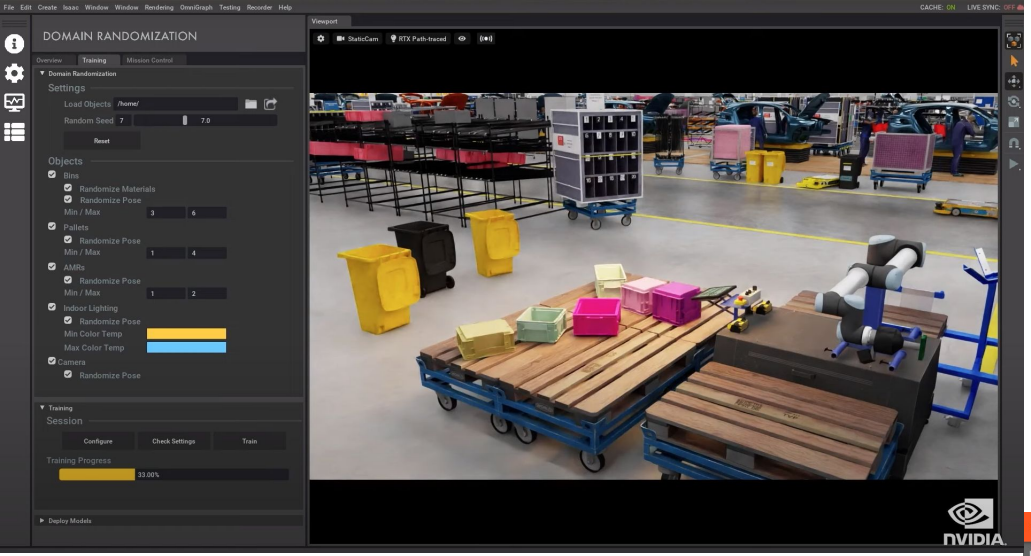
Best of Both Virtual Worlds: Bridging Ignition Gazebo & Isaac Sim

Alejandro Hernández Cordero & Brian Gerkey

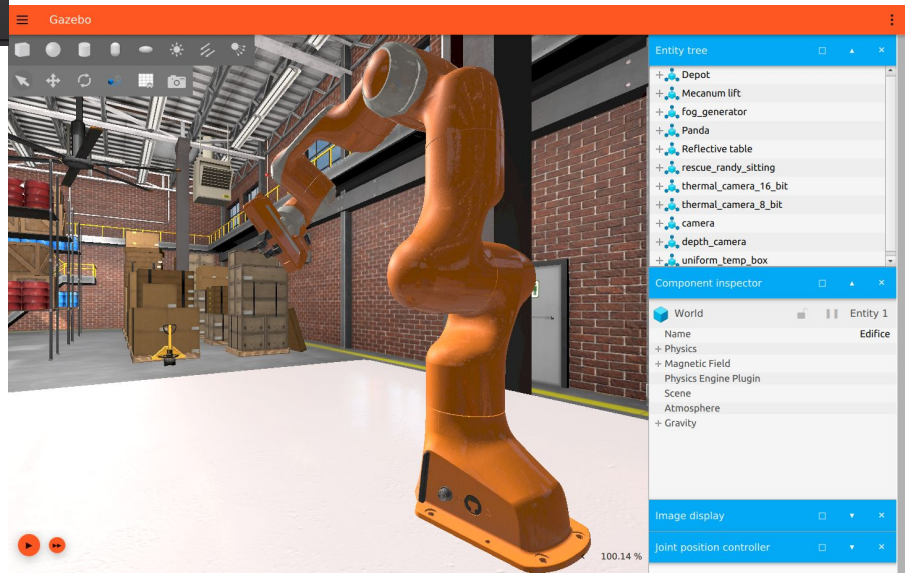
NVIDIA GTC 2022

open robotics



Isaac
Sim

Ignition
Gazebo



open
robotics

Ignition Gazebo

Open source toolbox of libraries and cloud services to make robot simulation easy

Iterate quickly in realistic environments with high fidelity sensors streams

Run simulation in continuous integration tests

open robotics

Isaac Sim

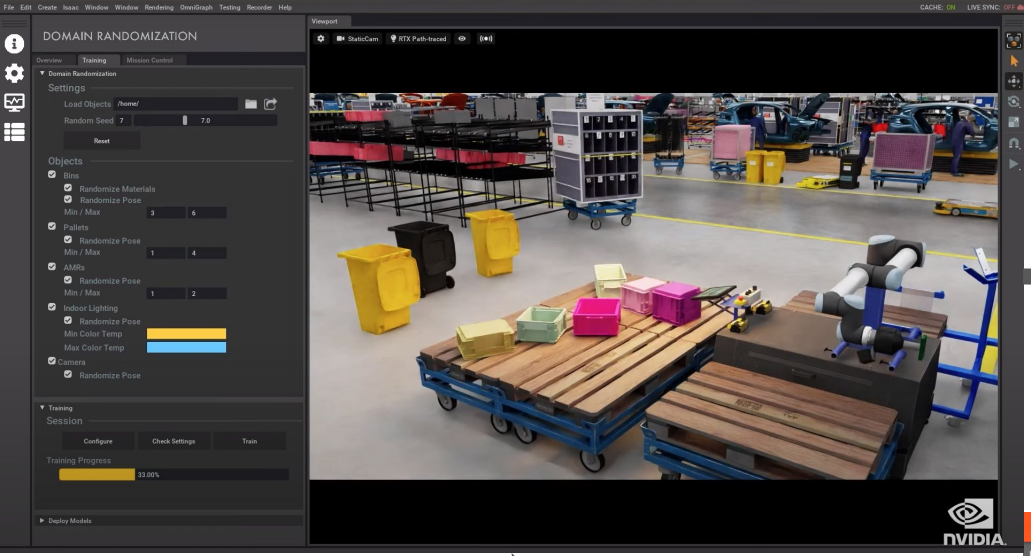
Scalable robotics simulation application and synthetic data generation tool

Photorealistic, physically-accurate virtual environments

Develop, test, and manage AI-based robots

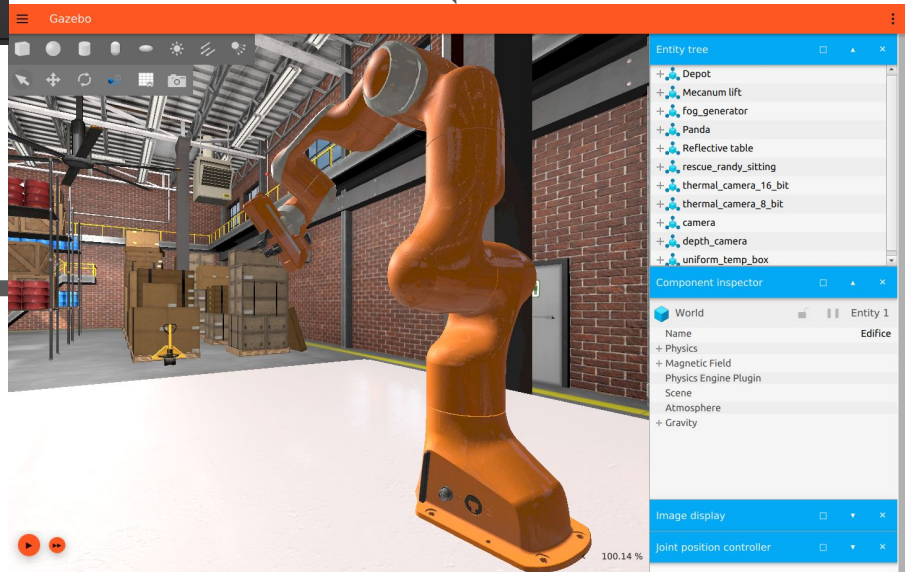
open robotics

"No simulator can address every robotics simulation challenge." - NVIDIA



Isaac Sim

Ignition Gazebo



open robotics

Part 1: Offline conversions

SDFormat

- SDF: used by Ignition Gazebo
- Describes objects and environments for robot simulators, visualization, and control
- Designed for scientific robot applications
- Extensible format describing all aspects of robots, static and dynamic objects, lighting, terrain, and physics.

```
<?xml version="1.0" ?>
<!--
  Demo containing a model with PBR materials, pulled from Ignition Fuel.
-->
<sdf version="1.6">
  <world name="fuel">
    <physics name="1ms" type="ignored">
      <max_step_size>0.001</max_step_size>
      <real_time_factor>1.0</real_time_factor>
    </physics>

    <scene>
      <ambient>1.0 1.0 1.0 1.0</ambient>
      <background>0.8 0.8 0.8 1.0</background>
      <grid>false</grid>
      <origin_visual>false</origin_visual>
    </scene>

    <plugin
      filename="ignition-gazebo-physics-system"
      name="ignition::gazebo::systems::Physics">
    </plugin>
    <plugin
      filename="ignition-gazebo-sensors-system"
      name="ignition::gazebo::systems::Sensors">
      <render_engine>ogre2</render_engine>
    </plugin>
    <plugin
      filename="ignition-gazebo-user-commands-system"
      name="ignition::gazebo::systems::UserCommands">
    </plugin>
    <plugin
      filename="ignition-gazebo-scene-broadcaster-system"
      name="ignition::gazebo::systems::SceneBroadcaster">
    </plugin>

    <light type="directional" name="sun">
      <cast_shadows>true</cast_shadows>
      <pose>0 0 10 0 0 0</pose>
      <diffuse>0.8 0.8 0.8 1</diffuse>
      <specular>0.2 0.2 0.2 1</specular>
      <attenuation>
        <range>1000</range>
        <constant>0.9</constant>
        <linear>0.01</linear>
        <quadratic>0.001</quadratic>
      </attenuation>
```

