HVAC2023技術講習会
配布用資料

2023.9.24

豊橋技術科学大学 次世代半導体・センサ科学研究所
垣内 洋平
Information Robotic System Laboratory
(IRSL, 垣内研究室)

ハンズオンの準備

- Dockerをインストールしてください
  - Linux, windows, (MacOS)いずれでも可
- 以下はハンズオンに関する説明(随時更新します)
- このスライドの末に追加します(配布用に追加)

今日の内容

- RobotAssemblerを使ってみる
- もう少し簡単なインタラクティブモデル構成
- ChoreonoidのPythonバインディングでロボットプログラミング

前提条件

- 大学で行っているロボット構成演習がもとになっています
  - https://irsl-tut.github.io/CPS-lecture/index.ja.html

- Choreonoidはrelease-2.0に少しパッチが当たっています
  - 順次choreonoidに取り込まれるようにがんばります

前提条件

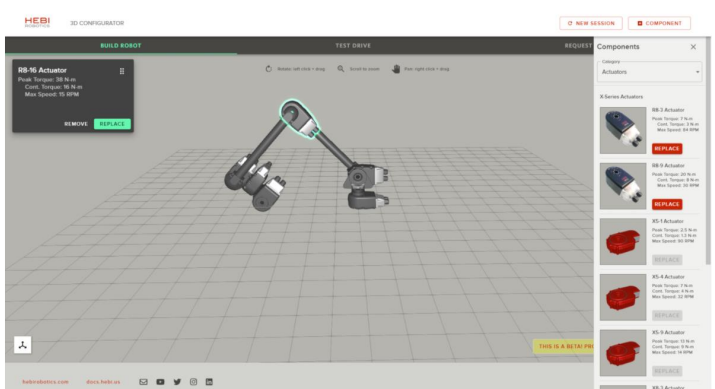
- まとめたDockerがあります irslrepo/irsl_system:noetic
  - https://hub.docker.com/r/irslrepo/irsl_system
  - https://github.com/IRSL-tut/irsl_docker_irsl_system
  - 以下構成レポジトリ
  - choreonoid (IRSL, stableブランチ)
    - https://github.com/IRSL-tut/choreonoid/tree/stable
    - (forked from https://github.com/choreonoid/choreonoid)
  - choreonoid_ros (IRSL, stableブランチ)
    - https://github.com/IRSL-tut/choreonoid_ros/tree/stable
    - (forked from https://github.com/choreonoid/choreonoid_ros)
  - irsl_choreonoid
    - https://github.com/IRSL-tut/irsl_choreonoid
  - irsl_choreonoid_ros
    - https://github.com/IRSL-tut/irsl_choreonoid_ros
  - robot_assembler_plugin
    - https://github.com/IRSL-tut/robot_assembler_plugin
  - jupyter_plugin
    - https://github.com/IRSL-tut/jupyter_plugin

今日の内容

- RobotAssemblerを使ってみる
- もう少し簡単なインタラクティブモデル構成
- ChoreonoidのPythonバインディングでロボットプログラミング

RobotAssemblerとは / 組立型ロボット構成システム

- ロボット設計は3DCADなどを用いて行う場合が多い
- 自由度は下がるが、あらかじめ決まったモジュールの組み合わせでロボットが構成できることに利点
  - 設計コストが下がる
  - 制御システムの再利用
  - 様々な構成を容易に試せる
- 構成したロボットをシミュレーション内において動作検証する（モデルファイルの書き出し）
- 部品を組み立てて実際にロボットを構築することができる
  - シミュレーションでのソフトウェアと同じものが動く

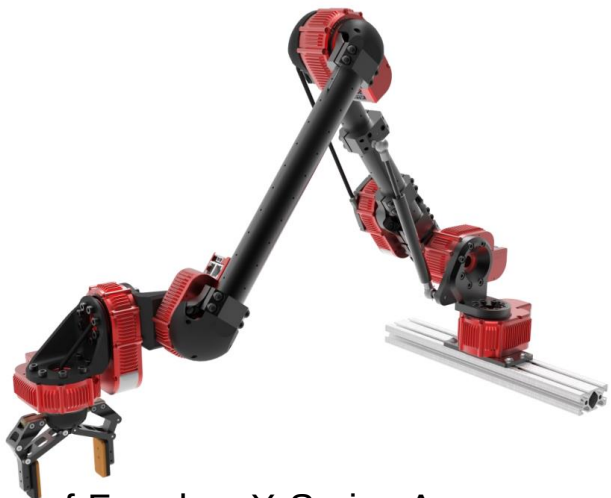


HEBI Configurator
https://www.hebirobotics.com/robot-builder

組み立てロボットモジュール

- HEBI Robotics社
- DynamixeI / Darwin(Robotis社)
- KXR (近藤科学(株))
- LEGO

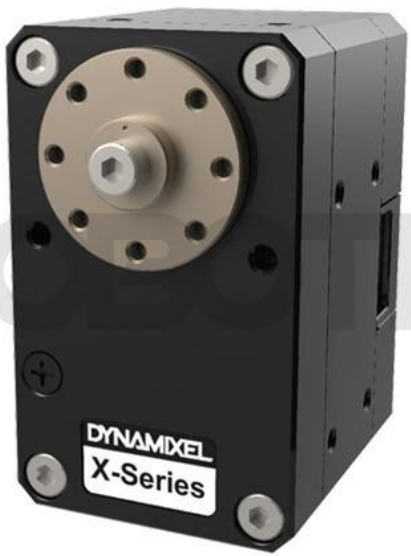
https://docs.hebi.us/resources/datasheets/X-SeriesDatasheet.pdf



