

Omniverse Base Workshop 1

KUKA X IPI

- 2x 4h Course with examples to
 - Learn the Basics in Omniverse, USD & Isaac Sim
 - Enable you to do Robotic model preparation & Simulation
 - Introduction to IsaacLab

Trainer Introduction



Fabian Fichtl



Julian Zürn

Institut für Produktion und Informatik

Technologietransferzentrum der Hochschule Kempten

- anwendungsorientierte Forschungseinrichtung
- F&E-Projekte im Kontext Digitalisierung in der Produktion
- Seit 2021 mit 30 Mitarbeitern
- Technologietransfer und Startup Inkubator
- VIBN und Industrial Metaverse in der Lehre



Introduction round

Please tell a few sentences about yourself,
what you expect from this workshop and
what you want to use Omniverse for

Agenda Tag 1:

- Introduction Metaverse
- Omniverse basics:
Nucleus | Data Formats | PhysX| extensions | Stage
- 5 min Pause
- Hands on: Scene manipulation, Kinematics and Assets
 - UI Manipulation Navigation, Manipulation (property window)
 - Mass & Mesh manipulation
 - First Robot
- 10 min Pause
- Import and Tune a Robot
- 5 min Pause
- First Robot Control (LULA test widget)
 - Adding Sensors and Kameras

Agenda Tag 2:

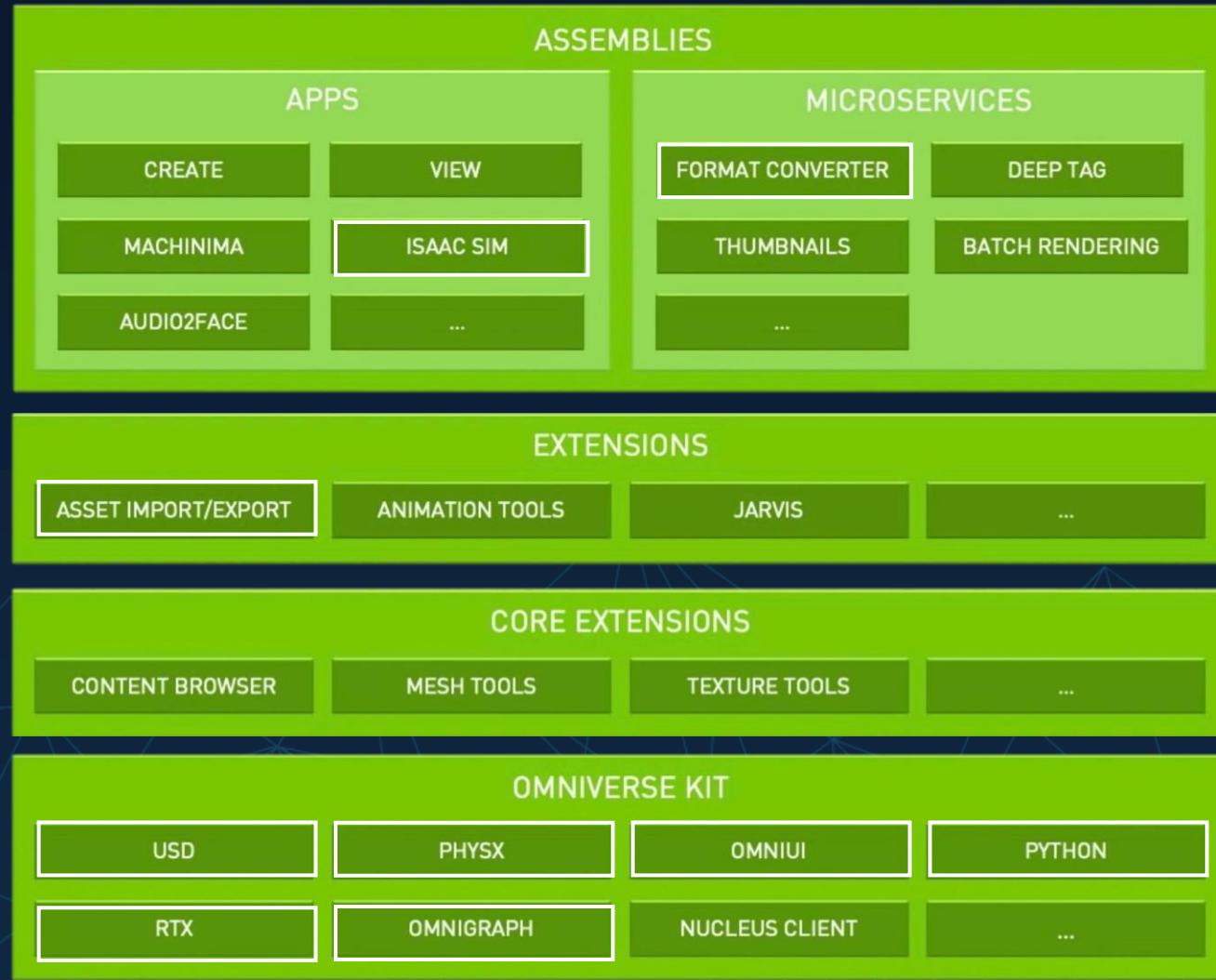
- Debugging Tools:
 - Console, VS Code, UI
- Scene Architecture
 - Build your Scenario: Robot, Gripper, Controller, Environment, Handle Object
- Pick & Place
- Information Channels | Further Courses

Introduction to Omniverse

Concept: Industrial Metaverse

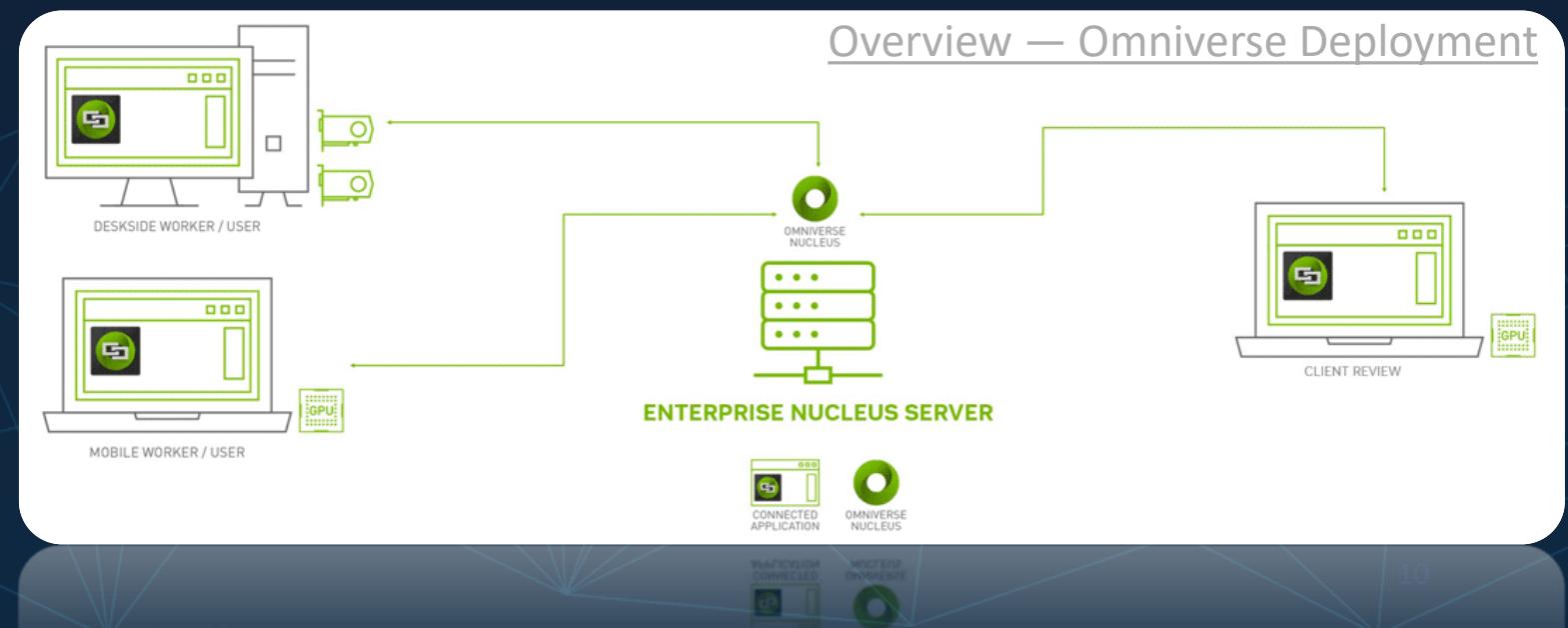
- Virtual Environment
- Real-time Interaction and Collaboration
- Purpose can span the Design, Simulation, Observation or Replay of arbitrary Objects and Processes in any Industry
- The *NVIDIA Omniverse* platform is optimized to run 3D physics Simulations in real-time on *NVIDIA RTX* Hardware

Omniverse Architecture

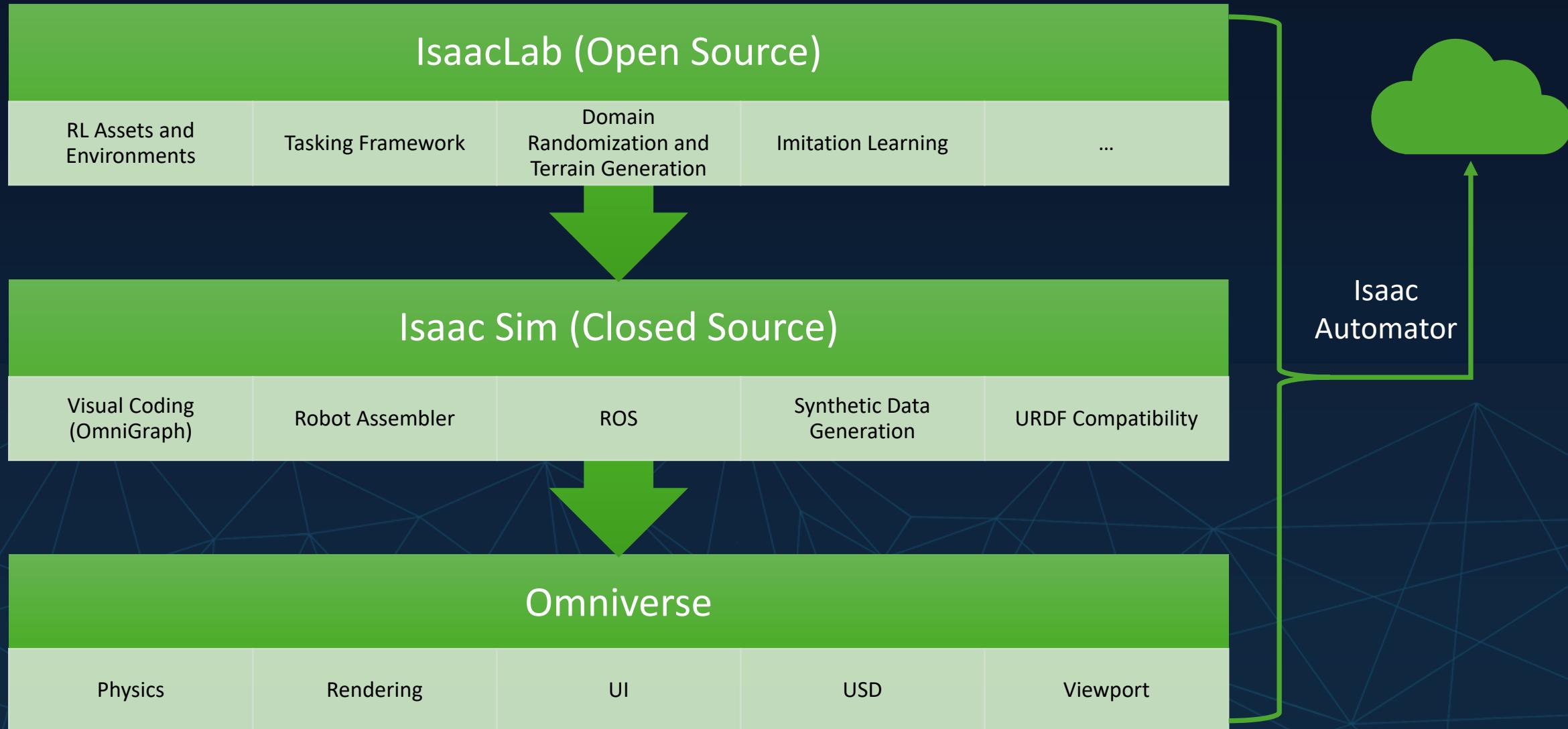


Nucleus – The Omniverse Database

- Enables Real-time Collaboration
- Comprehensive User/Permission Management
- Caching Mechanism for large Assets
- Wide API and Script Integration
- Enterprise Scalability



NVIDIA Isaac Ecosystem



Universal scene description



Interacting with USD (in Omniverse scene)

- UI Main Capabilities
 - Viewport:
 - Scene
 - Property Window:
 - Prim* Properties
 - Omnigraph:
 - Process Logic
 - ROS Controller
 - Various extensions
- API Main Capabilities
 - Python interface
 - Robot Controllers
 - URDF import
 - Create the scene for IsaacLab
 - Configure the policy
 - Start training

Robot Data formats

	URDF	SRDF	XRDF
Description Language	XML	XML	YAML
Purpose	Define Geometries and Kinematic Joints	Complements URDF Data with Semantic Grouping	Complements URDF Data with Kinematic Description
Collision Handling	Simple Shapes and Meshes	Collision Matrix (specifies handling for link pairs)	Collision Spheres along Robot Links
Joint Representation	Varying Types and Parameters	Defines Groups for Motion Planning	Advanced Configuration including Acceleration and Jerk Limits
Handling	Core Format in ROS and converted to USD upon import in Isaac Sim	ROS Moveit Assistant to generate Motion Plans	NVIDIA CuMotion and Lula to generate Motion Plans

Manual Modelling Phases

Robot Simulation

1. Import URDF and convert to USD
2. Config Joint drives -> Test
3. Add Gripper/ Cameras
4. Add details to the stage of:
 1. URDF for ROS and RL
 2. SRDF for Moveit
 3. XRDF for Cumotion and LULA / RMPflow

NVIDIA PhysX

- The PhysX world comprises a collection of Scenes, each containing objects called Actors;
- Each Scene defines its own reference frame encompassing all of space and time;
- Actors in different Scenes do not interact with each other;
- Characters and vehicles are complex specialized objects made from Actors;
- Actors have a physical state: position and orientation, velocity or momentum, energy, etc;
- Actor physical state may evolve over time due to applied forces, constraints such as joints or contacts, and interactions between Actors.

Omniverse Stage



Until now:

- We reviewed necessary Omniverse Basics and Resources

5 min Break

To be continued:

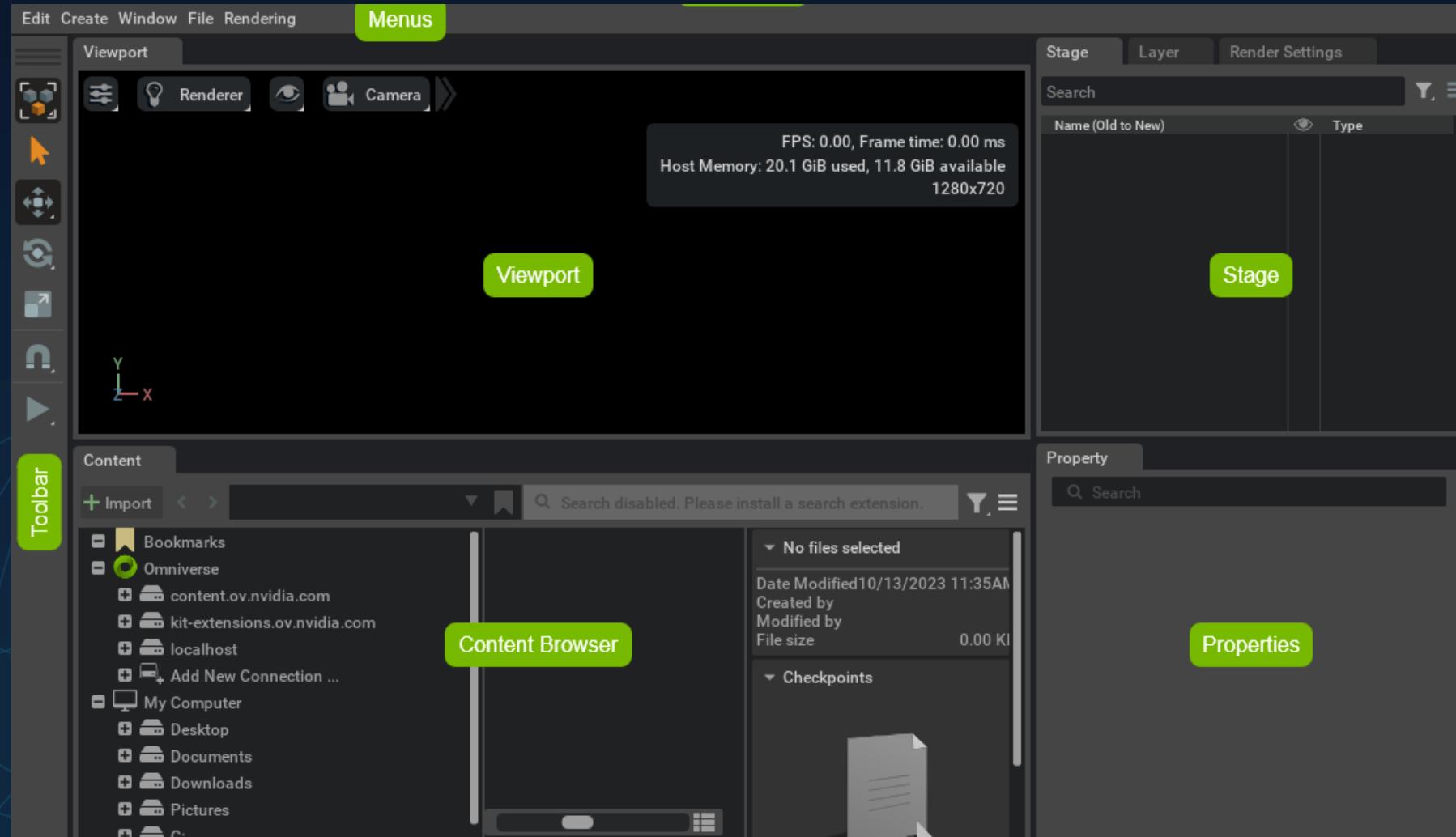
- Practical session:
 - Extensions installation
 - Scene Manipulation
 - Robot import + tuning

Hands on!

Getting started:

- Start Omniverse Isaac Sim on the Machine
 - Please give feedback if your application is loaded

Standard Layout

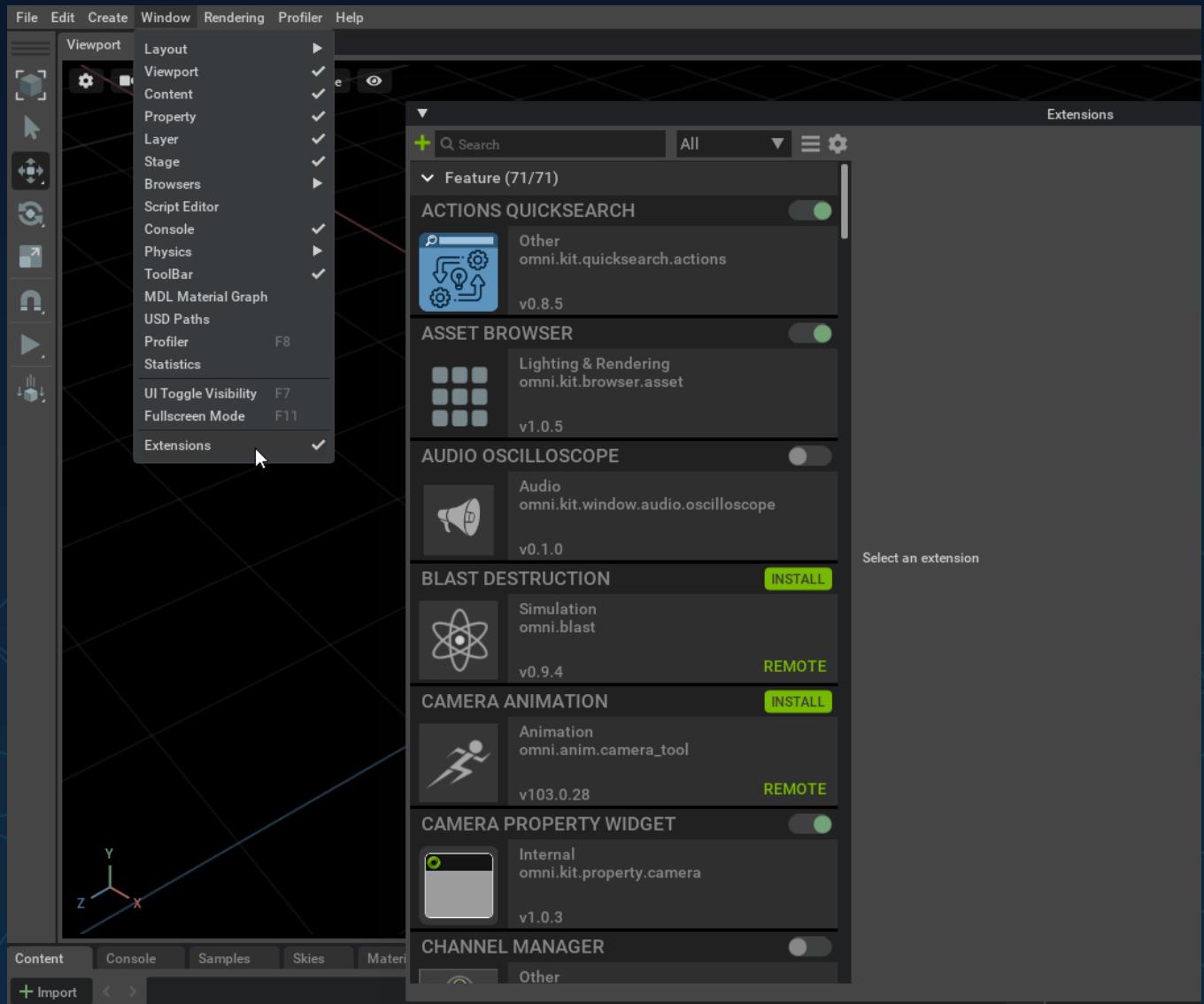


How-to: Viewport Navigation

- Orbit: Left mouse btn. + ALT
- Look: Right mouse btn. (RMB)
- Fly/ Walk: Hold RMB; WASD / Q up / E down
- Adjust Speed Hold RMB; Scroll
- Jump to Selection: F