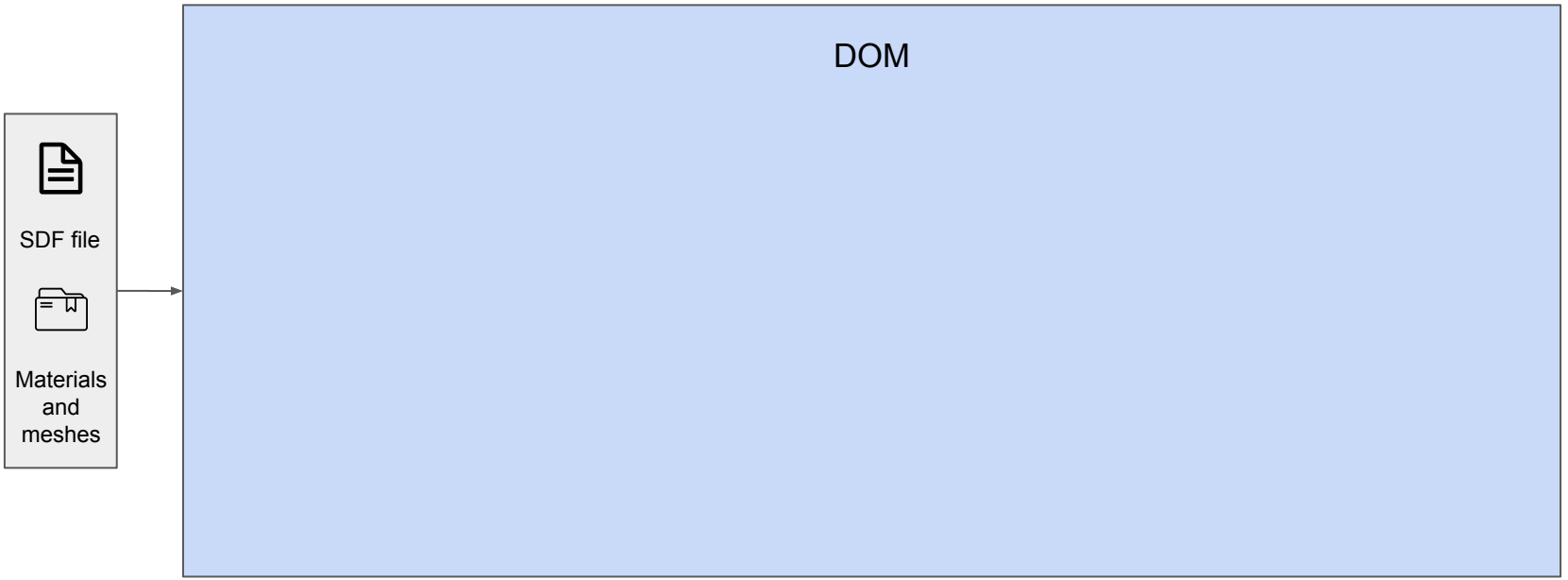
- USD: used by Isaac Sim
- High-performance extensible software platform for animated 3D scenes
- Designed to meet the needs of large-scale film and visual effects production
- Expanding set of schemas, covering geometry, shading, lighting, and physics

```
#usda 1.0
(
    endTimeCode = 100
    metersPerUnit = 1
    startTimeCode = 0
    timeCodesPerSecond = 24
    upAxis = "Z"
)

def "fuel"
{
    def PhysicsScene "physics"
    {
        vector3f physics:gravityDirection = (0, 0, -1)
        float physics:gravityMagnitude = 9.8
    }

    def Xform "panda" (
        prepend apiSchemas = ["PhysicsArticulationRootAPI"]
    )
    {
        float3 xformOp:rotateXYZ = (0, 0, 0)
        double3 xformOp:translate = (0, 0, 0)
        uniform token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ"]

        def Xform "panda_link0" (
            prepend apiSchemas = ["PhysicsRigidBodyAPI", "PhysicsMassAPI"]
        )
        {
            point3f physics:centerOfMass = (-0.025566, -0.0000287883, 0.057332)
            float3 physics:diagonalInertia = (0.0075390637, 0.010508018, 0.009864934)
            float physics:mass = 2.8142712
            float3 xformOp:rotateXYZ = (0, 0, 0)
            double3 xformOp:translate = (0, 0, 0)
            uniform token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ"]

            def Xform "panda_link0_visual"
            {
                float3 xformOp:rotateXYZ = (0, 0, 0)
                double3 xformOp:translate = (0, 0, 0)
                uniform token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ"]

                def "geometry" (
                    prepend apiSchemas = ["PhysicsCollisionAPI"]
                )
                {
                    rel material:binding = </Looks/Material_1>
```

