Isaac Simの利活用を簡単にするツールの実装と利用方法

土方　優明

X:@_toshizo
GitHub: hijimasa

https://github.com/hijimasa/isaac-ros2-control-sample

1. Isaac Simとは

- NVIDIAが提供している
  ロボットソフトウェア開発プラットフォーム
- シミュレータに加えて、ソフトウェア開発に
  必要な様々なツールを内包

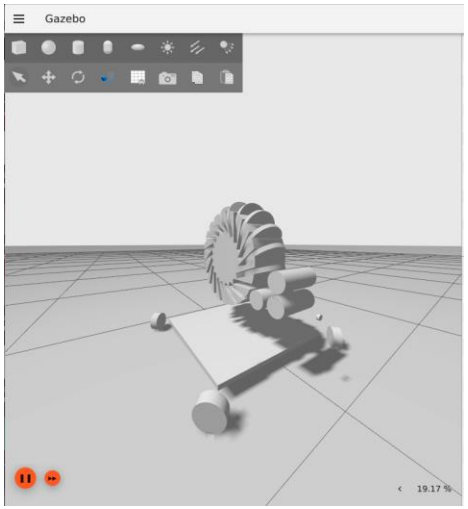
種類	Gazebo	Isaac Sim	Unity	Unreal Engine
価格	0 円	0 円 (有償サポートあり)	669,000 円/年 (1シート)	280,454 円/年 (1シート)

自分で環境構築・開発できれば無償で利用可能

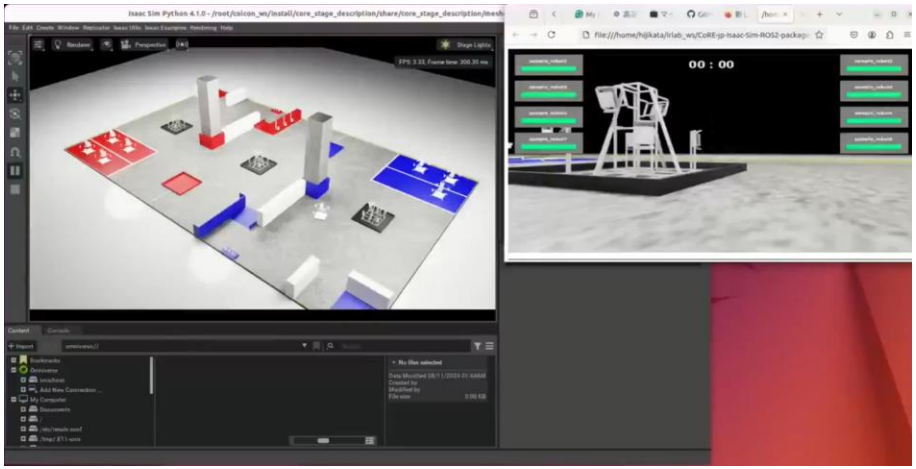
https://github.com/hijimasa/isaac-ros2-control-sample

