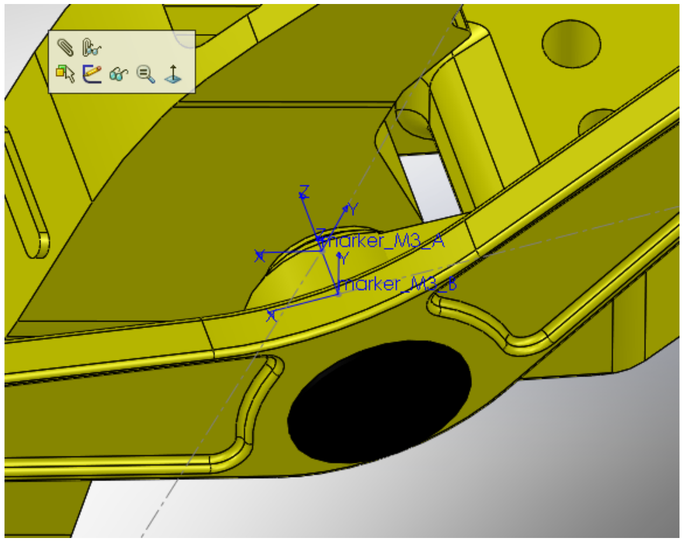




SolidWorks - actuators

- For our spider robot example:
  - Each leg is a 3 DOF arm (like a 'top loader' packaging robot, without gripper)
  - We need 3 actuators (three 'ChLinkEngine') per leg
  - Sorry, the SolidWorks "motor" object is not yet translated into a Chrono ChLinkEngine object (planned for future releases)..
  - ..so we just add couples of markers per each motor, and we'll add the ChLinkEngine objects programmatically, later from the Chrono side
    - Two for the vertical shoulder rotation
    - Two for the up-down rotation
    - Two for the forearm rotation

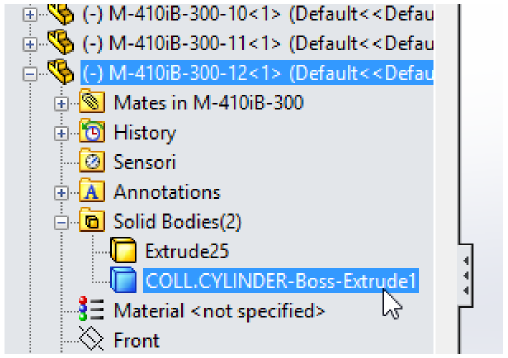




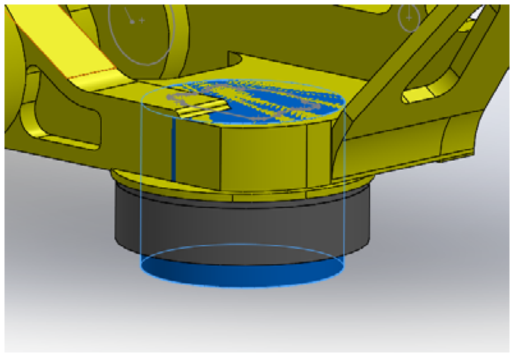
SolidWorks - collision shapes

Collision shapes can be defined in the SolidWorks interface

- Each part of SolidWorks can contain multiple “solid bodies”
- Use one SW “solid body” for the shape (mass, visualization), and create others for collision shapes
- Supported collision shapes:
    - Cylinders
    - Spheres
    - Boxes
    - Convex hulls
    - (compounds of the above)