6 Degree-of-Freedom X-Series Arm
https://docs.hebi.us/resources/kits/datasheets/x-series/A-2085-06G_Datasheet.pdf

組み立てロボットモジュール

- HEBI Robotics社
- Dynamixel (Robotis社)
- KXR (近藤科学(株))
- LEGO

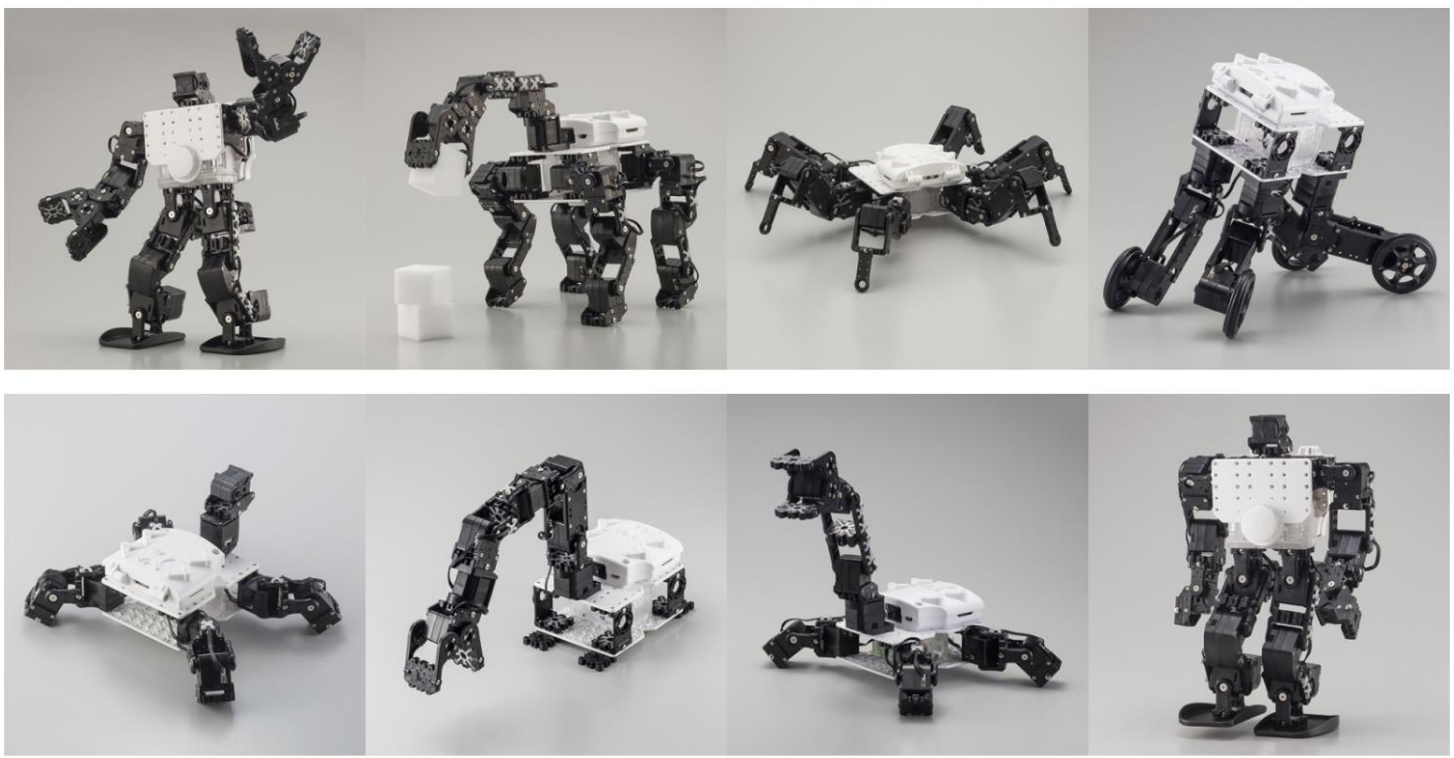


https://e-shop.robotis.co.jp/product.php?id=48

https://emanual.robotis.com/docs/en/platform/op3/introduction/

組み立てロボットモジュール

- HEBI Robotics社
- DynamixeI (Robotis社)
- KXR (近藤科学(株))
- LEGO

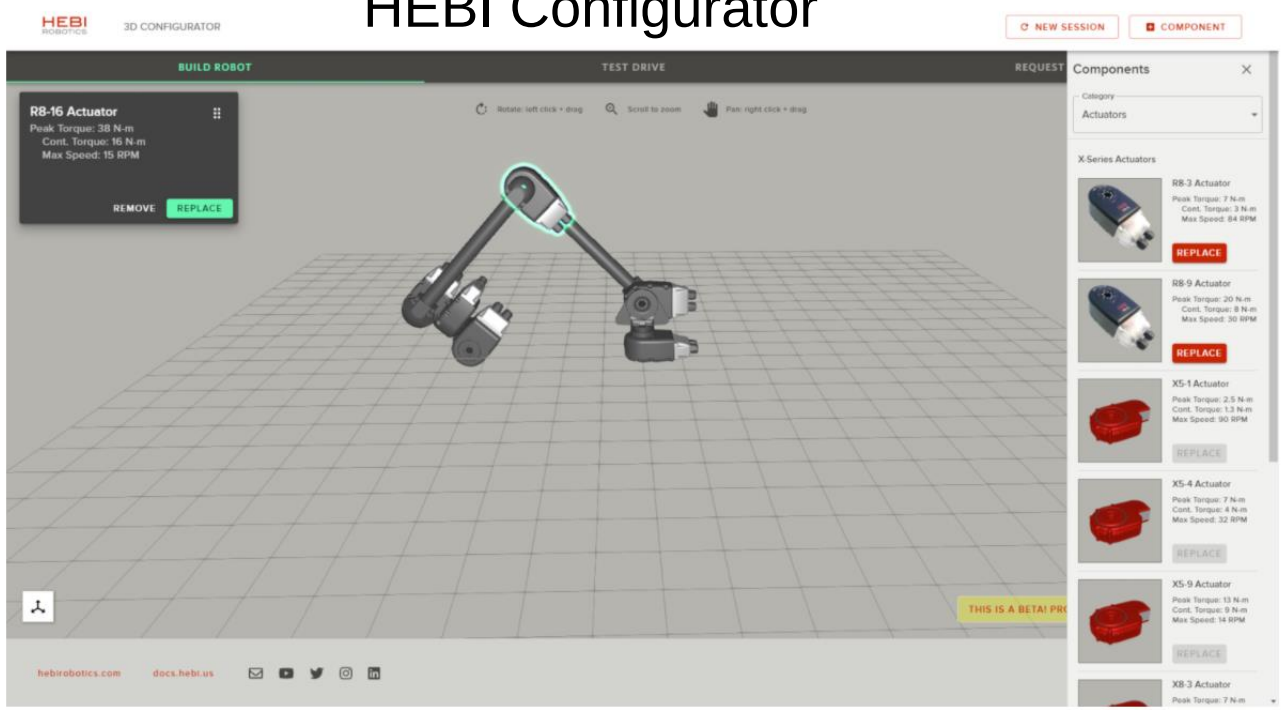


https://kondo-robot.com/product-category/robot/kxrseries

組立型ロボット構成システム

- モジュールを選択して構成する
- 実際に可能な構成に組み合わせに制限

HEBI Configurator

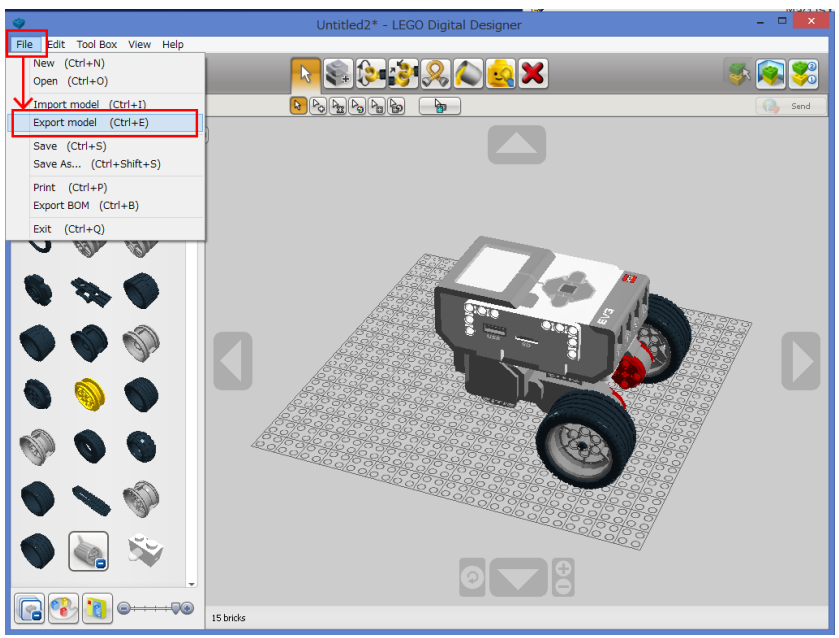


https://www.hebirobotics.com/robot-builder

組立型ロボット構成システム

- モジュールを選択して構成する
- 実際に可能な構成に組み合わせに制限

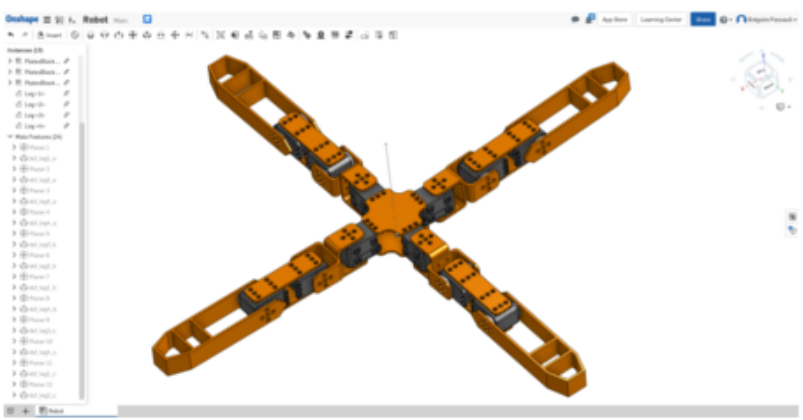
LEGO Degital Designer (https://www.lego.com/en-us/ldd)



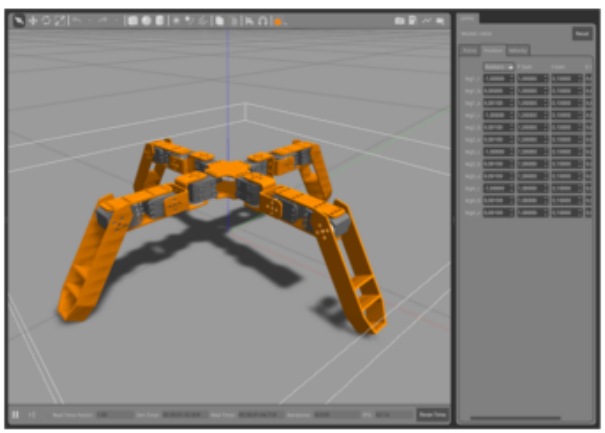
https://afrel.co.jp/archives/25916

ロボットの構成のシミュレーション検証及び実機の構成

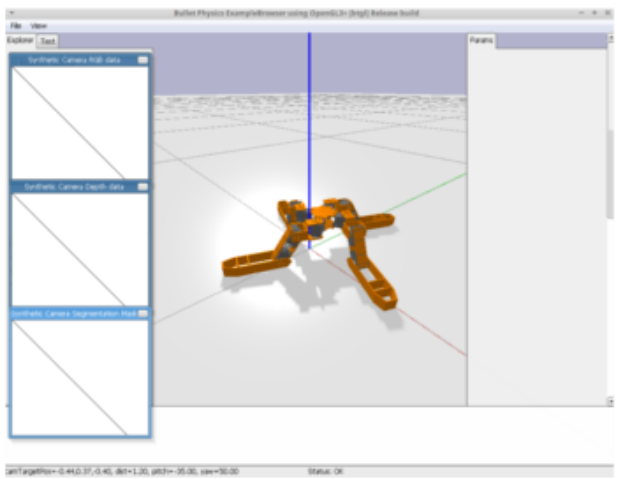
- Onshape (https://www.onshape.com/)
  - オンライン3DCAD、CADデータを公開、共有できる
  - DynamixelはここでファイルをOnshapeで公開している
- Onshape-to-robot (https://github.com/Rhoban/onshape-to-robot)
  - Onshapeで作ったCADファイルをロボットモデルファイルに変換
  - http://wiki.ros.org/sw_urdf_exporter 同じような試み