2. Gazeboとの相違点

- フォトリアリスティックなシミュレーション
- オブジェクト数が多いとき、Gazeboより高速



Gazeboでは1台のロボットでも
ディスク8枚程度で10fpsを切る



Isaac Simなら4台のロボットで
ディスク計80枚でも10fpsで動作

全体的にグラフィックや物理エンジンが高性能

https://github.com/hijimasa/isaac-ros2-control-sample

3. ROSユーザ目線の問題

- Isaac SimではUSDファイルでロボットモデルを管理
  - URDFとUSDの2重管理の問題

- センサ情報を変換したUSDに追記
  - ロボットモデル再生成の度に作業が発生

- ros2_controlではなくNvidiaの提供するコントローラを使用
  - 既存のros2_controlの資産が流用不可能

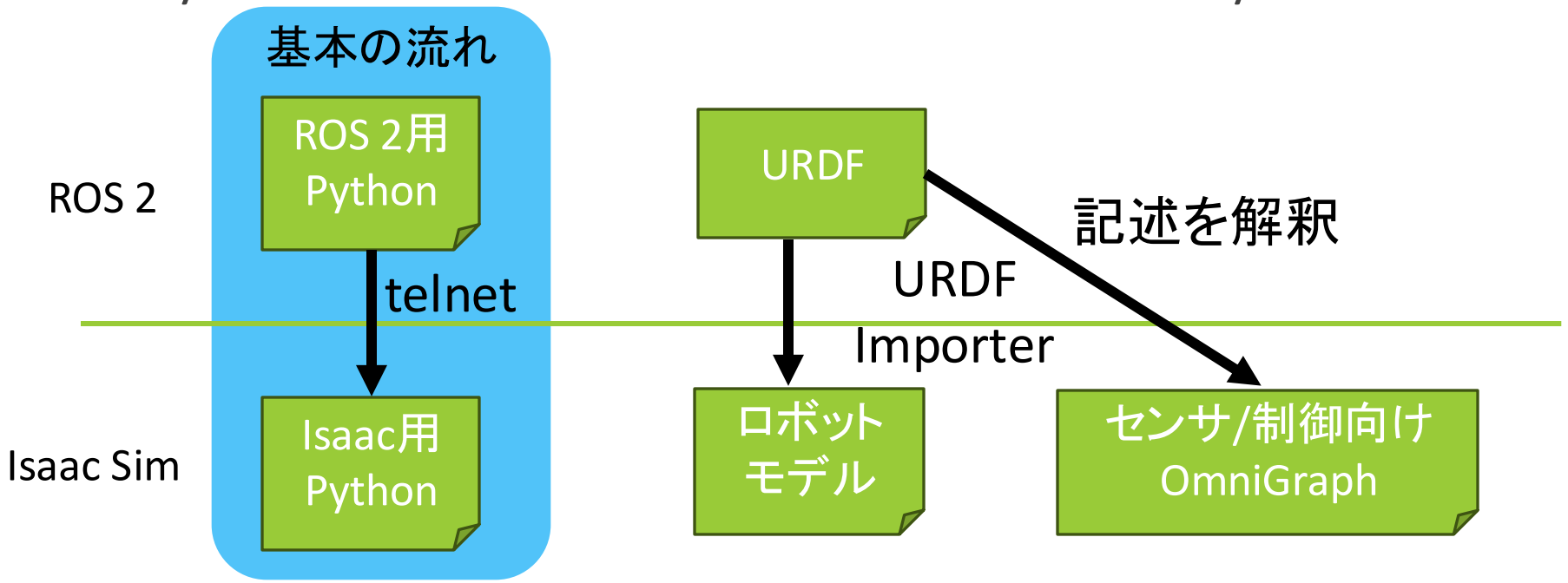
デフォルトのIsaac Simは作業が煩雑化し面倒

作業を簡単化するツールを開発

https://github.com/hijimasa/isaac-ros2-control-sample

4. 実装の方向性

- Isaac Simはほぼすべての機能をPython APIから実行可能
- Python REPL拡張機能で既存の環境に追加のPythonを実行



Python APIを活用してロボットモデルの生成を自動化

5. 代表的なURDFの記述方法

- センサ情報はgazeboタグの代わりにisaacタグに記述

```
<isaac>
 <sensor name="camera_link" type="camera">
  <topic>image_raw</topic>
  ...
 </sensor>

</isaac>
```

- ros2_controlにはトピックを活用するパッケージを活用

```
<ros2_control name="${name}" type="system">
 <hardware>
  <plugin>topic_based_ros2_control/TopicBasedSystem</plugin>
 </hardware>
</ros2_control>
```

