Omniverse ISAAC SIM 基础教程

0. 学习本课程所需的背景知识

- USD：Pixar Universal Scene Description (USD) | NVIDIA Developer 了解 USD Stage, Layer, Prim, reference 等概念

1. ISAAC SIM 概览：

- 官方文档：What Is Isaac Sim? — Omniverse Robotics documentation (nvidia.com)
- ISAAC SIM 能为智能机器人开发提供什么：
    - 导入场景，导入机器人模型，构建传感器：
        - 图形界面 GUI, 各种导入模块 Importers, 增益调节器 Gain Tuner, 关节检查器 Articulation Inspectors
    - 训练和调试算法：
        - Core API (专用于机器人应用的 API), Isaac Replicator (用于 AI 训练的 3D 合成数据生成)，ISAAC Gym(用于强化学习训练和推理的接口)
    - 与其他开发平台或者机器人通信：
        - ROS/ROS2 Bridges
    - 底层指令：（关节控制与避障）
        - Lula 运动学求解器， 运动策略，运动生成器
    - 上层指令：（行为选择与控制）
        - ISAAC Cortex （用于规划机器人行为的决策框架）

2. ISAAC SIM 开发流程：

- 使用 OV Launcher 打开 Isaac Sim，几种不同的本地可视化或者远程串流方法
    - 使用脚本打开 Isaac Sim：./local/share/ov/pkg/isaac_sim-2022.1.0/
    - 远程串流方法：串流客户端使用文档

- 图形界面 GUI:
    - 构建环境和机器人，添加传感器，并且保存场景为 USD
    - 您可以在下面菜单找到 ISAAC 相关资产和案例:
        - Create 菜单
        - Isaac Utils, Isaac Examples, Synthetic Data
        - Extensions 菜单
- Extension:
    - 方便地与现有仿真器进行交互，所有扩展都异步地执行，所有更改都会重新热加载；
    - 添加机器人，做个小任务，展示如何查看或者改写源码
- 使用 built-in 交互式脚本编辑器(Script Editor) : Windows -> Script Editor 打开

- Standalone python 脚本:

- 可以显式地控制物理时间步和渲染时间步，可以以无图形界面（headless）模式运行

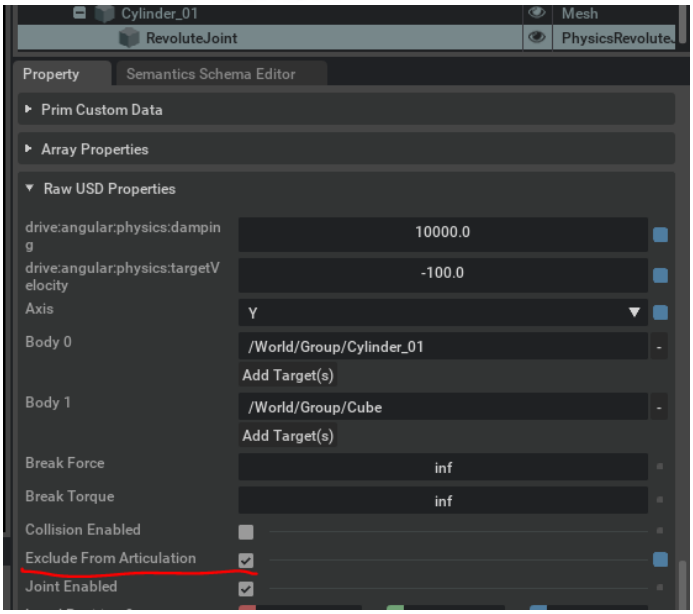
3. 常见的应用场景：

- 训练智能机器人：
  - 采用 Standalone python 脚本来定义环境和 Task, 采用 RL 算法训练机器人
- 调试智能机器人：
  - 通过 GUI 来配置场景和机器人，ROS bridges 进行通信， Extension 来与 Isaac SIM 仿真器交互

典型机器人应用	GUI /Omnigraph	Standalone Python	Extension
ROS	▪ 场景构建 ▪ 机器人导入与调试 ▪ 添加传感器 ▪ 构建 ROS bridges	▪ 控制消息发布频率	▪ 定义 Task，方便地与仿真器交互
强化学习	▪ 场景构建 ▪ 机器人导入与调试 ▪ 添加传感器	▪ 系统性地环境配置 ▪ 配置 Controller 和 Task ▪ 配置 RL 算法	