OnShape design



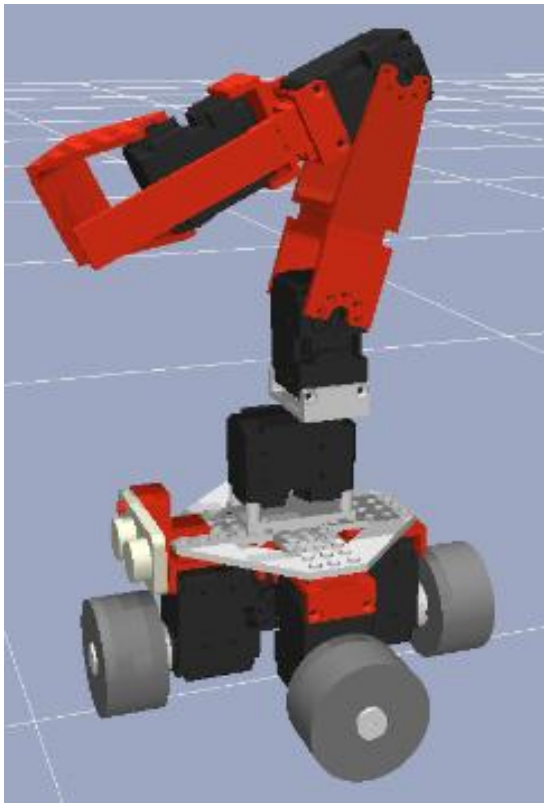
Gazebo (SDF)

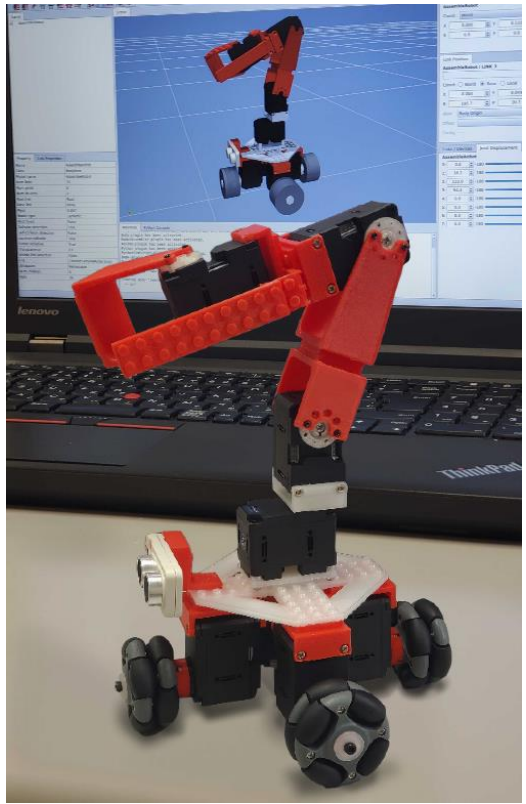


PyBullet (URDF)

ロボットの教育とロボット自動構成の研究

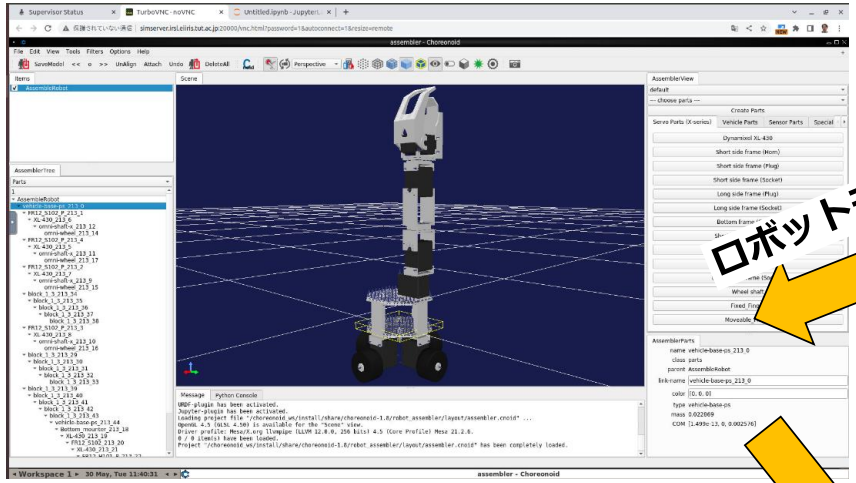
- ロボット構成インターフェースのChoreonoidのプラグインとしての実装
  - https://github.com/IRSL-tut/robot_assembler_plugin
  - Choreonoidのプラグインとしてモデルファイルを作成
  - シミュレーションまでシームレスに行えるようにしたい（作成中）