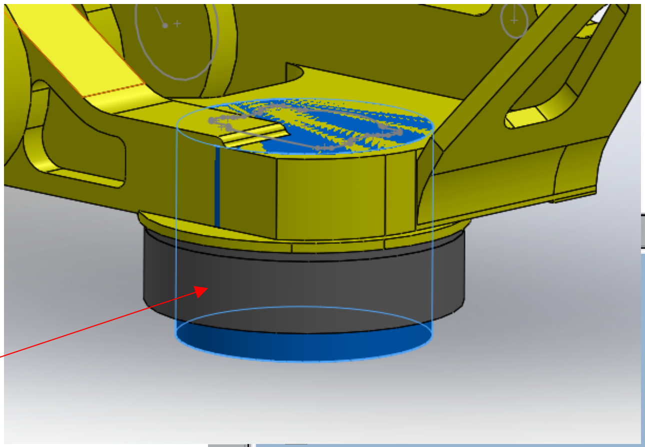
(-) M-410iB-300-10<1> (Default<<Defau
(-) M-410iB-300-11<1> (Default<<Defau
(-) M-410iB-300-12<1> (Default<<Defau
Mates in M-410iB-300
History
Sensori
Annotations
Solid Bodies(2)
Extrude25
COLL.CYLINDER-Boss-Extrude1
Material <not specified>
Front

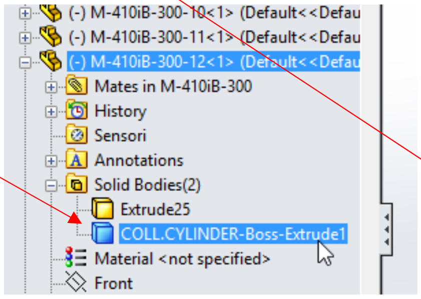


SolidWorks - collision shapes

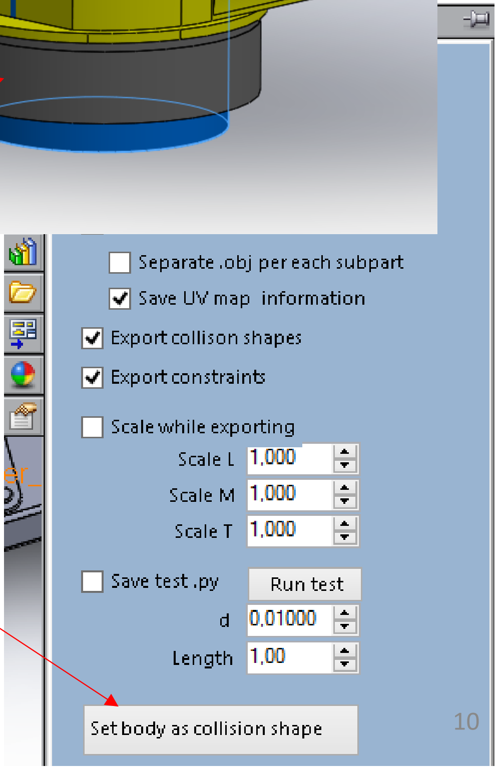
Collision shapes can be defined in the SolidWorks interface

- For the spider robot example: add a collision shape for the foot contact
  - Extrude a cylinder feature in the foot part
  - Do not merge the cylinder feature with the foot shape
  - Select the cylinder and use the “Set body as collision shape” button
  - Look how the name of the cylinder changed:





(-) M-410iB-300-10<1> (Default<<Defau
(-) M-410iB-300-11<1> (Default<<Defau
(-) M-410iB-300-12<1> (Default<<Defau
Mates in M-410iB-300
History
Sensori
Annotations
Solid Bodies(2)
Extrude25
COLL.CYLINDER-Boss-Extrude1
Material <not specified>
Front



- [ ] Separate .obj per each subpart
- [x] Save UV map information
- [x] Export collison shapes
- [x] Export constraints
- [ ] Scale while exporting

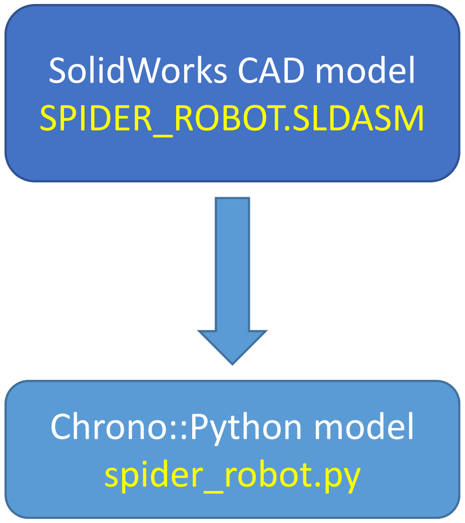
Scale L 1,000
Scale M 1,000
Scale T 1,000

