MetaTool
Robots inventing tools

POLITÉCNICA

UNIVERSIDAD POLITÉCNICA DE MADRID

Simulación de manipuladores y sistemas de visión en Isaac Sim con ROS 2

Mohamed Khalil Kahlaoui (UPM)

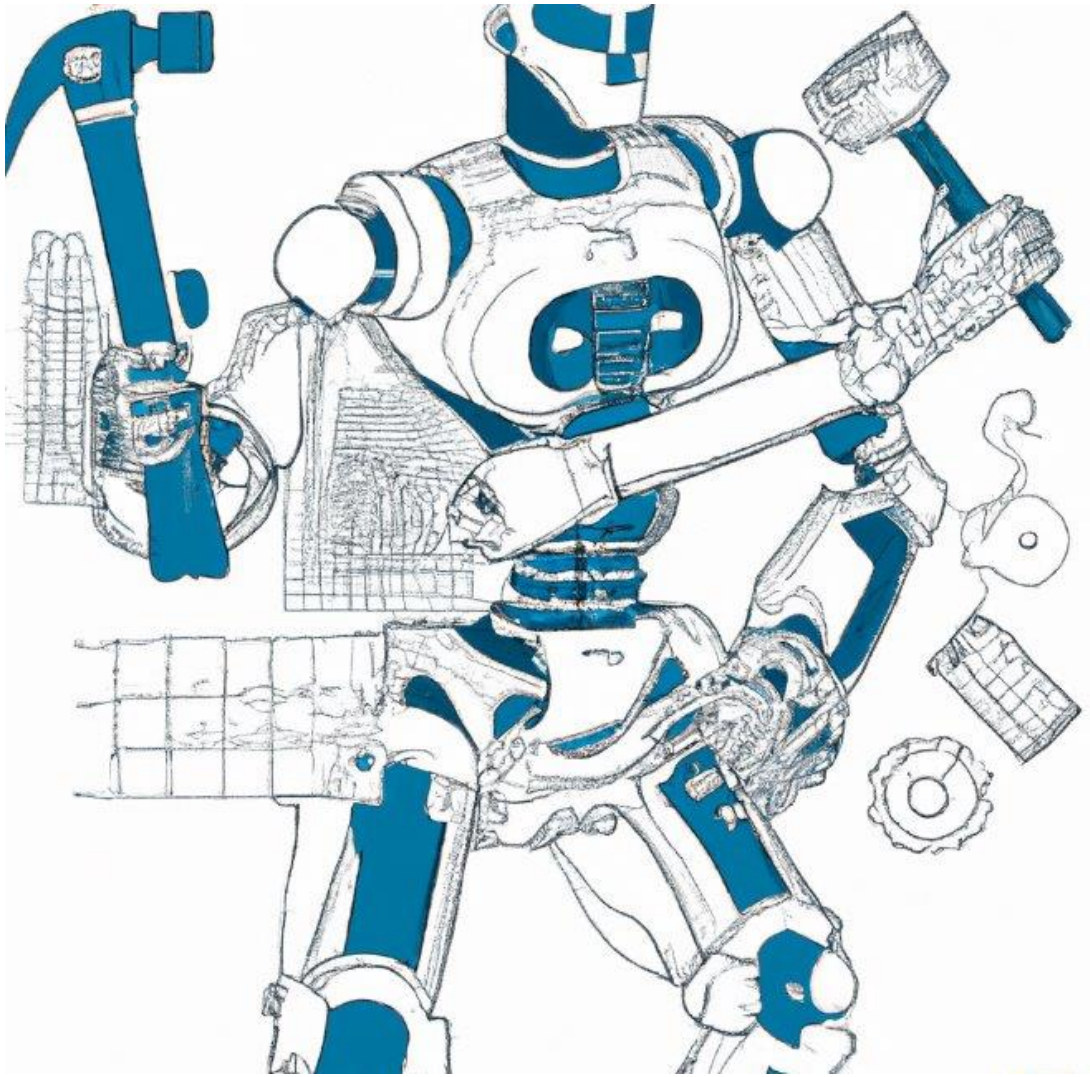
European Innovation Council

The METATOOL project has received funding from the European Innovation Council through the Pathfinder Challenges grant No. 101070940.

ROSCon ES 2024 SEVILLA

Contenido

- Introducción
- Simulación de Manipuladores
- Generación de movimiento
- Integración con sistemas de visión
- Integración Completa con ROS2
- Ejercicios Prácticos



Introducción

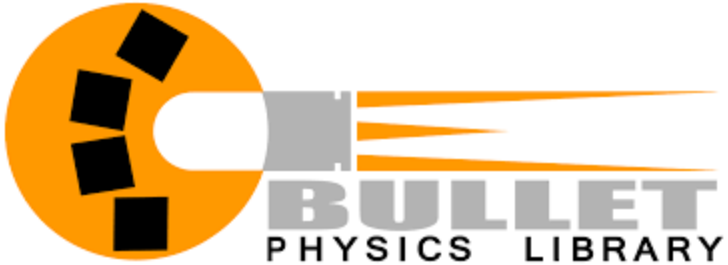
¿Cómo creen que la simulación contribuye al avance en el desarrollo de robots?

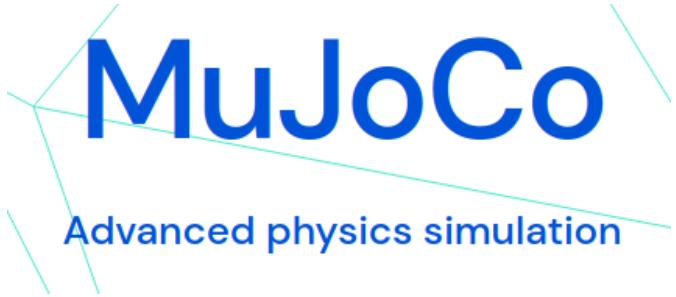
1. **Entorno seguro y económico:**
   - Permite probar diferentes configuraciones sin riesgos físicos.
   - Reduce costos al evitar dañar equipos reales.
2. **Aceleración del desarrollo**:
   - Facilita la iteración rápida de diseños y algoritmos.
   - Posibilidad de probar movimientos, trayectorias y sensores virtualmente.
3. **Pruebas y validación**:
   - Detecta fallos y errores de diseño antes de implementarlos en hardware real.
   - Prepara mejor los sistemas para su uso en situaciones del mundo real.