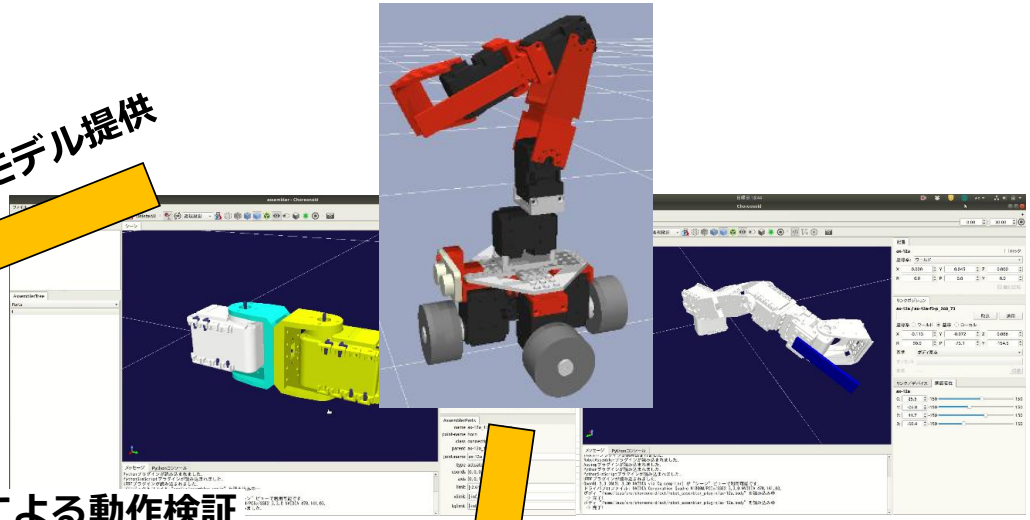


ロボット構成演習（大学1～3年生向け）

クラウドベース/ロボットプログラミング環境

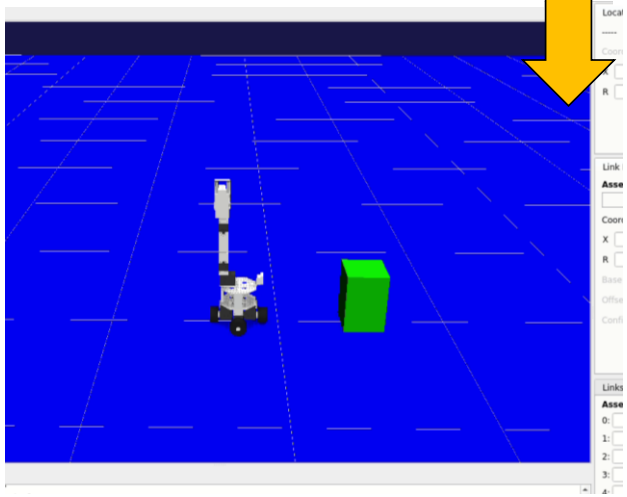


組立ロボットシステム組立構成ソフトウェア

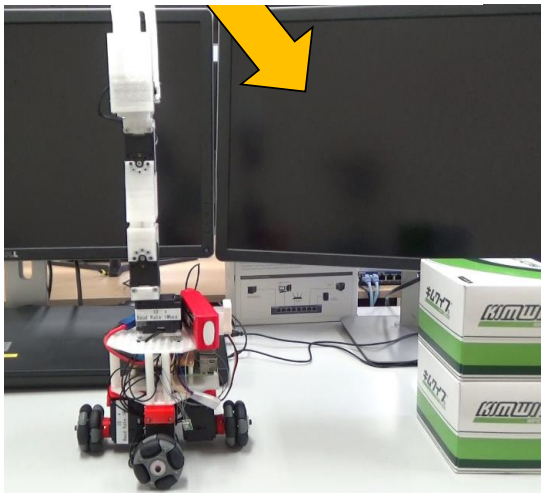


ロボットモデル提供

シミュレーターによる動作検証

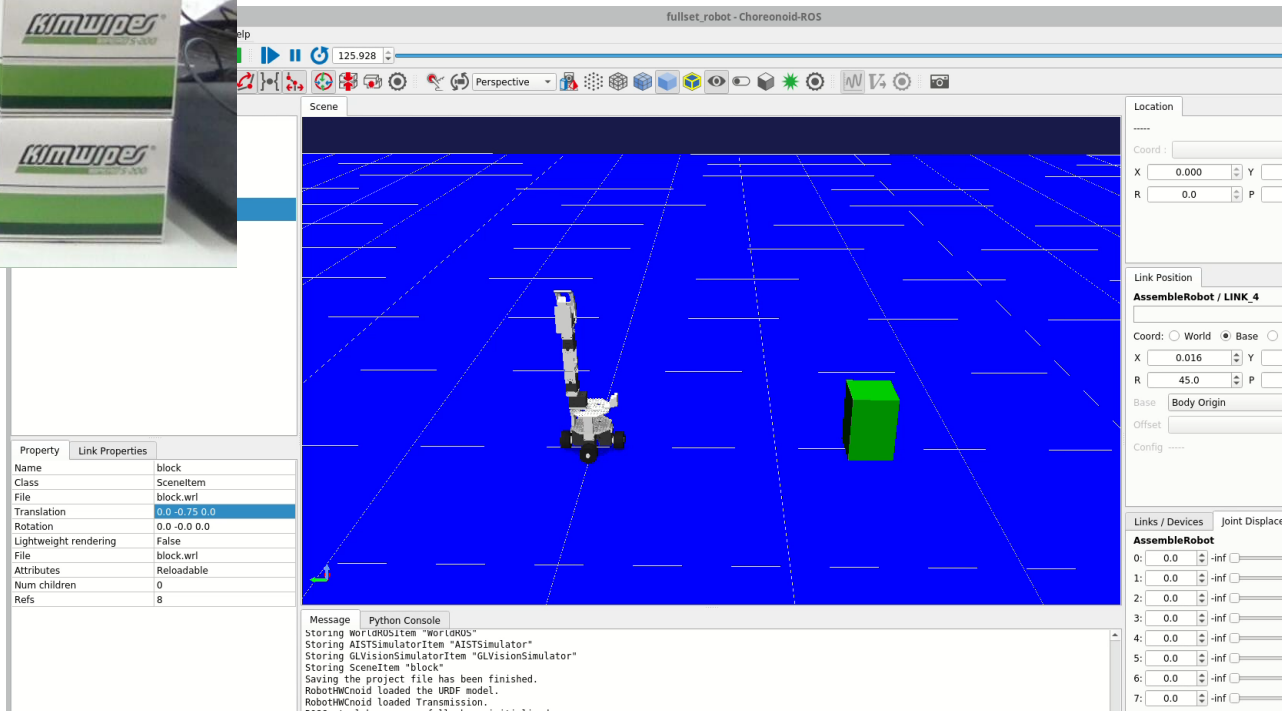


実機ロボットによる動作検証

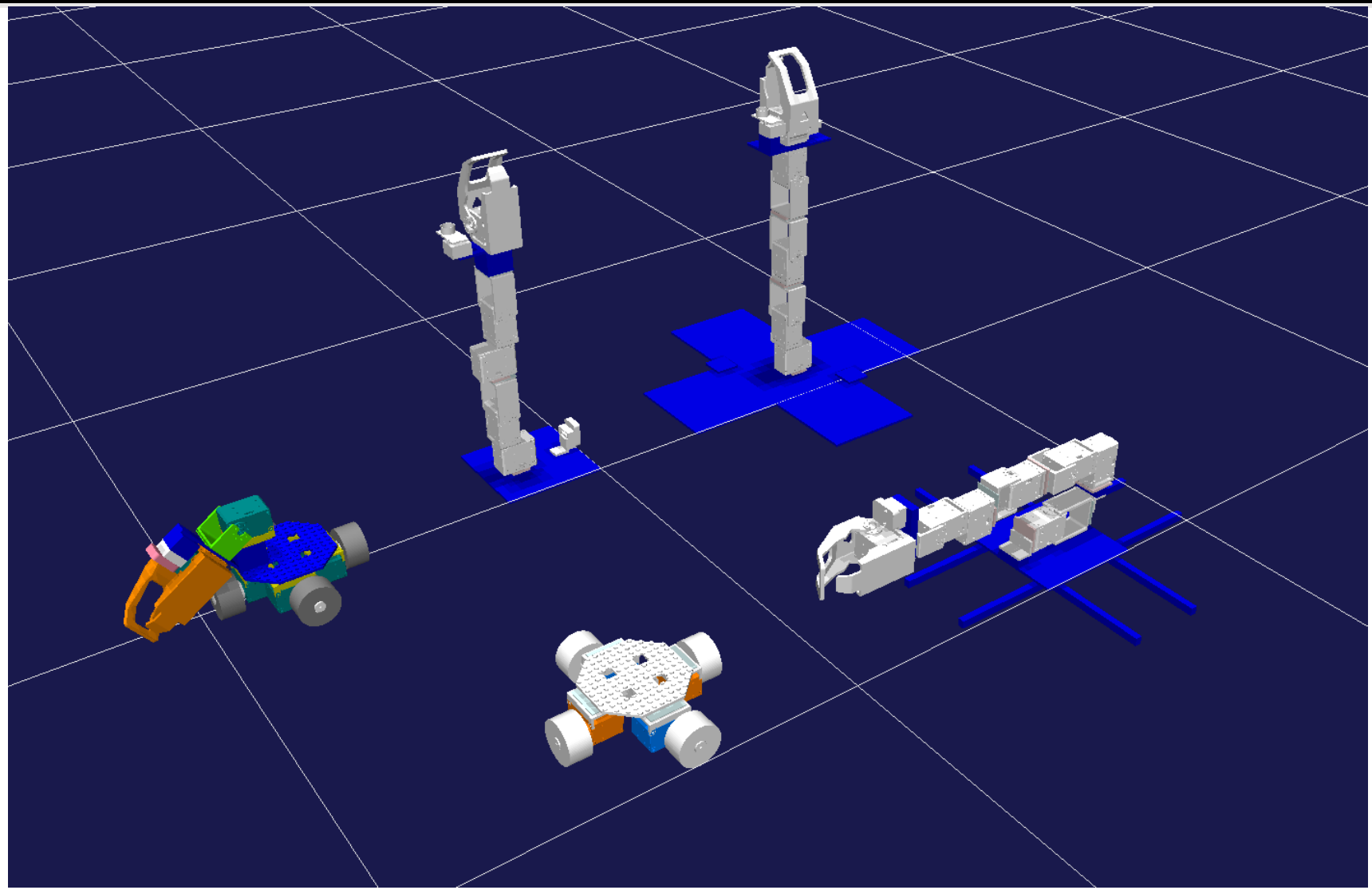


組立ロボットシステム実機ロボットの組立

一つの制御ソフトウェアで実機とシミュレーションの実施



RobotAssembler（演習の成果）



RobotAssemblerの使い方

- 実際の動かし方の資料
- https://github.com/IRSL-tut/CPS-lecture/wiki/RobotAssembler%E3%81%AE%E4%BD%BF%E3%81%84%E6%96%B9

- ロボットの組立ファイル(.roboasm)
- https://drive.google.com/file/d/1OJOhCeeP0Wed1Sx6VHMscAMQHiPjP_rn/view?usp=drive_link
- https://drive.google.com/file/d/1KzwFZDXM2uu2QZOQbZ12QMzwJHCEnW7q/view?usp=drive_link