- このほかにもmaterialを利用した摩擦係数の指定やジョイントの剛性パラメータの記述が存在

基本的な機能をURDFの記述から自動で変換

https://github.com/hijimasa/isaac-ros2-control-sample

6. 現状の問題点

- OmniGraphの数が増えると処理が重くなる
  - 単純にロボットの台数ではなく
    制御用のOmniGraphを立ち上げるとFPSが下がる

- FastDDSだとトピックが欠落することがある

- Python REPLが非推奨になっている

› **omni.isaac.repl**

› Changed

› Deprecate extension in favor of the vscode extension

参考URL:https://docs.omniverse.nvidia.com/isaacsim/latest/release_notes.html

7. まとめ

- Isaac Simにより容易にフォトリアリスティックで大規模なシミュレーションが可能
- 既存のIsaac SimはROSユーザにとって少々煩雑

- REPLを活用して既存環境にURDFをインポート
- URDFの記述をもとにセンサやros2_controlを自動生成

- ROSユーザに使いやすい開発環境を実現

みなさんもIsaac Simでシミュレーションしてみませんか

https://github.com/hijimasa/isaac-ros2-control-sample

検証用PCの構成

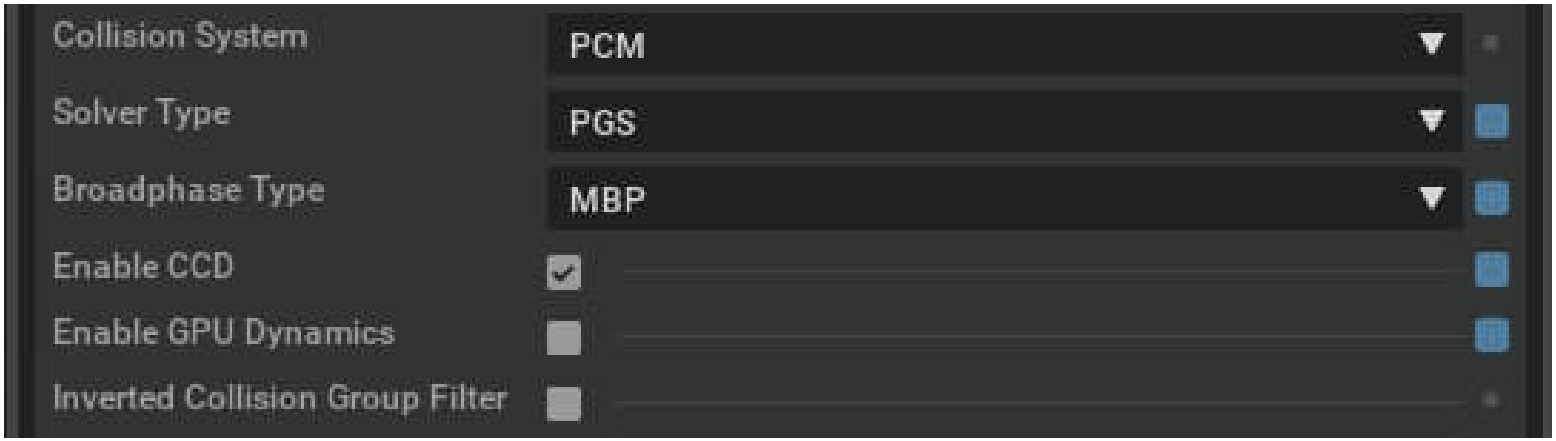
項目	詳細
CPU	AMD Rayzen9 5950x
メモリ	64GB
GPU	GeForce RTX 3090