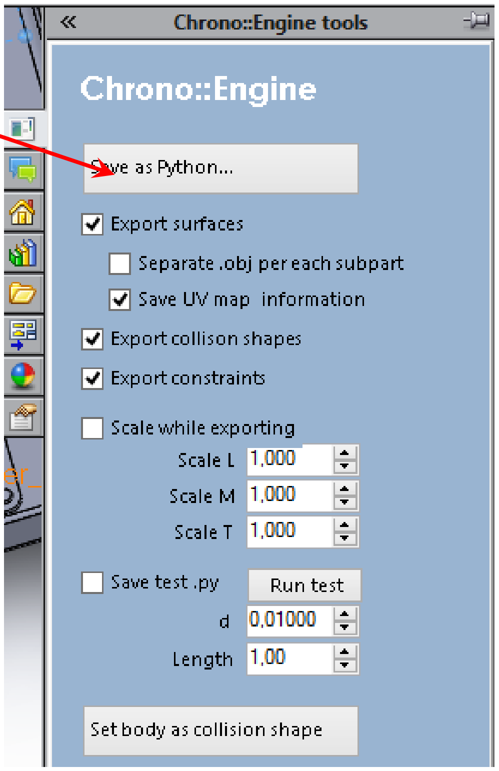
- [ ] Save test .py

d 0,01000
Length 1,00

Export the 3D CAD model to a .py file

- Use the “Save as Python” button to export the model into a .py file for later use in Chrono programs





Chrono::Engine tools

[✓] Export surfaces
[ ] Separate .obj per each subpart
[✓] Save UV map information
[✓] Export collison shapes
[✓] Export constraints
[ ] Scale while exporting

Scale L 1,000
Scale M 1,000
Scale T 1,000

[ ] Save test .py

d 0,01000
Length 1,00

A peek into the spider_robot.py model

```
import ChronoEngine_python_core as chrono
import builtins

shapes_dir = 'spider_robot_shapes/'

if hasattr(builtins, 'exported_system_relpath'):
    shapes_dir = builtins.exported_system_relpath + shapes_dir

exported_items = []

body_0= chrono.ChBodyAuxRef()
body_0.SetName('ground')
body_0.SetBodyFixed(True)
exported_items.append(body_0)

# Rigid body part
body_1= chrono.ChBodyAuxRef()
body_1.SetName('M-410iB-300 -9/M-410iB-300-12-1')
body_1.SetPos(chrono.ChVectorD(-4.07967425969663,1.53452994138092,0.579942165862929))
body_1.SetRot(chrono.ChQuaternionD(-0.0412727252884181,-0.000468569478897146,0.999147807992026,1.93556340954814e-05))
body_1.SetMass(16.1636065115335)
body_1.SetInertiaXX(chrono.ChVectorD(0.270733820597613,0.400877809867468,0.427371867141468))
body_1.SetInertiaXY(chrono.ChVectorD(0.0574221068634287,0.0377110505251339,-0.0625539343672037))
body_1.SetFrame_COG_to_REF(chrono.ChFrameD(chrono.ChVectorD(-0.0294847451383035,0.131793864973377,-0.039955675141979),chrono.ChQuaternionD(1,0,0,0)))

# Visualization shape
body_1_1_shape = chrono.ChObjShapeFile()
body_1_1_shape.SetFilename(shapes_dir +'body_1_1.obj')
body_1_1_level = chrono.ChAssetLevel()
body_1_1_level.GetFrame().SetPos(chrono.ChVectorD(0,0,0))
body_1_1_level.GetFrame().SetRot(chrono.ChQuaternionD(1,0,0,0))
body_1_1_level.GetAssets().push_back(body_1_1_shape)
body_1.GetAssets().push_back(body_1_1_level)

# Collision shapes
body_1.GetCollisionModel().ClearModel()
mr = chrono.ChMatrix33D()
mr[0,0]=0; mr[1,0]=0; mr[2,0]=-1
mr[0,1]=0; mr[1,1]=-1; mr[2,1]=0
mr[0,2]=-1; mr[1,2]=0; mr[2,2]=0
body_1.GetCollisionModel().AddCylinder(0.0891514706007005,0.0891514706007005,0.0737653824400992,chrono.ChVectorD(4.57739150408006E-17,0.0737653824400992,0),mr)
body_1.GetCollisionModel().BuildModel()
body_1.SetCollide(True)

exported_items.append(body_1)

.. .
```