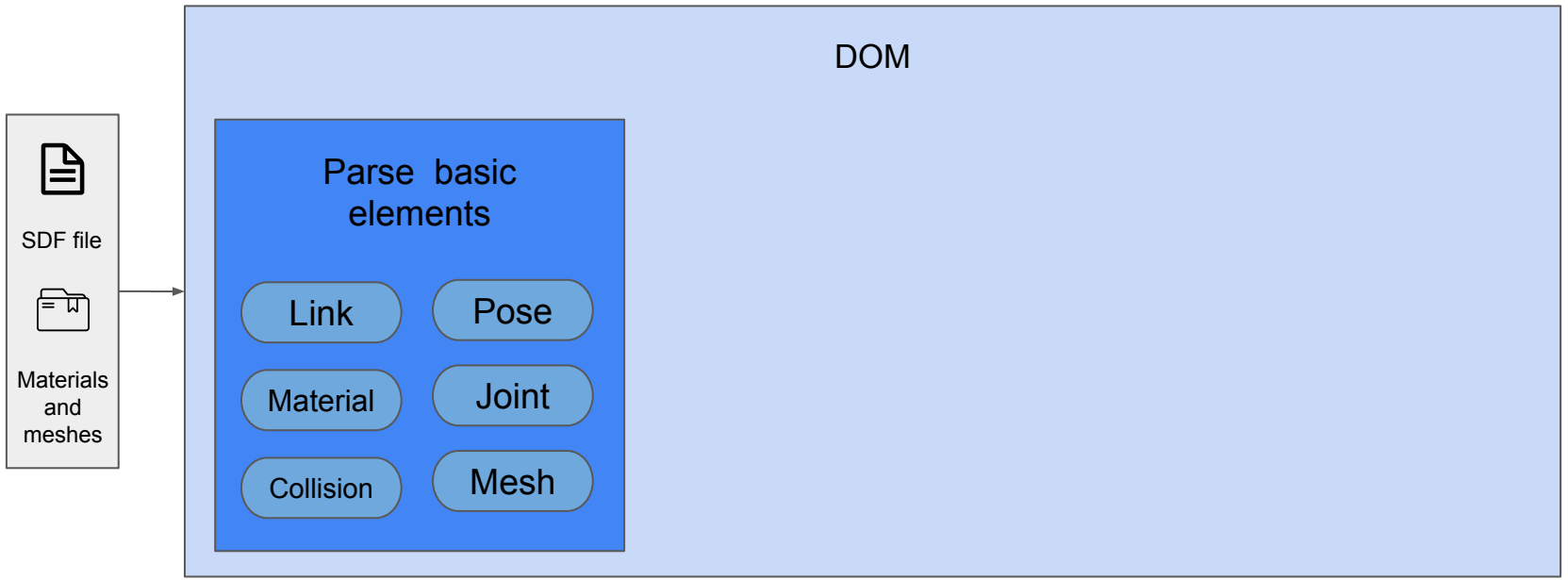
open robotics

Converters: SDF->USD

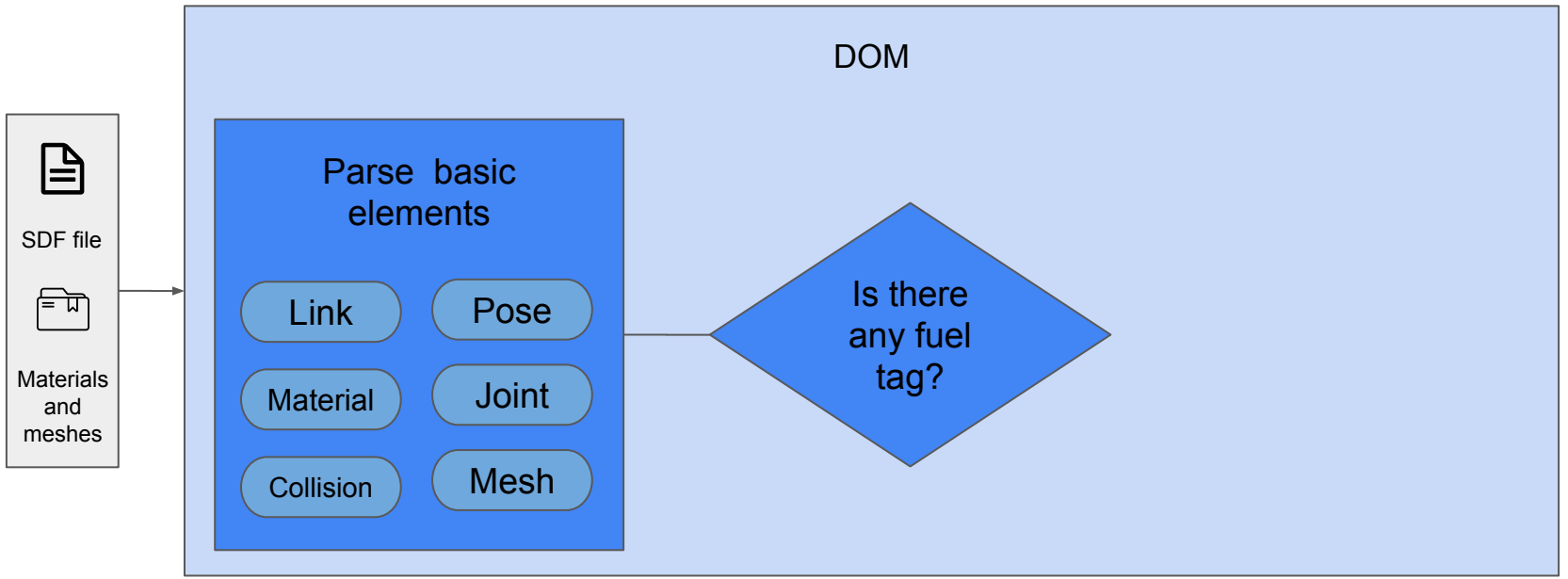
Converters: SDF->USD



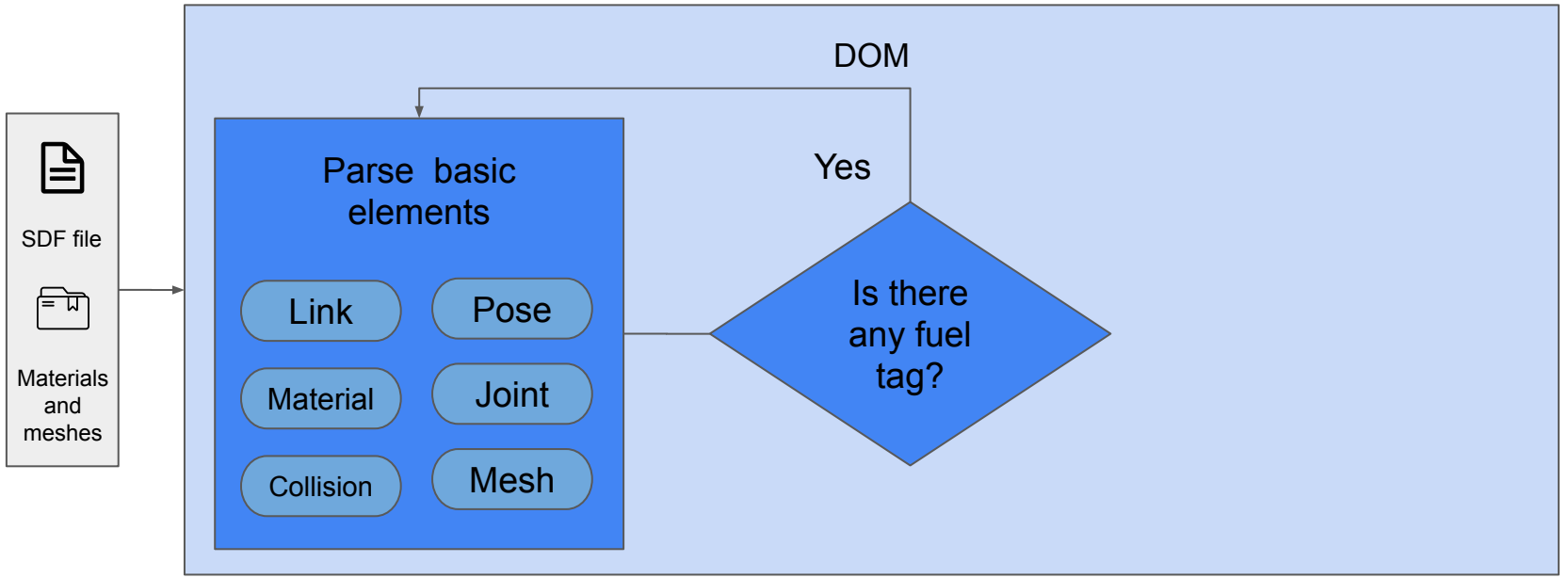
Converters: SDF->USD



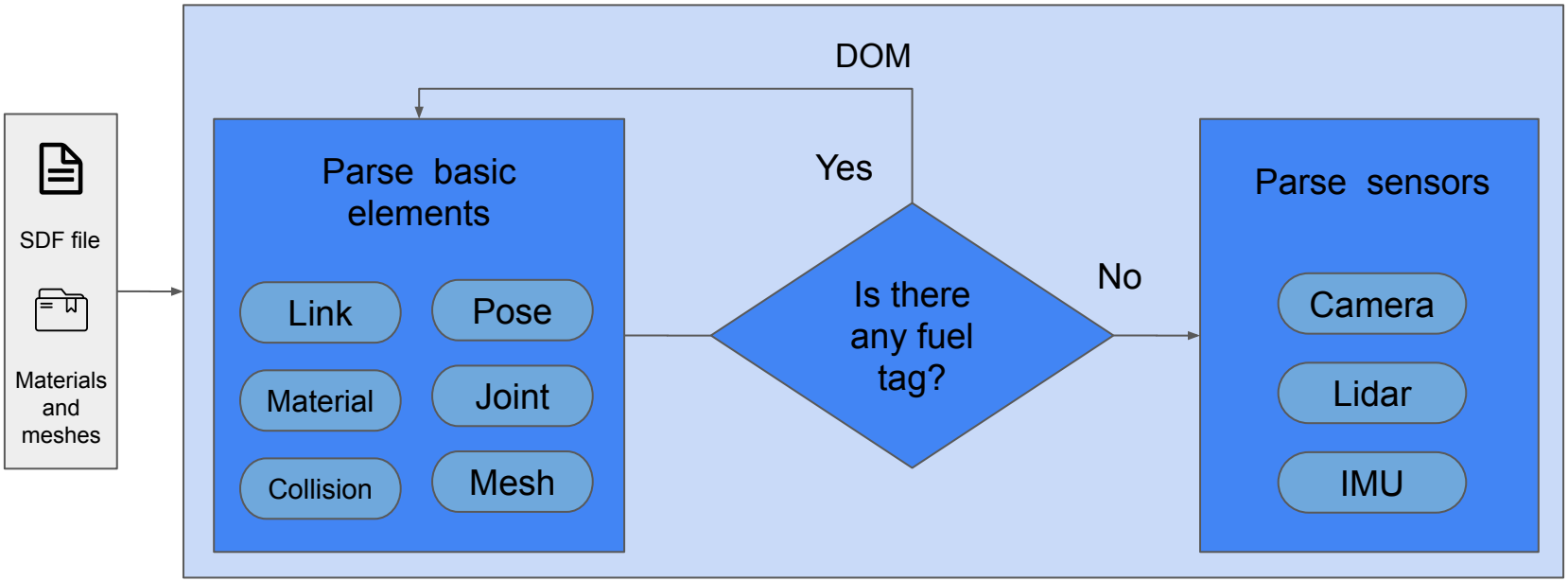
Converters: SDF->USD



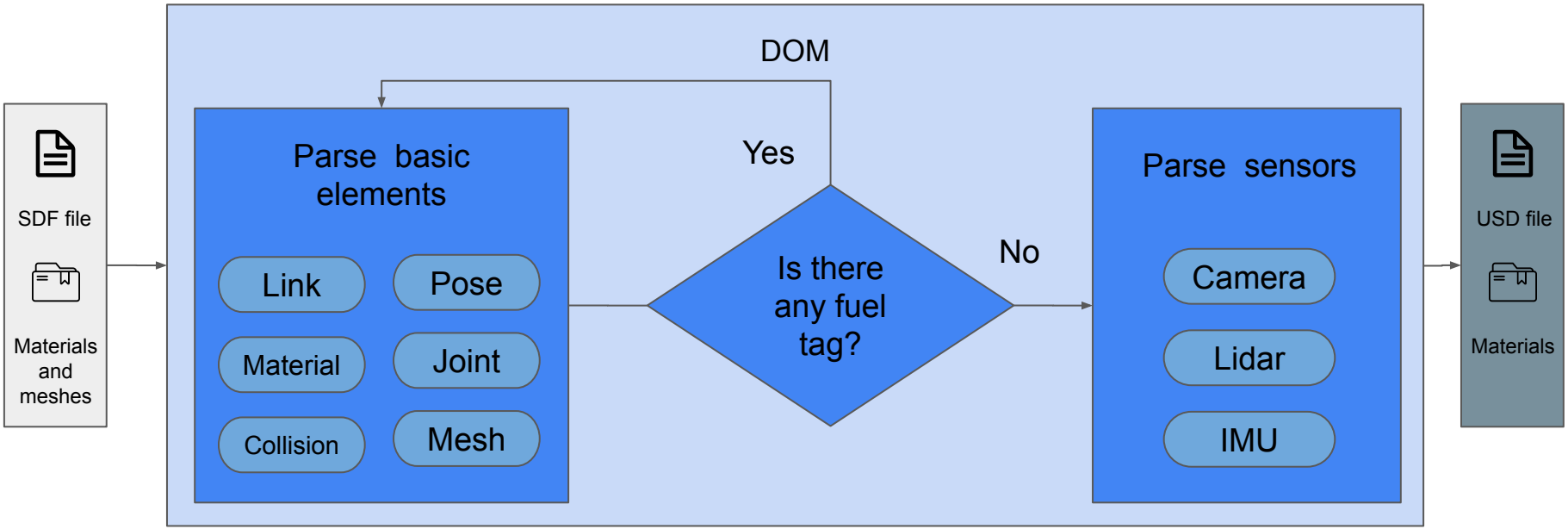
Converters: SDF->USD



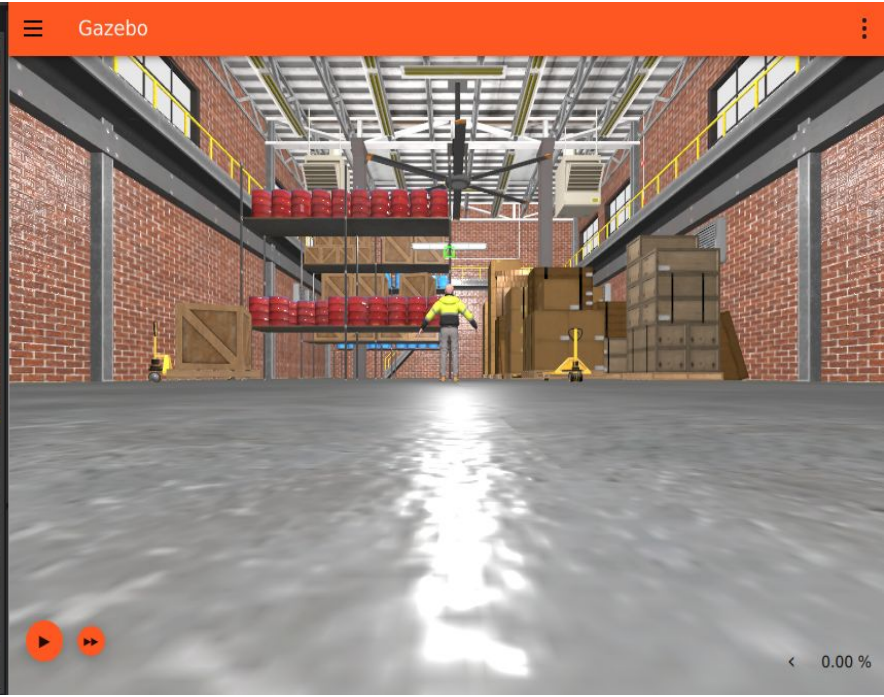
Converters: SDF->USD



Converters: SDF->USD



USD <- SDF



open robotics

Examples: SDF Fuel support

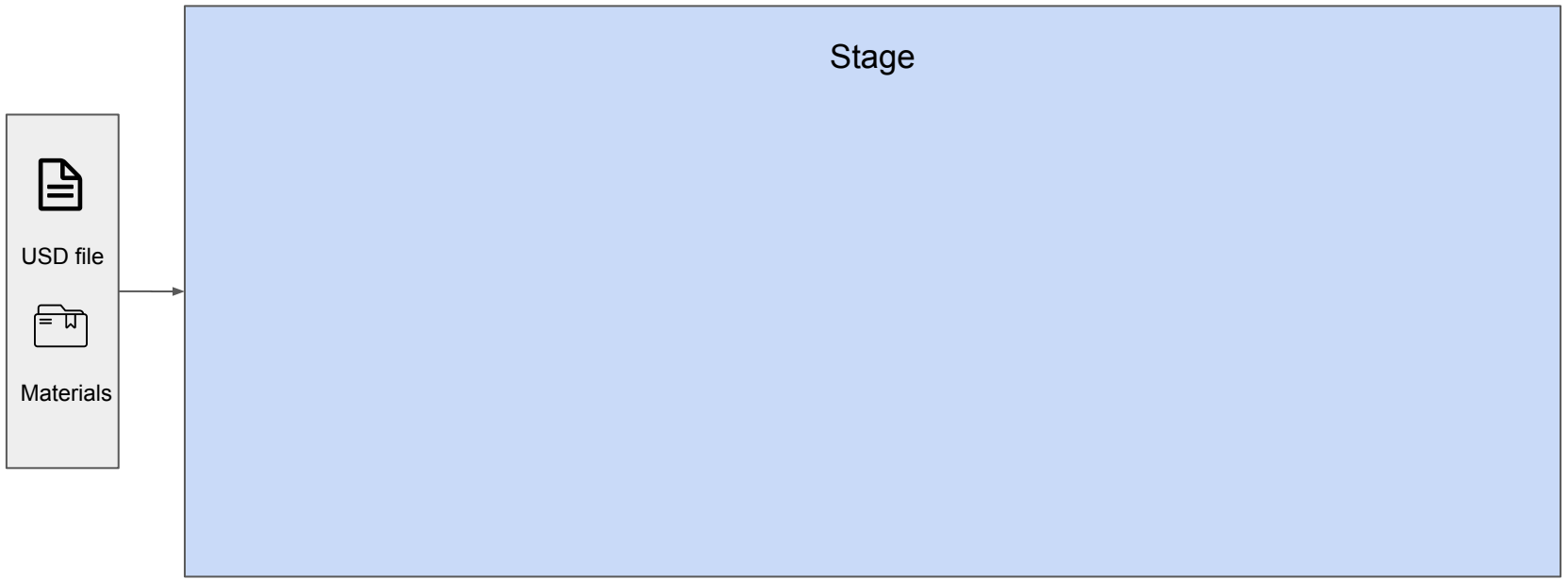
```
#usda 1.0
(
    endTimeCode = 100
    metersPerUnit = 1
    startTimeCode = 0
    timeCodesPerSecond = 24
    upAxis = "Z"
)

def "fuel"
{
    def PhysicsScene "physics"
    {
        vector3f physics:gravityDirection = (0, 0, -1)
        float physics:gravityMagnitude = 9.8
    }

    def Xform "panda" (
        prepend apiSchemas = ["PhysicsArticulationRootAPI"]
    )
    {
        float3 xformOp:rotateXYZ = (0, 0, 0)
        double3 xformOp:translate = (0, 0, 0)
        uniform token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ"]

        def Xform "panda_link0" (
            prepend apiSchemas = ["PhysicsRigidBodyAPI", "PhysicsMassAPI"]
        )
        {
            point3f physics:centerOfMass = (-0.025566, -0.0000287883, 0.057332)
            float3 physics:diagonalInertia = (0.0075390637, 0.010508018, 0.009864934)
            float physics:mass = 2.8142712
            float3 xformOp:rotateXYZ = (0, 0, 0)
            double3 xformOp:translate = (0, 0, 0)
            uniform token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ"]

            def Xform "panda_link0_visual"
            {