シミュレーションの設定

以下に普段の物理演算の設定を示す

基本的に物理演算自体はCPUで処理するほうが速い傾向にある

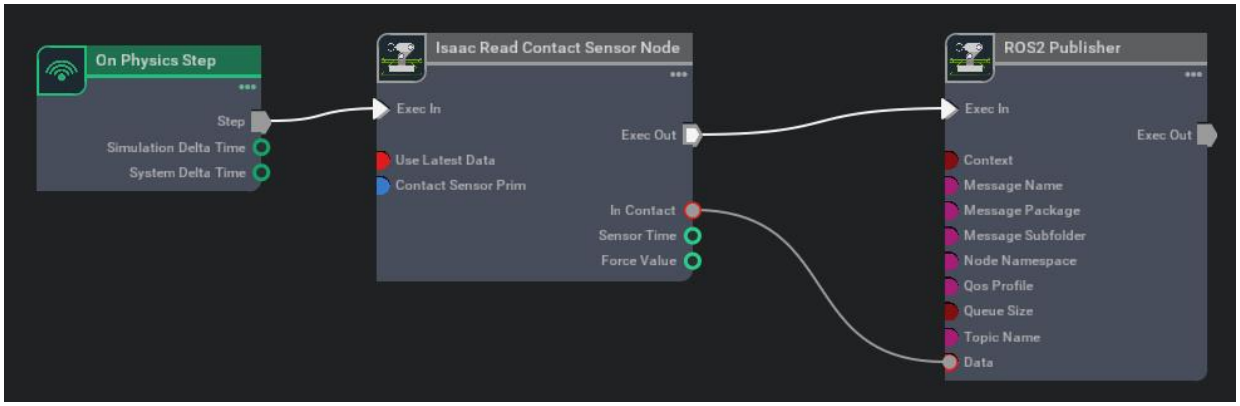
GPUは弾性体や流体のシミュレーションで使用すると思われる



https://github.com/hijimasa/isaac-ros2-control-sample

制御周期の改善

- 通常のチュートリアルでは、
「On Playback Tick」ノードによりイベントを生成している
- 上記ノードは基本的に描画ごとのイベントなので、
描画速度によって制御周期が律速される
- 「On Physics Step」ノードを用いると物理演算ステップごとのイベントになり、制御周期が改善できる



https://github.com/hijimasa/isaac-ros2-control-sample

独自ワークスペースへの適用

- Isaac SimとROS 2との連携部分は、
  isaac_ros2_utilsパッケージに分離
  https://github.com/hijimasa/isaac_ros2_utils
- 基本的にIsaac SimとROS 2のインストールされた
  実環境あるいはDocker環境のワークスペースに
  上記のパッケージを導入するだけで連携が可能

必要なAPIの検索

- 新しい機能を導入しようとするとき
  対応するPython APIを知る必要がある
- 基本はWebのドキュメントを参照することになるが、
  それだけでは扱い方がわからないAPIが存在する
- PythonAPIについては、Docker 環境であれば/isaac-sim/
  ディレクトリ内にすべてのPythonファイルがあるはずなので、
  grepコマンドで検索するとAPIがヒットすることがある

Implementation and Use of Tools to Simplify the Use of Isaac Sim

MASAAKI HIJIKATA

X:@_toshizo
GitHub: hijimasa

https://github.com/hijimasa/isaac-ros2-control-sample

1. What is Isaac Sim?

- Robot software development platform provided by NVIDIA
- In addition to the simulator, it contains various tools necessary for software development

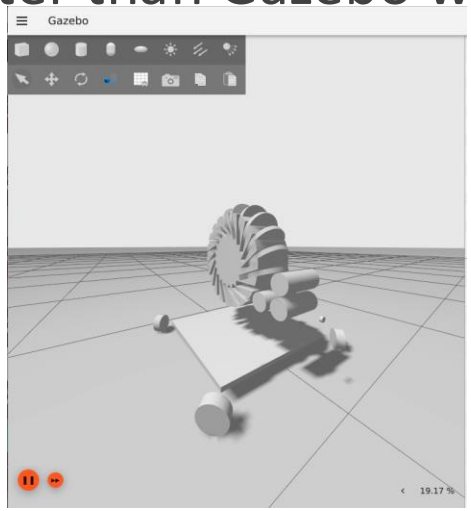
Kind	Gazebo	Isaac Sim	Unity	Unreal Engine
Price	$0	$0 (Paid support available)	$4,600 / year (1sheet)	$ 1,940 / year (1sheet)