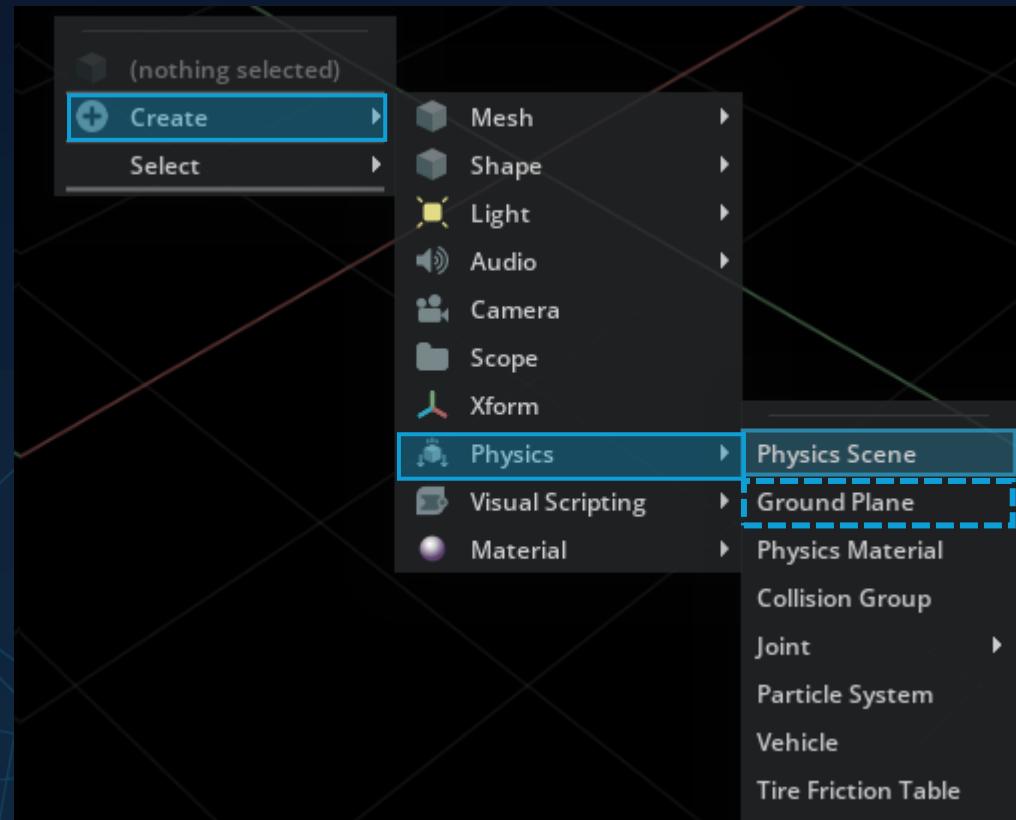
Scene Prep tools (extensions)

- Check for extensions:
 - Action Graph
 - URDF importer



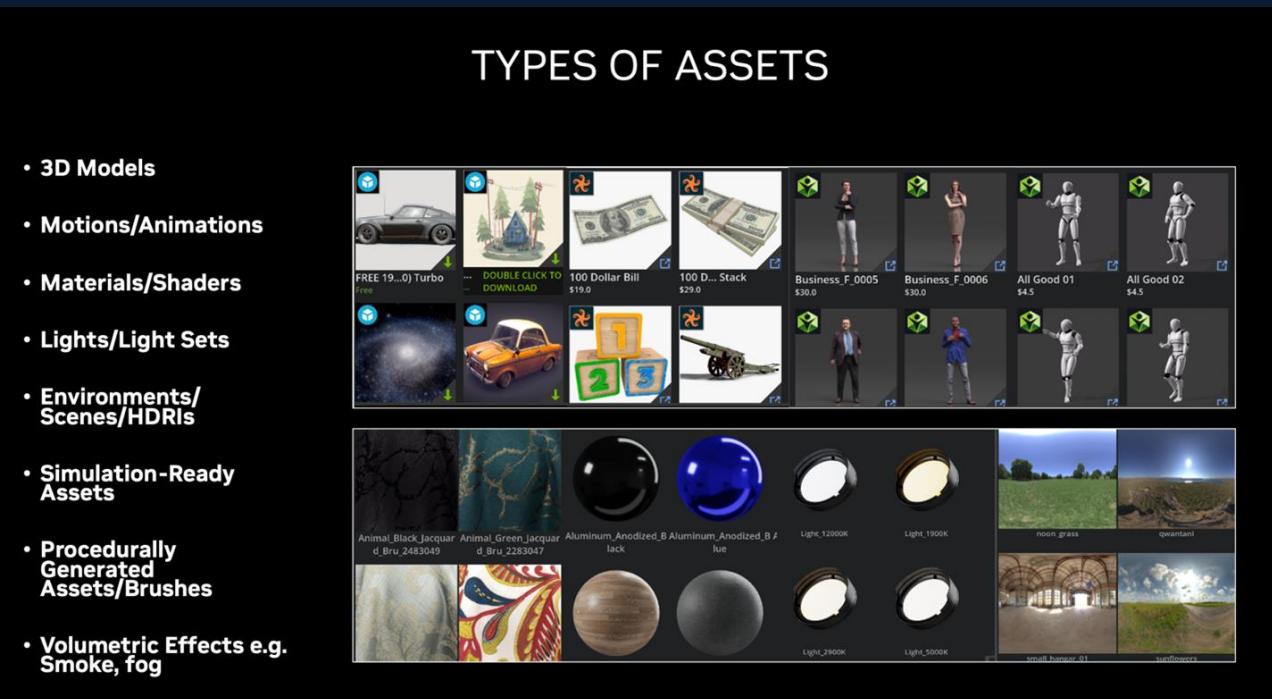
How-to: Add Physics Scene

- Create a Physics Scene
- Create a ground plane



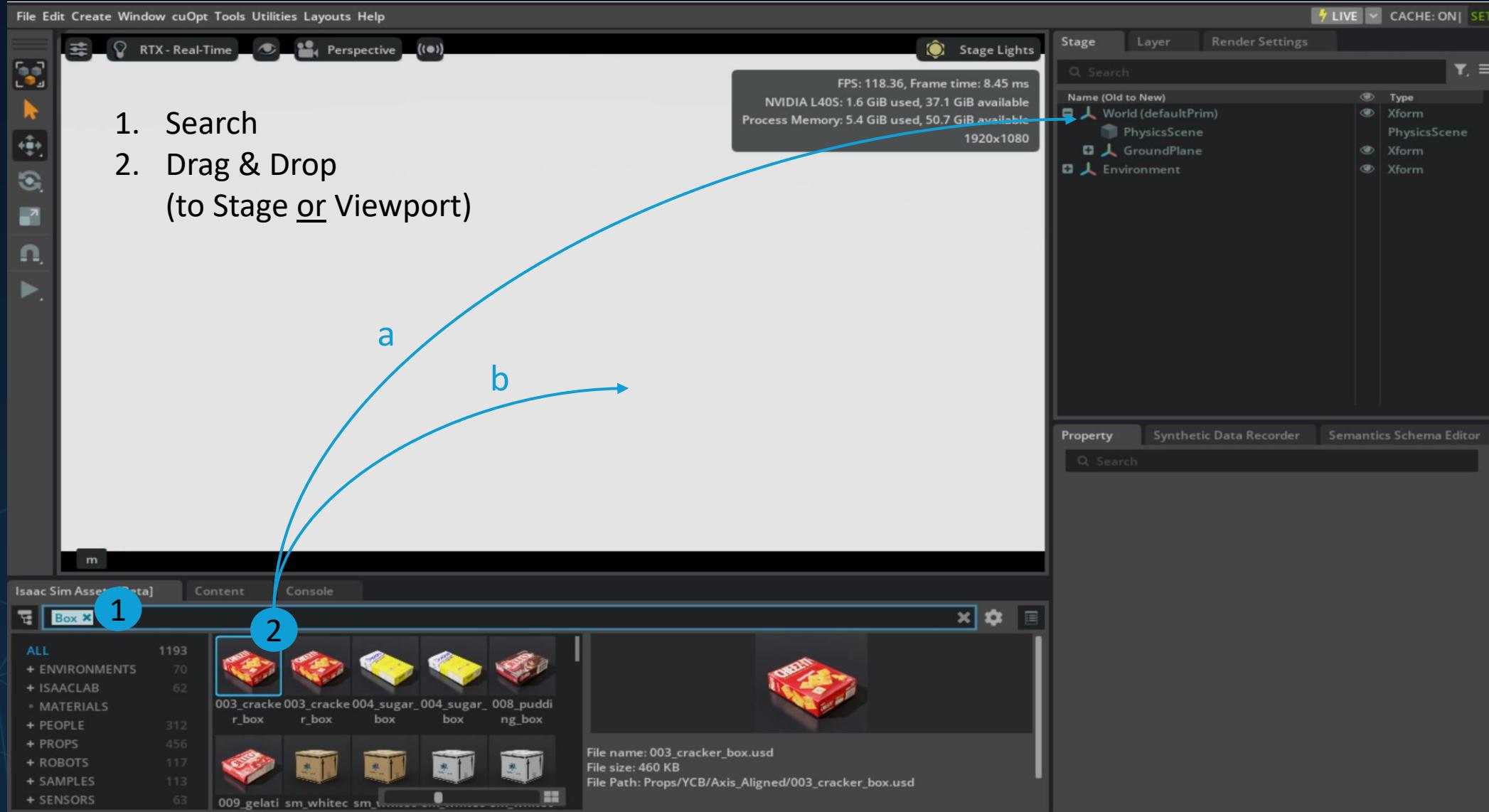
Asset Store

- Import a Box
 - Move above ground plane
 - Adjust the size
 - Add (Rigid Body and and Collider)
 - Press *Play* to start the Simulation

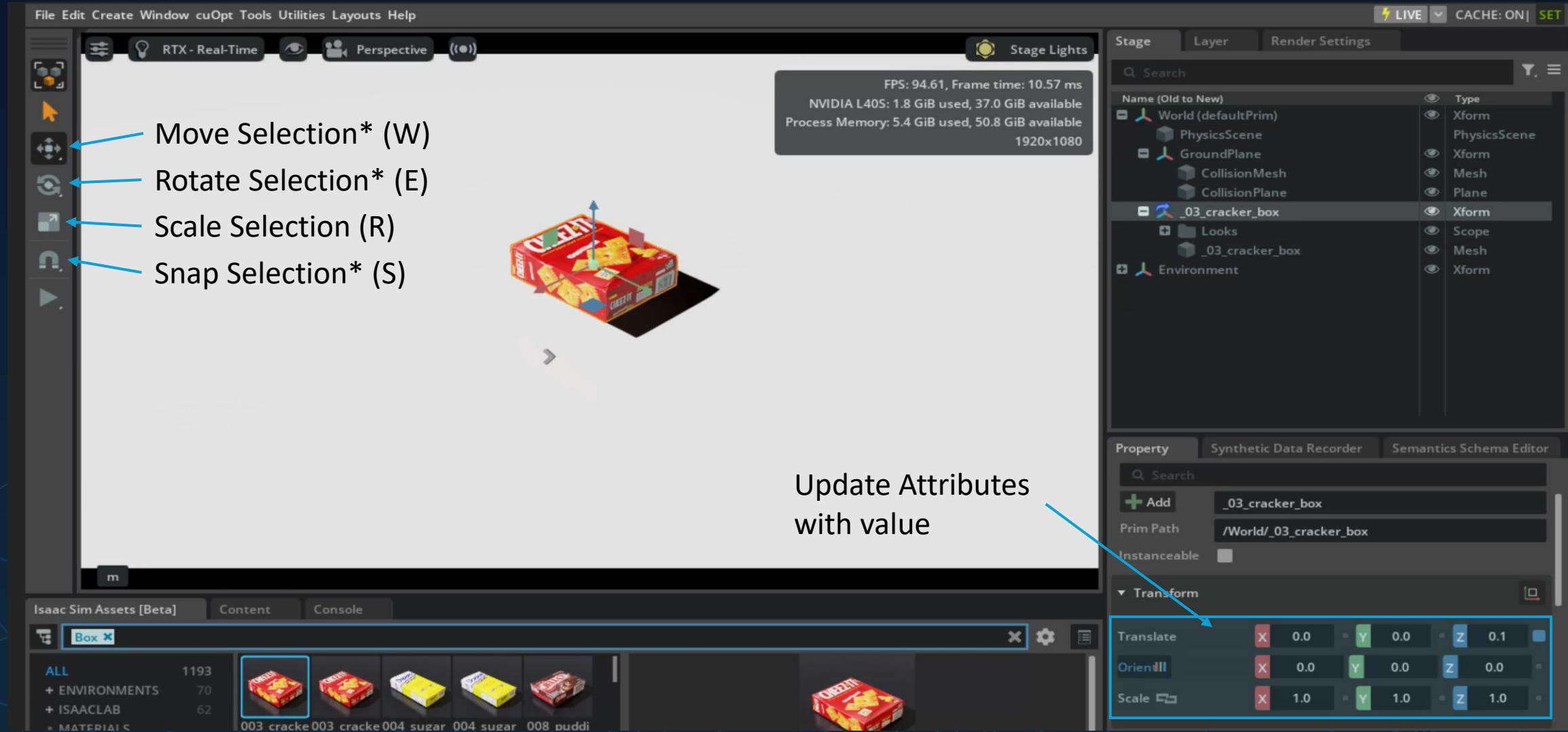


How-to ↓

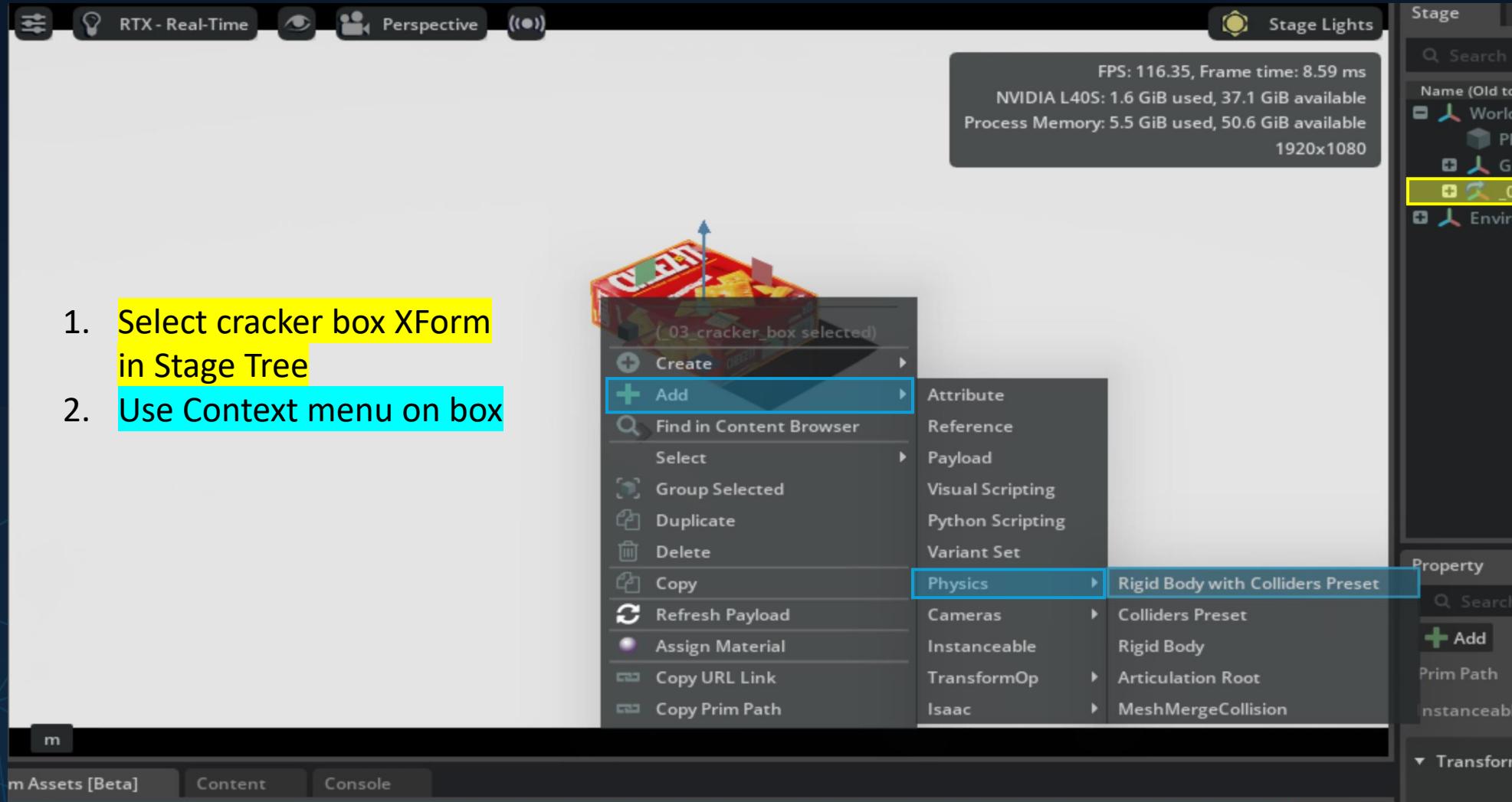
How-to: Use Asset Store



How-to: Manipulation in Viewport



How-to: Add Rigid-body & Collider



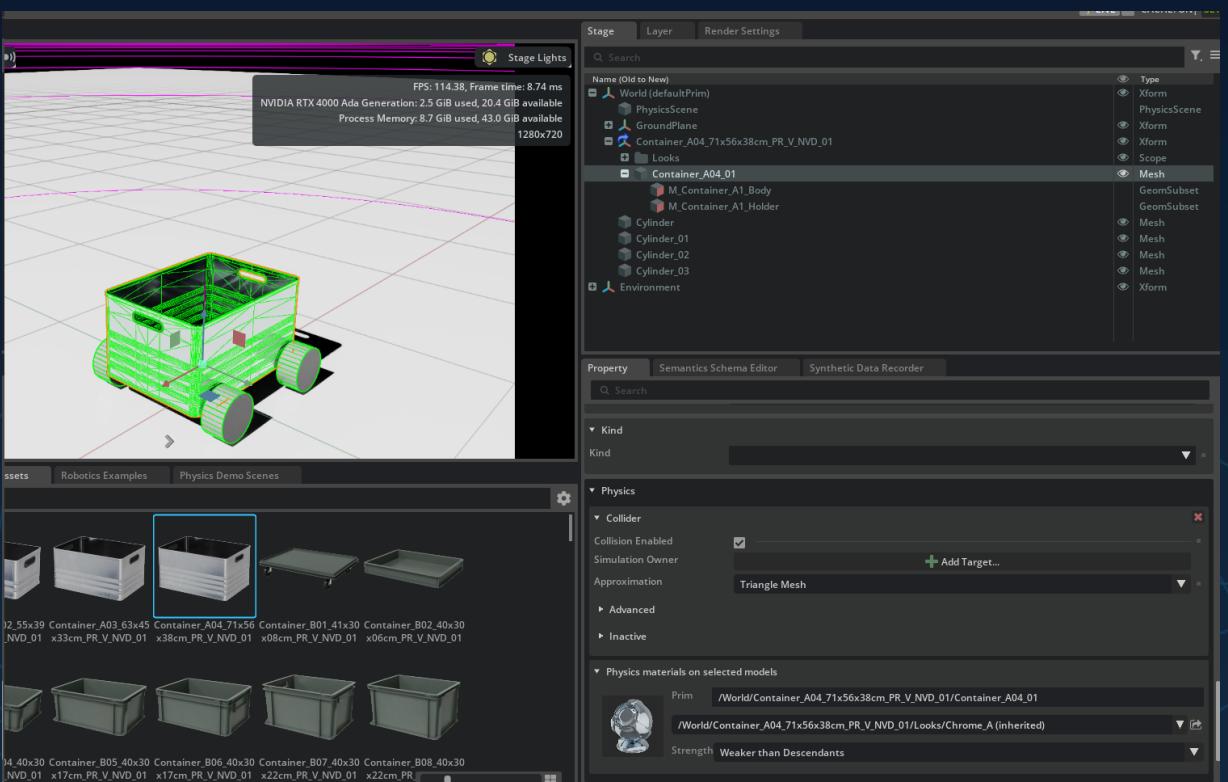
Asset Store

- Import a Box
 - Move above ground plane
 - Adjust the size
 - Add (Rigid Body and and Collider)
 - Press *Play* to start the Simulation



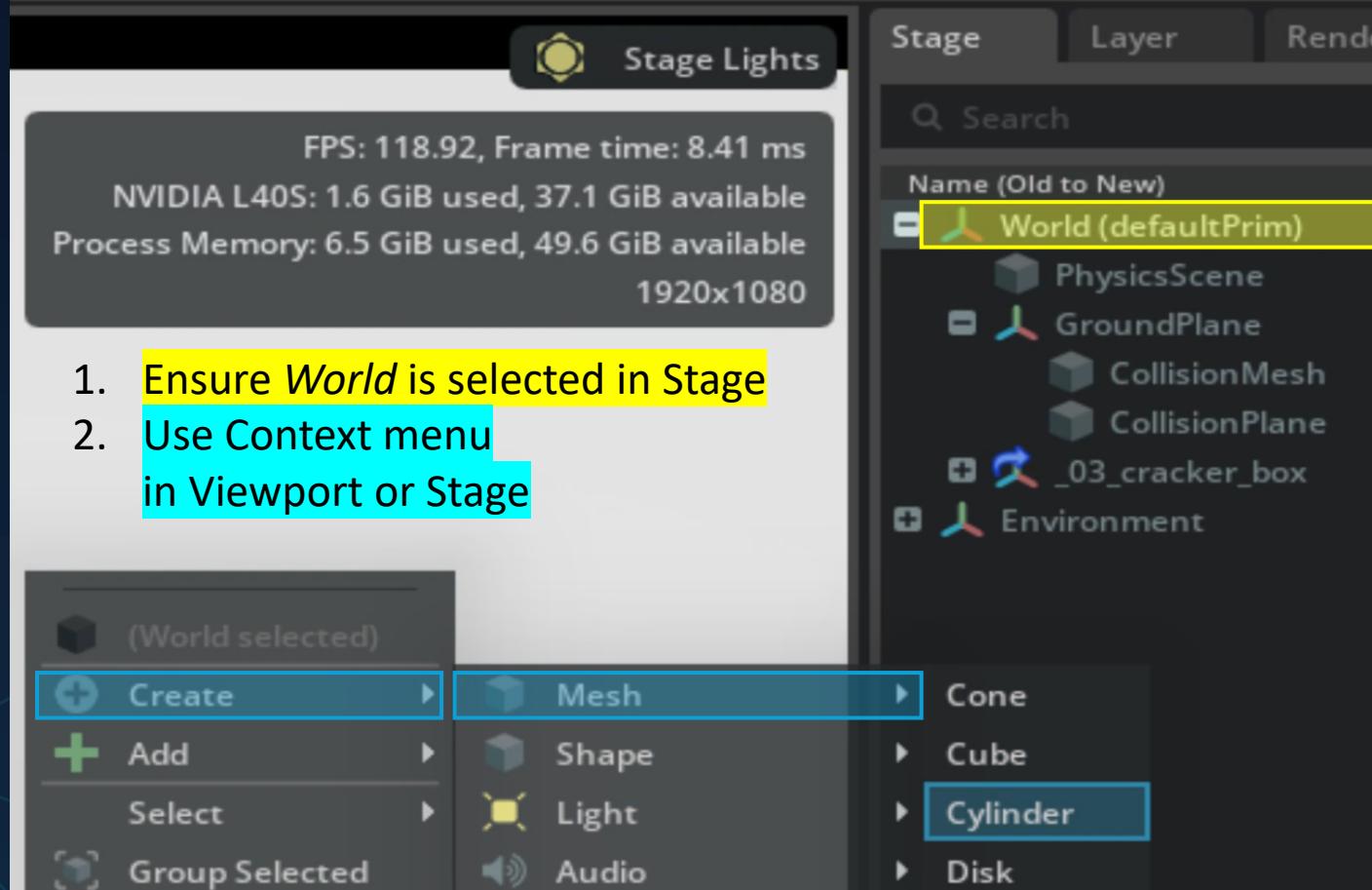
Assemble the first self-built Robot

- Stop the Simulation
- Create Cylinders
 - Move/ orient to the side
 - Create a revolute joint
 - Add Angular drive
 - Modify target velocity + stiffness
- Start the Simulation