Tipos de Simuladores de Robots

Webots

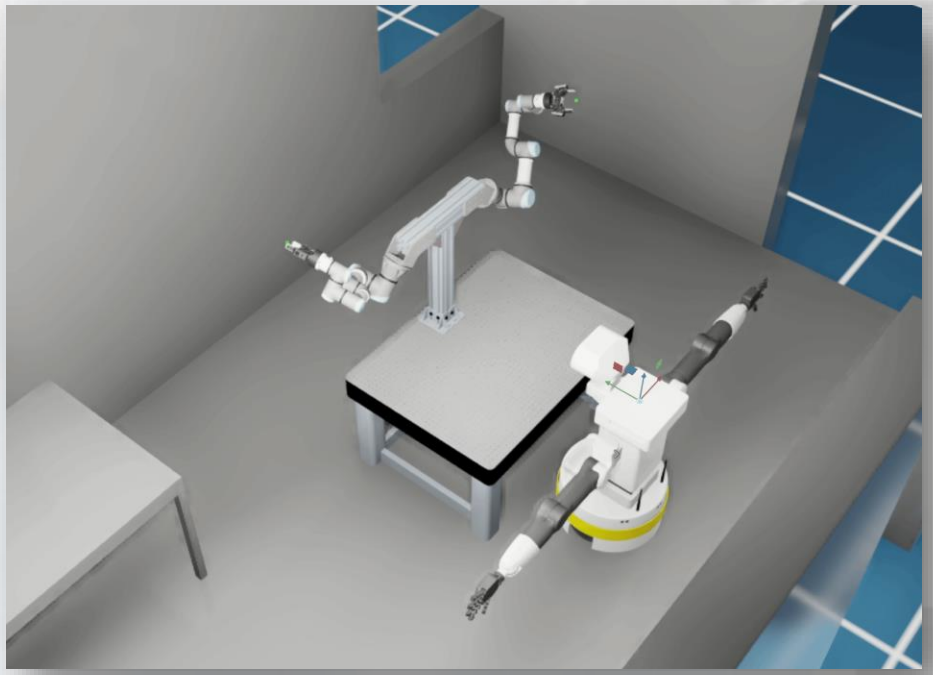
robot simulation



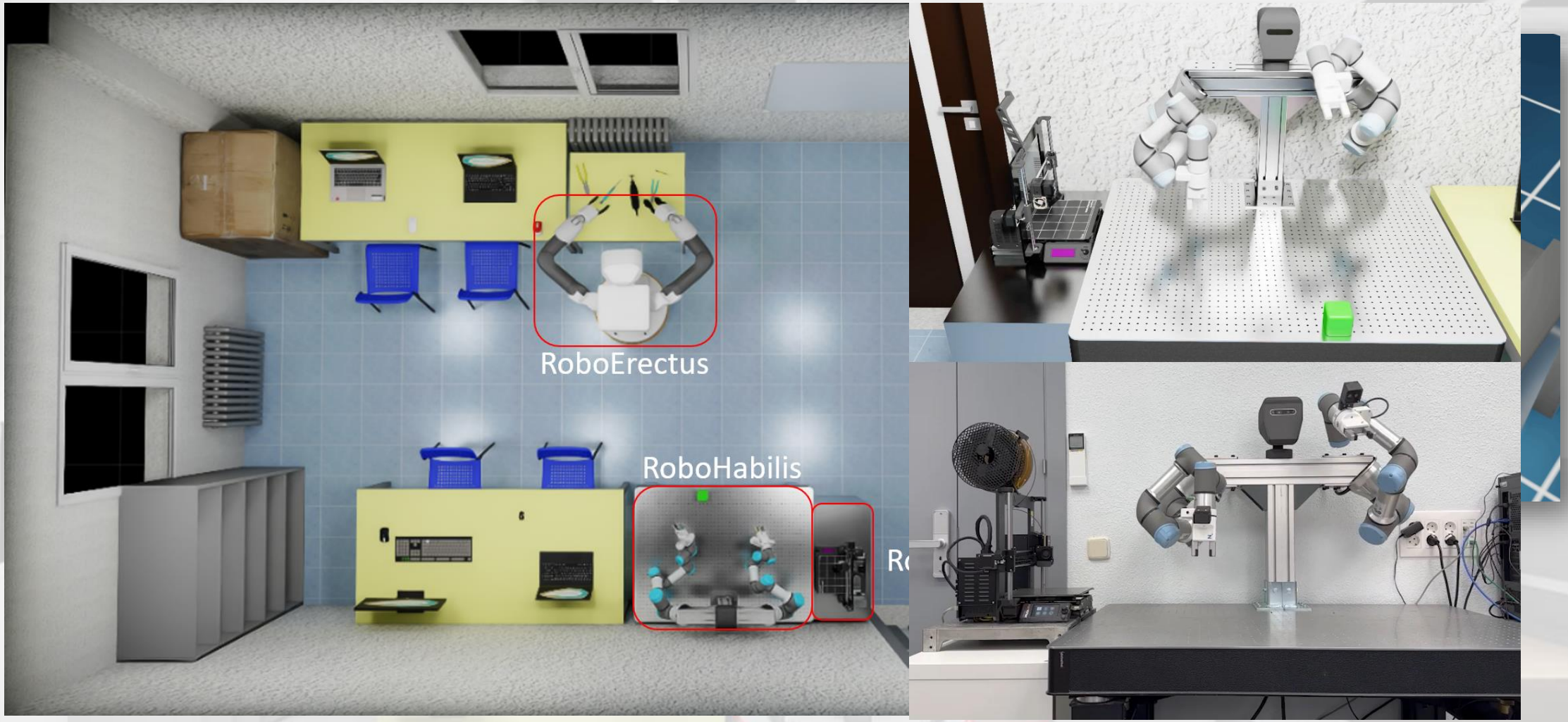


¿Por qué estamos usando Nvidia Isaac Sim y no otro simulador?

1. Simulación de Alta Calidad
2. Integración con ROS y ROS 2
3. Soporte para Escenarios Complejos
4. Flexibilidad y Extensibilidad
5. Soporte y Comunidad
6. Rendimiento y Escalabilidad



¿Por qué estamos usando Nvidia Isaac Sim y no otro simulador?