Free of charge if you can build and develop your own environment

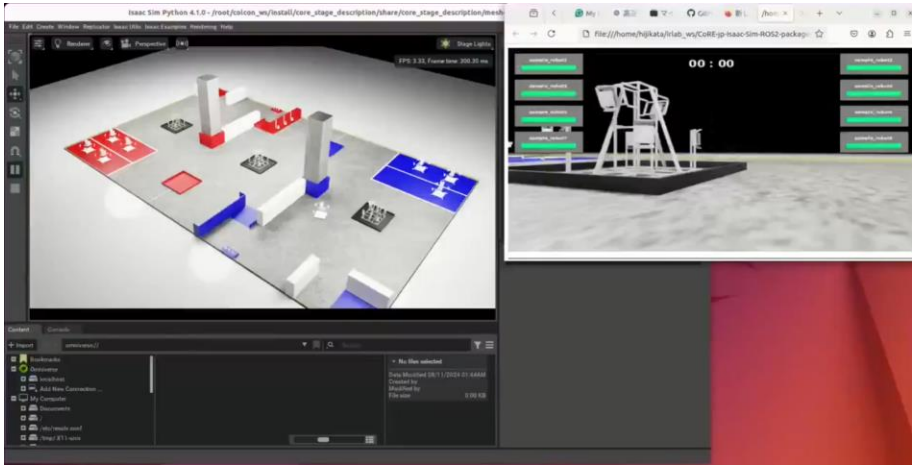
https://github.com/hijimasa/isaac-ros2-control-sample

2. Differents from Gazebo

- Photorealistic simulation
- Faster than Gazebo when the number of objects is large



On Gazebo, even with one robot, it takes about 8 discs to get below 10 fps.



Isaac Sim can run at 10 fps with 4 robots and 80 disks in total.

The graphics and physics engine are high-performance.

https://github.com/hijimasa/isaac-ros2-control-sample

3. ROS User Perspective Issues

- Isaac Sim manages robot models with USD files
  - Problem of dual management of URDF and USD
- Append sensor information to converted USD
  - Work occurs each time the robot model is regenerated
- Use the controller provided by Nvidia instead of ros2_control
  - Existing ros2_control assets cannot be diverted

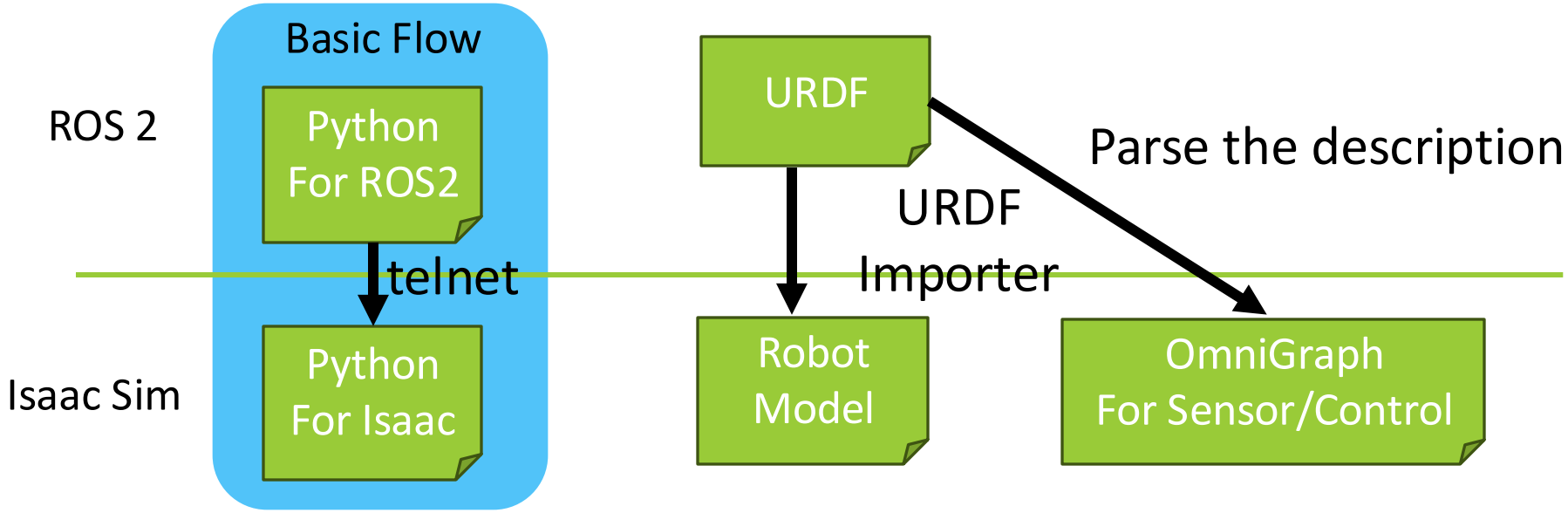
The default Isaac Sim is complexity and tedious to work with.