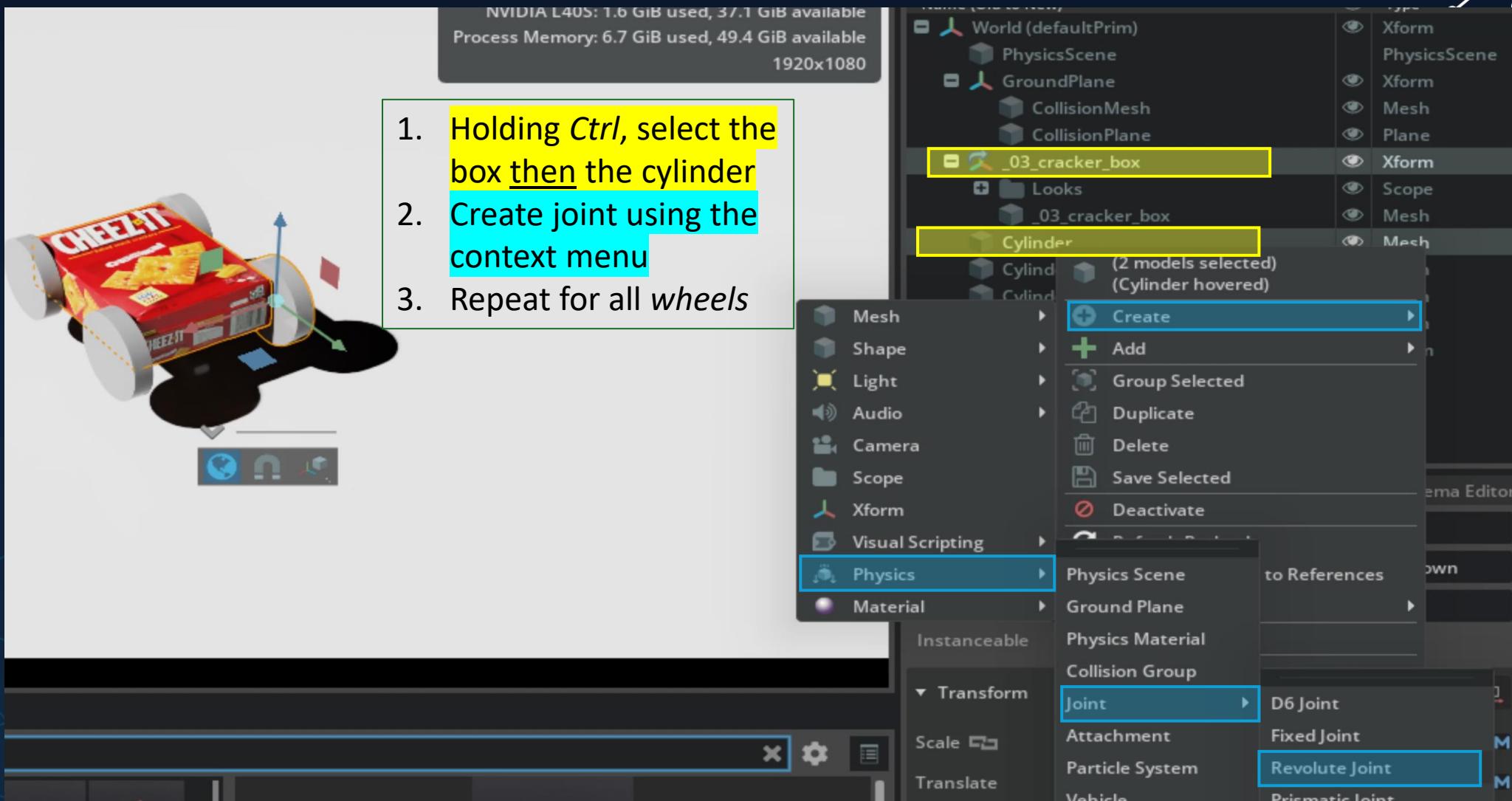
How-to ↓

How-to: Add Cylinder

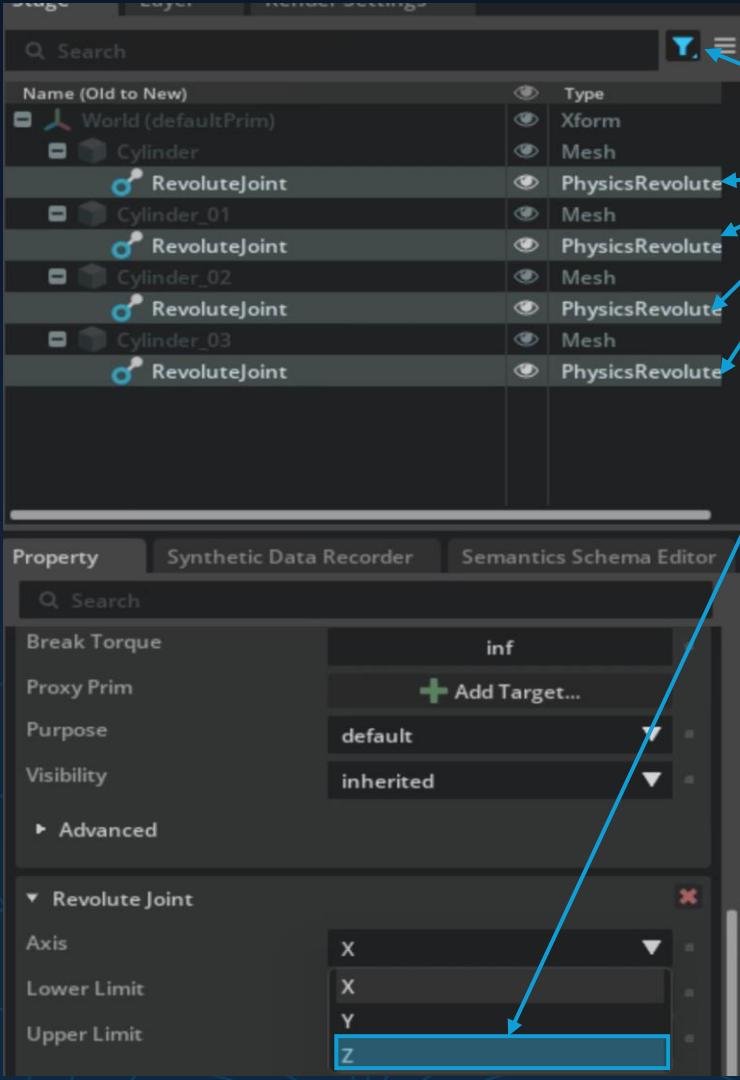


1. Ensure *World* is selected in Stage
2. Use Context menu in Viewport or Stage
3. Adjust scaling and orientation to represent a wheel
4. Add *Rigid Body & Collider*
5. Use *Ctrl+C* and *Ctrl+V* to replicate the wheel
6. Move the wheels to reasonable positions around the box

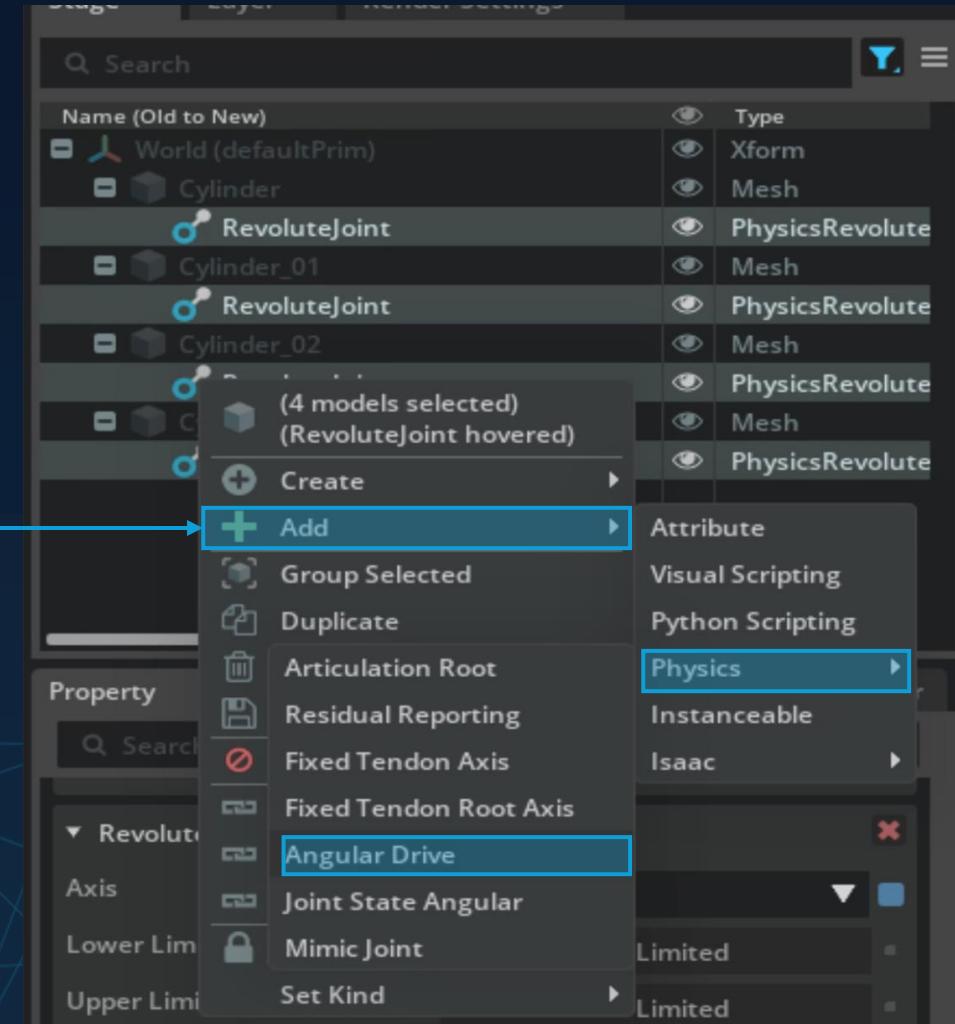
How-to: Add Joint



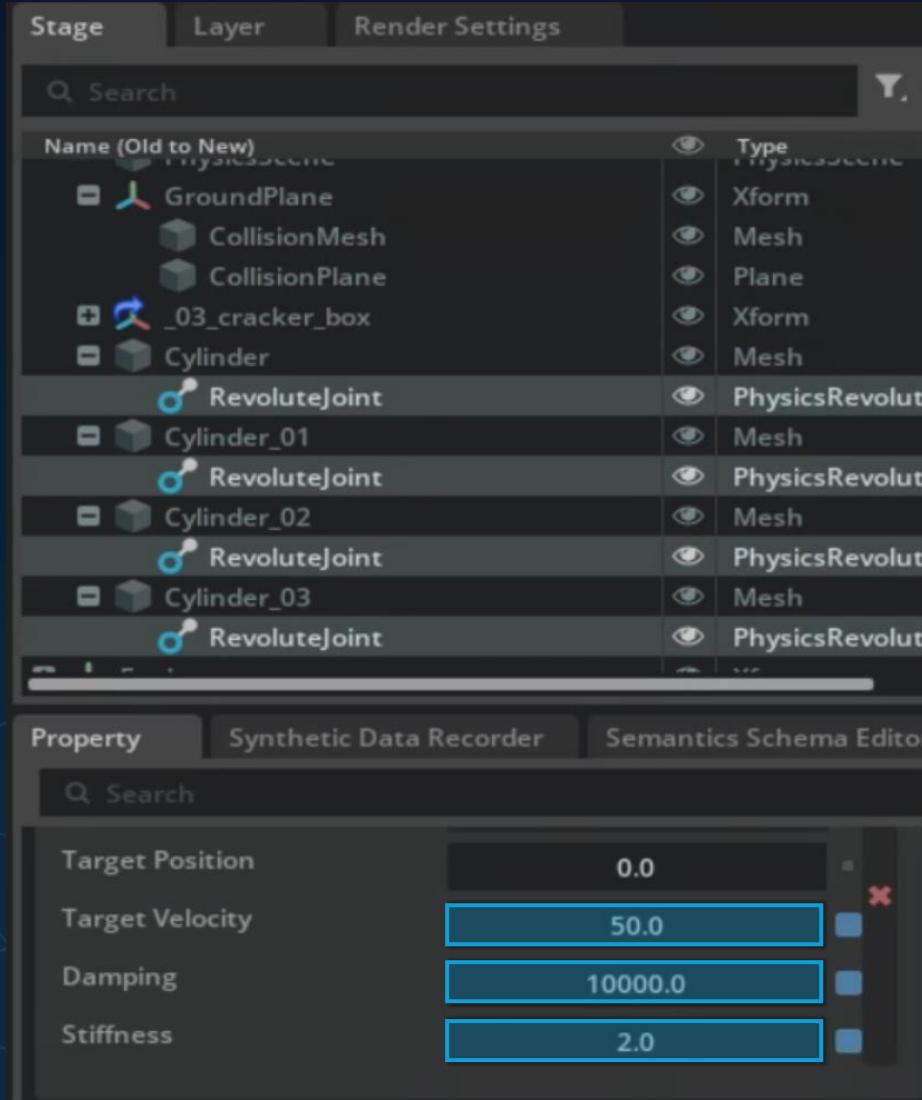
How-to: Adjust Joint and Add Drive



1. Filter for Joints
2. Select all joints
(via *Shift* or *Ctrl*)
3. Adjust joint rotation
4. Add Drives to joints
(via Context menu)



How-to: Adjust Drive

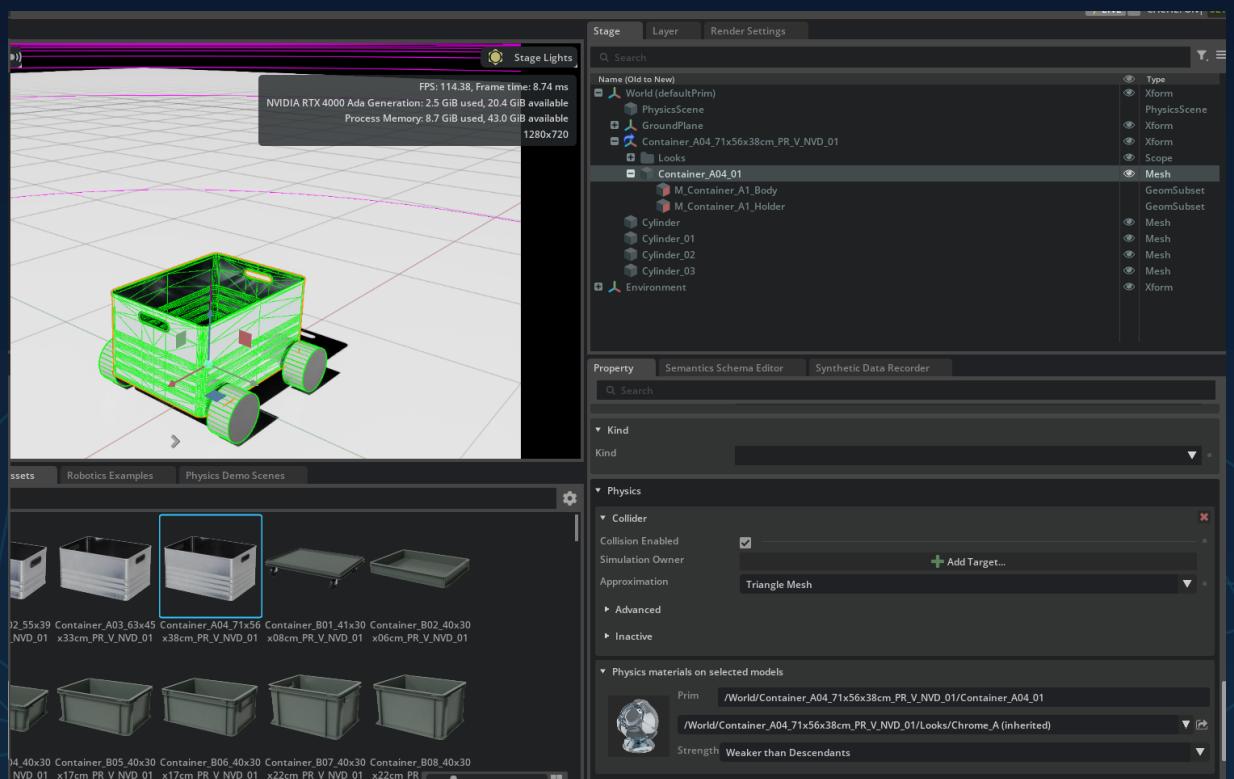


1. Set a Velocity(~20-100)
2. Iterate:
 1. Set Damping & Stiffness Values
 2. Restart Simulation for Testing*

*Not restarting the simulation leads to misleading results due to the forces that already applied before the update

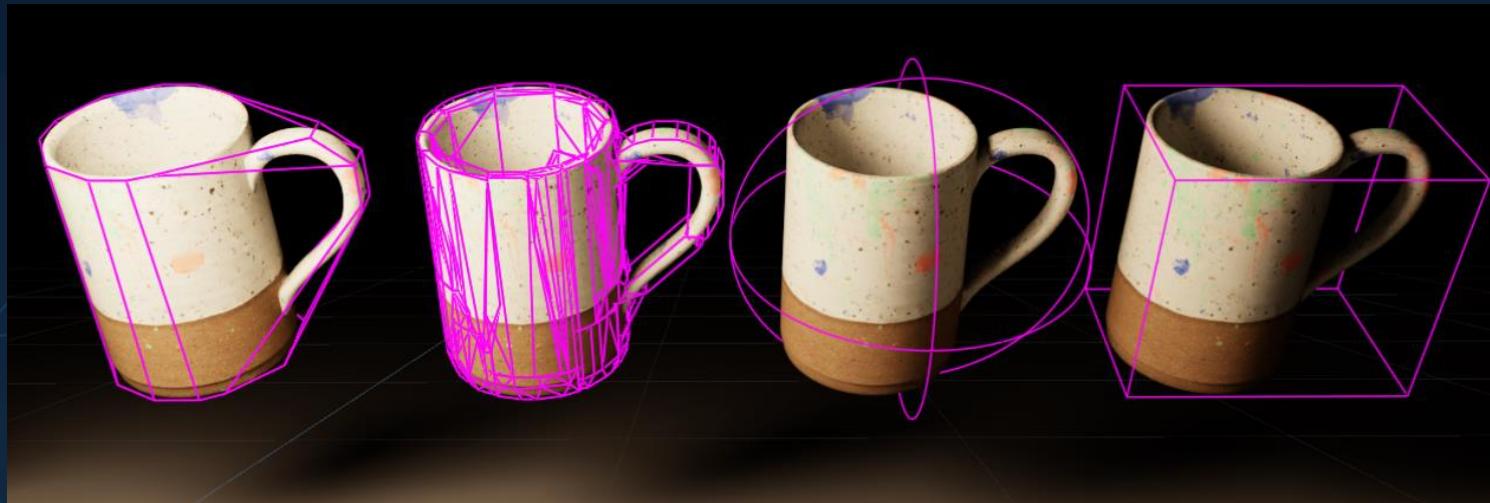
Assemble the first self-built Robot

- Stop the Simulation
- Create Cylinders
 - Move/ orient to the side
 - Create a revolute joint
 - Add Angular drive
 - Modify target velocity + stiffness
- Start the Simulation

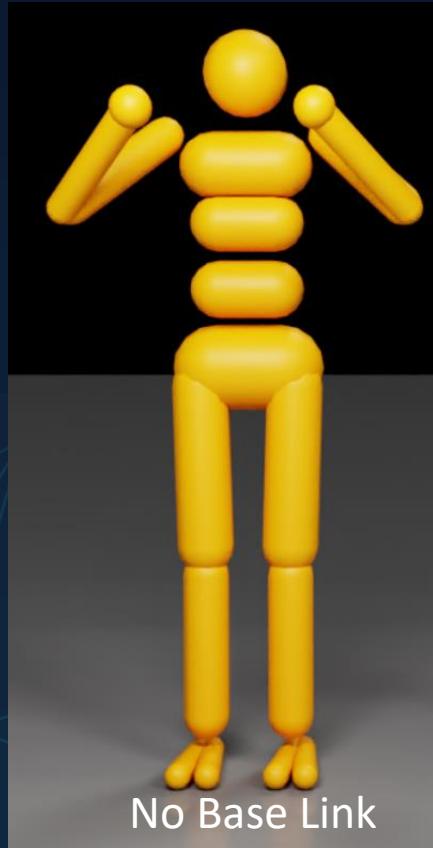


Rigid Bodies and Colliders

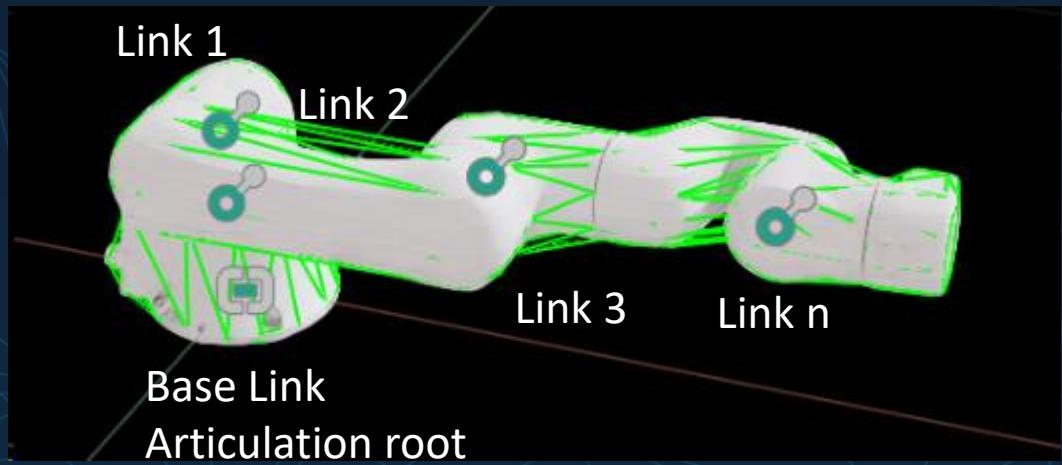
- Simpler approximation = faster Simulation