Import the Python model

There are two ways to load and simulate a .py file generated by the SolidWorks add-in:

- In a **C++ program** – use the pyparser library from the PYTHON unit of Chrono, to load **spider_robot.py** with `ImportSolidWorksSystem()`

- In a **PYTHON program** – just use the chrono python modules to load **spider_robot.py** with `ImportSolidWorksSystem()`

Used in this demo, for simplicity

(Not so many differences, by the way)

Import the Python model

- The Chrono::PyEngine modules must be correctly installed
  - Compile Chrono with the PYTHON unit enabled
  - If all is fine, Chrono modules will be available from the Python side, try demo_python_1.py
  - Suggested: use a Python IDE , ex. PyScripter

- Create a **demo_spider.py** program

- First: import the Chrono::PyEngine by writing

```
import os
import math
import ChronoEngine_python_core as chrono
import ChronoEngine_python_postprocess as postprocess
import ChronoEngine_python_irrlicht as chronoirr
```

Import the Python model

- Then, use the following to load the .py model:

```
mysystem  = chrono.ChSystem()

parts = chrono.ImportSolidWorksSystem('./spider_robot');

for ib in parts:
    mysystem.Add(ib);
```

- Note, do not pass ".py" suffix in ImportSolidWorksSystem()

Import the Python model

- Some components are not present in the .py model – the actuators!
- We will add them programmatically, connecting pair of **ChMarker** with ChLinkEngine
- Retrieve the ChMarker objects from their names as in:

```
bbody    = mysystem.SearchBody('Part3^SPIDER_ROBOT-1');
bbody.SetBodyFixed(False);
b1base    = mysystem.SearchBody('M-410iB-300 -1/ArmBase-1');
b1turret  = mysystem.SearchBody('M-410iB-300 -1/M-410iB-300-02-1');
b1bicept  = mysystem.SearchBody('M-410iB-300 -1/M-410iB-300-03-1');
b1forearm = mysystem.SearchBody('M-410iB-300 -1/M-410iB-300-06-1');
m1_1B = b1base.   SearchMarker('marker_M1_B');
m1_1A = b1turret. SearchMarker('marker_M1_A');
m1_2B = b1turret. SearchMarker('marker_M2_B');
. . .
```

- Note, look in SolidWorks GUI or in spider_robot.py if you do not remember the mnemonic names, and remember that `M-410iB-300 <1>` in GUI becomes `M-410iB-300 -1` in code

Add actuators or additional parts/links using Chrono

- Create some f(t) motion functions to be assigned to the actuators
- ChLinkEngines will use them in ‘impose rotation mode’
- Each represents the imposed rotation of the joint [rad], as a function of time [s]
- (In a more sophisticated simulator, one can put the ChLinkEngines in ZOH ‘impose torque’ and adjust torque in real time using some PID, IK, i.e. complex controller systems)
- The f(t) functions are created from the ChFunction class hierarchy
- Interesting: use ChFunction_Sequence to queue multiple curves, and ChFunction_Repeat to make periodic functions