Simulación de Manipuladores

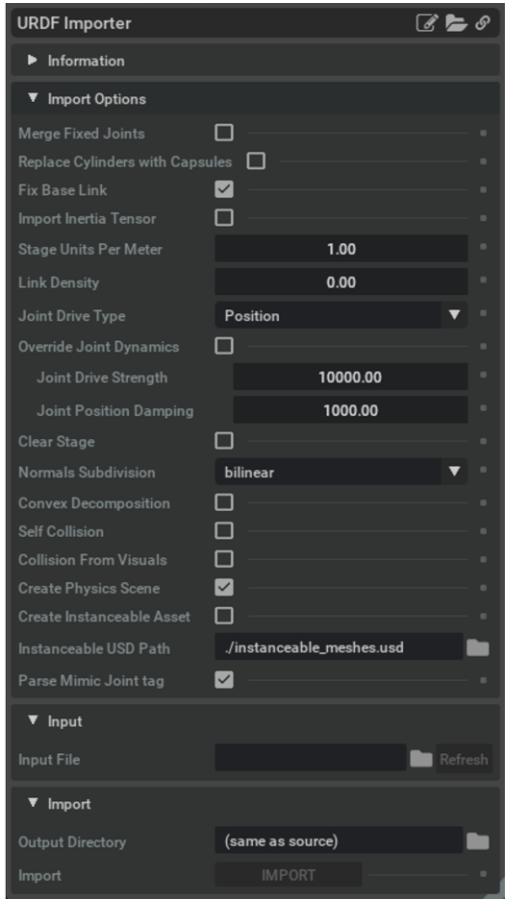
Instalación y Configuración:

- **Operating System**: Ubuntu 20.04 LTS or later (preferred); Docker is also available for Windows and macOS.
- **Docker**: Docker Engine and Docker Compose installed (version 1.27+ recommended).
- **CPU**: Multi-core processor (e.g., Intel Core i7 or equivalent).
- **GPU**: NVIDIA GPU with CUDA support (e.g., RTX 2060 or higher) and driver version **525.60.11** or later.
- **RAM**: Minimum 8 GB (16 GB recommended).
- **Storage**: At least 50 GB of free disk space for Docker images and data.
- **Network**: Stable internet connection for downloading Docker images and dependencies.

https://github.com/DarK404/Manipulators-simulation-workshop-roscon-es-2024

URDF importation

URDF (Unified Robot Description Format) es un formato basado en XML que se utiliza para describir modelos de robots en simulaciones y aplicaciones de robótica.




```xml
<robot name="simple_robot">
  <!-- Definimos el primer enlace (base_link) -->
  <link name="base_link">
    <inertial>
      <mass value="1.0"/>
      <origin xyz="0 0 0"/>
      <inertia ixx="0.1" ixy="0.0" ixz="0.0" iyy="0.1" iyz="0.0" izz="0.1"/>
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <box size="1 1 1"/>
      </geometry>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <box size="1 1 1"/>
      </geometry>
    </collision>
  </link>

  <!-- Definimos el segundo enlace (link_1) -->
  <link name="link_1">
    <inertial>
      <mass value="1.0"/>
      <origin xyz="0 0 0"/>
      <inertia ixx="0.1" ixy="0.0" ixz="0.0" iyy="0.1" iyz="0.0" izz="0.1"/>
    </inertial>
    <visual>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <cylinder length="1.0" radius="0.1"/>
      </geometry>
    </visual>
    <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <cylinder length="1.0" radius="0.1"/>
      </geometry>
    </collision>
  </link>

  <!-- Definimos la articulación entre base_link y link_1 -->
  <joint name="joint_1" type="revolute">
    <parent link="base_link"/>
    <child link="link_1"/>
    <origin xyz="0 0 0.5" rpy="0 0 0"/>
    <axis xyz="0 0 1"/>
    <limit lower="-1.57" upper="1.57" effort="10" velocity="1"/>
  </joint>
</robot>
```

Articulation Inspector

El **Articulation Inspector** en Nvidia Isaac Sim es una herramienta que facilita la inspección y el ajuste de robots articulados, como manipuladores UR o robots humanoides. Proporciona una interfaz gráfica para visualizar y modificar la configuración de las articulaciones del robot y sus propiedades físicas.

Robot Description File

¿Qué es un Robot Description File?

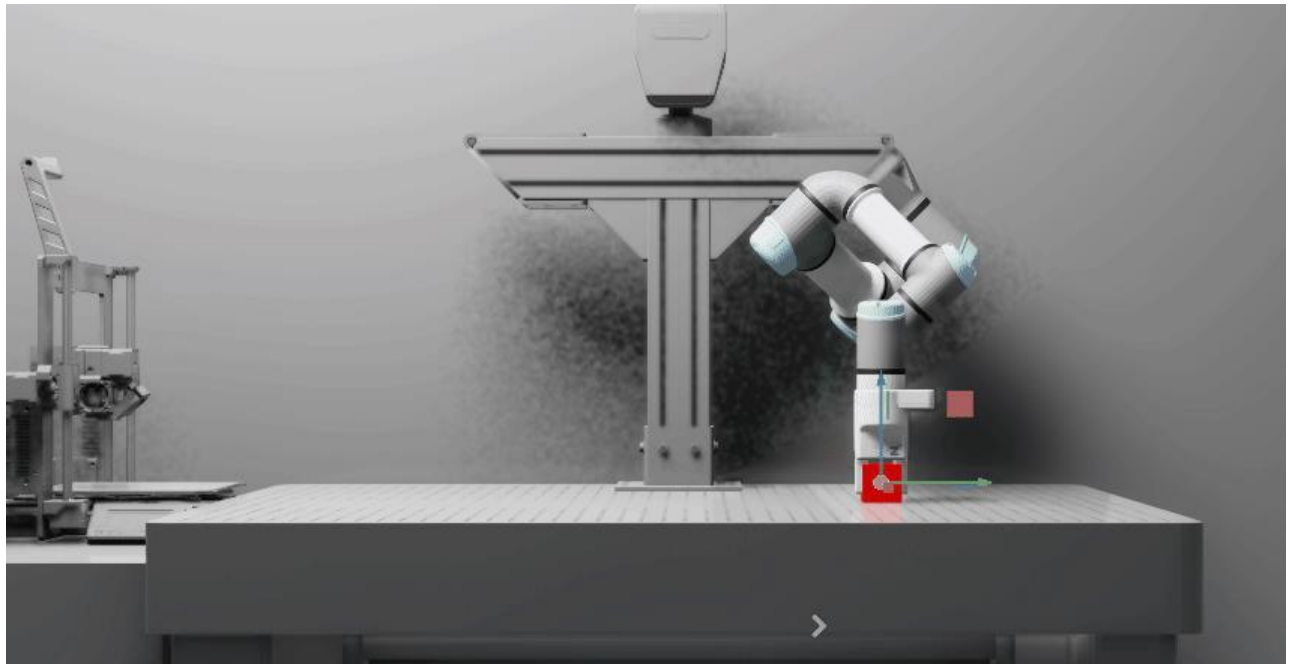
Es el archivo de configuración principal necesario para usar los algoritmos de **LULA**, junto con el archivo URDF del robot.

Definiendo el C-Space del Robot: Juntas Activas y Fijas

Juntas Activas: Son las que se controlan directamente mediante los algoritmos de LULA.

Juntas Fijas: Se consideran fijas desde la perspectiva de los algoritmos y no se controlan directamente.