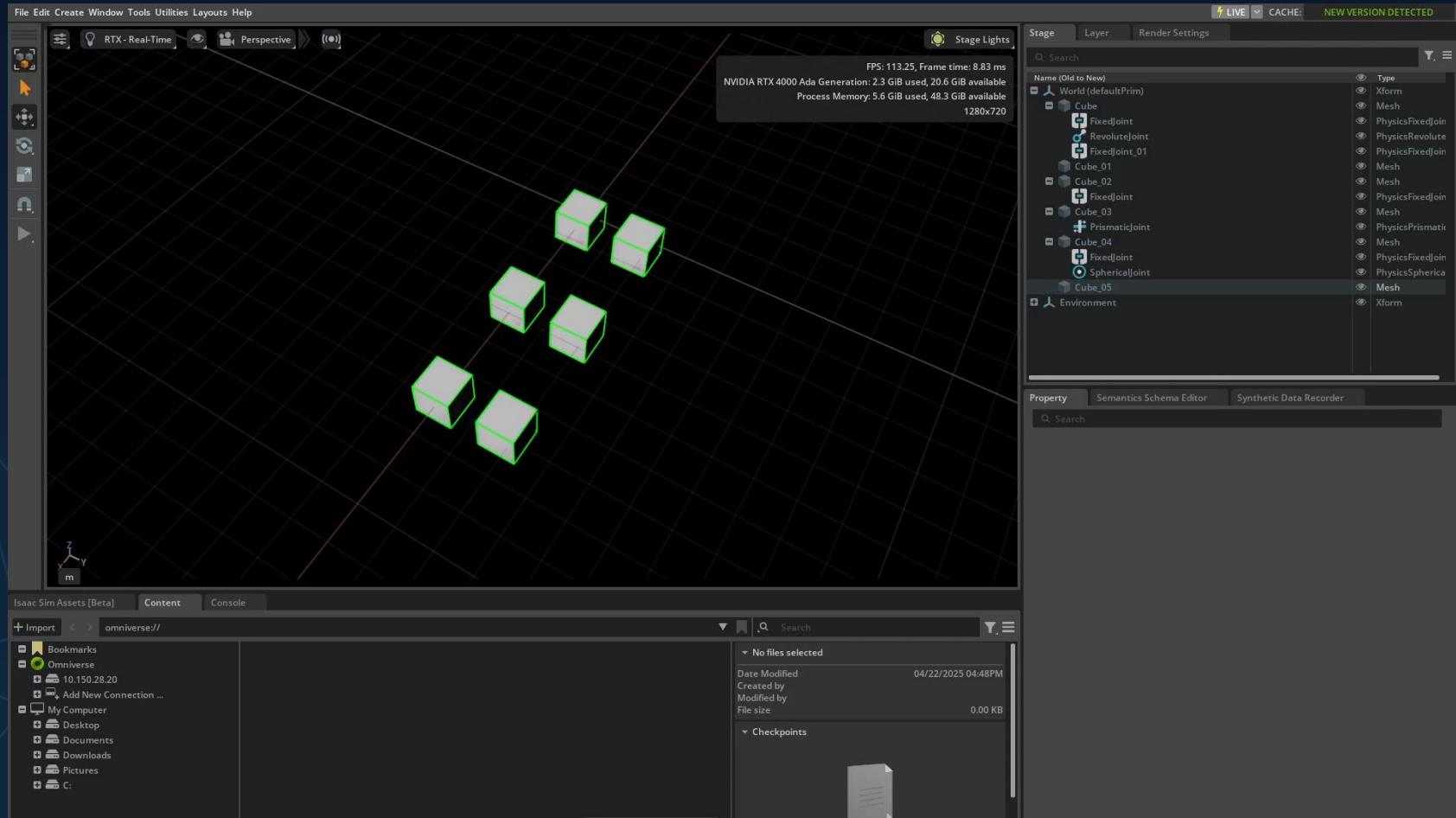
Robot assembly



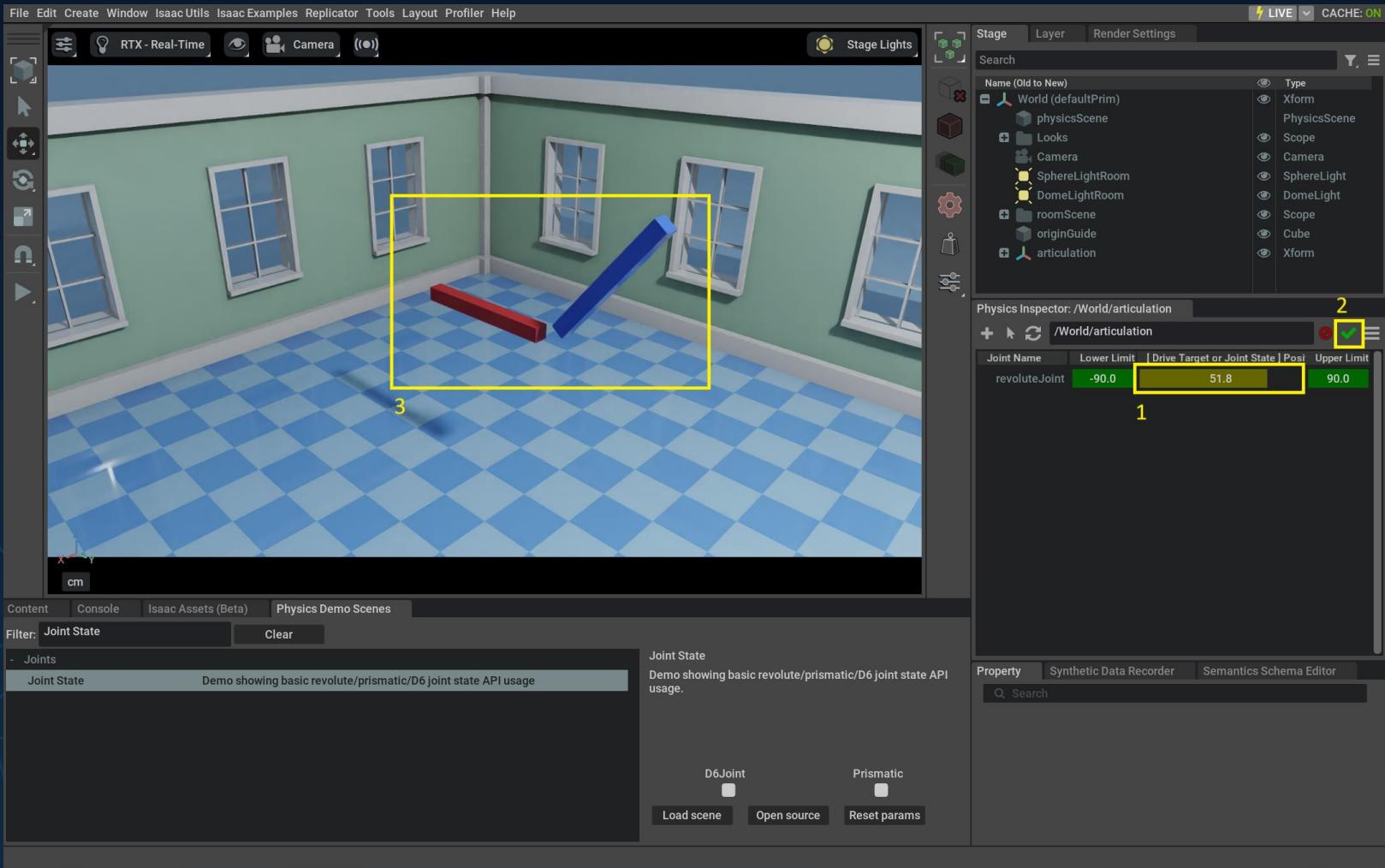
- Structure of mobile / static robot
 - Articulation chain
 - root link free / fixed



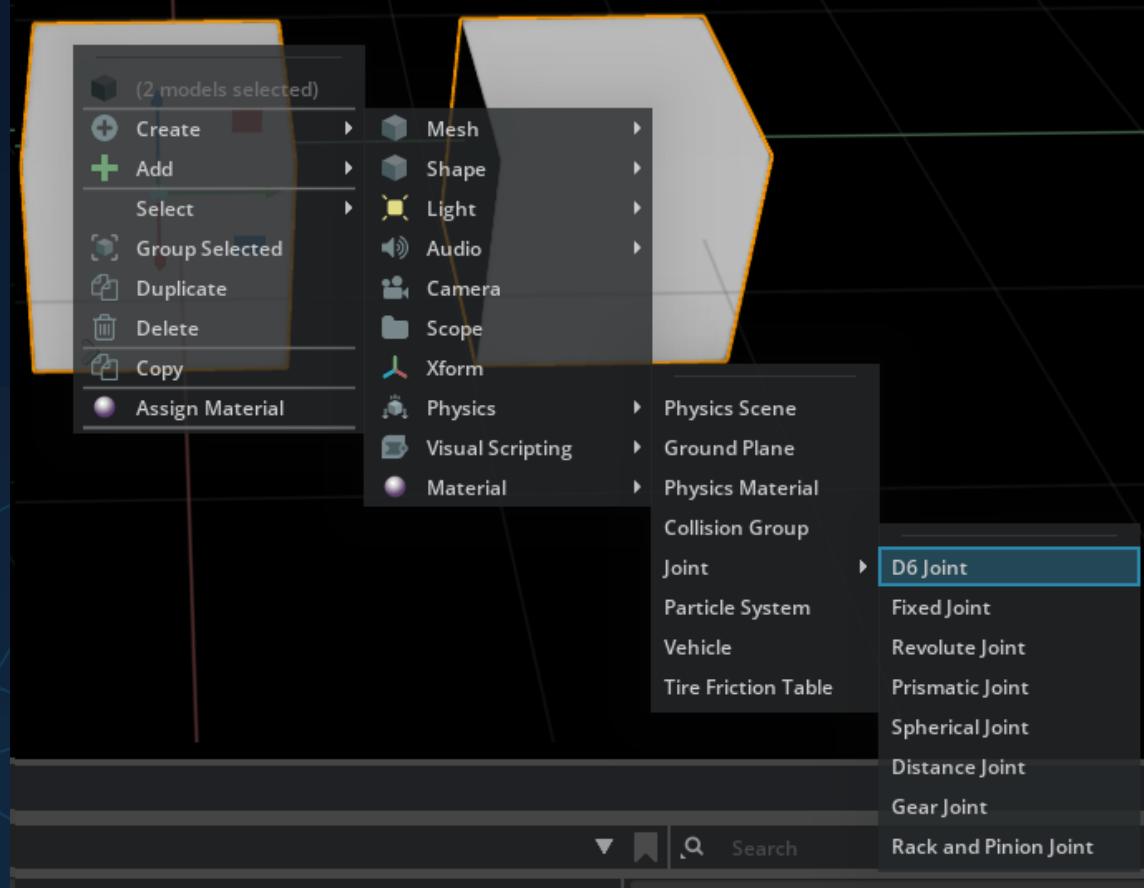
Joint types and properties



Joints



Joint types and properties



Joint drives are internally simulated with PD-Controllers

- *Stiffness* property <-> Proportional coefficient
- *Damping* property <-> Derivative coefficient

 This can lead to unexpected behavior
at low simulation frequencies

Until now:

- We reviewed necessary Omniverse Basics and Resources
- Practical session:
 - Extensions installation
 - Scene Manipulation

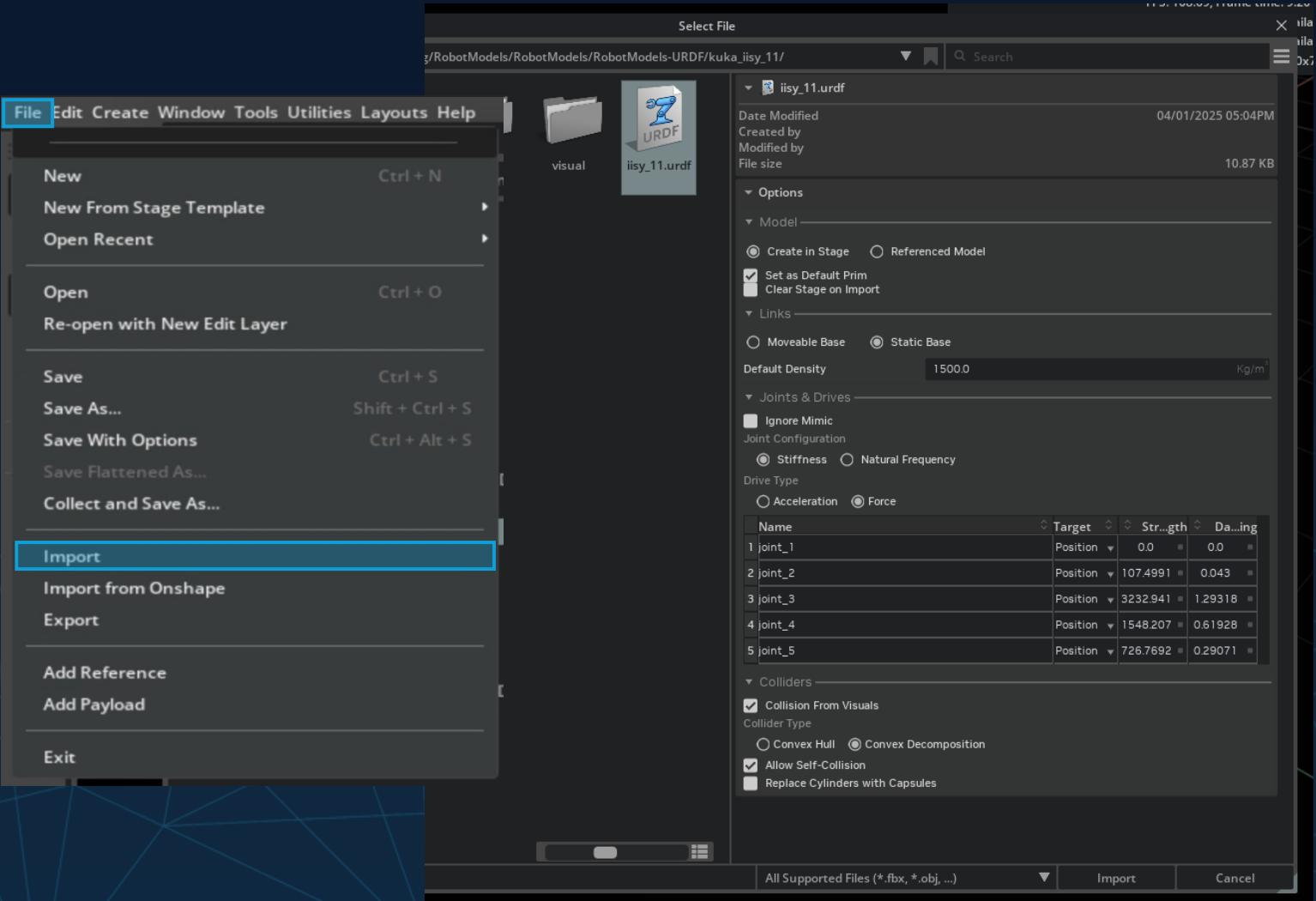
10 min Break

To be continued:

- Practical session:
 - Robot import + tuning

How-to: Starting with a real Robot

- Open *File->import*
- Navigate to a URDF File
- Modify the Options
 - Verify *Model* and *Links*
 - Adjust *Joints & Drives*
 - Verify *Colliders*
- Click *Import*

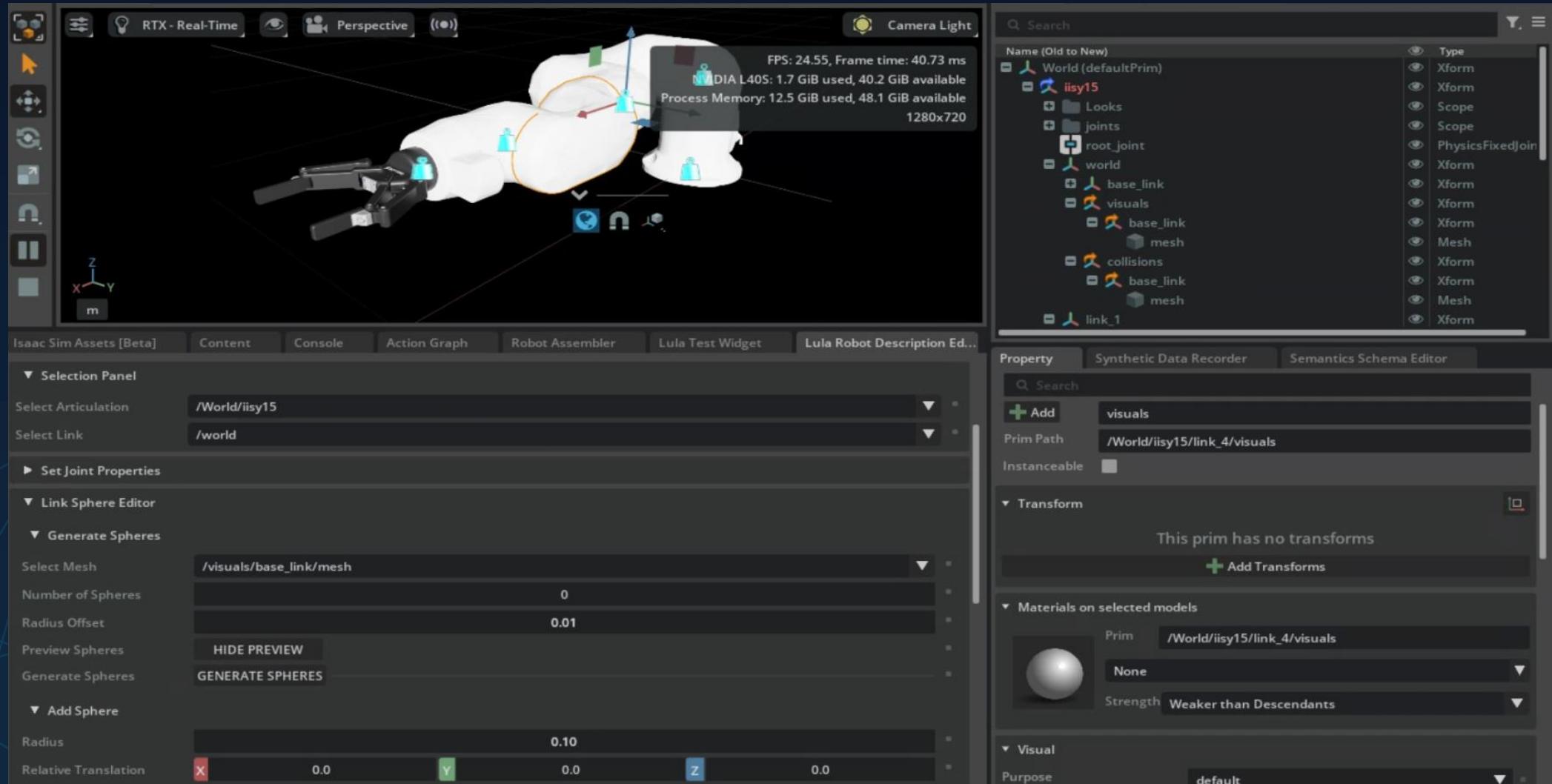


Robot import Tools

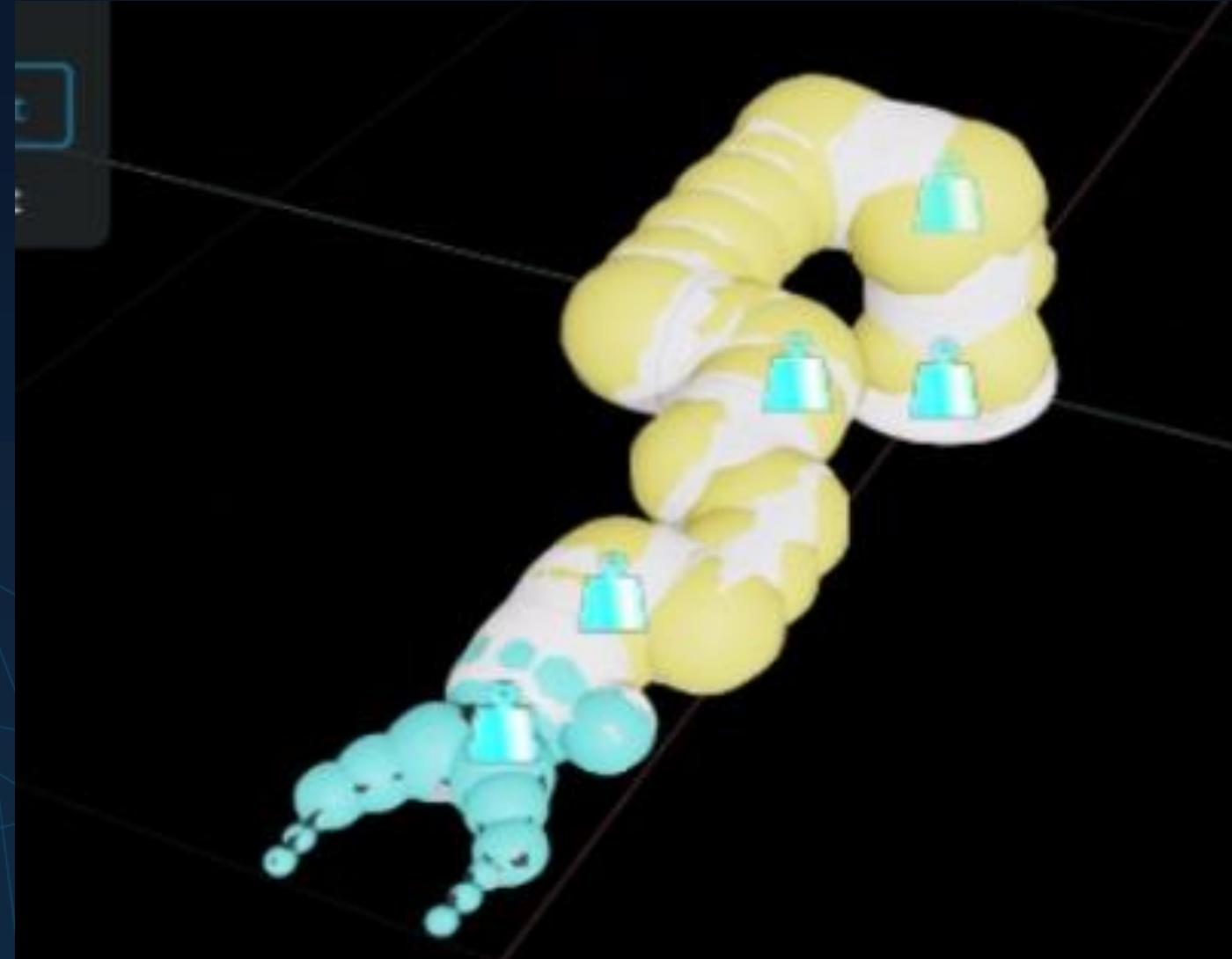
- LULA URDF Import Tool:
 - Import Robot URDF
 - Start scene -> look at joint drives
- Physics Inspector Tool:
 - Activate scene
 - Choose IISY to inspect-> control joint limits/ direction
- Gain tuner:
 - Adjust Stiffness & Damping and test the parameters
- Preparation of self and external collision shape:
 - Sphere approximation Collider tool for RMP