4. 模型和场景导入案例：导入一个机器人，然后简单控制它。

- GUI 使用教程：展示从零搭建一个物理小车的案例
  - 理解 visual geometry 跟 physical geometry 的含义
  - 理解 rigid body, joints, articulation 和 collider 的含义，添加和修改 Materials
  - 理解 stiffness 和 damping 参数的含义：
    - Joint Drives 会根据 stiffness 和 damping 参数对关节施加力（力矩）来维持给定的位置或者速度目标：
    - 施加的力（力矩）的大小正比于：*stiffness * (position - target_position) + damping * (velocity - target_velocity)*
  - P.S.: Articulation 结构中不能有环，如果有环，需要在对于特定的关节，在其 USD Properties 中勾选 *Exclude from Articulation*



- 机器人导入流程的改进：导入模型 + 增益调节器 + 关节检查器
    - URDF (ROS):
        - Xacro to urdf: `rosrun xacro xacro.py model.xacro > model.urdf`
    - MJCF (Mujoco): Isaac Utils -> Workflows -> MJCF importer
    - Onshape : File -> Import from Onshape
    - Other CAD: omni.kit.importer.cad

    - Gazebo SDF: The Ignition-Omniverse connector with Gazebo
    - 检查和调整：参考 GTC22 session , 导入第三方 abb 机械臂，并且检测和调整配置参数。
- 在 warehouse 场景中，通过 URDF 导入 turtlebot3 机器人，添加相机和 Lidar 传感器。

5. Core API 案例：

- **Core API 定义**: Omniverse Isaac Sim 利用 Omniverse™ Kit 工具包来构建其应用程序，它提供了用于脚本编写的 Python 解释器。每一个 GUI 命令以及许多附加功能都可以通过 Python API 调用。然而，使用 Pixar 的 USD Python API 与 Omniverse 工具包接口的学习曲线很陡峭，步骤往往很繁琐。因此，我们提供了一组设计用于**机器人应用程序的 API**，这些 API 可以抽象出 USD API 的复杂性，并将多个步骤合并为一个步骤，方便用于频繁执行的任务。
- World 是一个核心类，您能够以简单和模块化的方式与模拟器交互。它处理许多与时间相关的事件，例如添加回调函数、迭代物理时间步、重置场景、添加任务等。
    - World 包含 Scene 的实例。Scene 类管理 USD stage 中与仿真相关的资产。它提供了一个简单的 API，可以在 USD stage 中添加、操作、检查和重置不同的 USD 资产。
    - World 是单例，也就是说在 ISAAC SIM 运行时只能有一个 World 存在。
    - 通过 setup_scene()方法，从**空的** Stage 开始向 World 中添加 3D 资产。
- 展示 Hello World（BaseSample）案例，添加 jetbot_controller 功能。
- Task 类提供了一种将场景创建、信息检索和计算指标模块化的方法。使用高级逻辑来创建更复杂的场景。
    - 通过 set_up_scene()方法添加 3D 资产到场景，通过 get_observations()方法来从 simulator 获取观测信息， 通过 pre_step()方法来逐时间步执行控制指令。
- 展示 Hello World（BaseSample）案例，添加 Franka_Pick & Place_controller 功能。

6. Standalone example

- 开启 SIM，添加不同属性的 cubes，开始仿真：
    - standalone_examples/api/omni.isaac.core/add_cubes.py
    - VisualCuboid (GeometryPrim) DynamicCuboid (RigidPrim, VisualCuboid)
- headless 模式: 快速进行 RL 训练
    - standalone_examples/api/omni.isaac.jetbot/stable_baselines_example/train.py

7. ROS 应用案例：

7.1 基本介绍:

ROS 是最受欢迎的开源机器人操作系统，ROS 提供了一套标准的通讯方式来连接所有的传感器和执行器，并与控制软件相连通，您可以模块化地封装各种不同的传感器为 ROS 节点。ROS 为复杂的机器人感知和控制提供了各种开源的库，如地图构建和定位（SLAM）以及导航（Navigation）等等。

7.2 ROS2 与 ROS 的区别:

- ROS1: 集中式发现节点，可靠的 internet 协议（TCP IP）
    - roscore, rosmaster: 消息/主题可供所有其他节点查看
- ROS2 中间件接口(rmw): DDS/RTPS (数据分发服务/实时发布-订阅) - 分布式发现节点，多个逻辑网络共享同一物理网络
    - Domain ID: 只有同一逻辑网络中的节点可以自由传递消息

7.3 通过 OmniGraph 配置 ROS/ROS2 Bridges:

- 可视化脚本：在 Window -> Visual Scripting 打开 Omni Graph Editor: 搜索跟 ROS 相关的 Node，搭建 ROS bridges.
- Standalone ROS python 脚本: 通过 python 脚本来启动 SIM，配置 ROS bridges.