RobotAssemblerの設定ファイル

- 設定の書き方(ドキュメントほとんどないです)
- config/robot_assembler_parts_settings.yaml
- YAML記法のファイル (yaml-cppを用いて読み込み)
- https://github.com/agent-system/robot_assembler/blob/master/config/robot_assembler_kxr_settings.yaml
- 接続形式の設定とパーツの設定から構成される
- 接続形式設定部
  - 拘束の形態の名前付け
- パーツ設定部
  - 形状、マスパラメーター、接続形式の配置

config/robot_assembler_parts_settings.yaml の説明

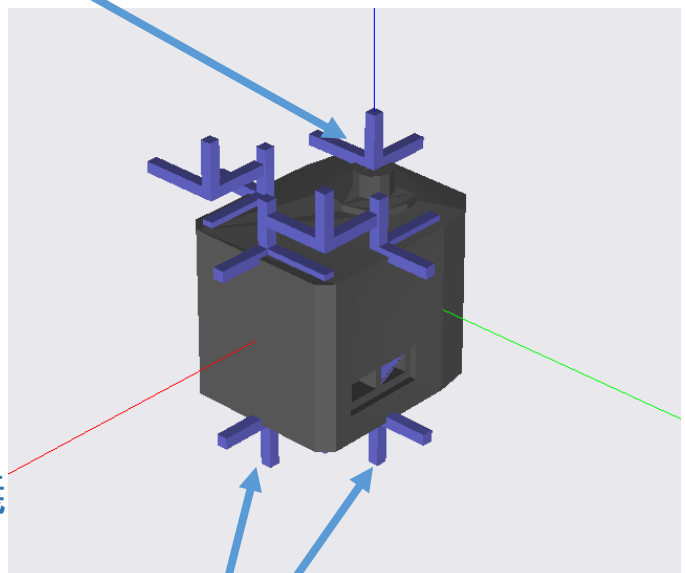
接続形式設定部

```
;; 接続可能位置のtypeのリスト (A) 名前の定義
fixed-point-type-list: [horn12, horn12-hole,

;; 接続可能位置のtypeの組み合わせリスト (B)
fixed-point-type-match-list:
  -
    pair: [horn12, horn12-hole] ;; (A)で定義した名前の組み合わせを書く
    allowed-configuration: [fixed, ;; 回転可能などの定義のリスト、(C)で定義

;; 接続時の回転の定義リスト (C)
pre-defined-configuration-list:
  -
    type: invert ;; (B)のリストのallowed-configurationで用いた名前
    description: 'rotate 180[deg] around z-axis'
    rotation: [0, 0, 1, 180] ;; どの方向に回転するか
```

horns(fixed-point-typeが設定されている)

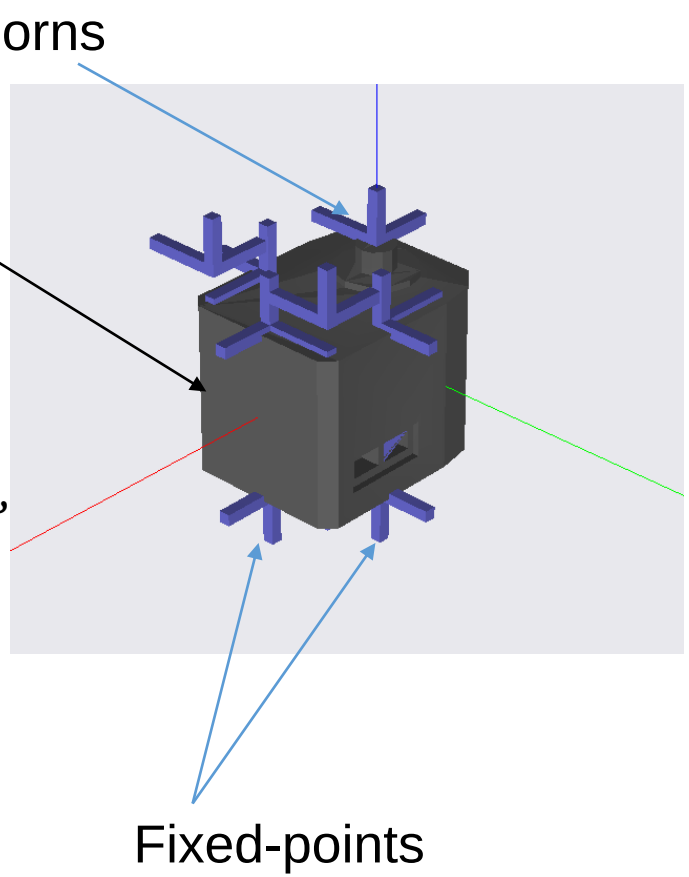


Fixed-point
(fixed-point-typeが設定されている)

config/robot_assembler_parts_settings.yaml の説明

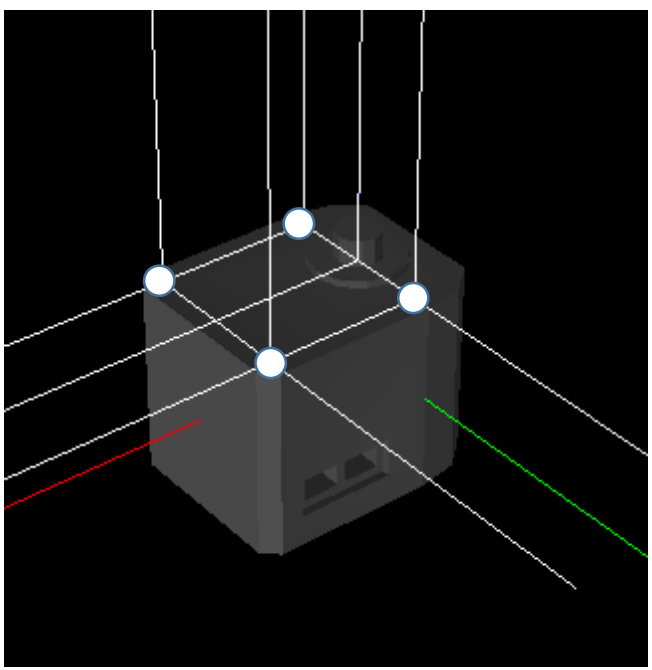
パーツ設定部の説明

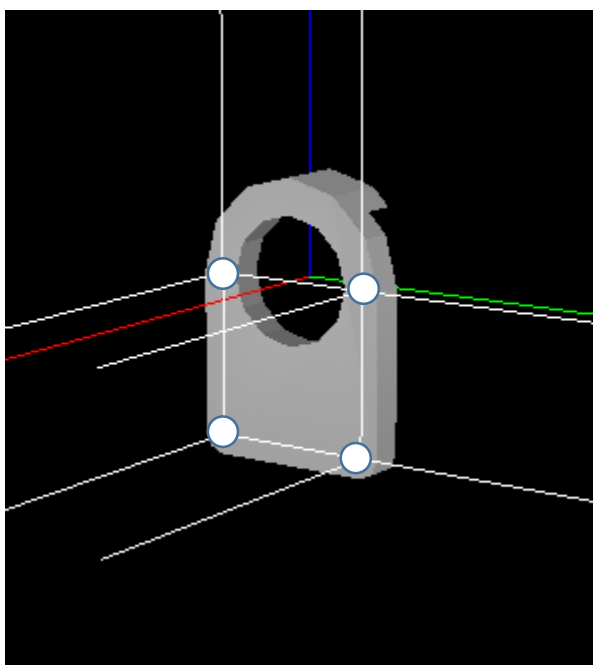
```
actuators:
  -
    type: xm430 ;; 形名
    geometry: ;; 形状定義
      -
        type: mesh ;; 今のところmeshのみ（assimpで読める種類）
        url: "meshes/xm430_dynamixel.dae"
        scale: 1000
    mass-param: ;; マスパラメータ（ここだけ m, kg系）
      mass: 0.1
      center-of-mass: [0.0, -0.012, 0.0015]
      inertia-tensor: [3.202708e-05, 0.0, 0.0,  0.0, 2.077708e-05, 0.0,
    horns: ;; ホーン(稼働部の定義)のリスト
      -
        name: horn
        types: [horn12] ;; 接続可能位置の形名，(A)のリストから選ぶ
        translation: [0, 0, 19]
    fixed-points: ;; 接続可能位置のリスト
      -
        name: left-side-tap
        types: [ bolt12_0-tap ]  ;; 接続可能位置の形名，(A)のリストから選ぶ
        translation: [ 14.25, -16, 0]
        rotation: [0, 1, 0, 90]
;; アクチュエータのない部品の定義（hornsが無いだけで actuatorsの定義と同じもの）
parts:
```