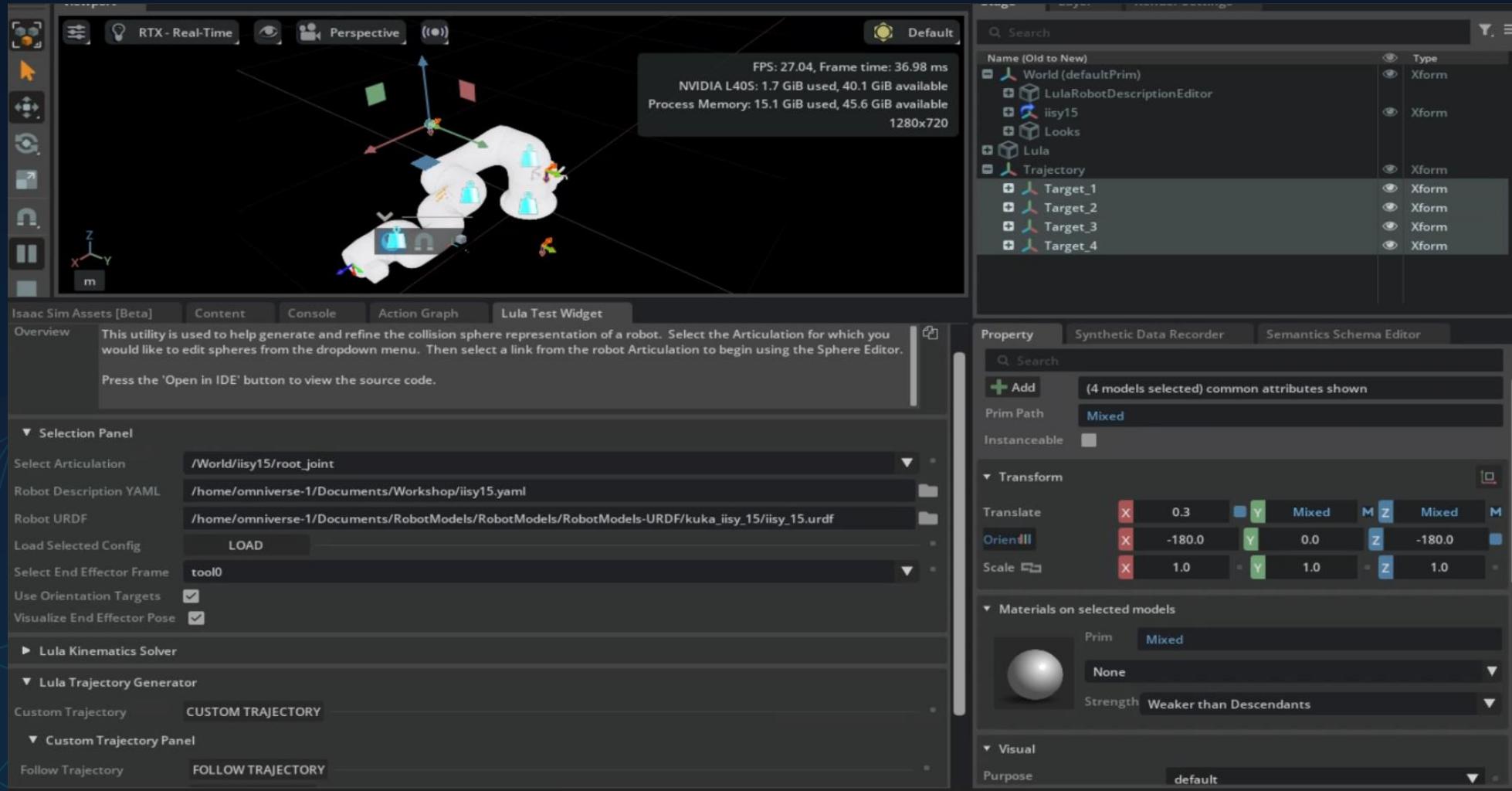
Robot Description Editor



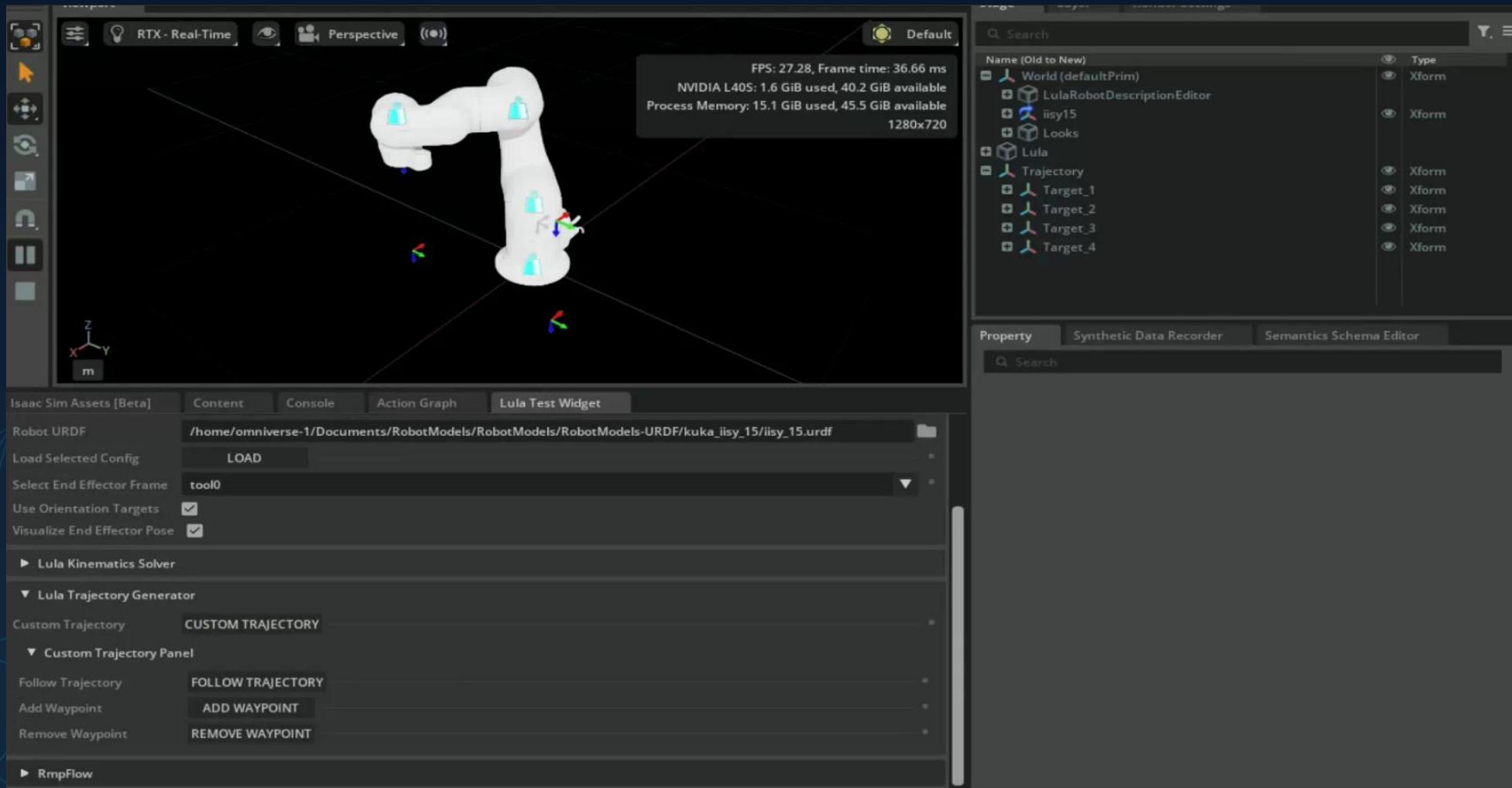
Sphere Approximation Result



Lula Test Widget



Lula Test Widget



Sensors

- Adding a Wrist Camera to the Robot
 - Tune the Camera

End of Day 1

Please provide some Feedback

Verbal or via:



For further questions mail us:
fabian.fichtl@hs-kempten.de
julian.zuern@hs-kempten.de

Omniverse Base Workshop 1

KUKA X IPI

- 2x 4h Course with examples to
 - Learn the Basics in Omniverse, USD & Isaac Sim
 - Enable you to do Robotic model preparation & Simulation
 - Introduction to IsaacLab

Review of Day 1

- We reviewed necessary Omniverse Basics and Resources
- Practical session:
 - Extensions installation
 - Scene Manipulation
 - Robot import + tuning

Agenda Tag 2:

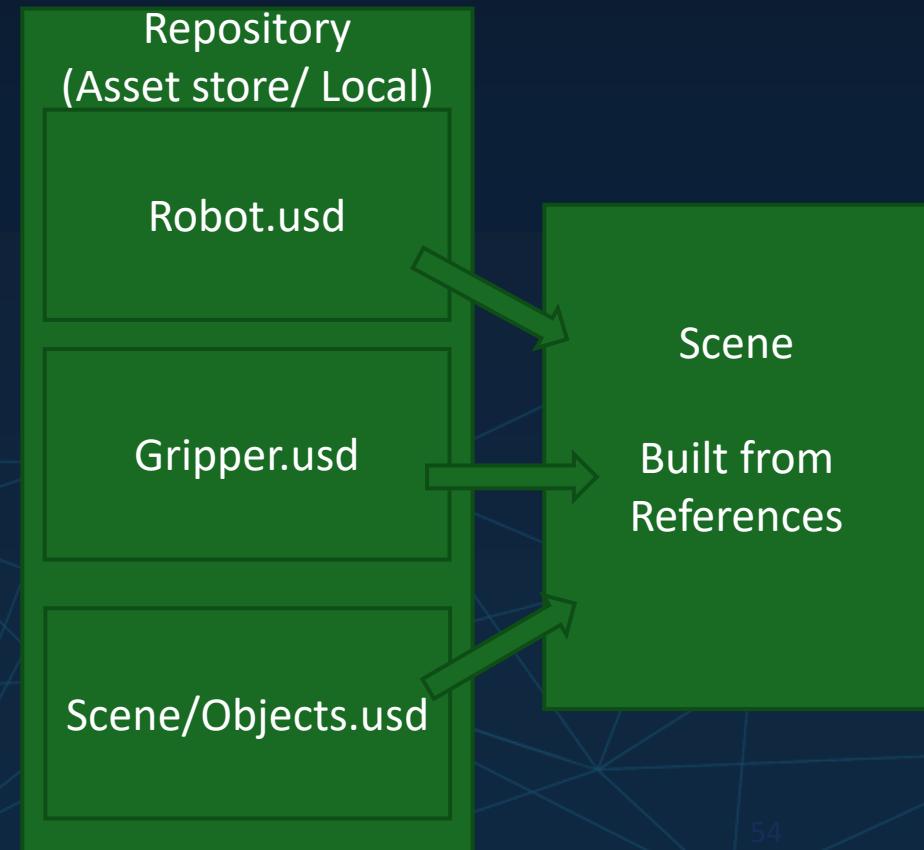
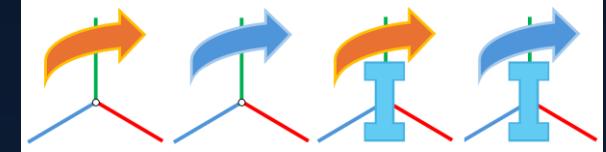
- Scene Architecture
 - Build your Scenario: Robot, Gripper, Controller, Environment
- Debugging Tools:
 - Console, VS Code, UI
- Materials
- Motion Planning & Motion Learning
- Practical Session: Pick & Place
- Information Channels | Further Courses

Interacting with USD (in Omniverse scene)

- UI Main Capabilities
 - Viewport:
 - Scene
 - Property Window:
 - Prim* Properties
 - Omnigraph:
 - Process Logic
 - ROS Controller
 - Various extensions
- API Main Capabilities
 - Python interface
 - Robot Controllers
 - URDF import
 - Create the scene for IsaacLab
 - Configure the policy
 - Start training

Scene Assembly

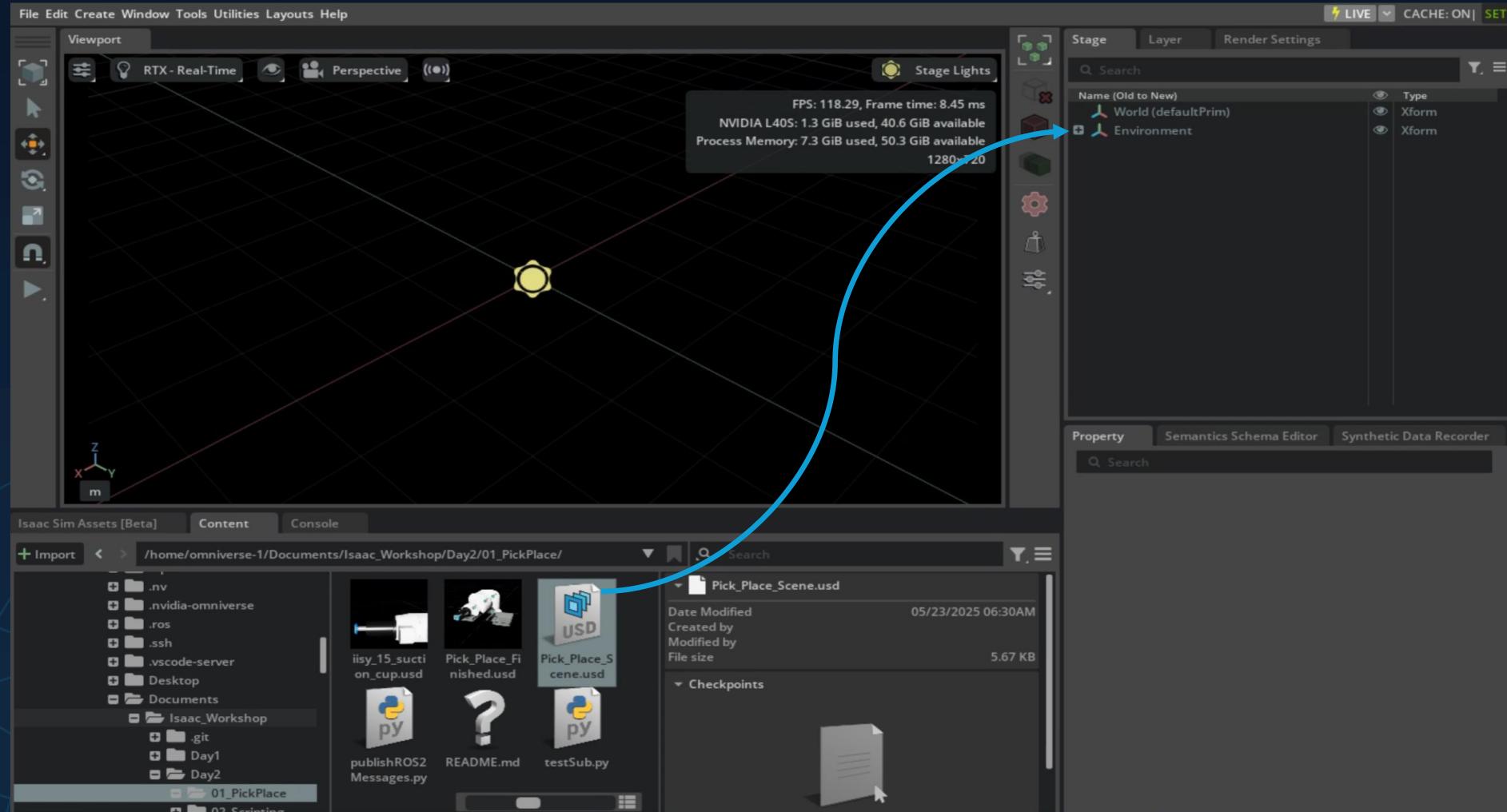
- Payload: blue ↔ Reference: orange
 - The user can choose to not load a payload
 - References always get loaded
 - I = Instancing active
- How to build a reference scene architecture
 - To test different robots/ grippers
 - Test Script of Robot Assembly



Assemble Your Scene

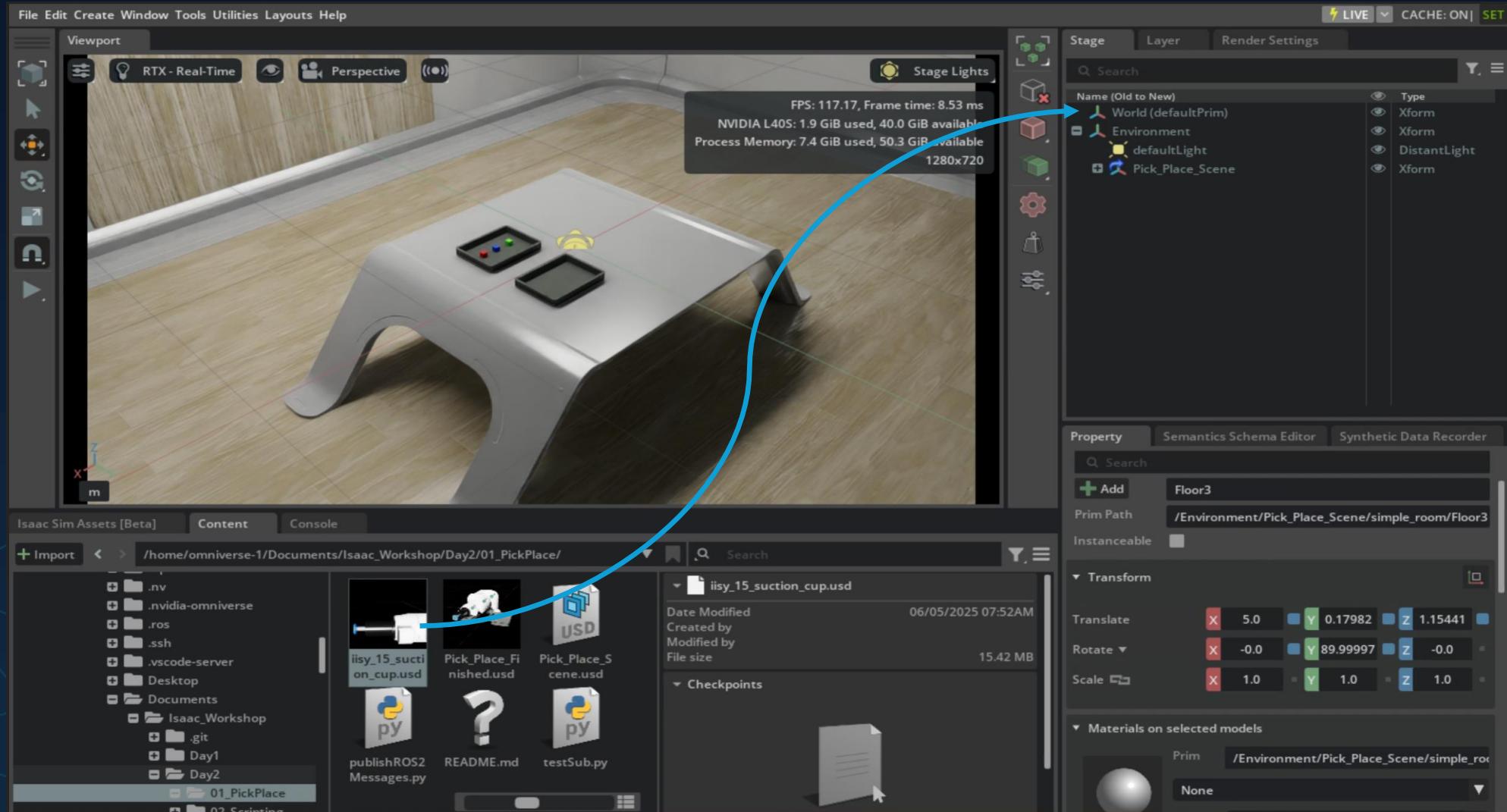


How-to: Add the Environment



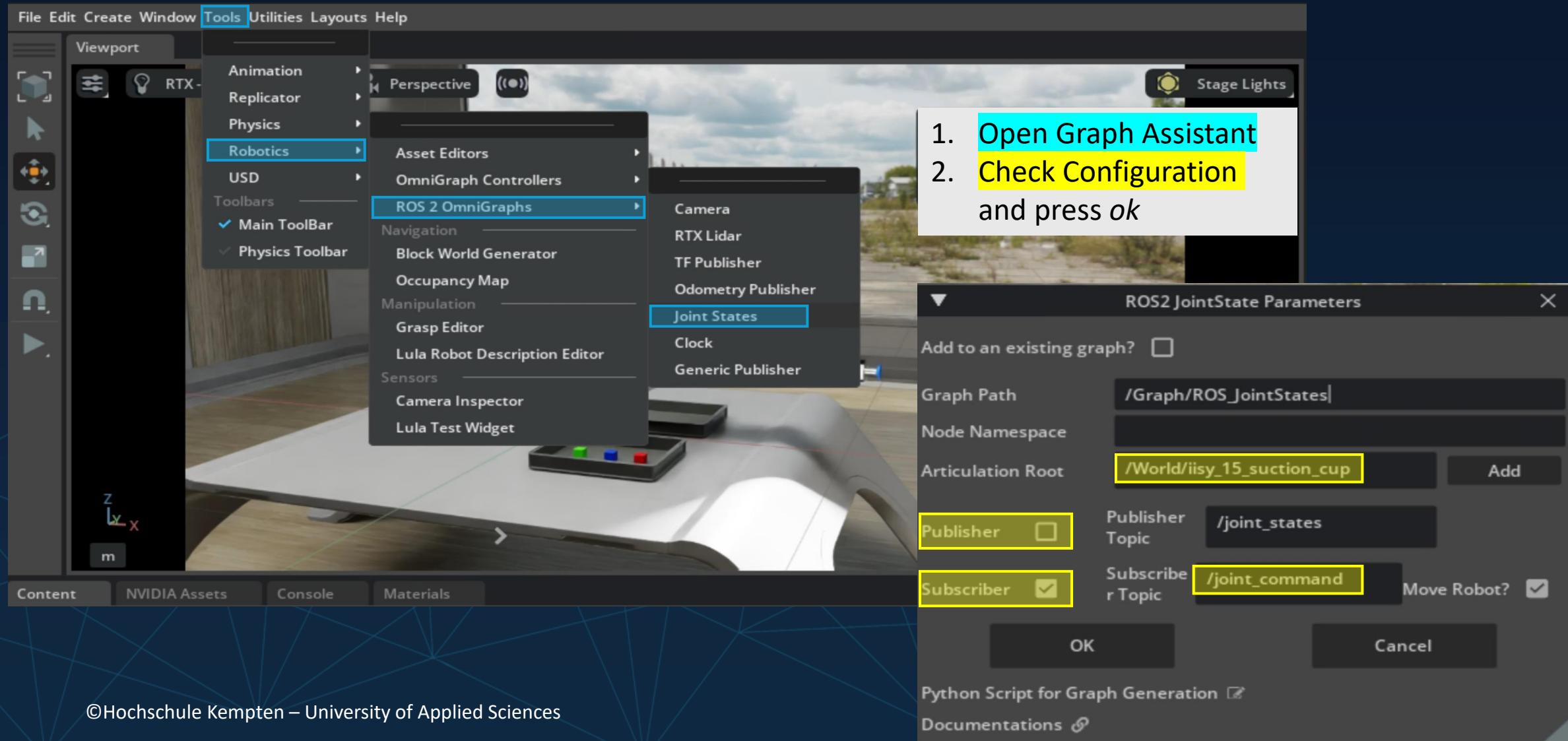
1. Drag USD scene from content browser onto Stage tree