A tool was developed to simplify the work.

https://github.com/hijimasa/isaac-ros2-control-sample

4. Implementation Direction

- Isaac Sim can perform almost functions from the Python API
- Use Python REPL extension to run additional Python in an existing environment



Automate generation of robot models using Python API

https://github.com/hijimasa/isaac-ros2-control-sample

5. Typical URDF Description

- Sensor information is written in isaac tags instead of gazebo tags

```
<isaac>
 <sensor name="camera_link" type="camera">
  <topic>image_raw</topic>
  …
 </sensor>

</isaac>
```

- Utilize topic_based_ros2_control package

```
<ros2_control name="${name}" type="system">
 <hardware>
  <plugin>topic_based_ros2_control/TopicBasedSystem</plugin>
 </hardware>
</ros2_control>
```

- There is a description of the friction coefficient using material and stiffness parameters of the joints.

Automatic conversion of basic functions from URDF descriptions

https://github.com/hijimasa/isaac-ros2-control-sample

6. Current Issues

- The more OmniGraph makes the process heavier.
  - FPS drops when launching OmniGraphs for control rather than simply the number of robots.

- With FastDDS, topics can be missing.

- Python REPL has been deprecated

> **omni.isaac.repl**
>
> > Changed
> >
> > > Deprecate extension in favor of the vscode extension

Reference URL:https://docs.omniverse.nvidia.com/isaacsim/latest/release_notes.html

7. Conclusion

- Isaac Sim Makes Large-Scale Photorealistic Simulations Easy
- Existing Isaac Sim is a bit complexity for ROS users

- Import URDF into existing environment using REPL
- Automatic generation of sensors and ros2_control based on URDF descriptions

- Easy-to-use development environment for ROS users

Why don't you also simulate with Isaac Sim?

https://github.com/hijimasa/isaac-ros2-control-sample

PC configuration for Verification

Item	Details
CPU	AMD Rayzen9 5950x
Memory	64GB
GPU	GeForce RTX 3090

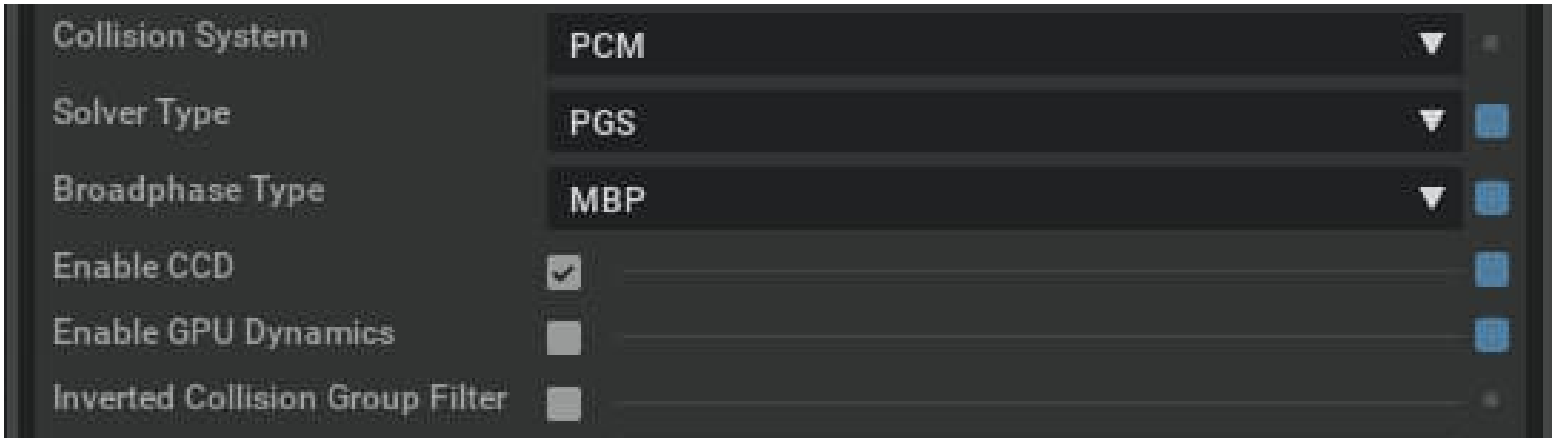
https://github.com/hijimasa/isaac-ros2-control-sample