Add actuators or additional parts/links using Chrono

- Example of f(t) functions for this example:

```
period = 2;
mfunc_sineS   = chrono.ChFunction_Sine(0, 1.0/period,  0.2);  # phase, frequency, amplitude
mfunc_swingSa = chrono.ChFunction_Repeat();
mfunc_swingSa.Set_fa(mfunc_sineS);
mfunc_swingSa.Set_window_length(period);
mfunc_swingSa.Set_window_start(0);
mfunc_swingSb = chrono.ChFunction_Repeat();
mfunc_swingSb.Set_fa(mfunc_sineS);
mfunc_swingSb.Set_window_length(period);
mfunc_swingSb.Set_window_start(period/2.0);
mfunc_sineD   = chrono.ChFunction_Sine(0, 1.0/period, -0.2);  # phase, frequency, amplitude
mfunc_swingDb = chrono.ChFunction_Repeat();
mfunc_swingDb.Set_fa(mfunc_sineD);
mfunc_swingDb.Set_window_length(period);
mfunc_swingDb.Set_window_start(period/2.0);
mfunc_swingDa = chrono.ChFunction_Repeat();
mfunc_swingDa.Set_fa(mfunc_sineD);
mfunc_swingDa.Set_window_length(period);
mfunc_swingDa.Set_window_start(0);
Etc. . .
```

Add actuators or additional parts/links using Chrono

- Create the actuators using **ChLinkEngine** objects between couple of ChMarker objects:

```
# Add actuators to Leg n.1

motor1_1 = chrono.ChLinkEngine();
motor1_1.Initialize(m1_1A, m1_1B);
motor1_1.Set_eng_mode(chrono.ChLinkEngine.ENG_MODE_ROTATION);
motor1_1.Set_rot_funct(mfunc_swingSa);
mysystem.Add(motor1_1);

motor1_2 = chrono.ChLinkEngine();
motor1_2.Initialize(m1_2A, m1_2B);
motor1_2.Set_eng_mode(chrono.ChLinkEngine.ENG_MODE_ROTATION);
motor1_2.Set_rot_funct(mfunc_updownA);
mysystem.Add(motor1_2);

motor1_3 = chrono.ChLinkEngine();
motor1_3.Initialize(m1_3A, m1_3B);
motor1_3.Set_eng_mode(chrono.ChLinkEngine.ENG_MODE_ROTATION);
motor1_3.Set_rot_funct(mfunc_const);
mysystem.Add(motor1_3);
```