How-to: Add the Robot



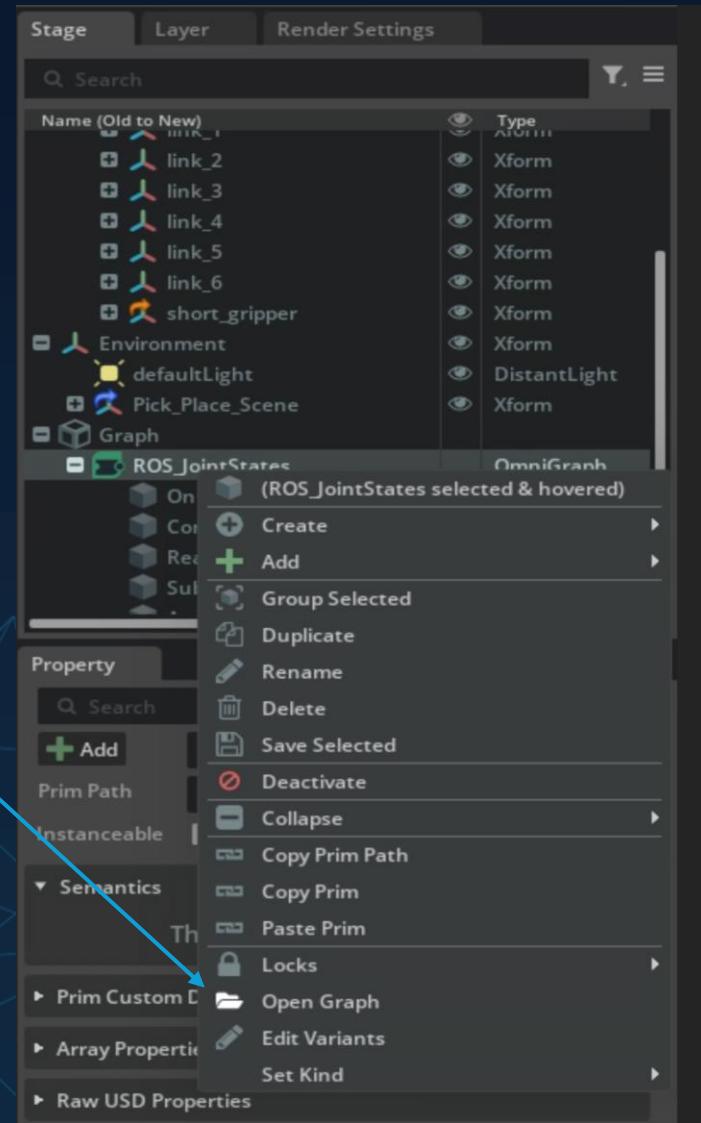
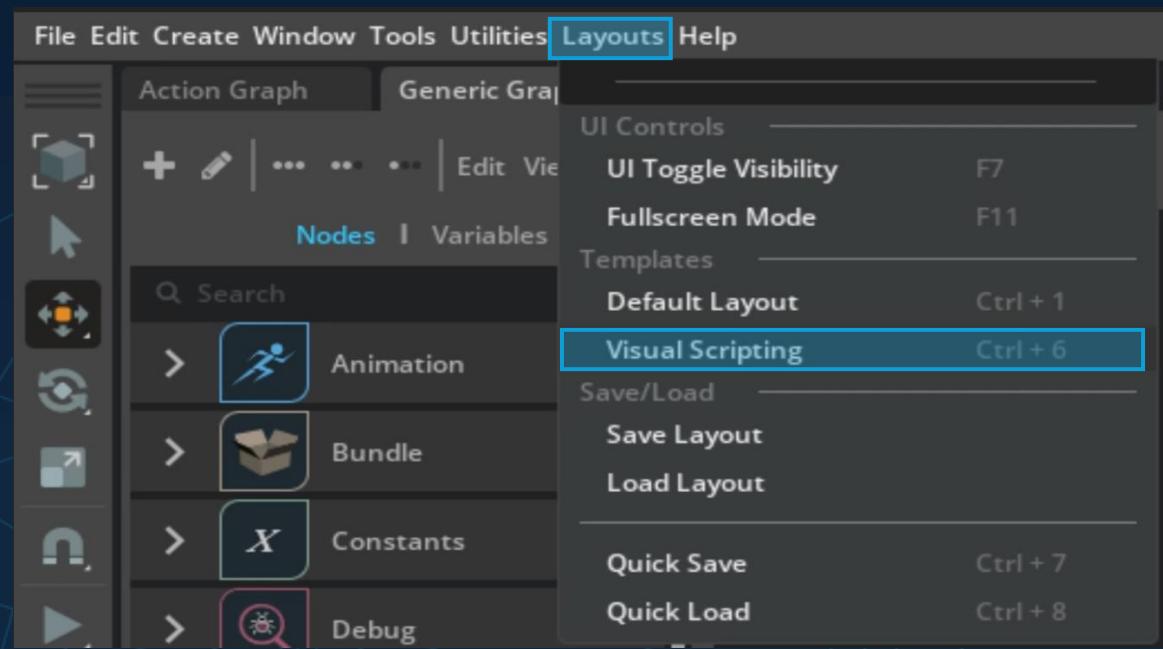
1. Drag robot from content browser onto Stage tree

How-to: Add the Controller



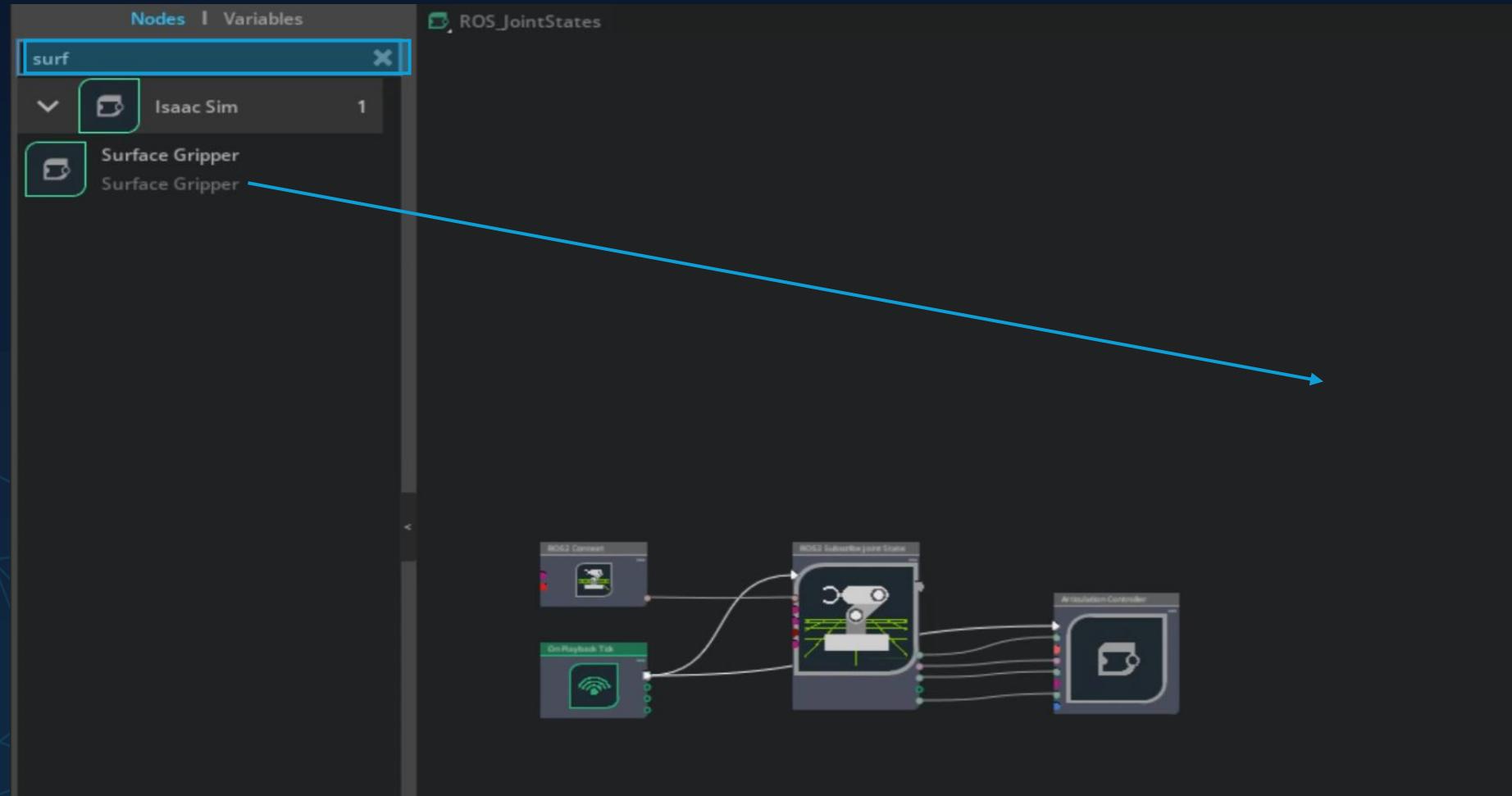
How-to: View the Controller

- Use Context menu in Stage
- or
- Use *Visual Scripting Layout*
(*Ctrl+6 -> 'Edit Graph'*)

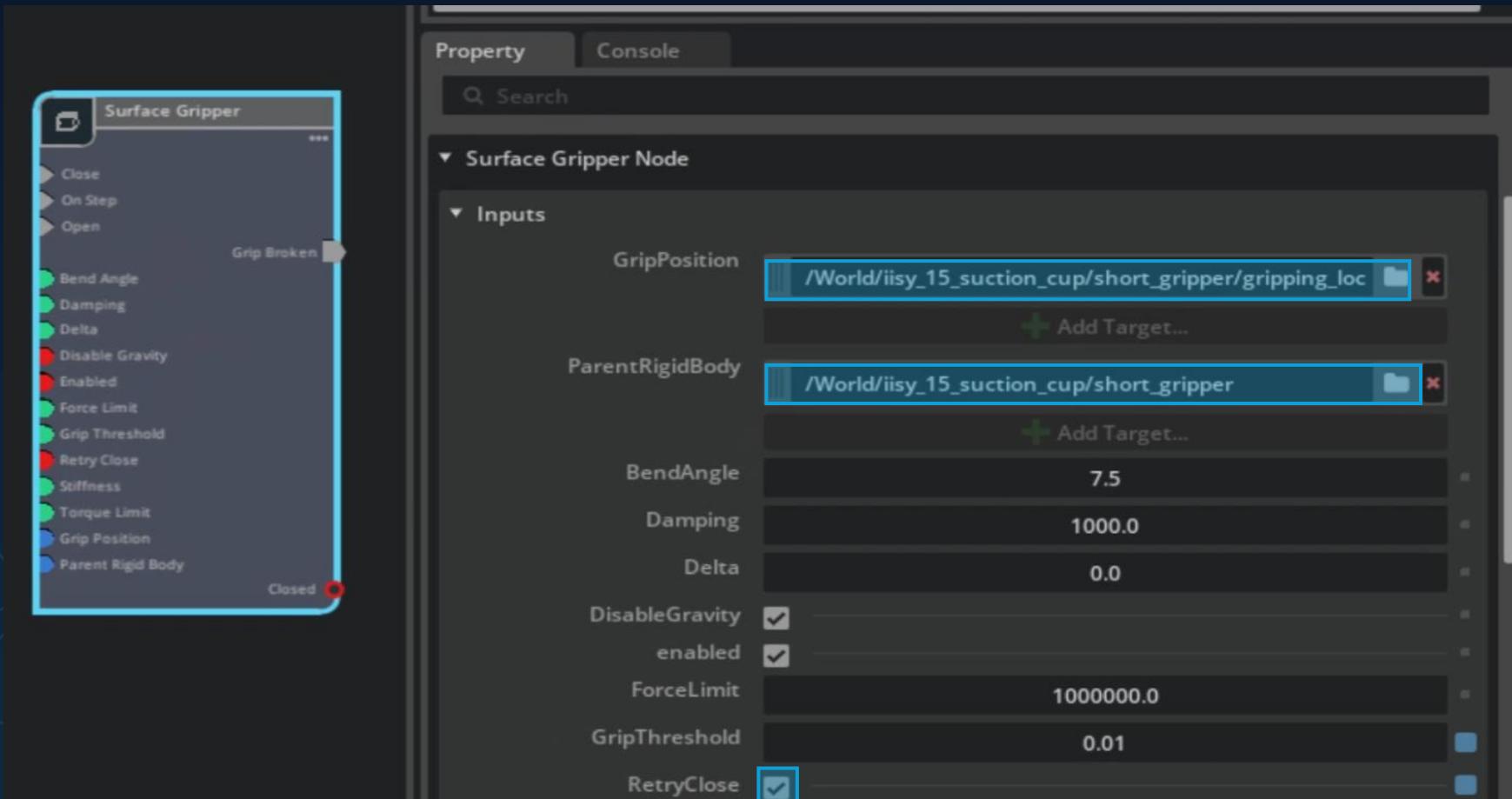


How-to: Add the Surface Gripper

- Search for node:
Surface Gripper
- Drag node into ActionGraph

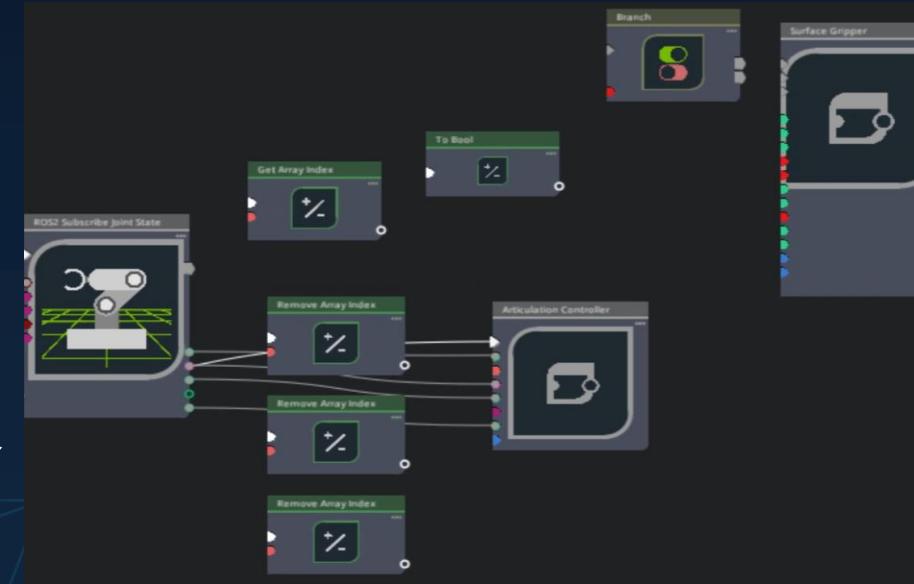


How-to: Configure the Surface Gripper



How-to: Process ROS Messages: Adding the Nodes

- Search for terms and drag the nodes to the designated location
 - “Branch”: Add Branch node in front of surface Gripper
 - “Index”: Add Get Array Index node after ROS2 Subscription
Set Property Inputs:Index to **6**
 - “Index”: Add 3x Remove Array Index node vertically aligned below *Get Array Index*
Set Property Inputs:Index to **6**
 - “Bool”: Add To Bool node in between *Get Array Index* and *Branch*



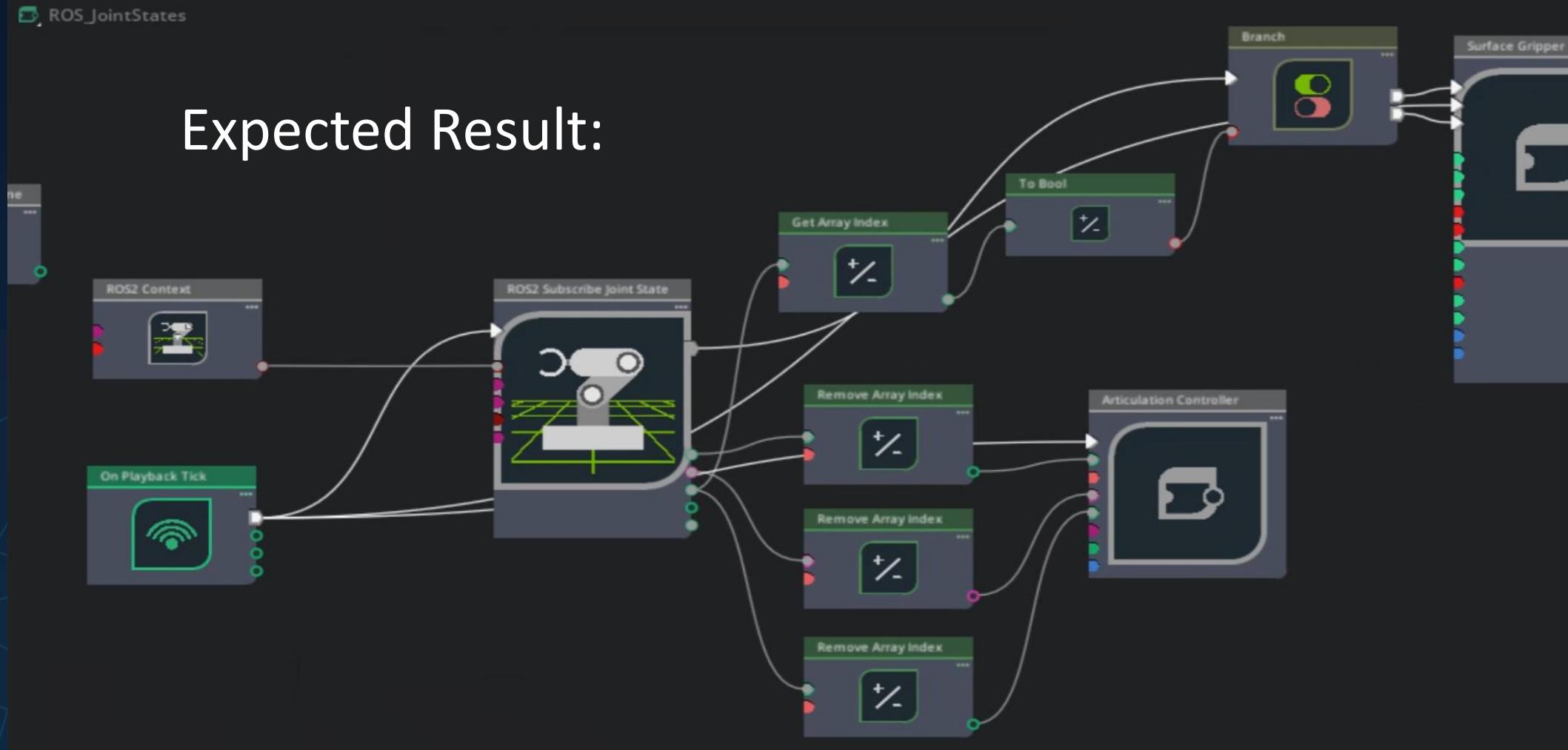
In front of = to the left of
After = to the right of

How-to: Process ROS Messages: Connecting the Nodes

- Connect Ports by Drag&Drop:
 - **OnPlaybackTick:Tick** → **SurfaceGripper:On Step**
 - **Branch:False** → **SurfaceGripper:Close**
 - **Branch:True** → **SurfaceGripper:Open**
 - **ROS2Sub.:Exec Out** → **Branch:Input execution**
 - **ToBool:Bool** → **Branch:Condition**
 - **GetArrayIndex:Value** → **ToBool:Value**
 - **ROS2Sub.:Position Command** → **GetArrayIndex :Array**
- Remove the **ROS2Sub.:Velocity Command** Connection (Context menu on connection)
- Replace remaining **ROS2Sub.** → **Artic.Contr.** connections with **ROS2Sub.:[x]** → **RemoveArrayIndex:Array** → **Artic.Contr.:[x]**

How-to: Process ROS Messages: Connecting the Nodes

Expected Result:



How-to: Run ROS2 Script in VS Code

- Connect to*/Open your Workspace in VS Code
- Open *publishROS2Messages.py*
- *Inspect it*
- *Start the Scene in Isaac Sim*
- *Run the Script from the VS Code Terminal:*

```
$ [your Isaac installation root]/python.sh ./Day2/01_PickPlace/publishROS2Messages.py
```

- *Watch the Pick and Place Process in Isaac Sim*

Assemble Your Scene



ee... end_effector

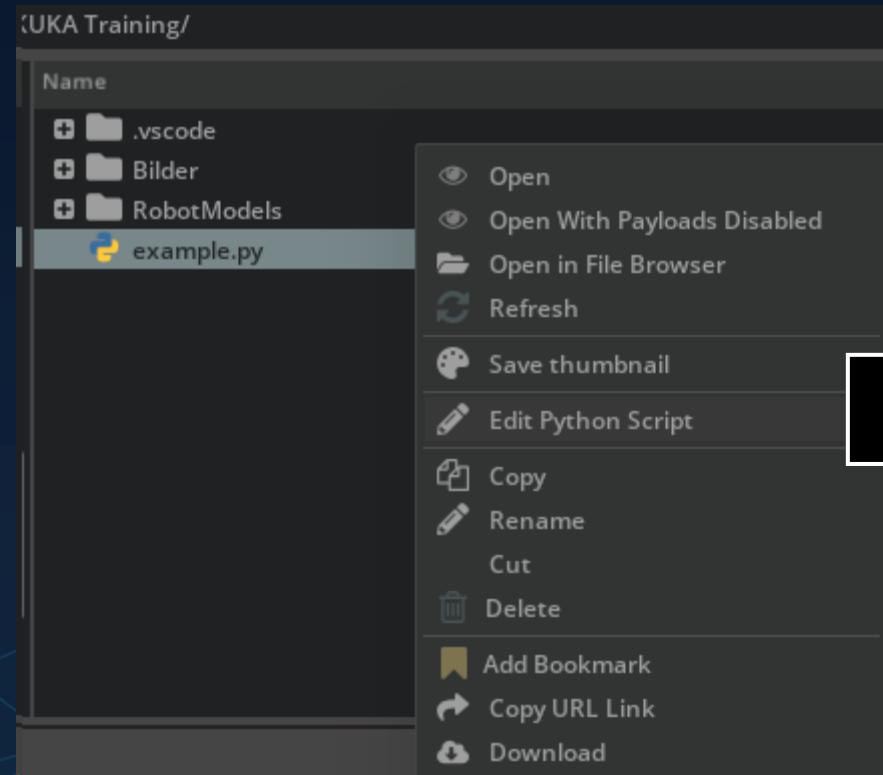
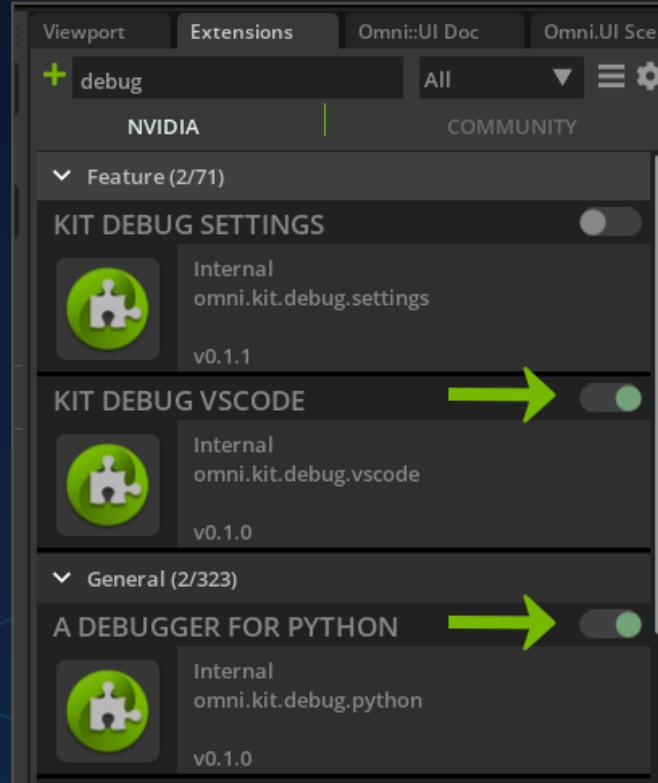
Pick & Place Phases

- Phase 0: Move above the cube center at the 'end_effector_initial_height'.
- Phase 1: Move *ee* down to encircle the target cube
- Phase 2: Wait for Robot's inertia to settle.
- Phase 3: Close grip.
- Phase 4: Move *ee* up again, keeping the grip tight (lifting the block).
- Phase 5: Move the *ee* toward the goal xy, keeping the height constant.
- Phase 6: Move *ee* vertically toward goal height at the 'end_effector_initial_height'.
- Phase 7: Loosen the grip.
- Phase 8: Move *ee* up again at the 'end_effector_initial_height'
- Phase 9: Move *ee* towards the old xy position.