                float3 xformOp:rotateXYZ = (0, 0, 0)
                double3 xformOp:translate = (0, 0, 0)
                uniform token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ"]

                def "geometry" (
                    prepend apiSchemas = ["PhysicsCollisionAPI"]
                )
                {
                    rel material:binding = </Looks/Material_1>

                    def Mesh "link0"
                    {
                        float3[] extent = [(-0.154086, -0.09459599, 0), (0.071544, 0.094573, 0.14)]
                        int[] faceVertexCounts = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3
                        3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
                        3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
```

```
          </camera>
          <always_on>1</always_on>
          <update_rate>30</update_rate>
          <visualize>true</visualize>
          <topic>camera</topic>
        </sensor>
      </link>
    </model>


    <!-- Rescue Randy -->
    <include>
      <static>true</static>
      <name>Rescue Randy</name>
      <pose>0 0 0 0 0 1.57</pose>
      <uri>https://fuel.ignitionrobotics.org/1.0/OpenRobotics/models/Rescu
    </include>

    <!-- Tube Light -->
    <include>
      <static>true</static>
      <name>Tube Light</name>
      <pose>1.5 0 3 0 0.78 0</pose>
      <uri>https://fuel.ignitionrobotics.org/1.0/openrobotics/models/Tube
    </include>

    <light type="spot" name="tube_light">
      <cast_shadows>false</cast_shadows>
      <pose>1.48 0 2.96 0 0.78 0</pose>
      <attenuation>
        <range>50</range>
        <linear>0</linear>
```