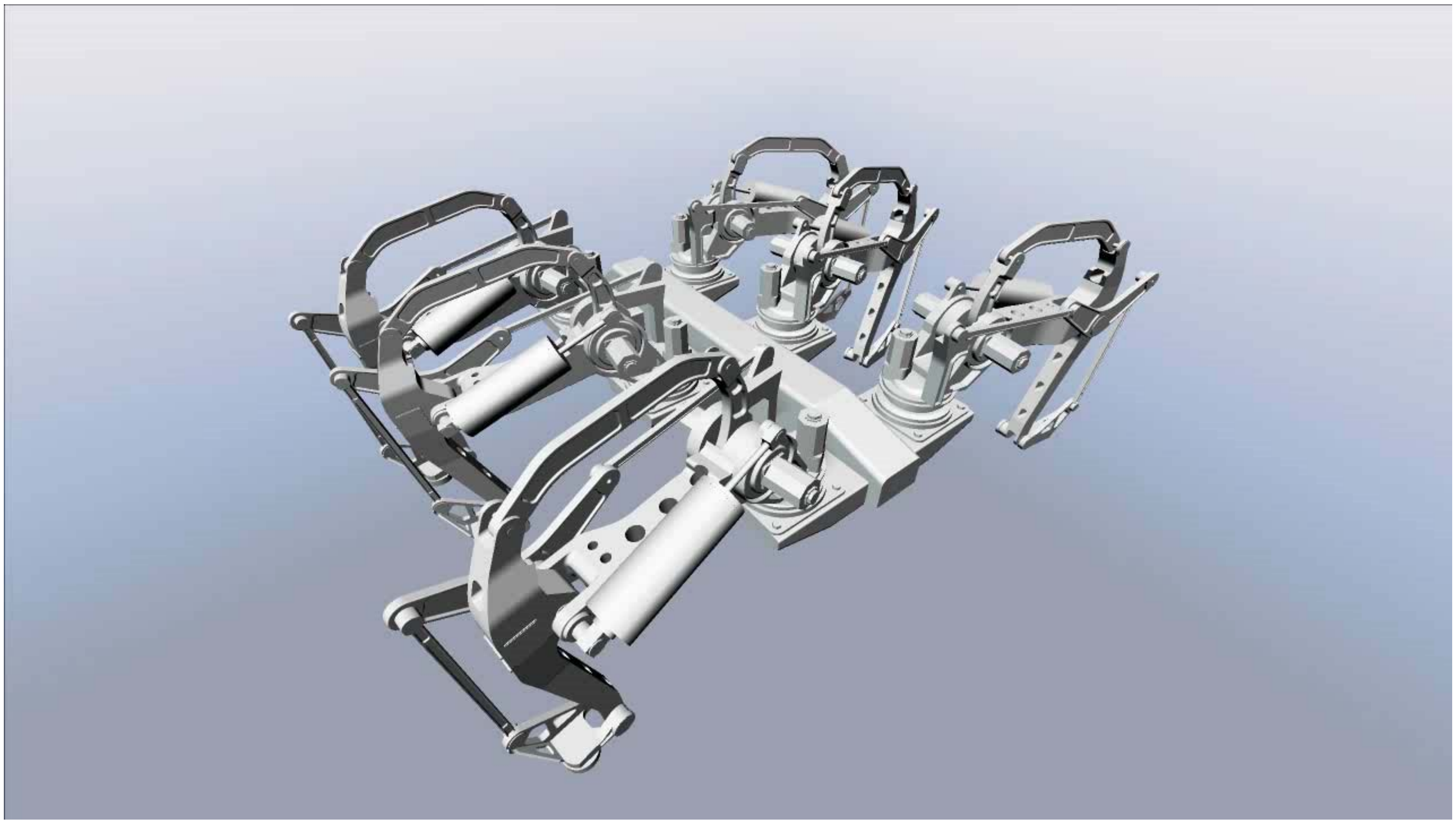
Setup Irrlicht visualization

```
myapplication = chronoirr.ChIrrApp(
                     mysystem,
                     'Test',
                     chronoirr.dimension2du(1280,720))

myapplication.AddTypicalSky('./data/skybox/')
myapplication.AddTypicalCamera(
                     chronoirr.vector3df(2.8,2.6,2.8),
                     chronoirr.vector3df(0.0,2.6,0.0))
myapplication.AddTypicalLights()
myapplication.AddLightWithShadow(
                     chronoirr.vector3df(10,20,10),
                     chronoirr.vector3df(0,2.6,0),
                     10 ,10,40, 60, 512);


myapplication.AssetBindAll();


myapplication.AssetUpdateAll();


myapplication.AddShadowAll();
```