Simulation Settings

Below are the usual settings for physics calculations

Basically, the CPU tends to be faster for the physics calculations themselves.

GPU is supposed to be used for elastic and fluid simulations.

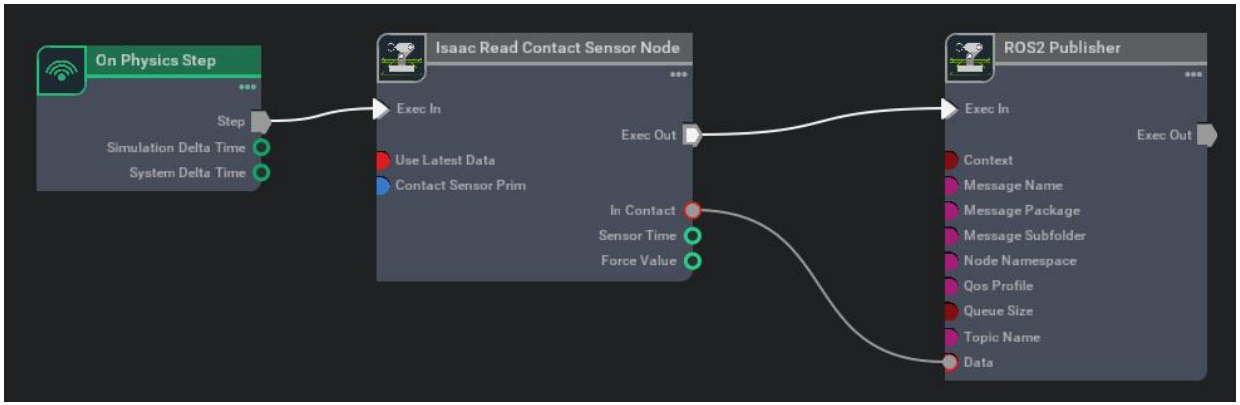


Improved Control Cycle

In the normal tutorial, events are generated
by the “On Playback Tick” node.

The above node is basically an event for each drawing,
so the control cycle is slowed down by the drawing speed.

- The “On Physics Step” node generates an event for each physics step, which improves the control cycle.



https://github.com/hijimasa/isaac-ros2-control-sample

Apply to Your Own Workspace

- Isaac Sim and ROS 2 integration part is separated to isaac_ros2_utils package.
  https://github.com/hijimasa/isaac_ros2_utils
- Basically, the above packages can be installed in the workspace of a real or Docker environment with Isaac Sim and ROS 2 installed.

How to Find the API You Need

- When trying to introduce new functionality, you need to know the corresponding Python API.
- Basically, you will refer to the documentation on the web, but there are some APIs that you may not know how to handle only from the documentation.
- For Python APIs, if you are in a Docker environment, all Python files should be in the /isaac-sim/ directory, so you can search for APIs with the grep command.