Robot Description File

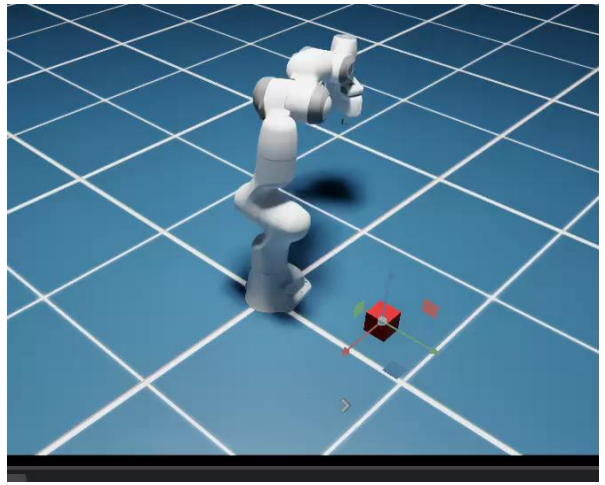


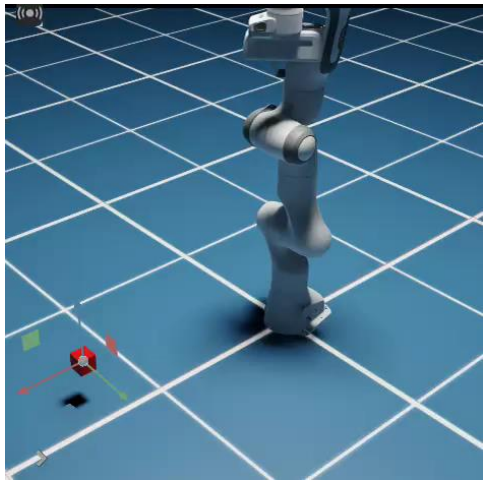
Esferas de Colisión

Configuración : Se deben definir esferas de colisión que cubran la superficie del robot.

El Robot Description Editor proporciona herramientas para definir rápidamente estas esferas.

```
collision_spheres:
    shoulder_link:
        - "center": [0.0, 0.0, 0.0]
        "radius": 0.1
    upper_arm_link:
        - "center": [-0.416, -0.0, 0.143]
        "radius": 0.078
        - "center": [-0.015, 0.0, 0.134]
        "radius": 0.077
        - "center": [-0.14, 0.0, 0.138]
        "radius": 0.062
        - "center": [-0.285, -0.001, 0.139]
        "radius": 0.061
        - "center": [-0.376, 0.001, 0.138]
        "radius": 0.077
        - "center": [-0.222, 0.001, 0.139]
        "radius": 0.061
        - "center": [-0.055, 0.008, 0.14]
        "radius": 0.07
        - "center": [-0.001, -0.002, 0.143]
        "radius": 0.076
```





¿Qué pasa si tenemos una mala configuración?

Recape: Simulación de Manipuladores

Importación URDF: Cargar el archivo URDF del robot en Nvidia Isaac Sim.

Articulation Inspector: Usar la herramienta Articulation Inspector para verificar y ajustar las articulaciones del robot.

Robot Description File: Definir un archivo de descripción del robot, asignando juntas activas y fijas, y configurando esferas de colisión.

Generación de movimiento

Algoritmos de Planificación de Rutas (Path Planning)

•**Definición**: Estos algoritmos se encargan de generar una ruta geométrica desde un punto inicial hasta un punto final, pasando por puntos intermedios predefinidos, ya sea en el espacio articular (c-space) o en el espacio operativo del robot.

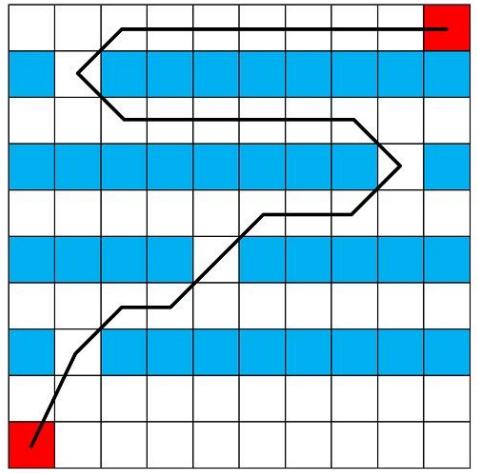
•**Objetivo**: Encontrar un camino que el manipulador debe seguir sin considerar el tiempo ni las dinámicas del movimiento.

•**Ejemplo**: Un algoritmo de planificación de rutas podría calcular el trayecto más corto o evitar obstáculos estáticos en un entorno.

Rapidly-exploring Random Trees (RRT)

*A (A-Star)**

Probabilistic Roadmap (PRM)



Algoritmos de Planificación de Trayectorias (Trajectory Planning)

•**Definición**: Una vez que se tiene la ruta geométrica, la planificación de trayectorias añade información temporal, determinando cómo moverse a lo largo de esa ruta, es decir, con qué velocidad y aceleración.

•**Objetivo**: Optimizar el movimiento del robot a lo largo del tiempo, garantizando un control suave y eficiente del manipulador.

•**Ejemplo**: Calcular la velocidad y aceleración adecuadas para llegar a un objetivo en un tiempo determinado sin superar las limitaciones físicas del robot.

Polynomial Trajectories

Linear Quadratic Regulator (LQR)

Time-Optimal Trajectories (TOT)

Generación de Movimiento (Motion Generation)

•**Definición**: La generación de movimiento toma los resultados de la planificación de rutas y trayectorias y ejecuta esos movimientos en tiempo real, considerando tanto la dinámica del robot como los cambios en su entorno.

•**Objetivo**: Asegurar que el robot sigue el camino planificado mientras evita obstáculos dinámicos y adapta su movimiento a las condiciones del entorno.

•**Diferencia Clave**: A diferencia de los algoritmos de planificación, la generación de movimiento se encarga de la ejecución real del movimiento, adaptándose a los imprevistos durante la operación.

•**Ejemplo**: Un robot que ajusta su trayectoria cuando detecta un obstáculo en movimiento o cambia la velocidad según las condiciones del entorno.