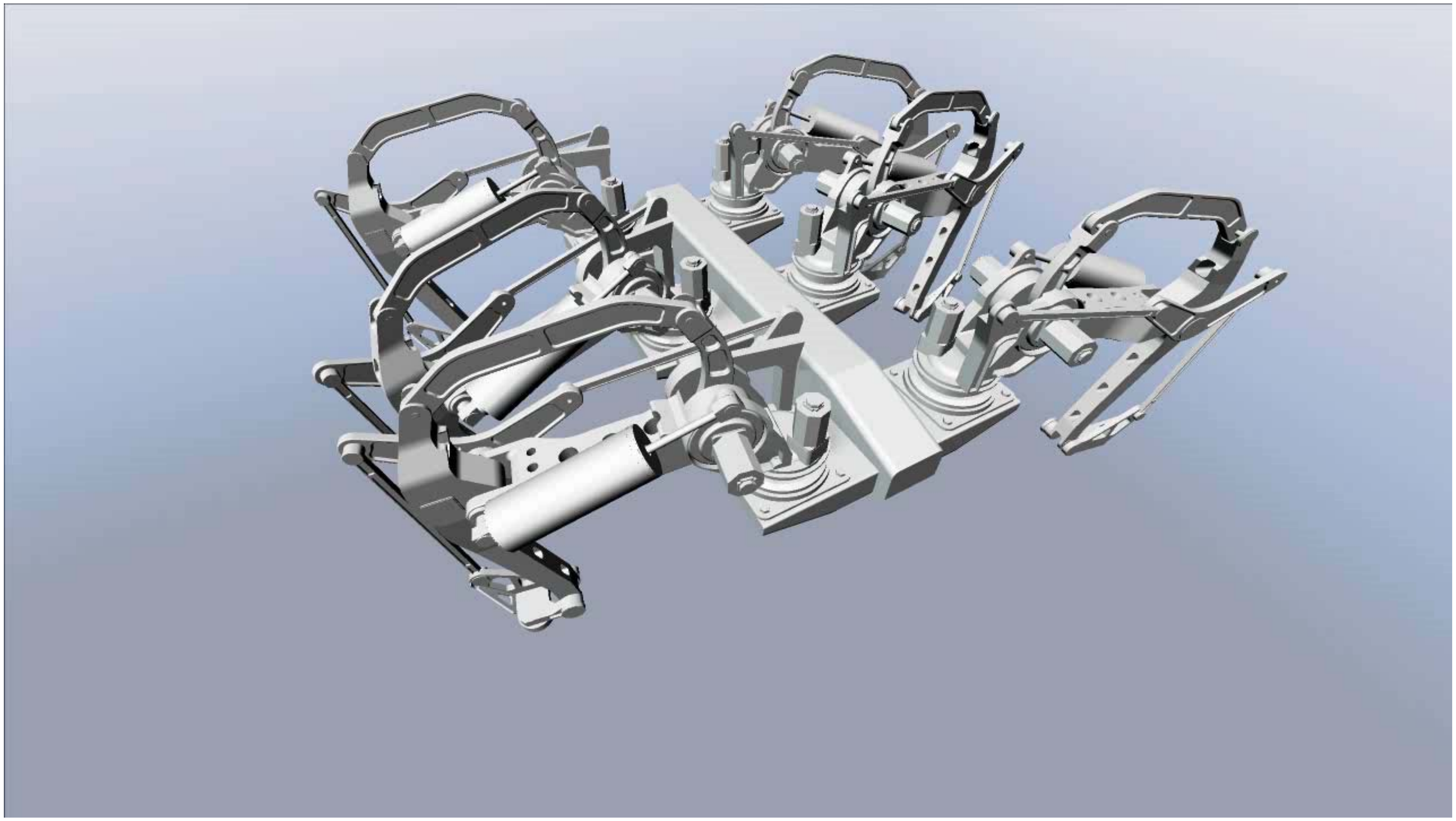
The simulation loop

```
mysystem.SetMaxItersSolverSpeed(600);
mysystem.SetSolverWarmStarting(True);
mysystem.SetSolverType(chrono.ChSystem.SOLVER_BARZILAIBORWEIN);
myapplication.SetTimestep(0.002);


while(myapplication.GetDevice().run()):
    myapplication.BeginScene()
    myapplication.DrawAll()
    myapplication.DoStep()
    myapplication.EndScene()
```