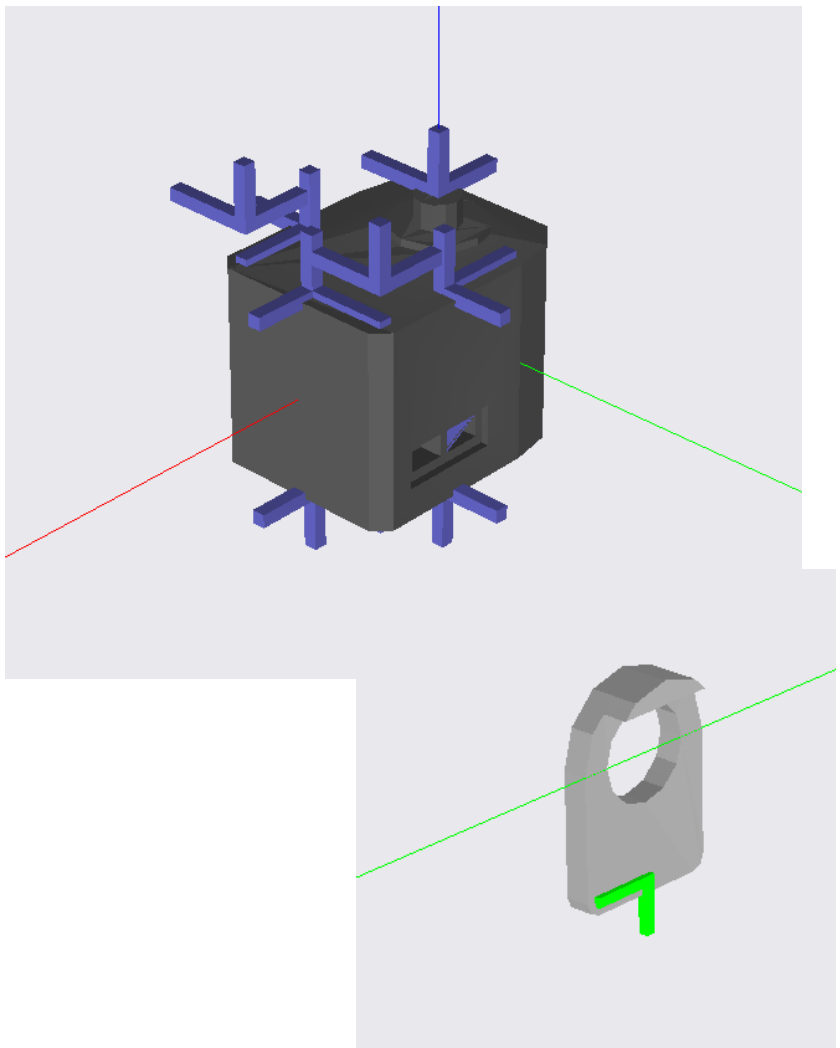


RobotAssemblerの接続の設定

- 本来のネジ穴
- 接続形式への変換(現在手動)