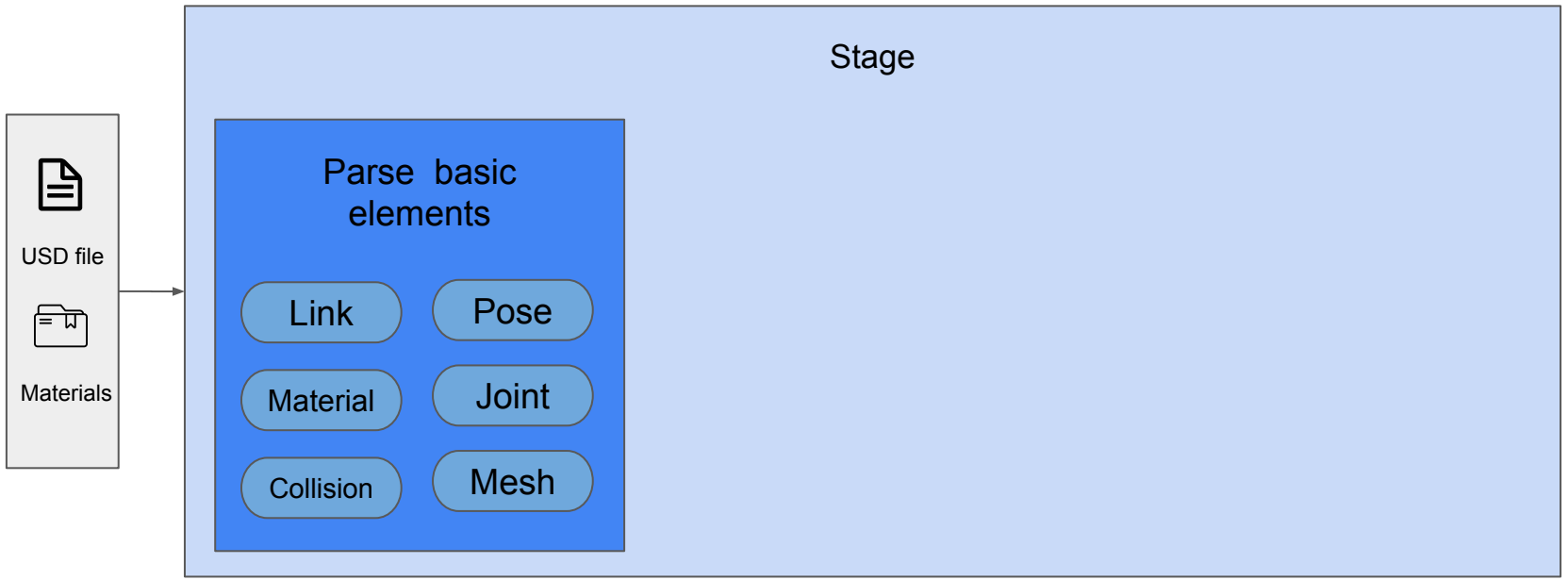
SDF TO USD
FRANKA EMIKA PANDA+ MOVEIT

Converters: USD->SDF

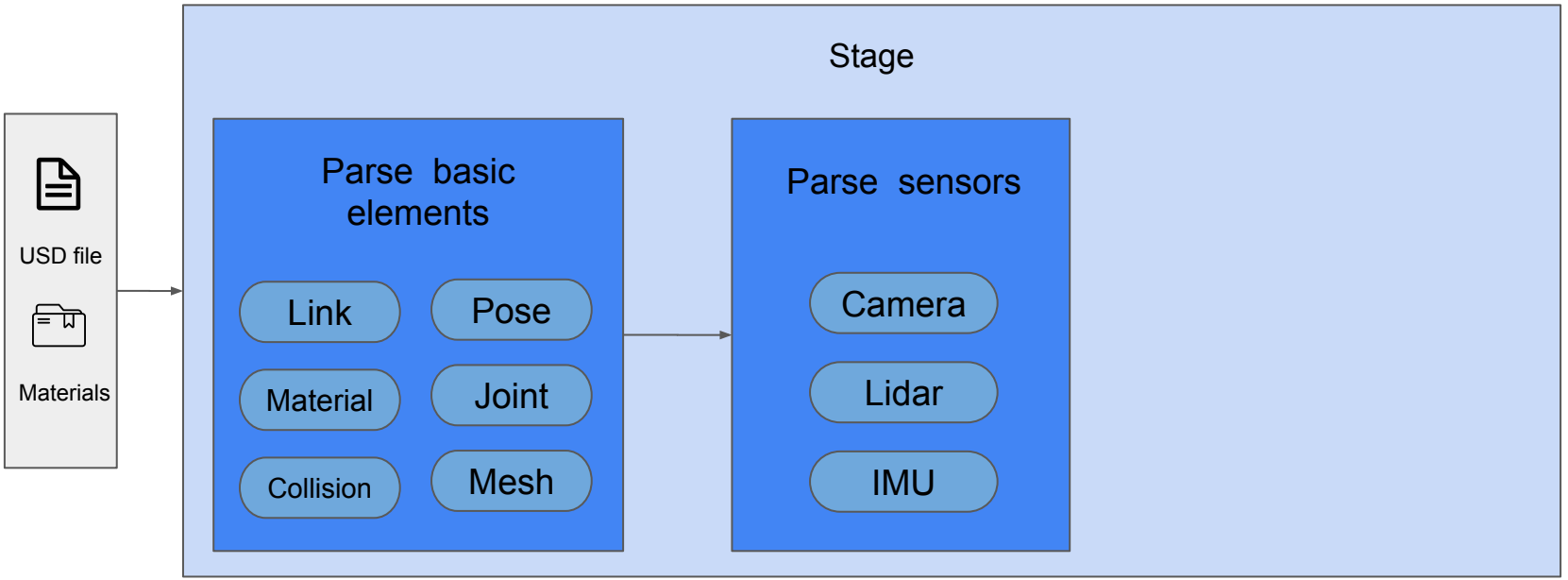
Converters: USD->SDF



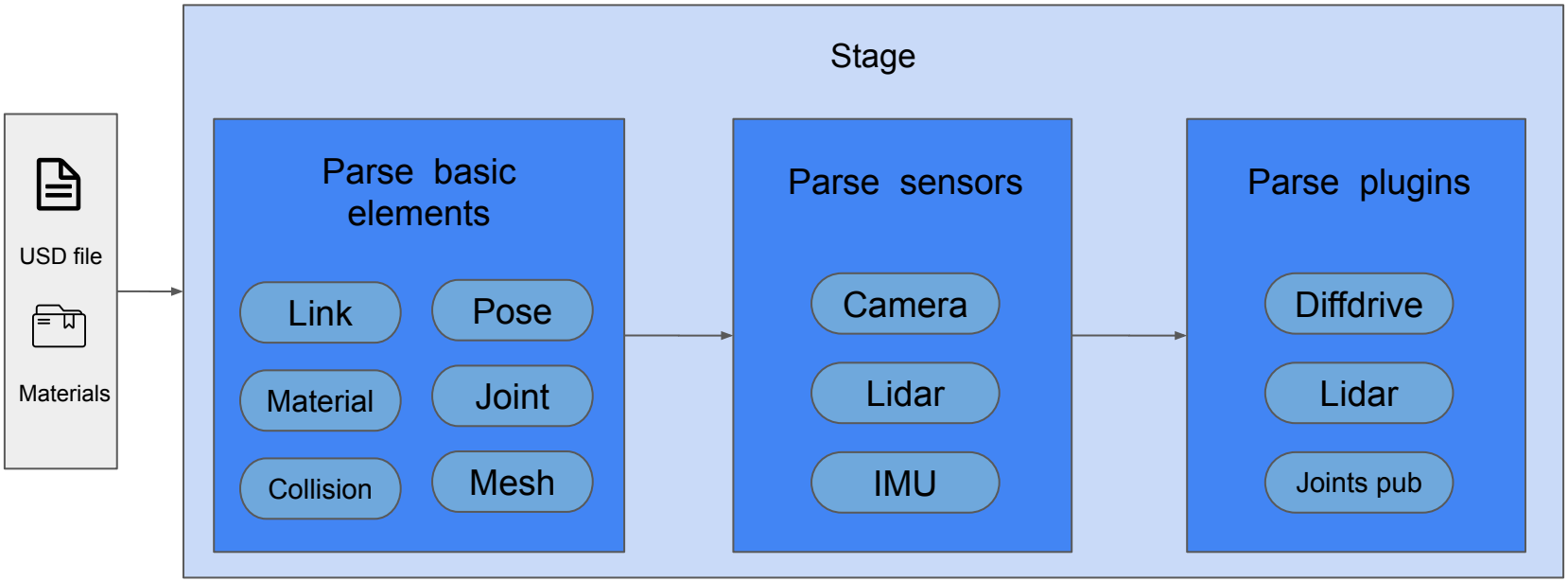
Converters: USD->SDF



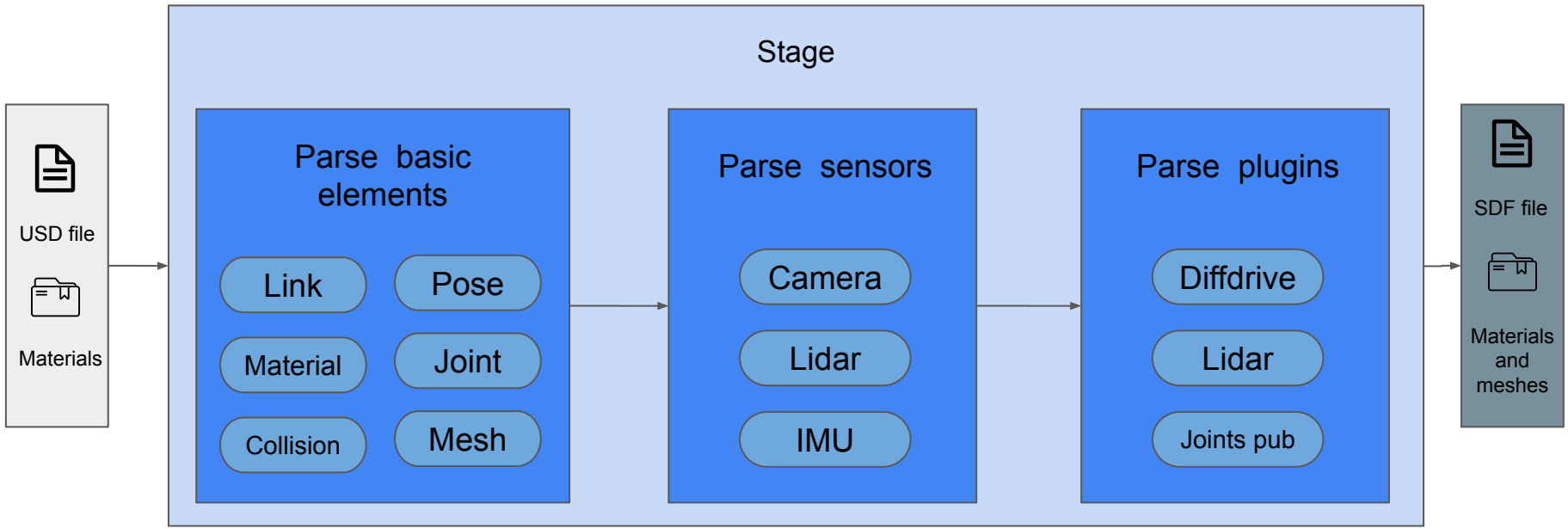
Converters: USD->SDF



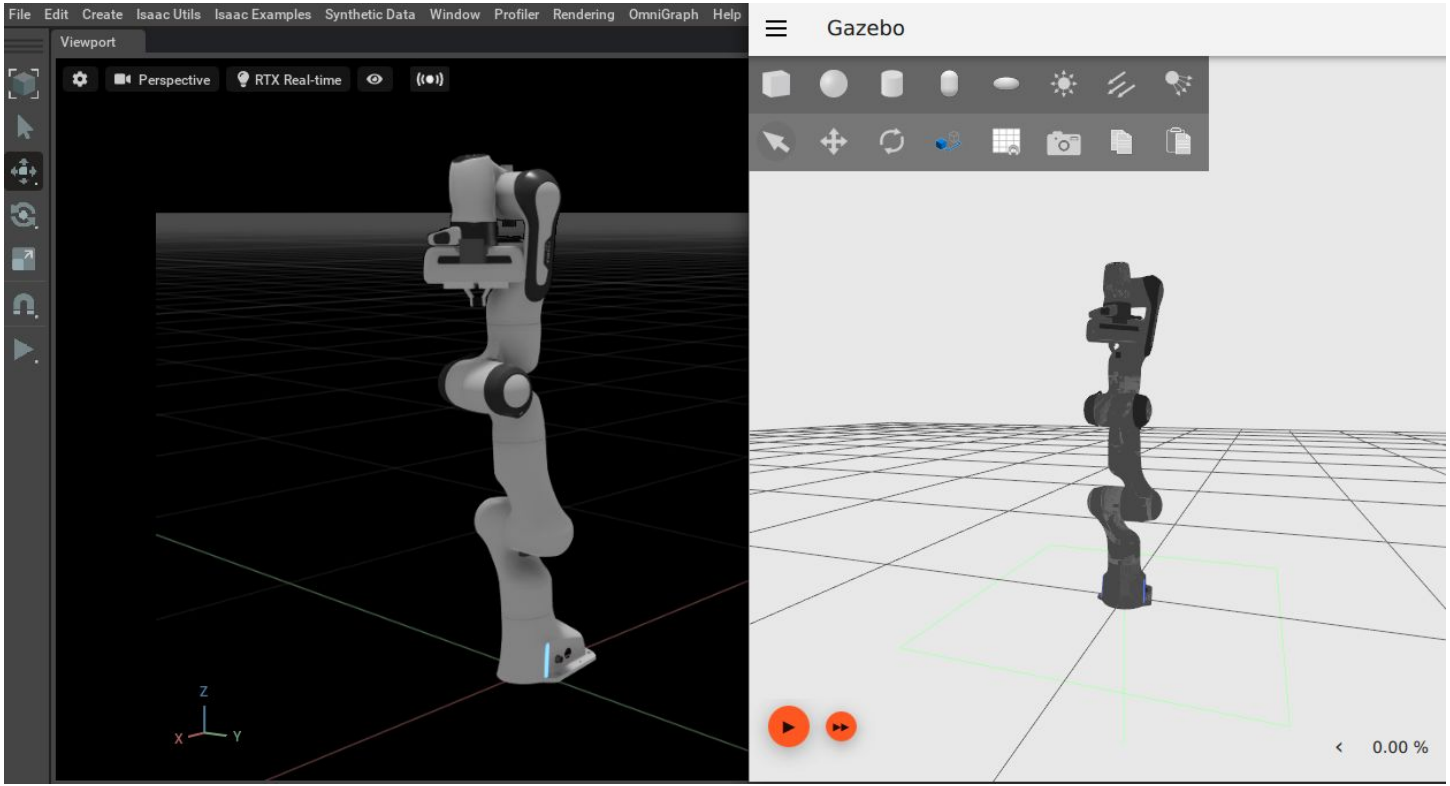
Converters: USD->SDF



Converters: USD->SDF

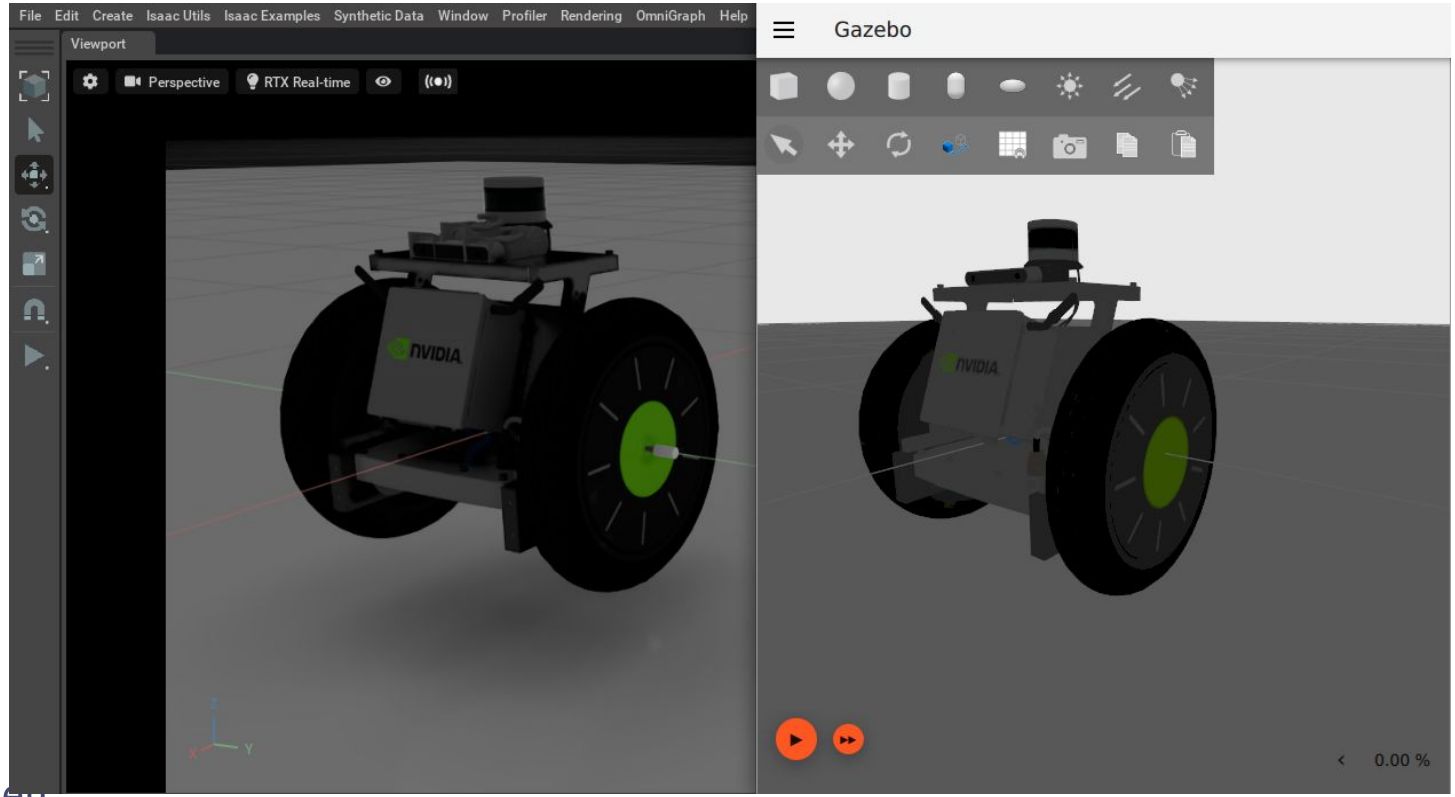


USD -> SDF



open robotics

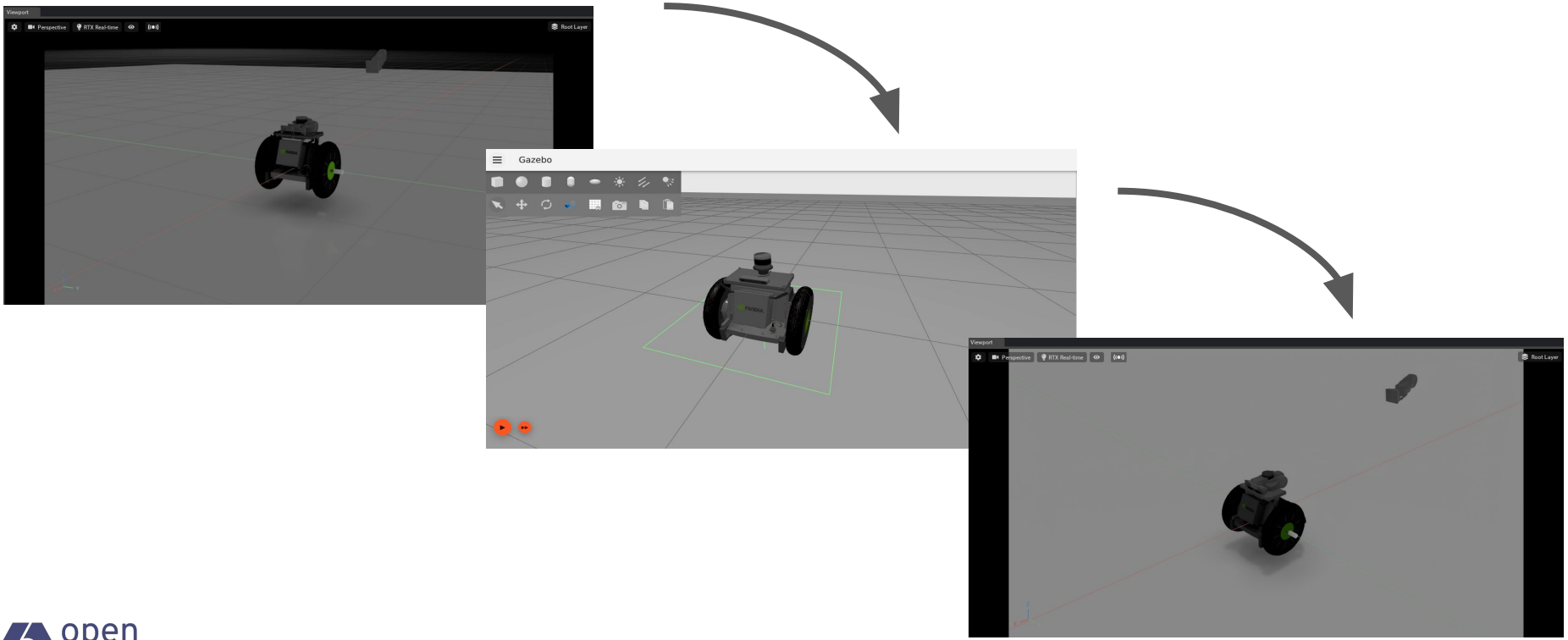
USD -> SDF



open robotics

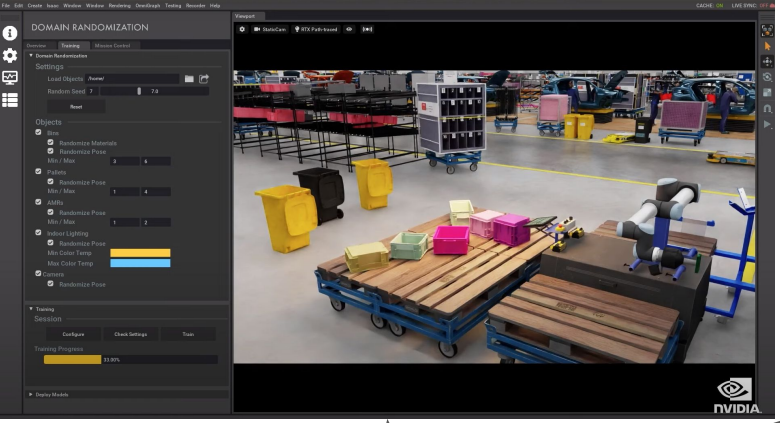
USD TO SDF
CARTER ROBOT

Repeated conversion USD->SDF->USD



open robotics

Part 2: Run-time connections



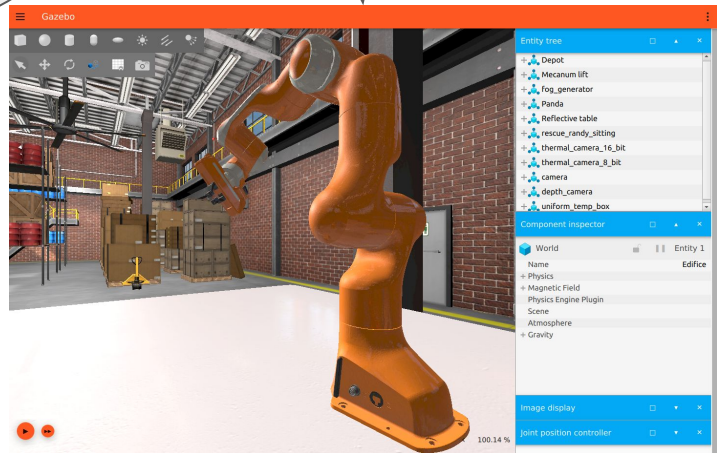
omniclient

Ignition
Gazebo

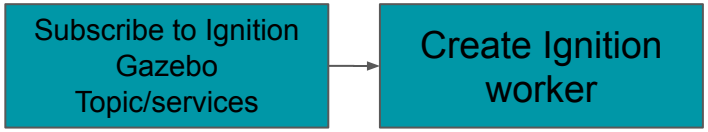
Ign-omniverse connector

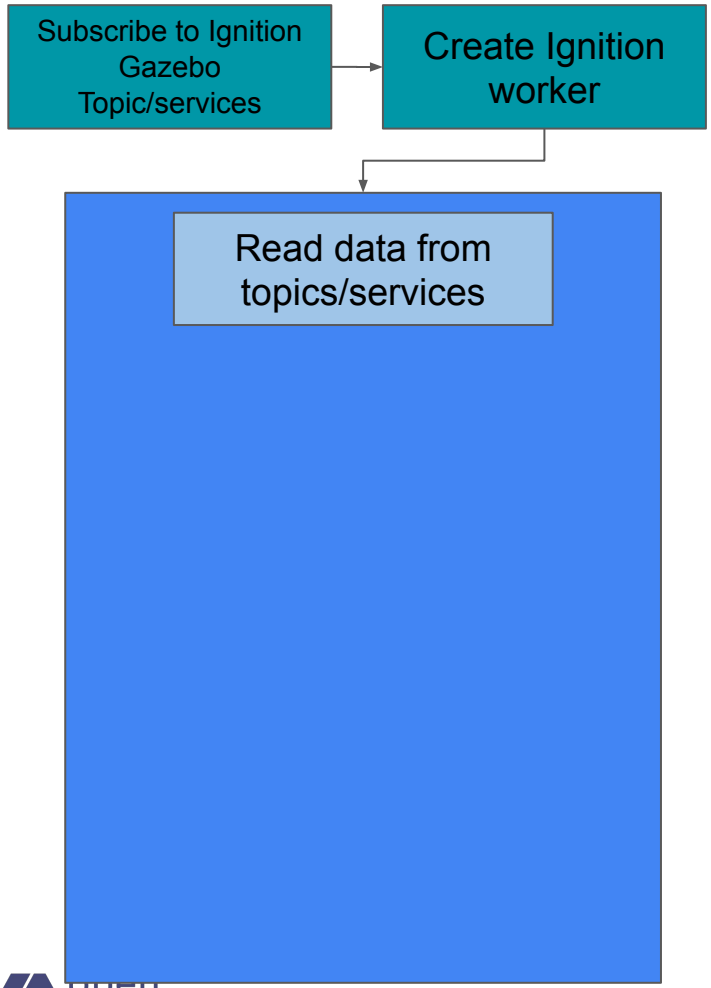
Isaac
Sim