7.4 OmniGraph 简介

- 连接和组织基于 OV 的应用程序
- 为灵活性和模块化而设计
- 粒度范围大
- 可视化编程（Visual Scripting）
    - 节点基于 C++/Python/Cuda 实现，图构建通过 Python 或 GUI 实现。
- 清晰的数据流和计算
- 清晰的输入/输出、全局和本地数据
- 后台调度和 GPU 数据处理

7.5 ROS2 OmniGraph 节点:

- Tick: (**On Playback Tick Node**)
- Timestamp: (**Isaac Read Simulation Time Node**)
- 配置 Domain ID: (**ROS2 Context Node**)
- TF: (**ROS2 Publish Transform Tree Node**)
    - 适用于具有父子关系的坐标系框架
    - 配置 inputs:parentPrim, inputs:targetPrims
    - 关节对象的 Transform: 关节树中所有的后继坐标框架都自动添加到 TF 中
        - 确保添加的是 articulation root prim
    - 使用 sim time :
        - 注意：在计算两个对象之间的 TF 时，它们会获取时间戳最接近的两条消息并计算坐标变换，因此如果时间戳显示的差别太大，则它将不起作用.

- Raw TF: (**Isaac Compute Odometry Node + ROS2 Publish Raw Transform Tree Node**)
    - 添加跟没有 prim 对应的虚拟坐标系框架之间的 TF 关系
    - 比如 odom 并不是一个真实存在的框架，所以 base_link 与 odom 之间的 TF 需要依赖 Raw TF Tree Node
    - **Isaac Compute Odometry Node:** 配置 inputs:chassisPrim (articulation root prim)
    - **ROS2 Publish Raw Transform Tree Node**: 配置 childFrameId: base_link, parentFrameId: odom, topicName: tf
    - 使用 sim timestamp
- 里程计 Odometry: (**Isaac Compute Odometry Node + ROS2 Publish Odometry Node**)
    - **Isaac Compute Odometry Node:** 配置 inputs:chassisPrim (articulation root prim)
    - **ROS2 Publish Odometry Node:** 配置 chassisFrameId: base_link, odomFrameId: odom, topicName: odom
    - 使用 sim timestamp
- 机器人控制器:
    - 移动机器人底盘：(**ROS2 Subscribe Twist Node + Differential Controller Node + Articulation Controller Node**)
        - **ROS2 Subscribe Twist Node:** 配置 topicName: cmd_vel
        - **Differential Controller Node:** 配置 wheelDistance, wheelRadius
        - **Articulation Controller Node:** 配置 inputs:TargetPrim, Joint Names
    - 多关节机械臂：
        - **ROS2 Publish Joint States Node + ROS2 Subscribe Joint States Node:** 配置 topicName: joint_states/joint_command
- Clock: 发布仿真时间
    - **Isaac Read Simulation Time Node**
    - **ROS2 Publish Clock Node**
- 其他传感器:
    - Lidar: (**Isaac Read Lidar Beams Node + ROS2 Publish Laser Scan**)
        - 配置 inputs:lidarPrim
        - 配置 frameId: carter_lidar, topicName: scan
    - Camera:
        - **Isaac Create Viewport Node:** 配置 viewportId
        - **Set Active Camera Node:** 配置 Camera Path
        - **ROS2 CameraHelper Node**
            - 与 viewport 绑定
            - 配置 topicName, type: camera_info, rgb, depth, depth_pcl, bbox_2d, bbox_3d, instance_segmentation, semantic_segmentation
            - Camera Helper Node 为用户抽象出一个复杂的后处理过程：当开始仿真后，ActionGraph List 中会出现一个新的 omnigraph 图:

/Render/PostProcessing/SDGPipeline，这是有 Camera Helper Node 自动创建的图，它从渲染器中检索相关数据，对其进行处理，并将其发送到相应的 ROS 发布服务器。此图仅在您正在运行的会话中创建。它不会保存为您 USD 资产的一部分，也不会出现在 USD Stage 中。

  - 可视化方法 1）Image_view: ros2 run image_view image_view image:=/rgb
  - 可视化方法 2）Rviz2: rviz2

- **与之前版本的 ROS Bridges 主要区别：**
  - 不再从 Create 菜单中添加，而是从 OmniGraph Node List 中查找
  - 更加模块化：
    - 过去的 Differential Base ROS Bridge 会发布包括 odom, tf, joint_states 等多个消息，现在都有独立的 OGN 来发布