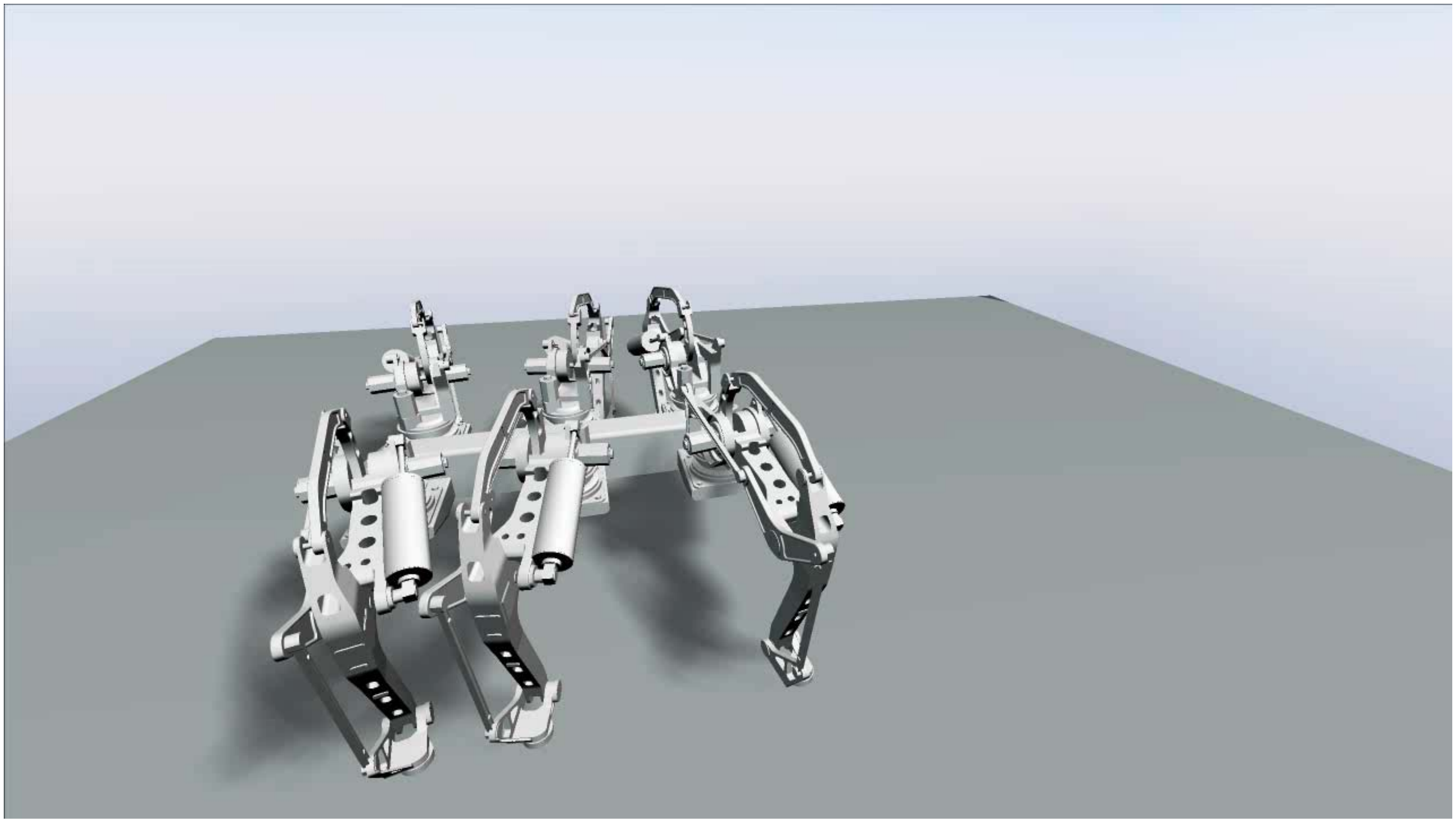
Simulate the mechanism (walk pattern A)



Simulate the mechanism (walk pattern B – better)



Simulate the mechanism



Conclusions

- Used Chrono::SolidWorks add-in as a preprocessor
- This add-in is under development: some SolidWorks mates are not yet translated into equivalent Chrono constraints
- Exported models in .py files can be load in C++ programs or Python programs
- One can add engines, actuators etc. programmatically
- The demo_spider.py program could be written as a C++ program with minimal differences
- This is a quick demo: a more advanced simulator could include sophisticated gait algorithms, path following, PID based actuators, Simulink co-simulation, etc.

Future developments

- We are developing a new ROBOTICS module for Chrono

- Some features are already in development
  - Support for artificial vision
  - Support for sensors
  - Path following, controllers,
  - Visualization tools
  - Etc.