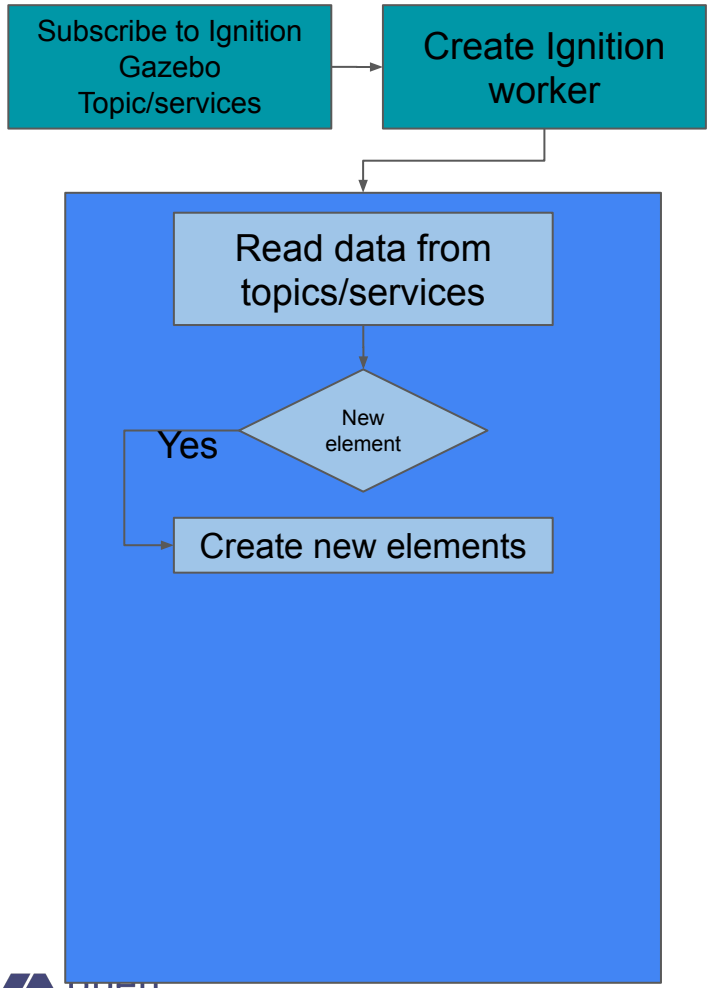
ignition-transport

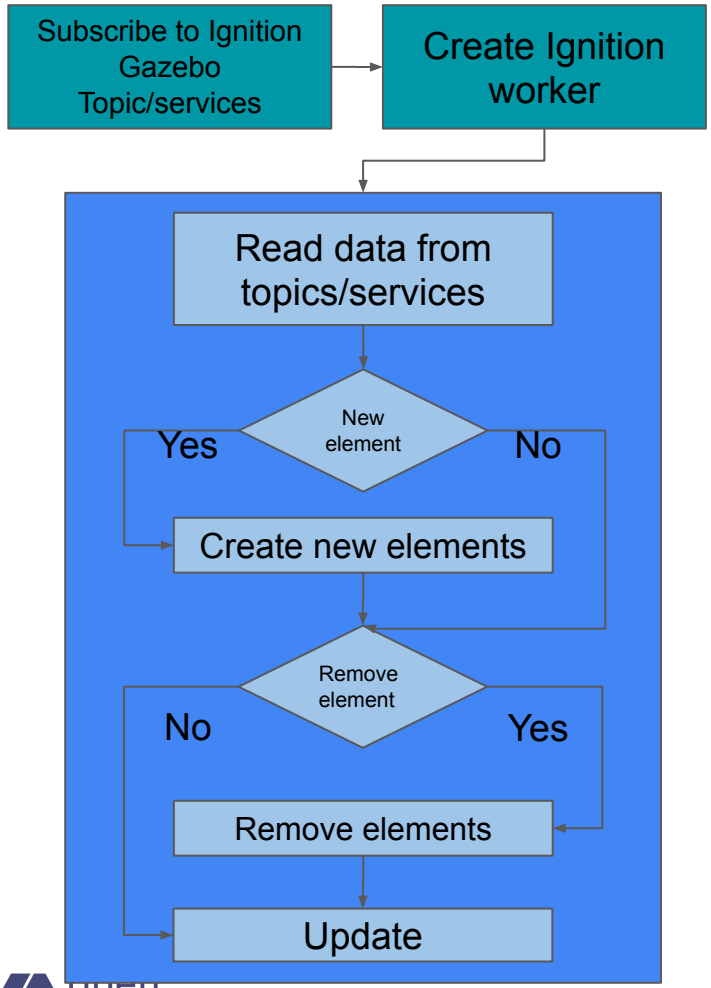


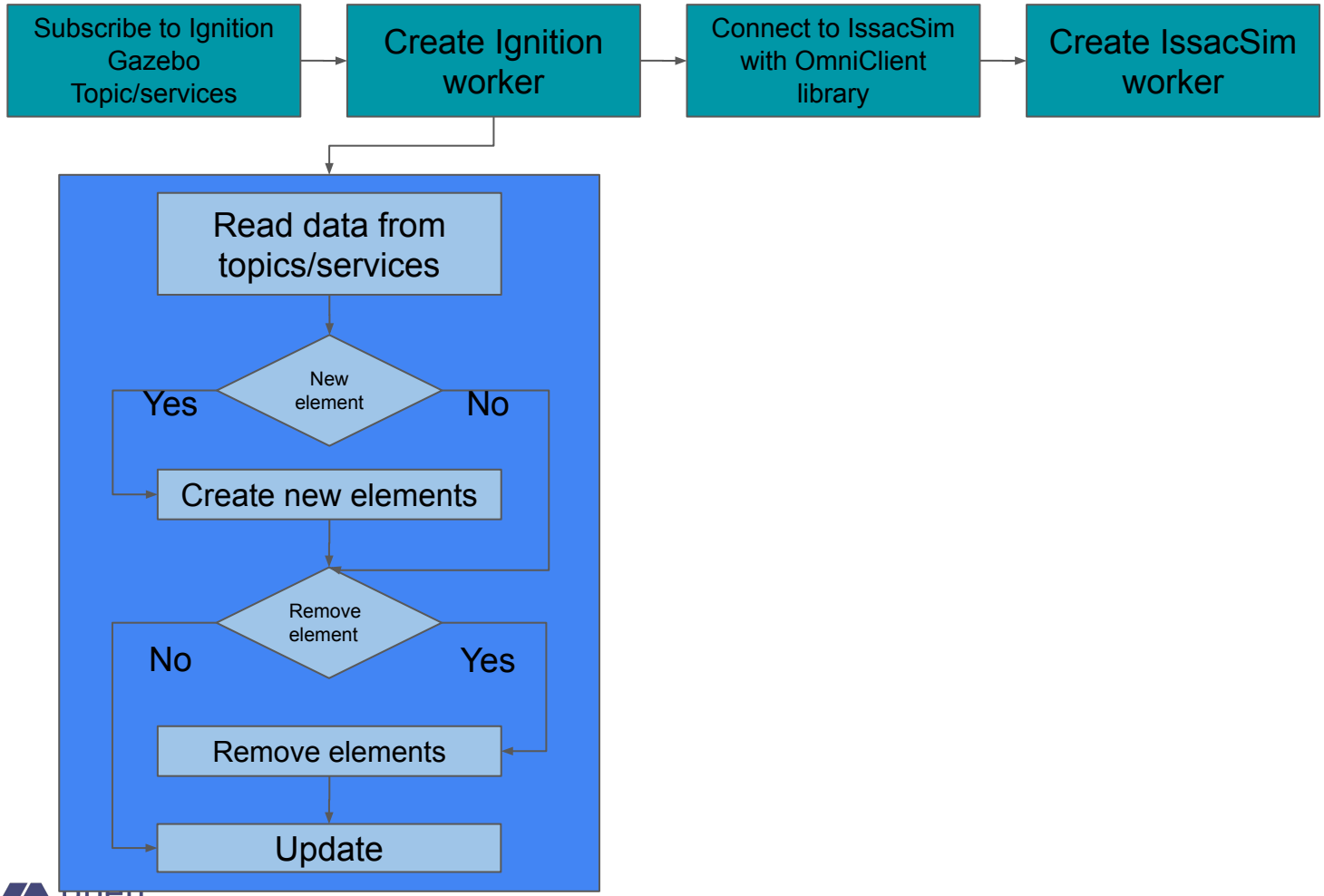
open
robotics

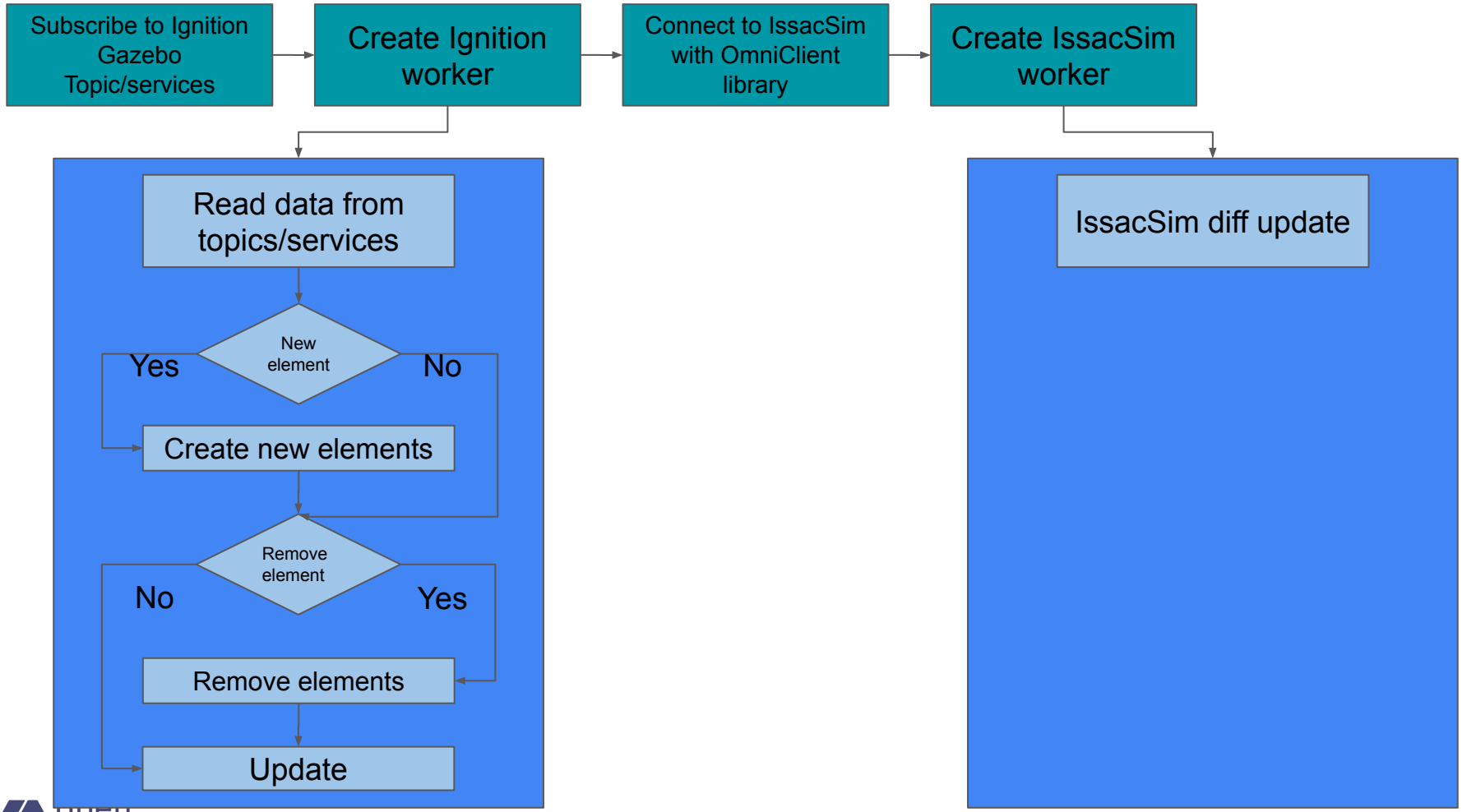

Subscribe to Ignition Gazebo Topic/services
Create Ignition worker

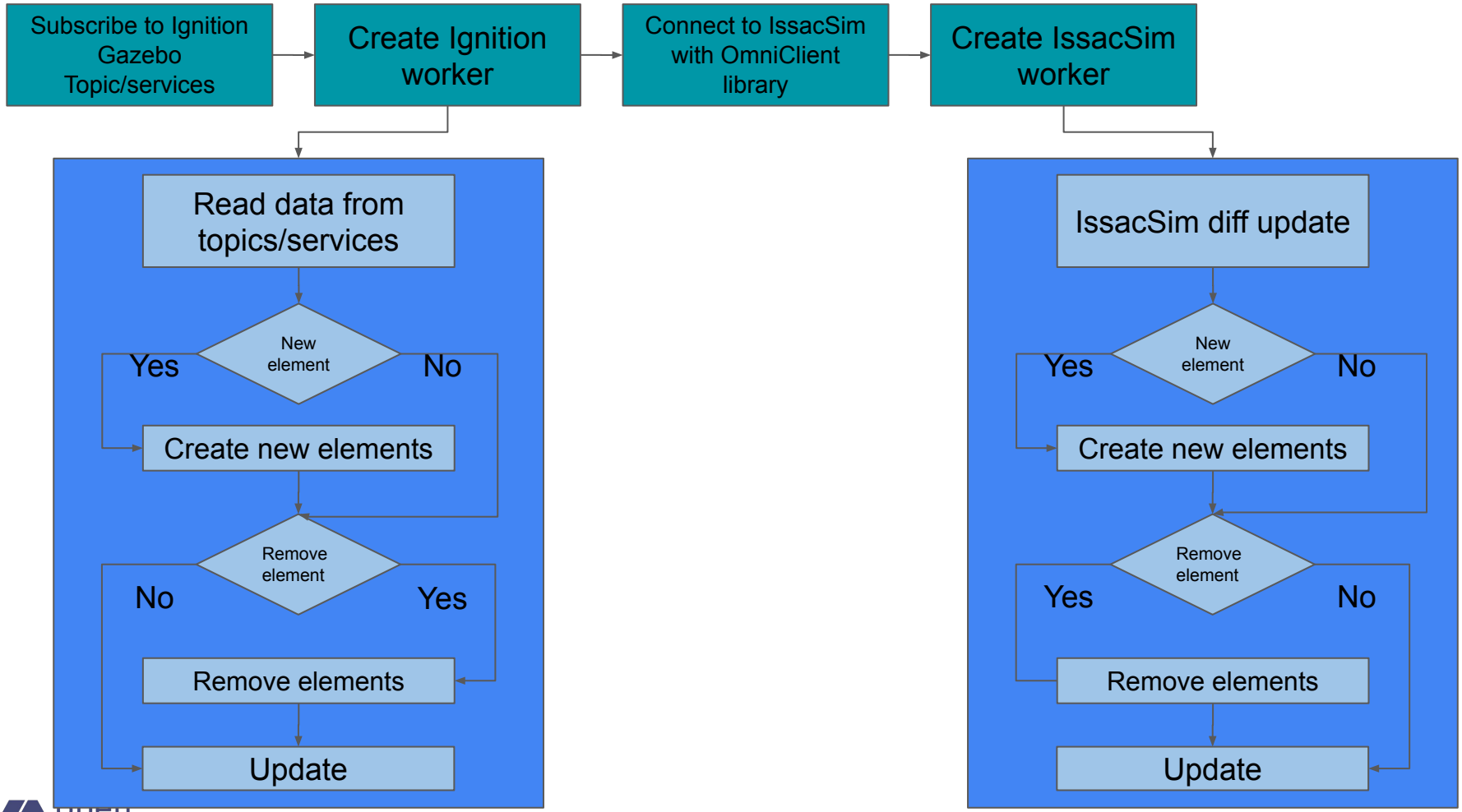

Subscribe to Ignition Gazebo Topic/services
Create Ignition worker
Read data from topics/services

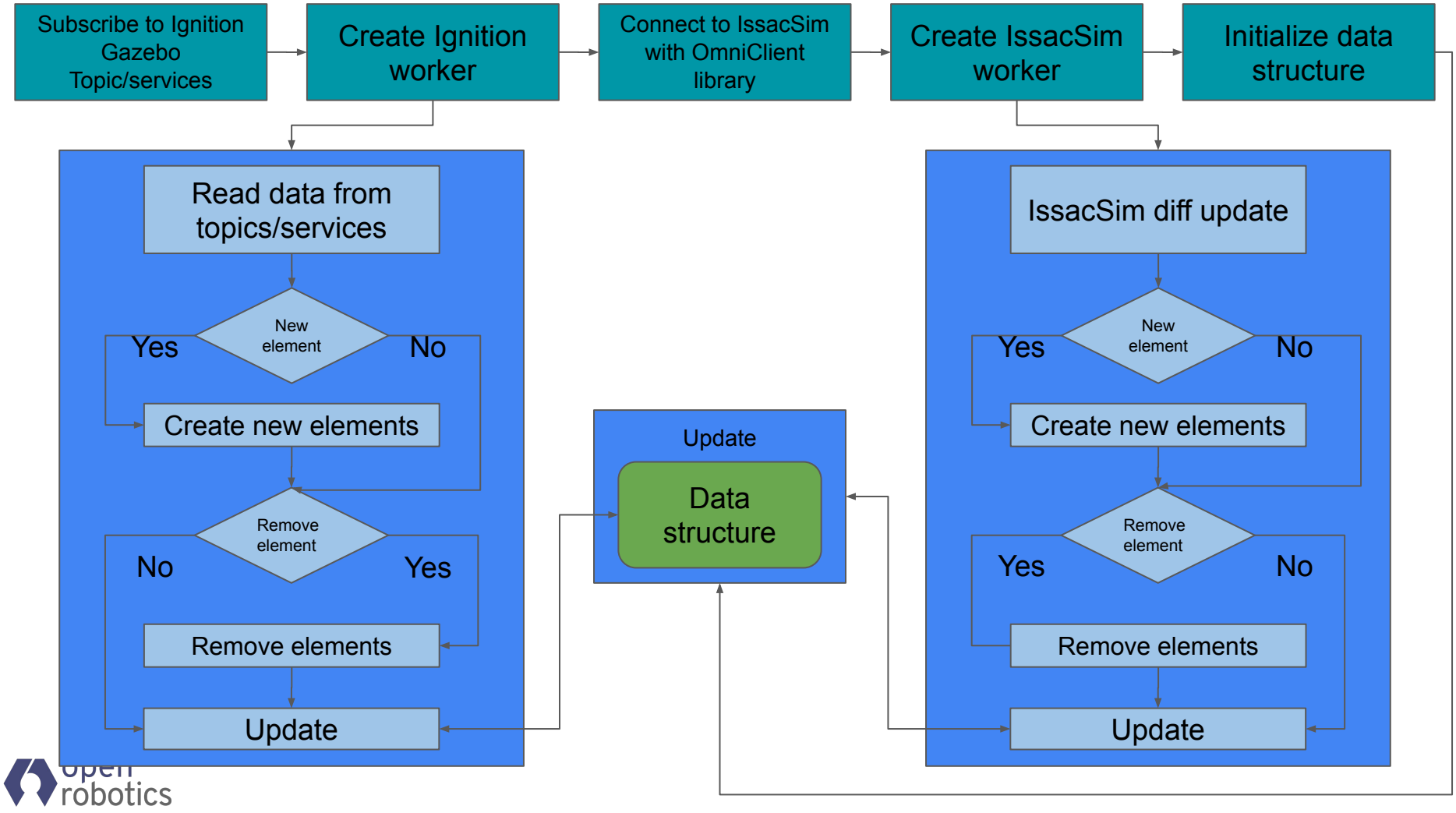

Subscribe to Ignition Gazebo Topic/services
Create Ignition worker
Read data from topics/services
New element
Yes
Create new elements


Subscribe to Ignition Gazebo Topic/services
Create Ignition worker
Read data from topics/services
New element
Yes
No
Create new elements
Remove element
No
Yes
Remove elements
Update


Subscribe to Ignition Gazebo Topic/services
Create Ignition worker
Connect to IssacSim with OmniClient library
Create IssacSim worker