Dynamic Movement Primitives (DMPs)

RMPflow (Reactive Motion Planning Flow)

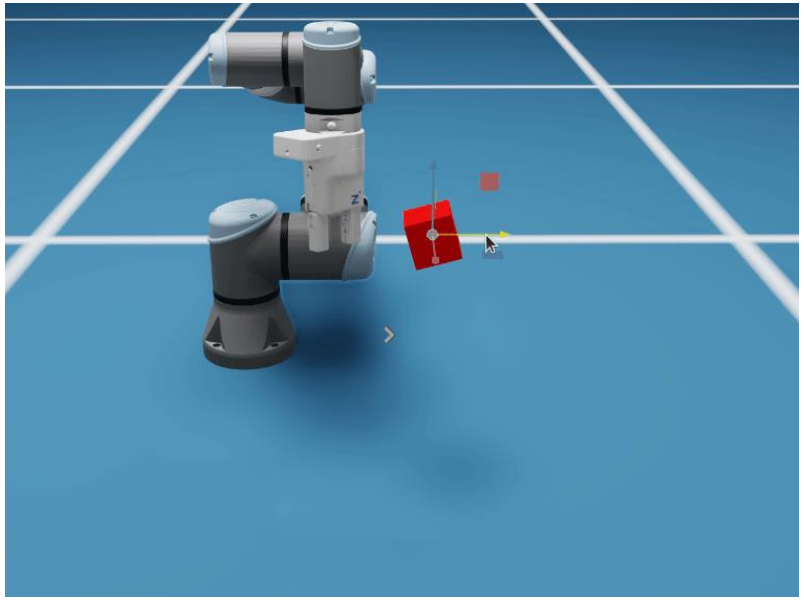
Model Predictive Control (MPC)

¿Qué es Curobo?

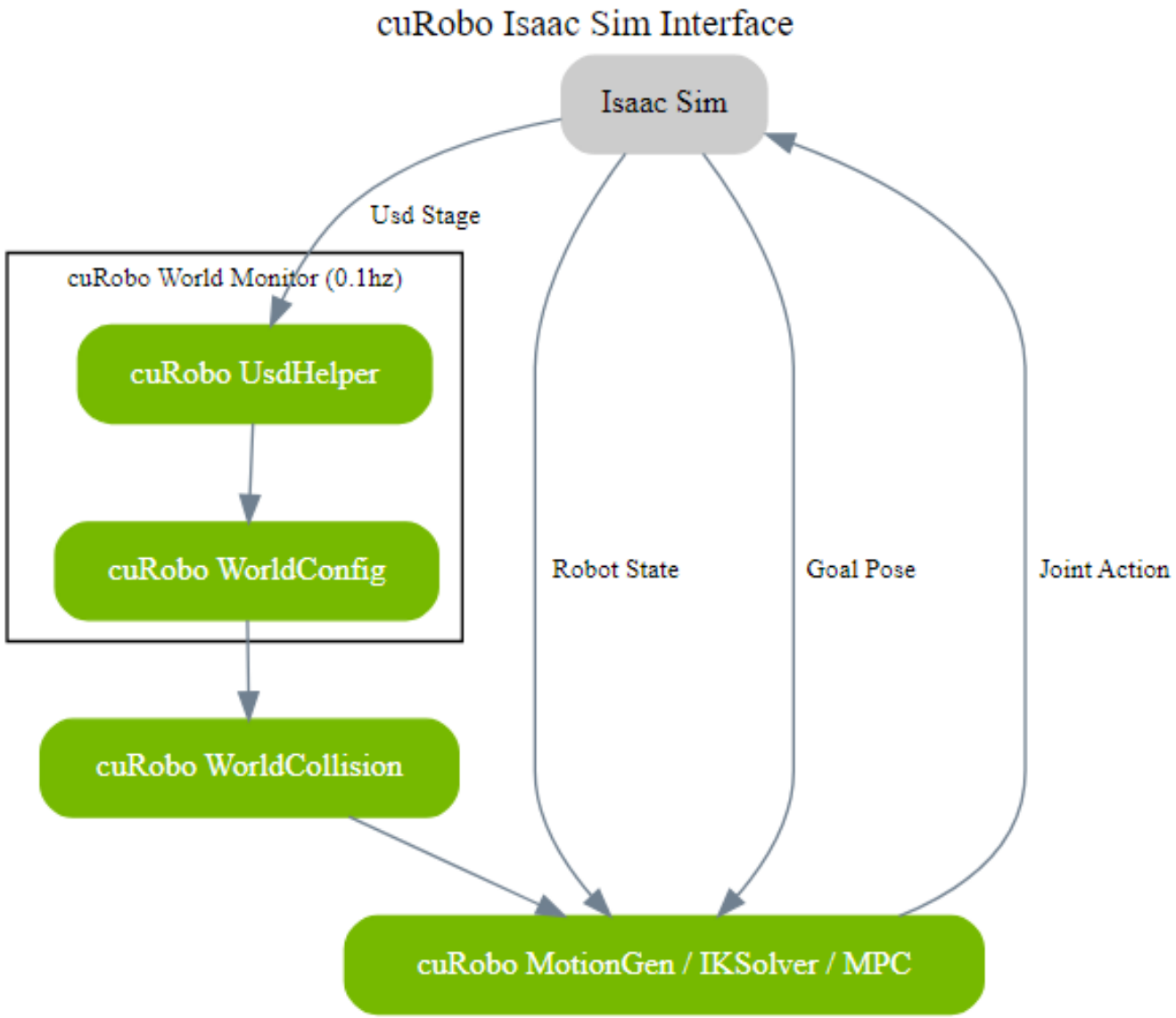
Curobo: CuRobo es una biblioteca de generación de movimiento acelerada por GPU que se integra con Omniverse Isaac Sim, ofreciendo soluciones avanzadas para la planificación de movimientos robóticos.

2. Características Principales:

- **Cinemática Inversa Sin Colisiones**
- **Planificación de Movimiento Sin Colisiones**
- **Control Dinámico**
- **Cinemática Inversa Sin Colisiones por Lotes (Batched)**