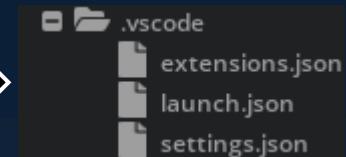
Save Scene

- Use the USDA format
- Inspect the structure and *payloads*

Debugging Tools



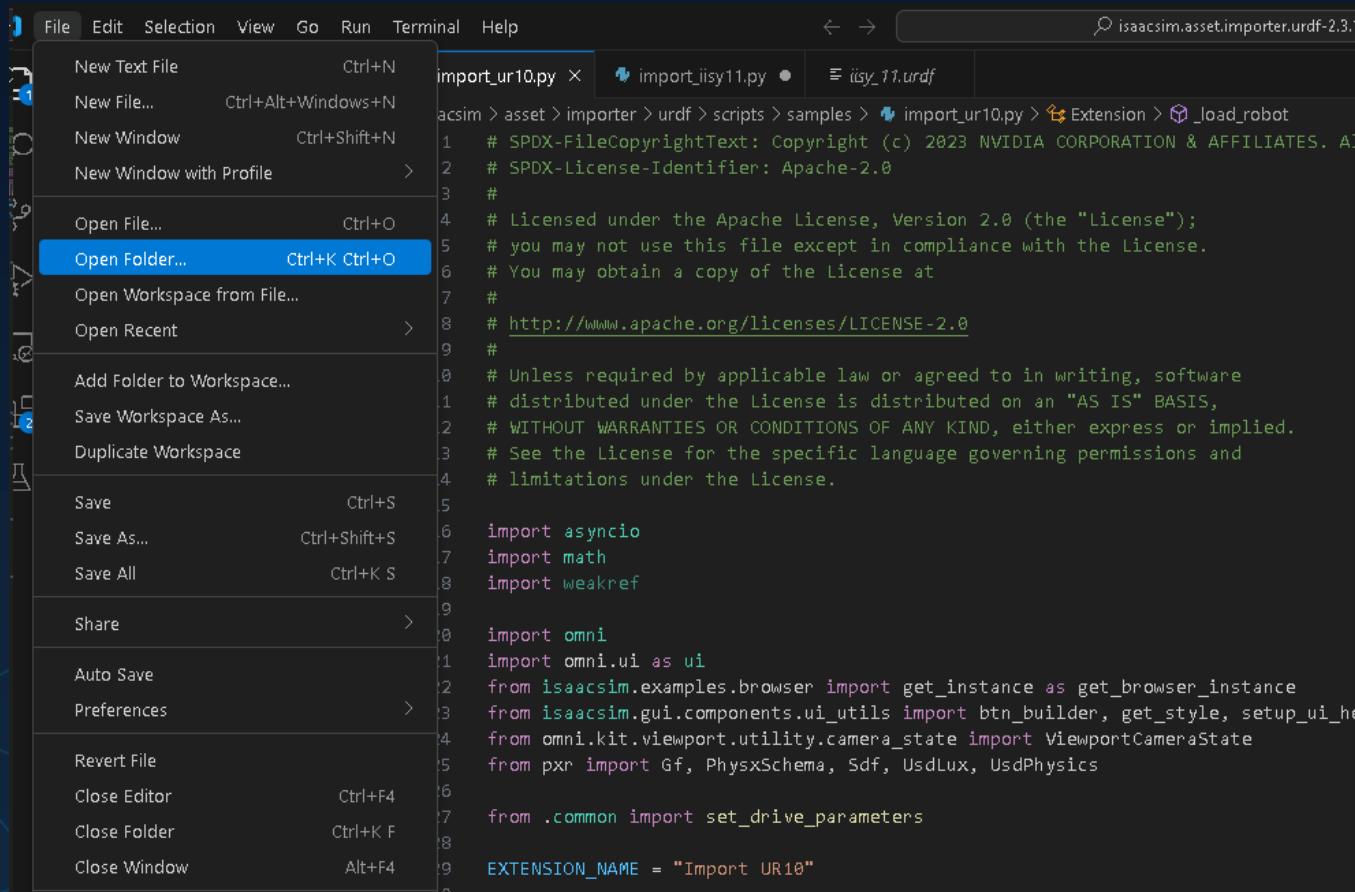
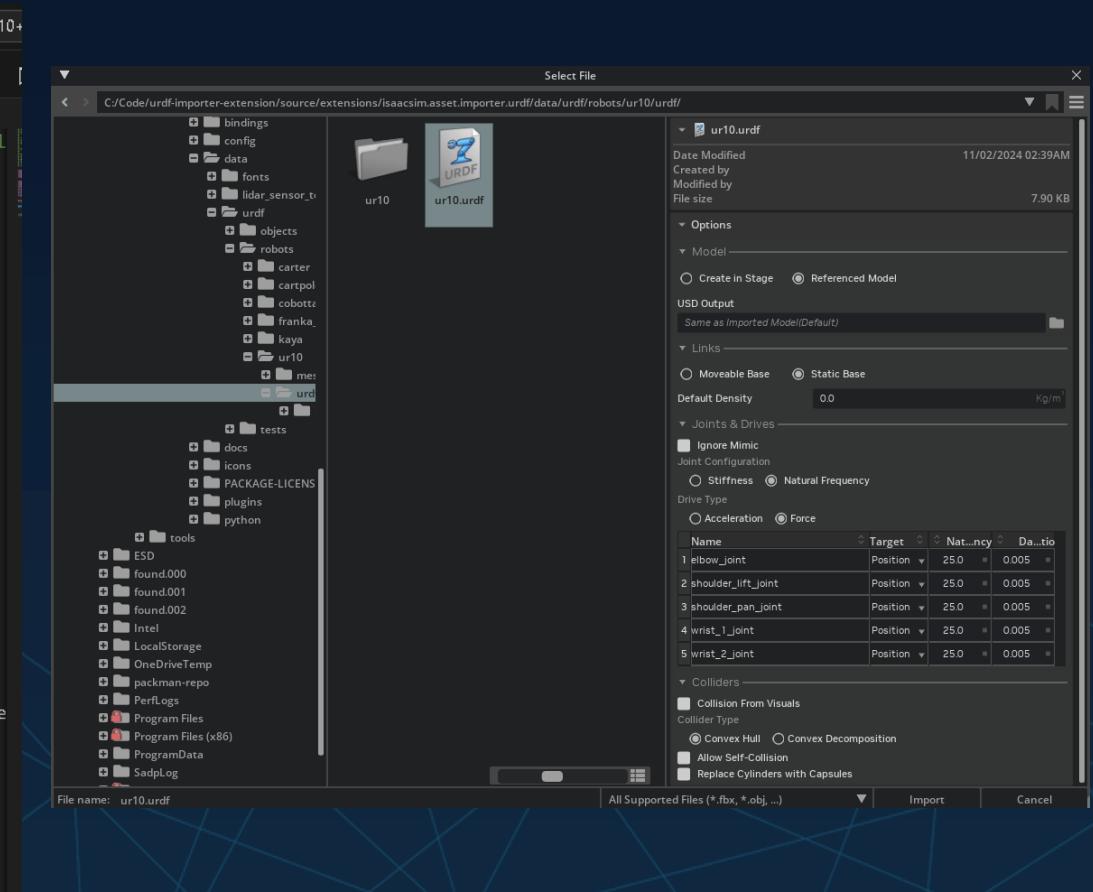
creates



https://docs.omniverse.nvidia.com/extensions/latest/ext_vs-code-link.html

If you open .py in OV, the VS Code Link dependencies are created automatically
-> Enables hot reload of code

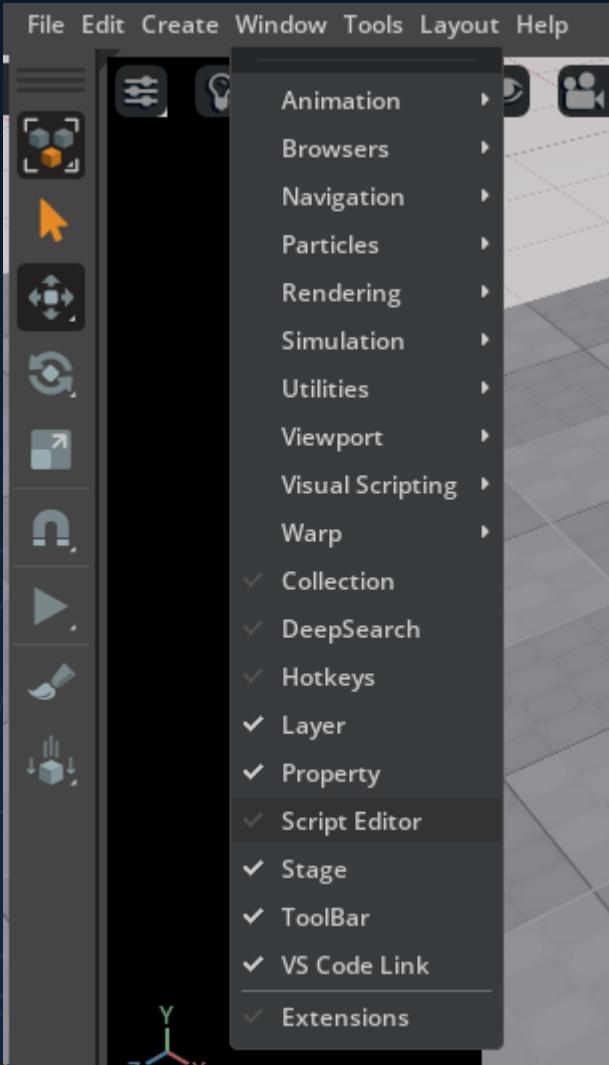
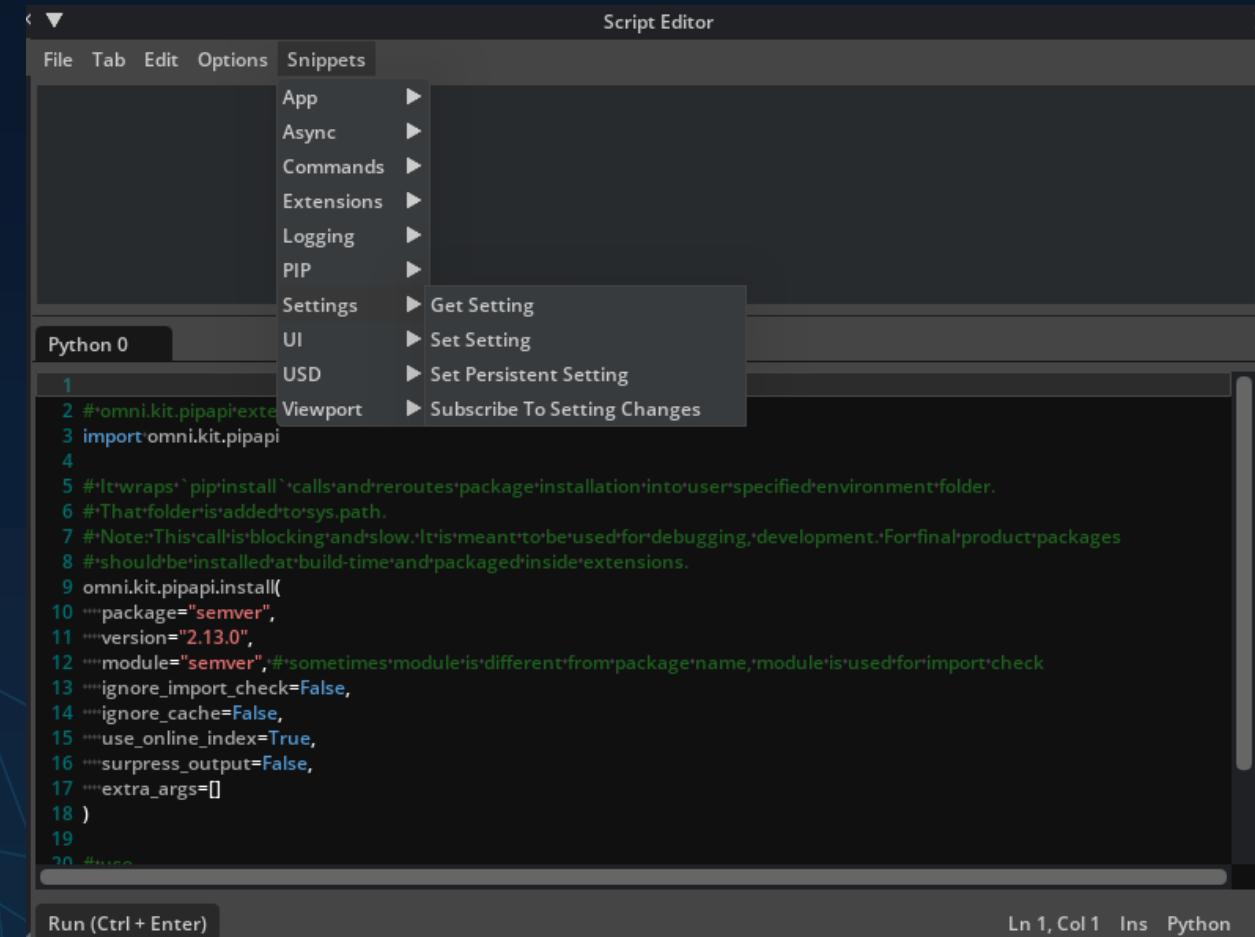
Debugging Tools

The image shows two windows illustrating the use of debugging tools. On the left is a code editor with several tabs open, displaying Python code related to IsaacSim asset importers and URDF imports. One tab shows a license header and another shows import statements and function definitions. On the right is a file browser window showing the structure of a USD file containing URDF data. The browser displays a tree view of assets and specific URDF files for a UR10 robot, along with various configuration options and joint parameters.

Helpful to reuse example code or look into USDA file

Debugging Tools

The screenshot shows a 'Script Editor' window titled 'Script Editor'. The menu bar includes 'File', 'Tab', 'Edit', 'Options', 'Snippets', and 'Python 0'. The 'Snippets' menu is open, showing options like 'App', 'Async', 'Commands', 'Extensions', 'Logging', 'PIP', 'Settings', 'UI', 'USD', and 'Viewport'. A context menu is also open over the 'Settings' option, with choices 'Get Setting', 'Set Setting', 'Set Persistent Setting', and 'Subscribe To Setting Changes'. The main editor area contains the following Python code:

```

1 #`omni.kit.pipapi`executes
2 #`pip`install`calls`and`reroutes`package`installation`into`user`specified`environment`folder.
3 import omni.kit.pipapi
4
5 #`It`wraps` `pip`install`calls`and`reroutes`package`installation`into`user`specified`environment`folder.
6 #`That`folder`is`added`to`sys.path.
7 #`Note:`This`call`is`blocking`and`slow.`It`is`meant`to`be`used`for`debugging`.`development`.`For`final`product`packages
8 #`should`be`installed`at`build`time`and`packaged`inside`extensions.
9 omni.kit.pipapi.install(
10     package="semver",
11     version="2.13.0",
12     module="semver", #`sometimes`module`is`different`from`package`name,`module`is`used`for`import`check
13     ignore_import_check=False,
14     ignore_cache=False,
15     use_online_index=True,
16     suppress_output=False,
17     extra_args=[]
18 )
19
20 #`...

```

At the bottom of the editor are buttons for 'Run (Ctrl + Enter)' and 'Ln 1, Col 1 Ins Python'.

Script Editor

Manipulate the Scene by Python API

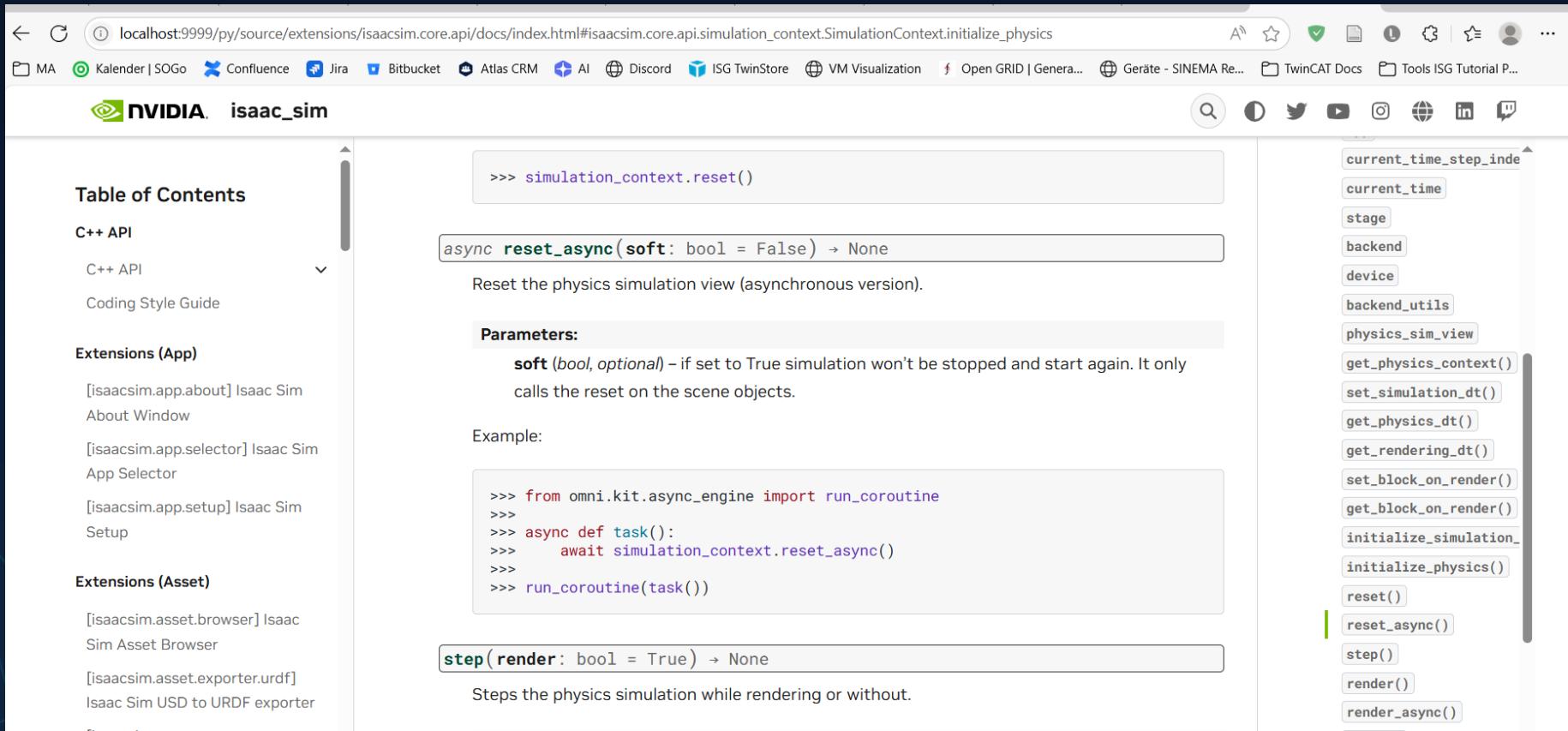
```
import omni.usd
import omni.kit.commands from pxr
import Gf
# Get current stage
stage = omni.usd.get_context().get_stage()
# Create cube with specified parameters
result, prim_path = omni.kit.commands.execute(
    "CreateMeshPrim",
    prim_type="Cube",
    prim_path="/World/Cube",
    object_origin=Gf.Vec3f(0, 0, 0),
    half_scale=50.0,
    u_verts_scale=1,
    v_verts_scale=1,
    w_verts_scale=1 )
```

- Open the Script Editor

- Open the file *scene_Manipulation.py*.
- Run it
- Try to create another prim for example a cone:

```
# Create cone with specified parameters
result, prim_path = omni.kit.commands.execute(
    "CreateMeshPrim",
    prim_type="Cone",
    prim_path="/World/Cone",
    object_origin=Gf.Vec3f(0, 0, 0),
    half_scale=50.0,# 1m Height/Diameter (dep. on def.)
    u_verts_scale=1,
    v_verts_scale=1,
    w_verts_scale=1 )
```

Documentation



The screenshot shows a browser window displaying the Isaac Sim API documentation. The URL in the address bar is `localhost:9999/py/source/extensions/isaacsim.core.api/docs/index.html#isaacsim.core.api.simulation_context.SimulationContext.initialize_physics`. The page title is "NVIDIA isaac_sim". On the left, there is a "Table of Contents" sidebar with sections for C++ API, Coding Style Guide, Extensions (App), and Extensions (Asset). The main content area shows the `reset()` method documentation. It includes a code example for the synchronous version:

```
>>> simulation_context.reset()
```

and for the asynchronous version:

```
async reset_async(soft: bool = False) → None
```

The description states: "Reset the physics simulation view (asynchronous version)." Below this, there is a "Parameters:" section and an "Example:" section with the following code:>>> from omni.kit.async_engine import run_coroutine
>>>
>>> async def task():
>>> await simulation_context.reset_async()
>>>
>>> run_coroutine(task())

On the right side of the page, there is a vertical sidebar listing various methods and functions, likely part of the API documentation structure.

<https://docs.isaacsim.omniverse.nvidia.com/4.5.0/py/index.html>

Online or Local
(latter might not be available)

\$ cd isaacsim/docs
\$ python3 -m http.server 9999

Referencing Prims

- Create new scene through GUI
- Open Script Editor
- Open File *assemble_scene.py*
- Analyze Content...
- Run Script

Materials & physics Materials

Materials:

- Color Map
- Reflectivity
- Absorption
- Many others and mixtures/ Shaders



Physics Materials:

- Static Friction
- Dynamic Friction
- Restitution
- Density



Until now:

- Scene Architecture
 - Build your Scenario: Robot, Gripper, Controller, Environment
- Debugging Tools:
 - Console, VS Code, UI
- Materials

10 min Break

To be continued:

- Motion Planning & Motion Learning
- Practical Session: Pick & Place

Motion Planning Libraries

- Lula

- Kinematics Solver (fast-forward and inverse kinematics)
- Trajectory Generator (using one or more planners)
- Global Planners (e.g. RRT-Connect, JT-RRT)
- Local Planners (e.g. RMPflow)
- CPU-based

- CuMotion

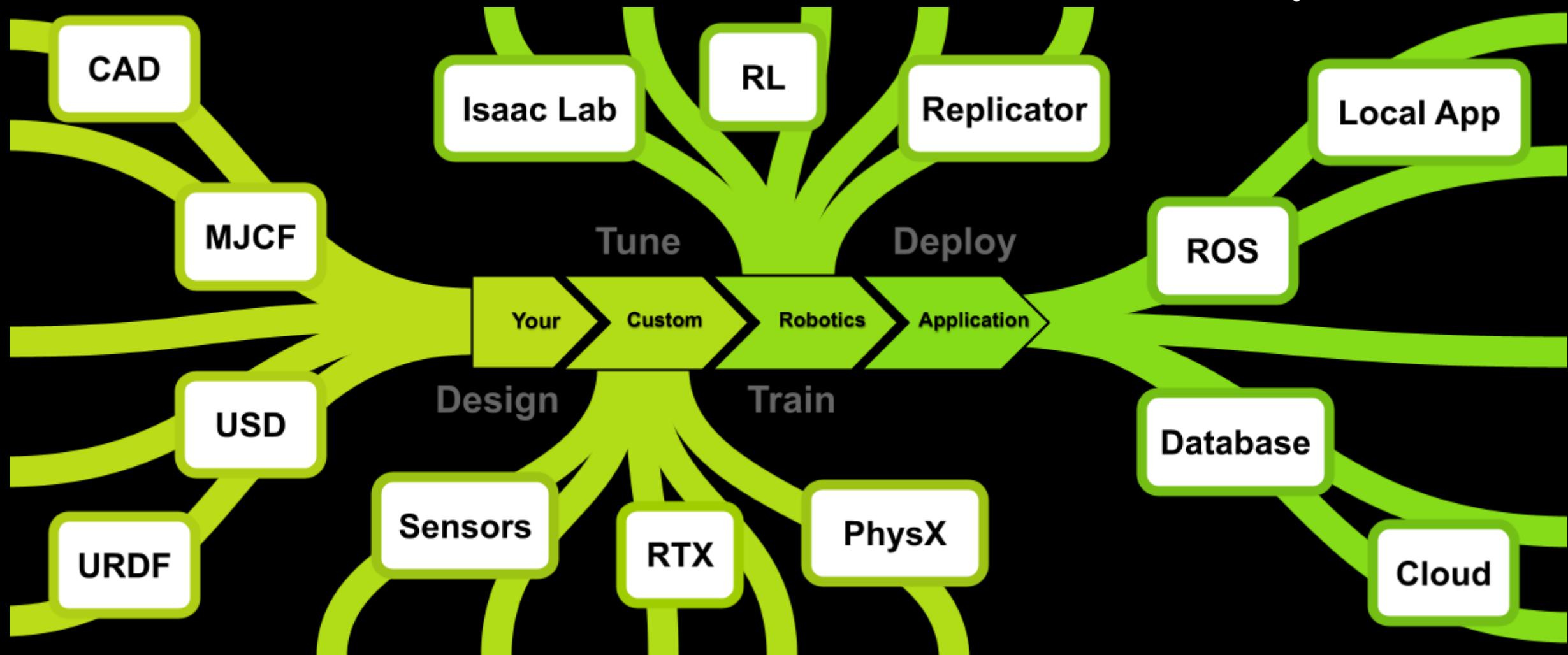
- Similar capabilities as Lula
- Supports real-time control
- Builds upon CuRobo (CUDA-accelerated)

⚠ Lula algorithms require a *robot_description.yaml*. It contains active and fixed joints, default joint positions, collision spheres and more. This format will be superseded by the XRDF-format.