7.6 ROS2 机器人导航案例：

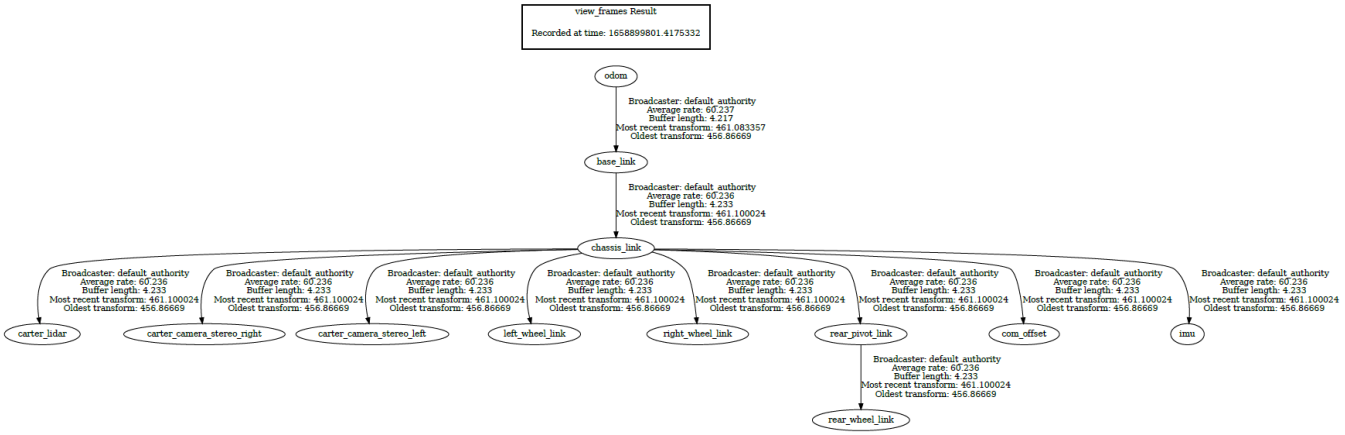
Carter navigation in warehouse

- 工作空间路径：~/.local/share/ov/pkg/isaac_sim-2022.1.0/ros2_workspace/src/navigation/carter_navigation/

source /opt/ros/foxy/setup.bash

source ~/.local/share/ov/pkg/isaac_sim-2022.1.0/ros2_workspace/install/setup.bash

ros2 run tf2_tools view_frames.py



ros2 launch carter_navigation carter_navigaiton.launch.py

7.7 ROS2 机械臂运动规划案例：

利用 Moveit2 帮助仿真中 Franka Emika Panda 机械臂进行运动规划:

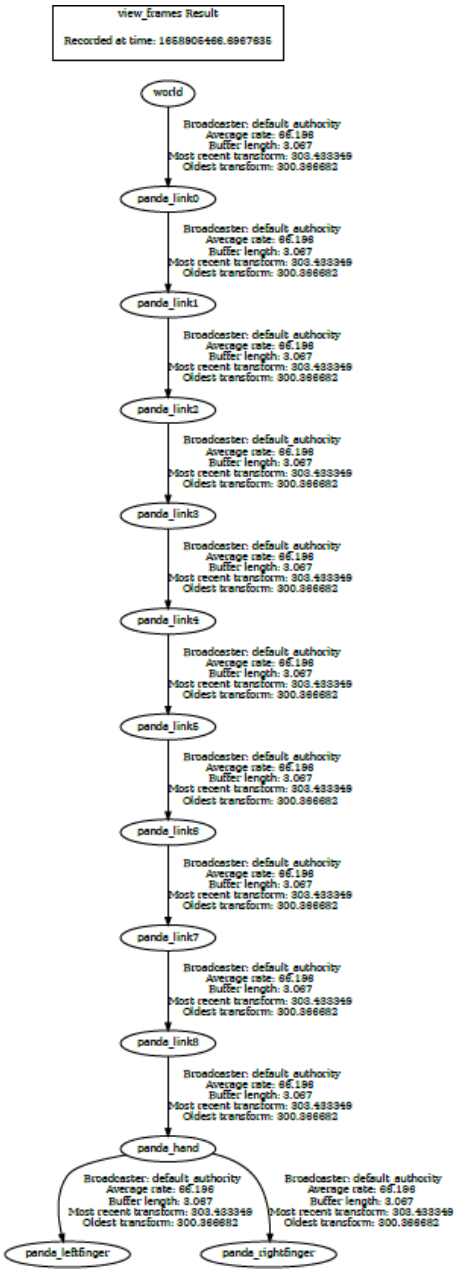
- 工作空间路径：~/.local/share/ov/pkg/isaac_sim-2022.1.0/ros2_workspace/src/isaac_moveit/

source /opt/ros/foxy/setup.bash

source ~/.local/share/ov/pkg/isaac_sim-2022.1.0/ros2_workspace/install/setup.bash

ros2 launch isaac_moveit franka_isaac_execution.launch.py

ros2 run tf2_tools view_frames.py



8. 大作业:

使用上面导入到 warehouse 中的 turtlebot3 机器人，完成各种传感器的 ROS bridges 的配置，实现在 SIM 中调试 Navigation 算法的功能。