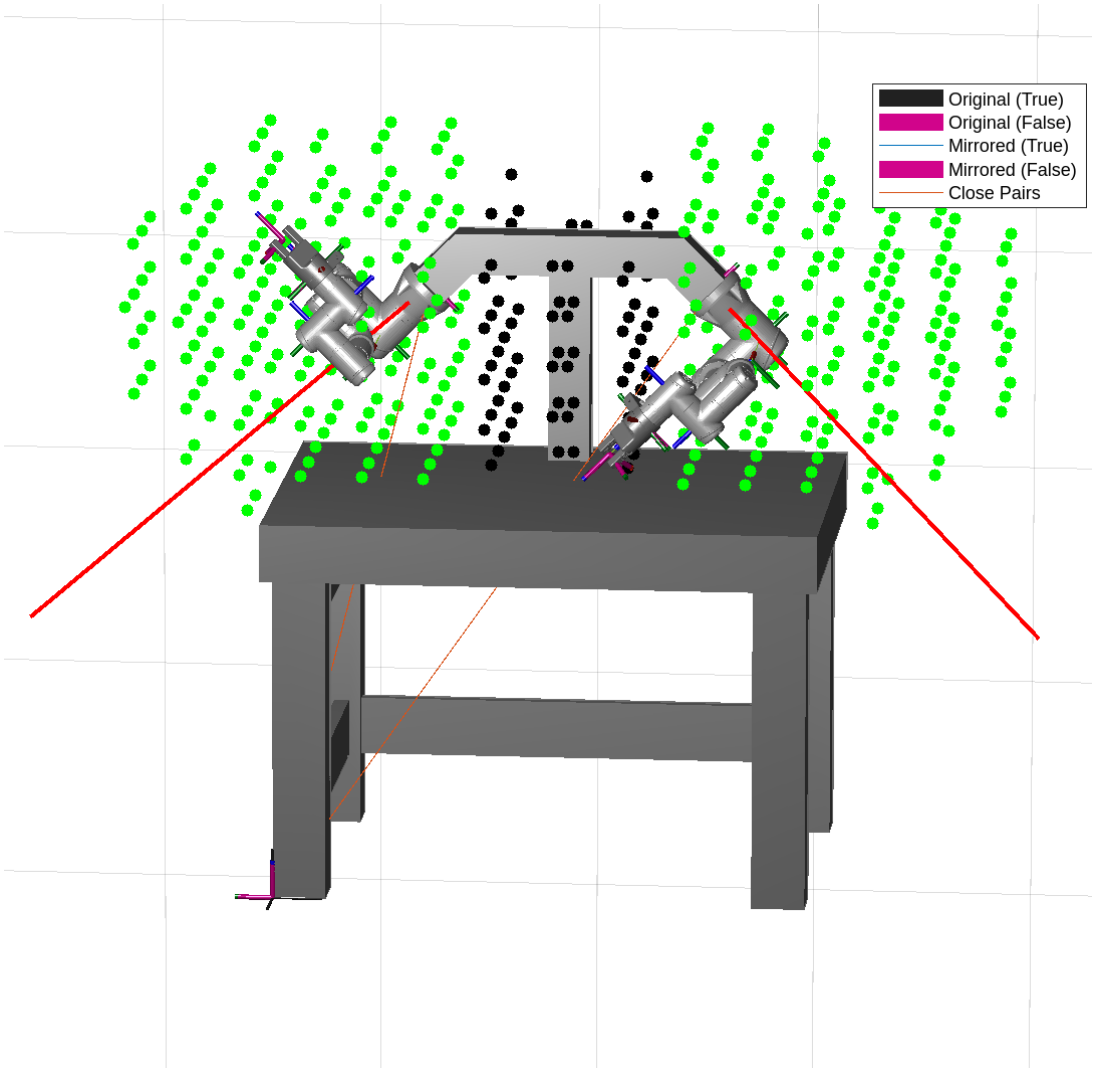


2024/09/19

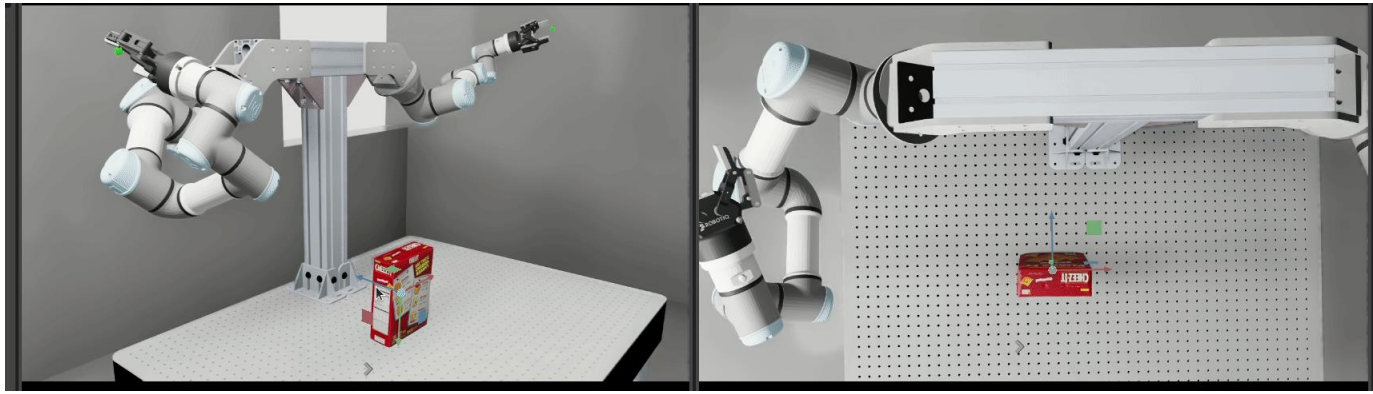

cuRobo Isaac Sim Interface
Isaac Sim
Usd Stage
cuRobo World Monitor (0.1hz)
cuRobo UsdHelper
cuRobo WorldConfig
cuRobo WorldCollision
Robot State
Goal Pose
Joint Action
cuRobo MotionGen / IKSolver / MPC

Ejemplo de Aplicación de Cinemática Inversa Sin Colisiones por Lotes (Batched)

La **cinemática inversa sin colisiones por lotes** es una herramienta poderosa para evaluar la alcanzabilidad de un robot en su espacio de trabajo de manera eficiente. Esta técnica permite calcular soluciones de cinemática inversa para múltiples poses simultáneamente, utilizando la aceleración de GPU para optimizar el proceso.



Visión por Cámara en Simulaciones



Concepto: La visión por cámara permite a los robots "ver" su entorno mediante cámaras integradas. En la simulación, esto se traduce en el uso de modelos virtuales de cámaras para captar y analizar imágenes del entorno simulado.

Ventajas:

- **Percepción del Entorno**

  Mejora la capacidad del robot para identificar y localizar objetos.

- **Interacción Dinámica**

  Permite la interacción con objetos y obstáculos en tiempo real.

cómo agregar y configurar una cámara en el entorno de Isaac Sim utilizando Python y el API de USD.