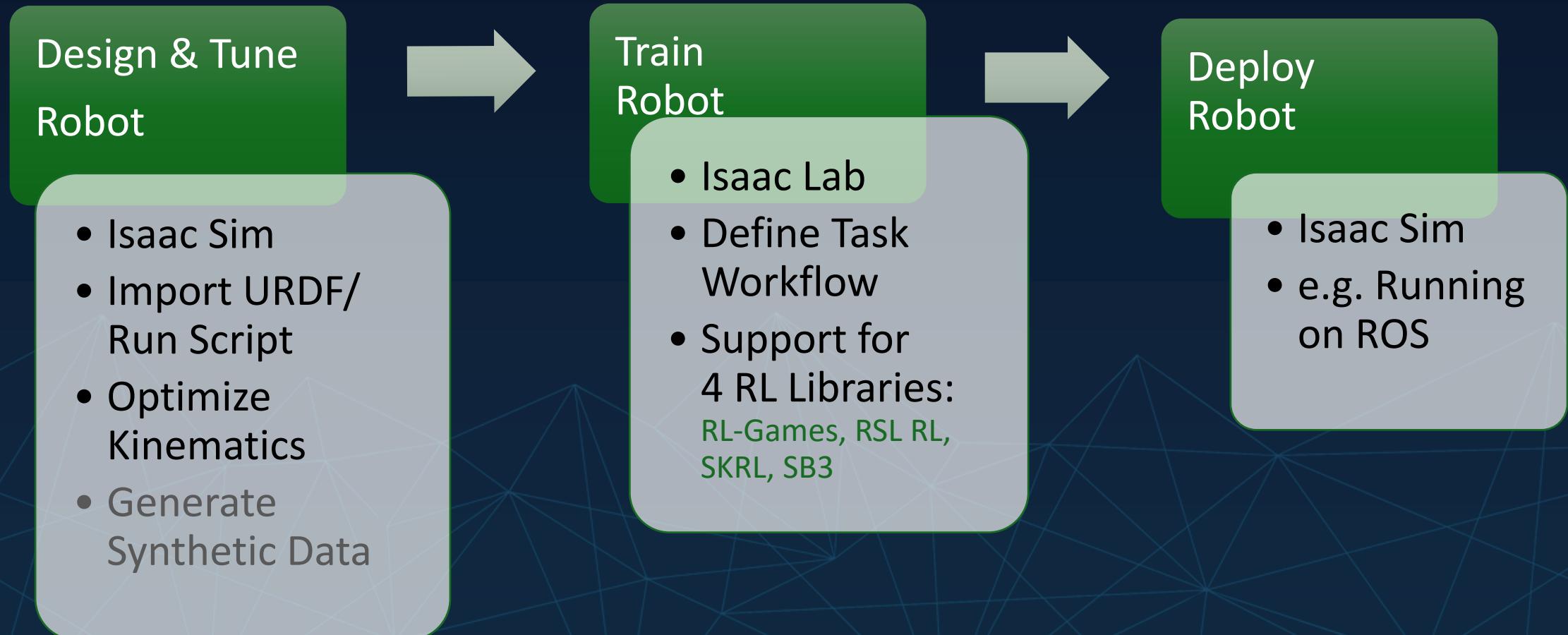
Prerequisites for CuMotion with RMP

- An URDF file, used for specifying robot kinematics as well as joint and link names. Position limits for each joint are also required.
- An (X)RDF file which can be generated using the Lula Robot Description Editor UI tool. (configure drives, robot density and collision meshes)
- A RMPflow configuration file in YAML format, containing parameters for all enabled RMPs.*

*For algorithms other than RMPflow different configurations and are required



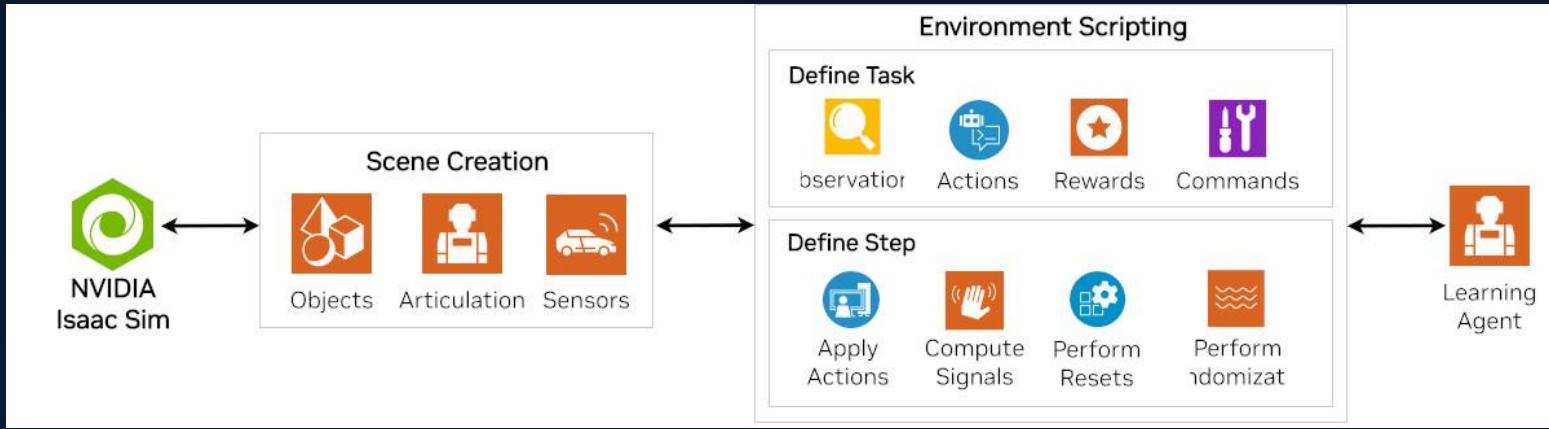
IsaacLab Workflow – High Level



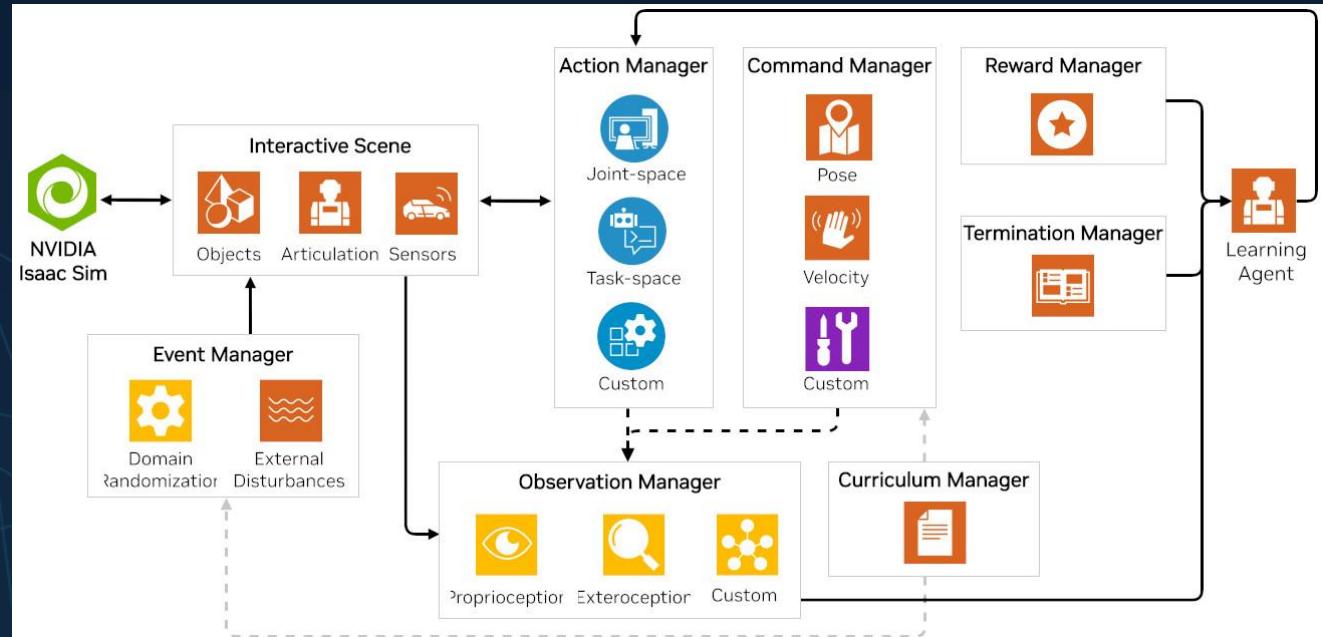


Task Workflows in IsaacLab

Direct RL Workflow



Manager-based RL Workflow



Integrating ML Models

- Training models on synthetic data generated in Isaac Sim or COSMOS
- Foundation model as perception or reasoning module within the Isaac Lab workflow
- Isaac Lab RL policy optimization
- Groot N1:
 - Requires Trajectory Data, Video
 - <https://github.com/NVIDIA/Isaac-GR0OT/blob/main/media/model-architecture.png>
- π 0.5
 - Sensory Data
 - <https://pi.website/blog/pi05>
- OCTO
 - Generalist Robotic Policy
 -  [Octo: An Open-Source Generalist Robot Policy](#)

How to generate Trajectory Data

- Classic: Real Robot „imitation learning“
- Trajectory generators
- Applied RL policies
- Teaching sim. Robot with imitation learning

Assemble a gripper to your robot

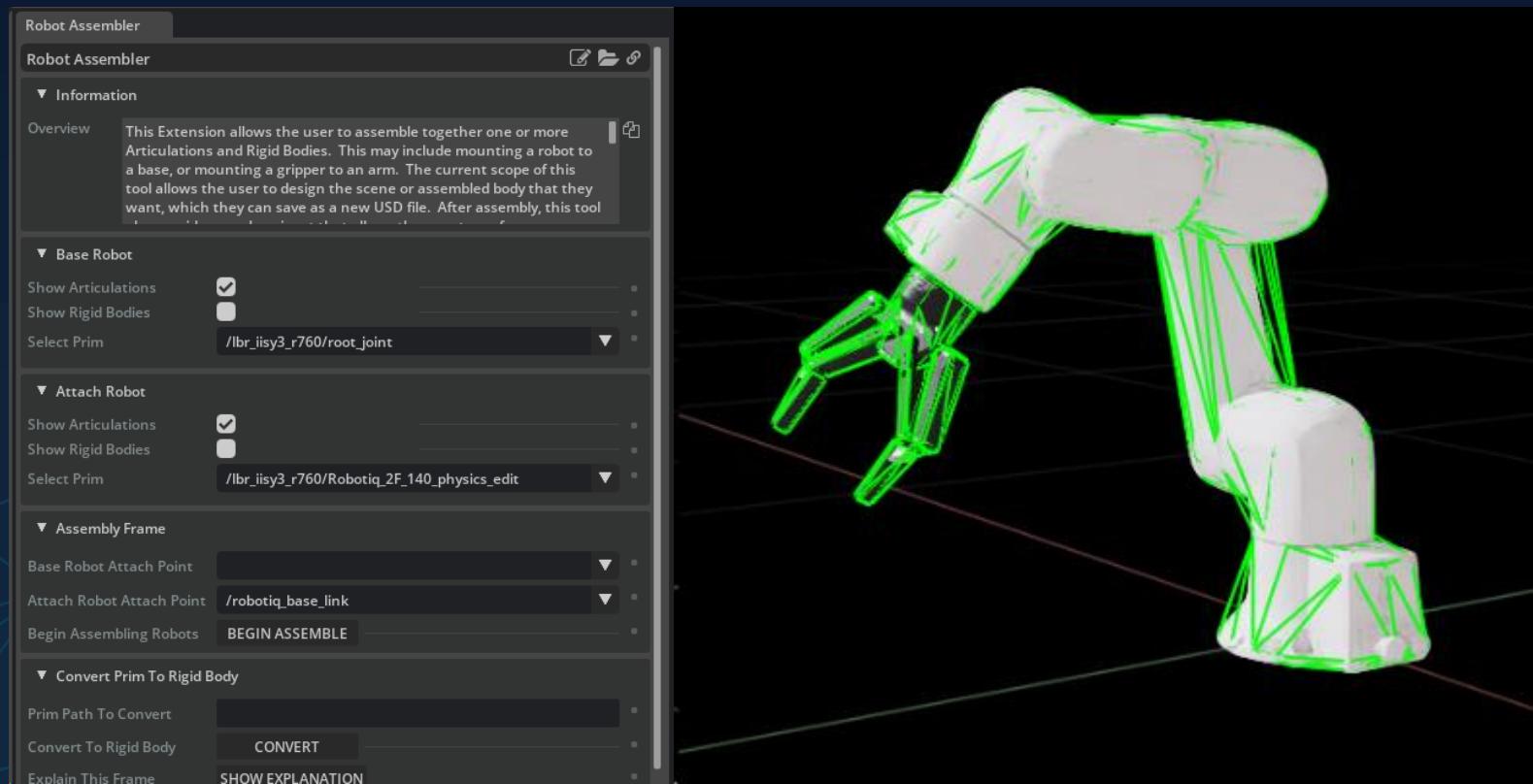
1. UI:

The Tool joins two articulation trees to one.

Guides you how

2. Python API

Code is created automatically from the UI. This can be reused for later automatically create a stage + assembly



Until now:

- Scene Architecture
- Build your Scenario: Robot, Gripper, Controller, Environment
- Debugging Tools:
 - Console, VS Code, UI
- Materials
- Motion Planning & Motion Learning

5 min Break

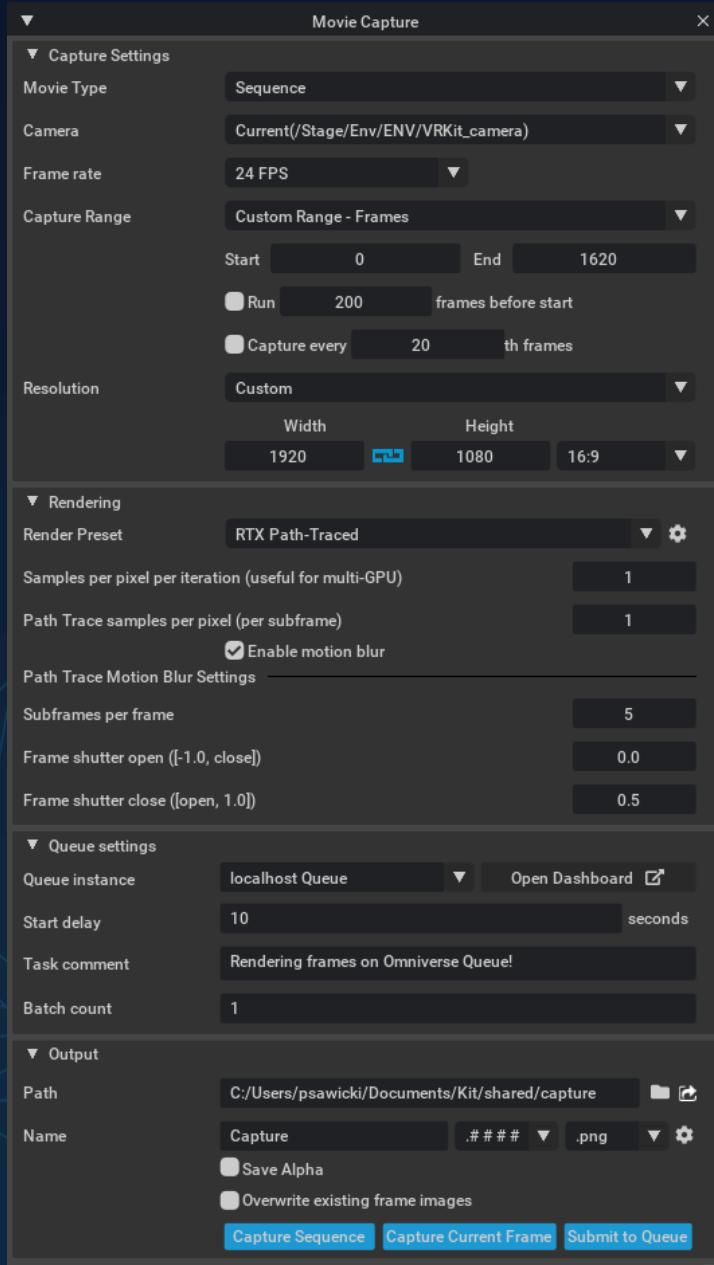
To be continued:

- Practical Session: Pick & Place
- Movie Capture

Movie Capture Tool

- Navigate to Window -> Extensions Manager using the top-level menu
- Search *Movie Capture* in the text field, to reveal the *omni.kit.window.movie_maker*.
- Click the toggle to enable the extension.

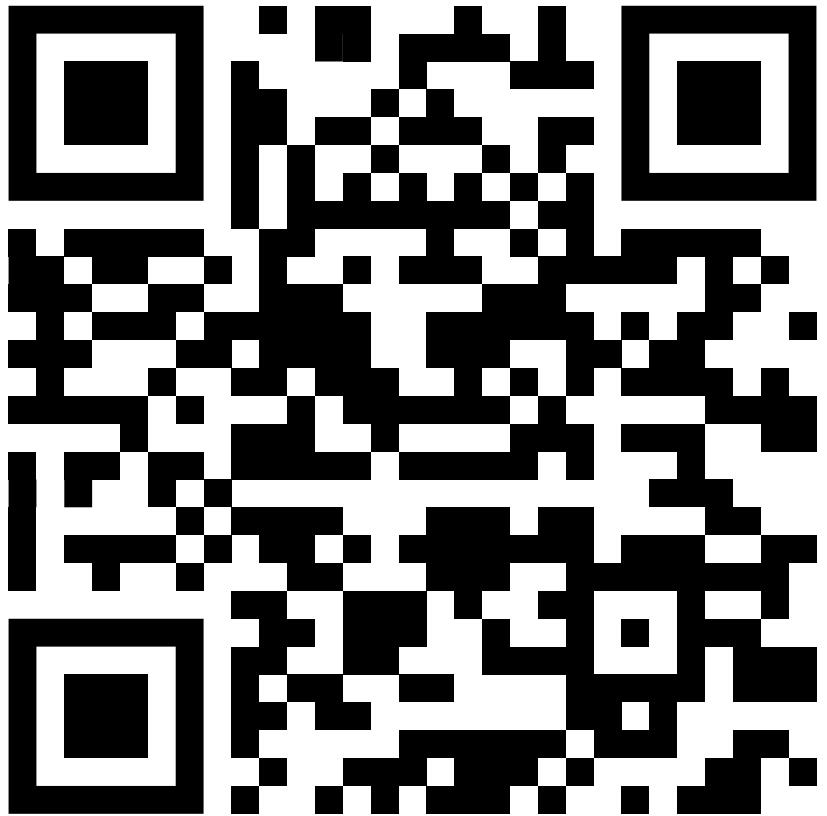
- Open the extension in rendering -> Movie Capture



Movie Capture + simple Animation



Feedback Workshop



For further questions mail us:
fabian.fichtl@hs-kempten.de
julian.zuern@hs-kempten.de

Ressources & further Courses



Omniverse Glossary

- Omniverse Glossary of Terms — Omniverse Developer Workstations

USD

Payload vs Reference:

<https://forums.developer.nvidia.com/t/usd-payload-vs-reference-vs/252658>

Joints

- https://nvidia-omniverse.github.io/PhysX/physx/5.3.0/_downloads/6acf3afb8f69452757e0e766b5a22978/implicitDrives.pdf

URDF Import

- https://docs.isaacsim.omniverse.nvidia.com/4.5.0/robot_setup/ext_isaacsim_asset_importer_urdf.html

Gripper

- https://docs.isaacsim.omniverse.nvidia.com/latest/robot_setup/assemble_robots.html
- https://docs.isaacsim.omniverse.nvidia.com/latest/robot_simulation/ext Isaacsim_robot_surface_gripper.html
- cobotta demo und anschließend mit iisy
- https://docs.isaacsim.omniverse.nvidia.com/latest/robot_setup/import_manipulator.html
- https://docs.isaacsim.omniverse.nvidia.com/latest/manipulators/manipulators_configure_rmpflow_denso.html#isaac-sim-app-tutorial-configure-rmpflow-denso

https://docs.isaacsim.omniverse.nvidia.com/latest/robot_setup/grasp_editor.html

Roboter konfiguration

- [https://docs.omniverse.nvidia.com/isaacsim/latest/advanced tutorials/tutorial_configure_rmpflow_denso.html](https://docs.omniverse.nvidia.com/isaacsim/latest/advanced_tutorials/tutorial_configure_rmpflow_denso.html)
- Articulation Root
 - > Mobil oder stationär
- https://docs.omniverse.nvidia.com/extensions/latest/ext_physics/articulations.html

Movelt

- <https://moveit.picknik.ai/main/index.html>
- <https://moveit.picknik.ai/main/doc/concepts/concepts.html>

Cumotion

- <https://nvidia-isaac-ros.github.io/concepts/manipulation/index.html#nvidia-cumotion>
- Isaac Manipulator consists of a set of components and reference workflows for advanced perception-driven manipulation. These components include state-of-the-art packages for object detection and object pose estimation, as well as obstacle-aware motion generation.

Task Space Controller

- https://isaac-sim.github.io/IaacLab/main/source/tutorials/05_controllers/run_diff_ik.html

Self paced Courses:

- https://learn.nvidia.com/courses/course?course_id=course-v1:DLI+S-OV-31+V1&unit=block-v1:DLI+S-OV-31+V1+type@vertical+block@bd7da7c3339a42c69f073ca845778391
- https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-OV-31+V1
- https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-OV-15+V1
- https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-OV-27+V1
- https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-OV-29+V1
- https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-OV-20+V1
- https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-OV-28+V1
- <https://academy.nvidia.com/en/course/nvidia-ai-enterprise-administration/?cm=55144>