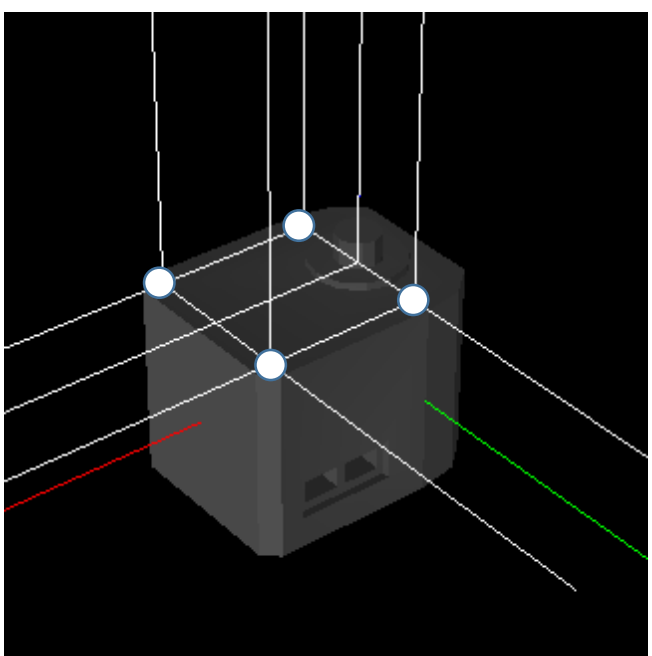


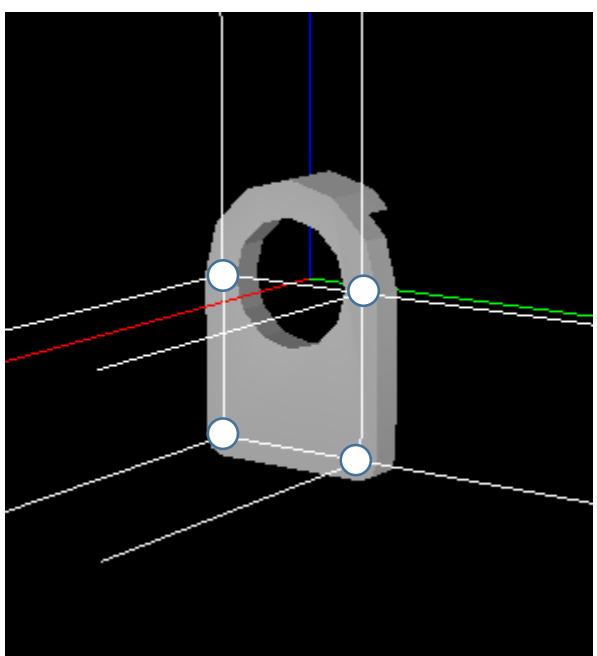


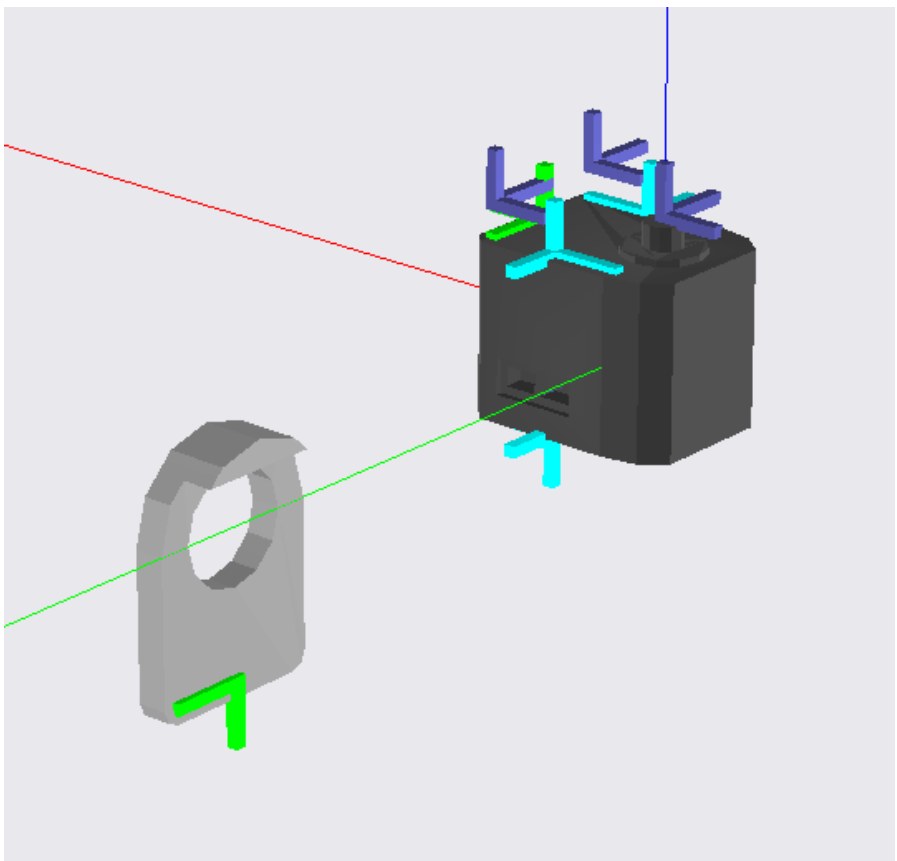


RobotAssemblerの接続の設定

- 本来のネジ穴
- 接続形式への変換(現在手動)
- 接続可能位置の対応

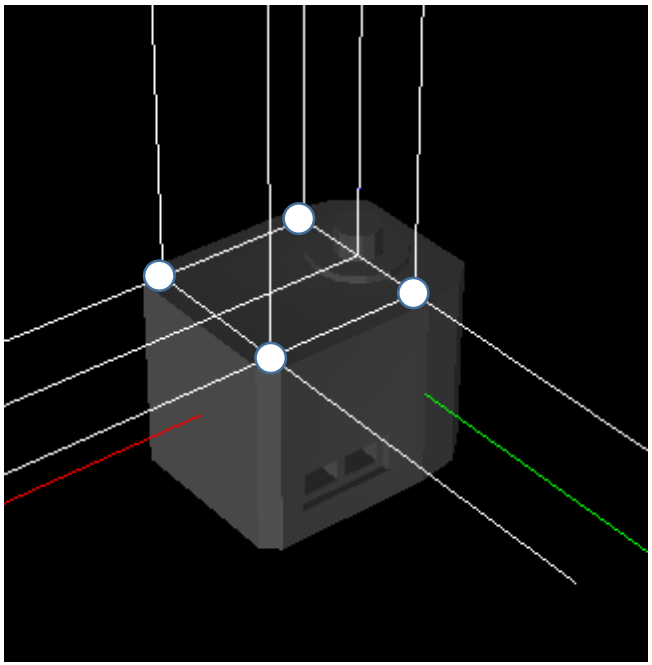


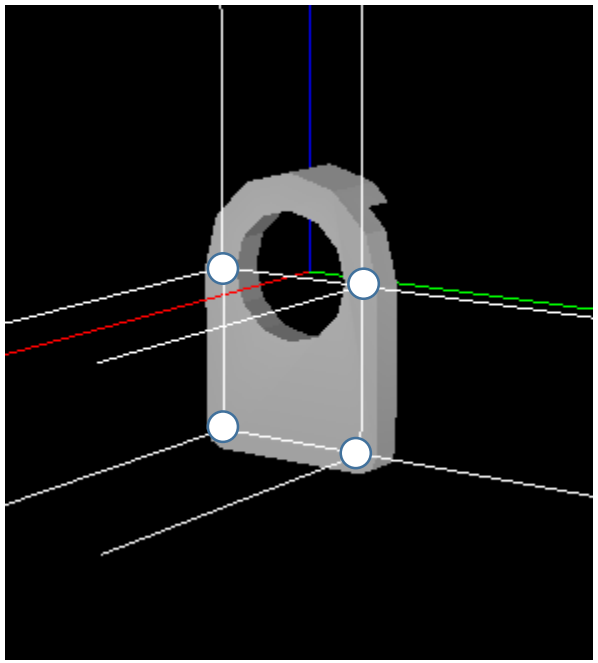


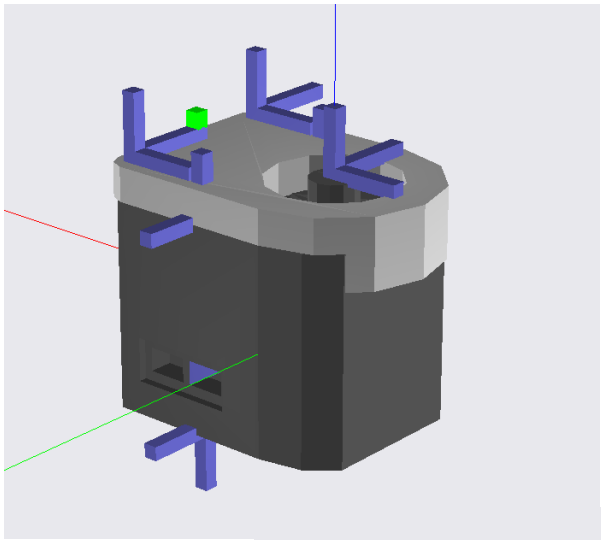


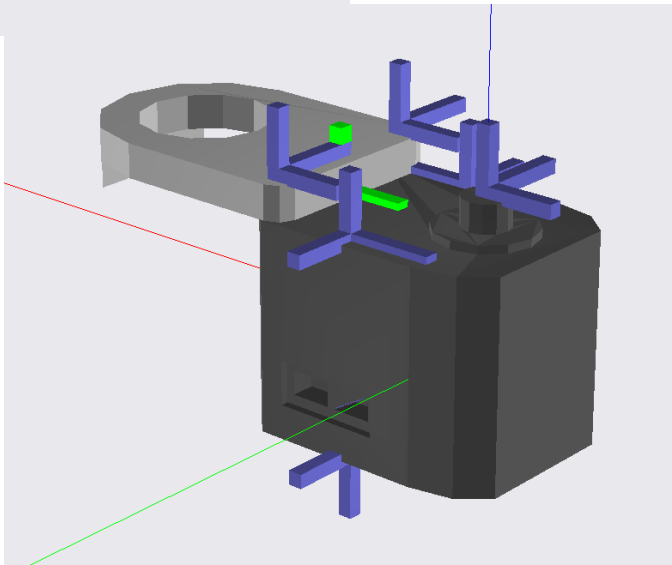
RobotAssemblerの接続の設定

- 本来のネジ穴
- 接続形式への変換(現在手動)
- 接続可能位置の対応
- 接続の方向









RobotAssemblerの設定ファイル

- 講義で用いた設定
  - https://github.com/IRSL-tut/robot_assembler_config_IRSL

- Kxrのサンプル
  - https://github.com/IRSL-tut/robot_assembler_plugin/tree/main/config/kxr

- 注意

robot_assembler_pluginはwindowsでのbuildは現状できていません

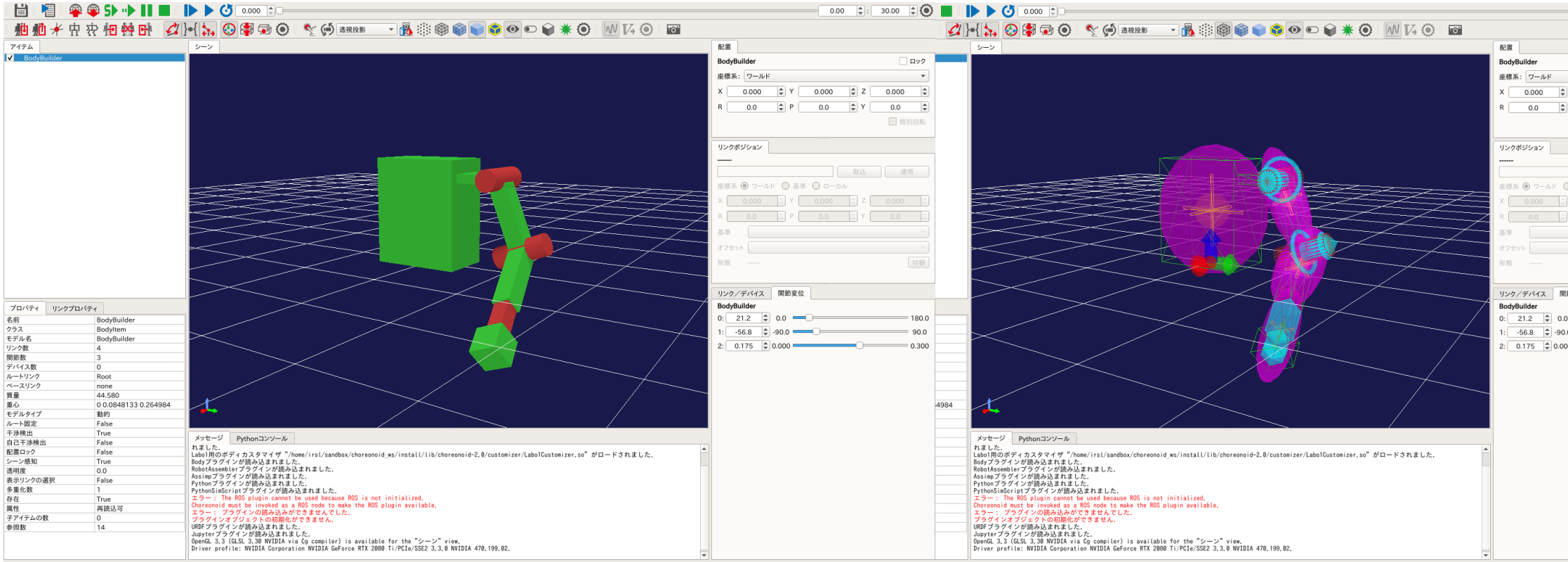
（対応はできると思います）

今日の内容

- RobotAssemblerを使ってみる
- もう少し簡単なインタラクティブモデル構成
- ChoreonoidのPythonバインディングでロボットプログラミング