Read data from topics/services
New element
Yes
No
Create new elements
Remove element
No
Yes
Remove elements
Update


Subscribe to Ignition Gazebo Topic/services
Create Ignition worker
Connect to IssacSim with OmniClient library
Create IssacSim worker
Read data from topics/services
New element
Yes
No
Create new elements
Remove element
No
Yes
Remove elements
Update
IssacSim diff update




Subscribe to Ignition Gazebo Topic/services
Create Ignition worker
Connect to IssacSim with OmniClient library
Create IssacSim worker
Initialize data structure
Read data from topics/services
New element
Yes
No
Create new elements
Remove element
No
Yes
Remove elements
Update
Update
Data structure
IssacSim diff update
New element
Yes
No
Create new elements
Remove element
Yes
No
Remove elements
Update

CONNECTOR: IGNITION - ISSAC SIM
TURTLEBOT3 + NAV2

Summary

User benefits: choice & interoperability

Assets & worlds

- Access content in two widely used formats

Rendering systems

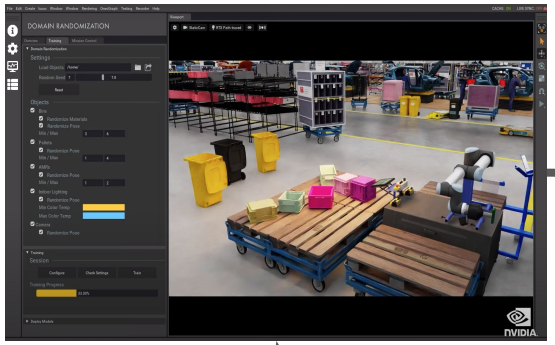
- Omniverse, OGRE, OGRE2

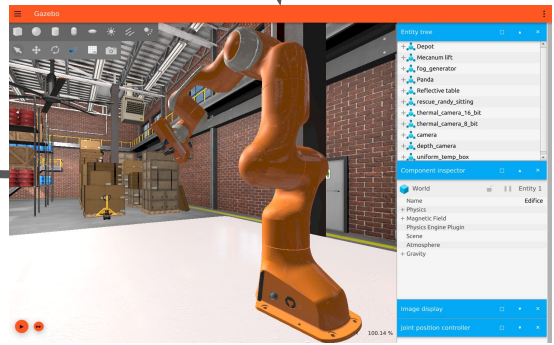
Physics engines

- ODE, DART, Bullet, PhysX

Developer tools

- UIs, log, plot, sensor viz





open robotics

Future work

- **Collisions**: support PhysX 5 collision binary format
- **Joints**: add ball, revolute2, screw, universal, gearbox
- **Shaders**:
  - USD->SDF: convert additional mdl shaders
- **Connector**: improve data flow Isaac Sim <-> Ignition Gazebo

Conclusion

Resources:

- **USD <-> SDF Converters**
  - https://github.com/ignitionrobotics/sdformat
- **ign-omni (Connector)**
  - https://github.com/ignitionrobotics/ign-omni

Feedback:

- We invite everyone to try it and test it! We are happy to receive your feedback

IGNITION
robotics.org

Thank you!