1. **Definición del Prim de la Cámara**

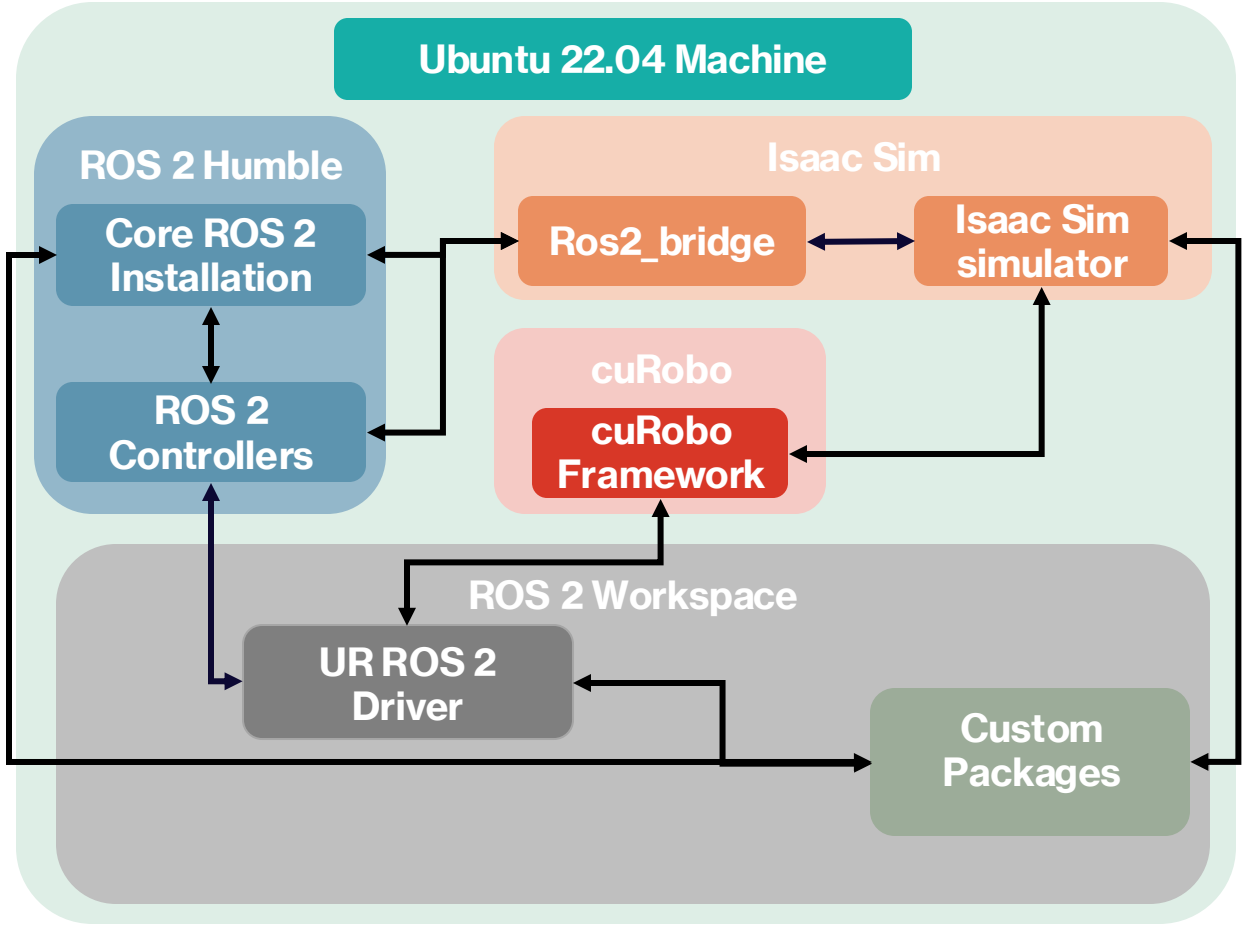
```
path="/World/Camera/MyCamera"
prim_type="Camera"
camera_prim = stage.DefinePrim(path, prim_type)
```

2. **Configuración de la Transformación de la Cámara**

```
translation=(0,0,5)
rotation=(0,0,-90)
xform_api=UsdGeom.XformCommonAPI(camera_prim)
xform_api.SetTranslate(translation)
xform_api.SetRotate(rotation)
```

Prim Class	Prim Type Name
UsdGeom.Mesh	Mesh
UsdGeom.Camera	Camera
UsdLux.RectLight	RectLight

Integración Completa con ROS2


Ubuntu 22.04 Machine
ROS 2 Humble
Core ROS 2 Installation
ROS 2 Controllers
Isaac Sim
Ros2_bridge
Isaac Sim simulator
cuRobo
cuRobo Framework
ROS 2 Workspace
UR ROS 2 Driver
Custom Packages

Integración con ROS2 usando omni.isaac.ros2_bridge

Descripción General

- **omni.isaac.ros2_bridge** es una extensión que conecta Omniverse Isaac Sim con ROS2, facilitando la integración y comunicación entre el simulador y los sistemas basados en ROS2.

Publicación y Suscripción de Temas:

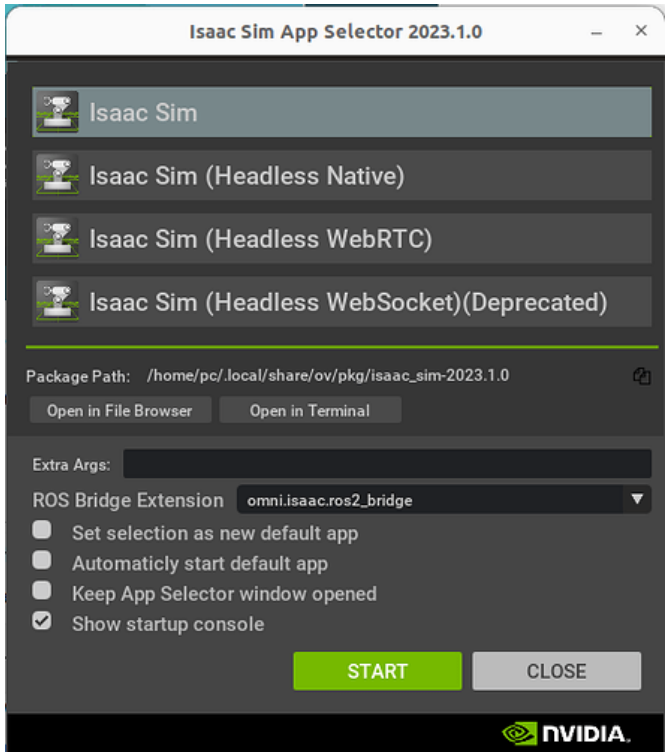
Publicación: Envía datos desde el simulador a ROS2.

Suscripción: Recibe comandos y datos desde ROS2 para controlar el simulador.

Servicios y Acciones:

Servicios: Permiten ejecutar comandos específicos en el simulador.

Acciones: Manejan tareas complejas y proporcionan una interfaz para ejecutar algoritmos en el simulador.



Nodos de Omnigraph

- **omni.isaac.ros2_bridge** es una extensión que conecta Omniverse Isaac Sim con ROS2, facilitando la integración y comunicación entre el simulador y los sistemas basados en ROS2.

Publicación y Suscripción de Temas:

Publicación: Envía datos desde el simulador a ROS2.

Suscripción: Recibe comandos y datos desde ROS2 para controlar el simulador.

Servicios y Acciones:

Servicios: Permiten ejecutar comandos específicos en el simulador.

Acciones: Manejan tareas complejas y proporcionan una interfaz para ejecutar algoritmos en el simulador.

```
from omni.isaac.core.utils.extensions import enable_extension

enable_extension("omni.isaac.ros2_bridge")
```

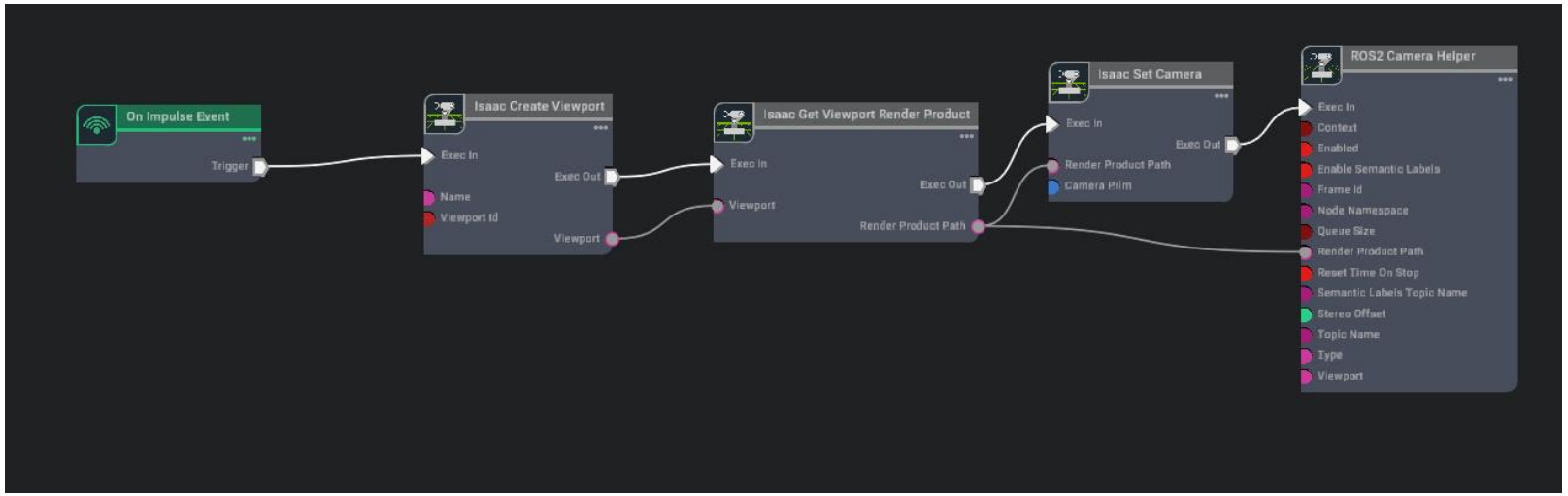
Nodos de Omnigraph

•Las funciones relacionadas con ROS y ROS2 están disponibles en forma de nodos en Omnigraph.

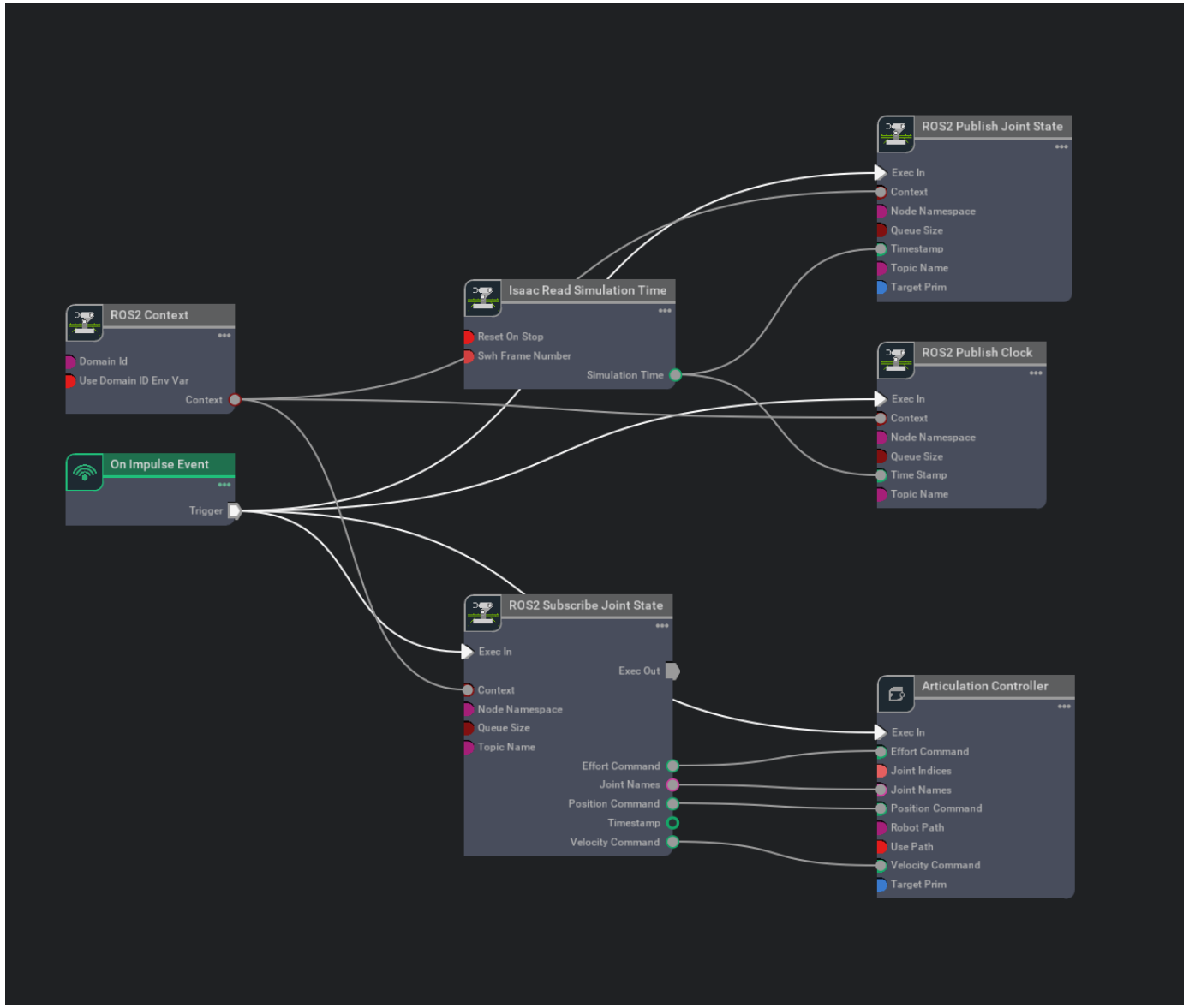
•Para usar Omnigraph, ve a **Window > Visual Scripting > Action Graph**.

•Los nodos relacionados con ROS se encuentran bajo **Isaac Ros** y los relacionados con ROS2 bajo **Isaac Ros2**.

Creación del gráfico para un publicador RGB:



Creación del graph para control del manipulador:



Ejercicios Prácticos

Gracias por su atención !

MetaTool
Robots inventing tools

https://www.metatool-project.eu/