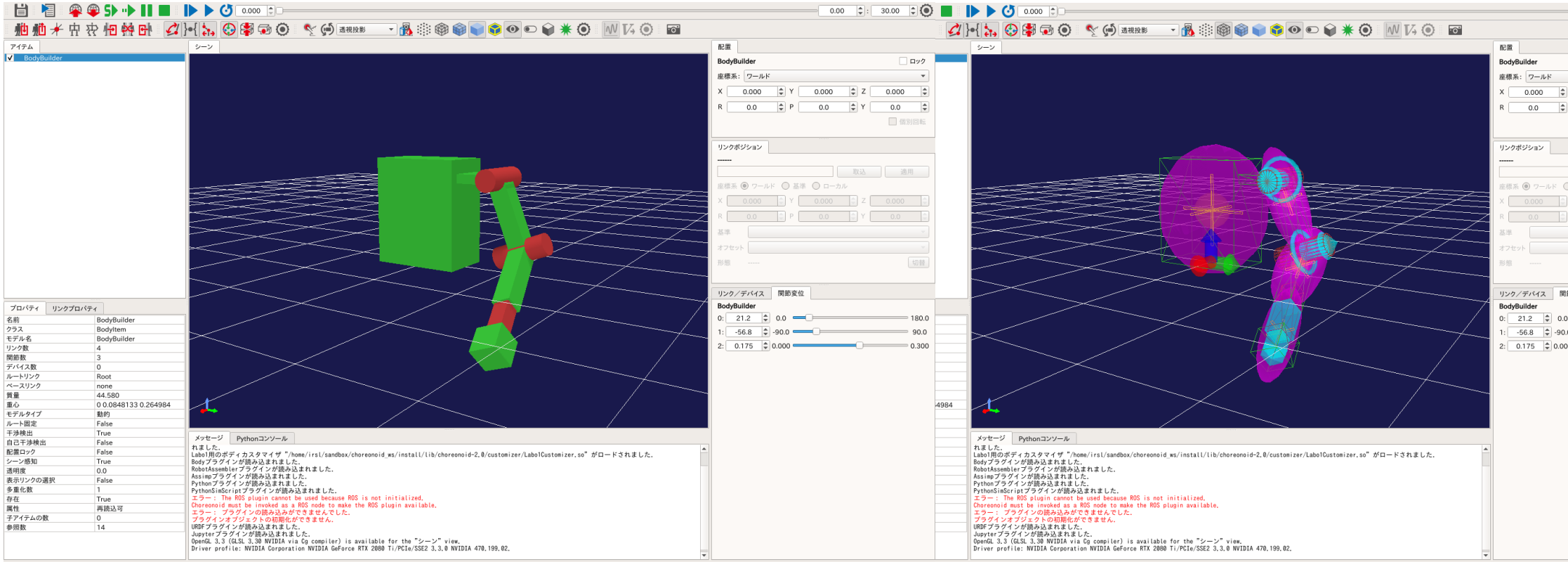
インタラクティブモデル構成

- インタラクティブなpythonプログラミングで簡単なロボットモデルを作る方法を解説します
- モデルの整合性の確認方法なども説明します。



インタラクティブモデル構成

- モジュール（ドキュメントがまだありません）
  - https://github.com/IRSL-tut/irsl_choreonoid/blob/main/python/irsl_choreonoid/RobotBuilder.py



Jupyterの使い方

- ノートブック(ipynb)の使い方
  - Restart kernel
  - Shutdown kernel

- モードの違い
  - エディットモード
  - コマンドモード

- タイプの違い
  - Codeタイプ
  - Markdownタイプ

- 資料
  - https://github.com/IRSL-tut/CPS-lecture/wiki/Jupyter%E3%81%AE%E4%BD%BF%E3%81%84%E6%96%B9

インタラクティブモデル構成

- 以下のノートブック・モデルファイルをダウンロードしてjupyterへアップロード
- ノートブック

https://drive.google.com/file/d/1CapY2RSewvp265CqIC4UhulhYYXjbIsz/view?usp=drive_link

https://drive.google.com/file/d/1mghKqtnyZwaX9yw7xcBIN4iRiLyMrdoc/view?usp=drive_link

- モデルファイル

https://drive.google.com/file/d/1XsHcL5IcRpUcZ5E72dCd-JUIIi9B0DnX/view?usp=drive_link

今日の内容

- RobotAssemblerを使ってみる
- もう少し簡単なインタラクティブモデル構成
- ChoreonoidのPythonバインディングでロボットプログラミング

ChoreonoidのPythonバインディングでロボットプログラミング

- ロボットモデルを使ったインタラクティブロボットプログラミング
- 以下のノートブックをダウンロードしてjupyterへアップロード
- 3次元座標系のプログラミング
  - ノートブック
    - https://github.com/IRSL-tut/CPS-lecture/blob/main/notebooks/cps_lecture_coords00.ipynb
    - https://github.com/IRSL-tut/CPS-lecture/blob/main/notebooks/cps_lecture_coords01.ipynb
    - https://github.com/IRSL-tut/CPS-lecture/blob/main/notebooks/cps_lecture_coords02.ipynb
- ロボットモデルのプログラミング
  - https://github.com/IRSL-tut/CPS-lecture/blob/main/notebooks/cps_lecture_robot_model.ipynb

- 以下の参考資料のP.8以降くらいの話（初歩の話が多いです）
  - https://github.com/IRSL-tut/CPS-lecture/blob/main/documents/CPS-lecture%E8%AA%AC%E6%98%8E.pdf

Choreonoid Jupyter-plugin

- https://github.com/IRSL-tut/jupyter_plugin
- Jupyter-xeusライブラリを使っている
  - https://github.com/jupyter-xeus/xeus
  - Buildが少し面倒(たぶんもっと簡単になる。。。)
  - WindowsでnativeでbuildはNしそう

ドキュメント

- Python-binding ドキュメント
  - https://irsl-tut.github.io/irsl_documents/ja/

- Jupyterの使い方
  - tab補完
  - Ctrl-I / documentを見る
  - Console (jupyterのwebアプリではなくコンソールで使う)
    - Jupyter console -kernel=choreonoid

準備(配布用追加)

実行環境が必要な場合はDockerをインストールしておいてください

Linux(Ubuntuでしかテストしていません)

- Docker-engine
- https://docs.docker.com/engine/install/ubuntu/#install-using-the-convenience-script
- Nvidia-docker(nvidiaのグラボを使う場合 / なくても構いません)
- https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html

Windows 以下どちらでも大丈夫

- Doker-desktop(windows-native)
  - https://docs.docker.jp/docker-for-windows/install.html
- WSL
  - linuxの方法と同じ

準備(配布用追加)

- 実行環境が必要な場合はDockerをインストールしておいてください

Dockerインストール後に以下の2つをしておいてください。

- (数GBのダウンロードが行われます)

docker pull irslrepo/browser_vnc:20.04

docker pull irslrepo/irsl_system:noetic

準備(配布用追加)

Dockerインストール後に、
以下の「ブラウザで使う」を行う
https://github.com/IRSL-tut/irsl_docker_irsl_system/wiki/IRSL%E3%81%AE%E3%82%BD%E3%83%95%E3%83%88%E3%82%A6%E3%82%A7%E3%82%A2%E3%82%92%E4%BD%BF%E3%81%86

<ファイルの準備>
docker-compose-xxxx.yamlファイルのダウンロード

<実行> ターミナルで実行してください
docker compose -f <downloaded-compose>.yaml up

<実行後に>
Jupyterへのアクセス http://localhost:8888

画面表示へのアクセス http://localhost:9999/code.html

メモ(配布用追加)

ローカルマシンでの実行(Linux, Windows-wsl)

https://github.com/IRSL-tut/irsl_docker_irsl_system/wiki/IRSL%E3%81%AE%E3%82%BD%E3%83%95%E3%83%88%E3%82%A6%E3%82%A7%E3%82%A2%E3%82%92%E4%BD%BF%E3%81%86

git clone https://github.com/IRSL-tut/irsl_docker_irsl_system.git

cd irsl_docker_irsl_system

./run.sh -hub choreonoid

./run.sh -hub assembler

./run.sh -hub jupyter

./run.sh -hub choreonoid-console

メモ(配布用追加)

- この資料を作ったときに試したDockerバージョン

試しているDockerバージョン(Ubuntu, 多分最新です)

$ docker --version

Docker version 24.0.2, build cb74dfc

composeのバージョン (Docker Inc., v2.18.1)

$ docker --help

(結果は抜粋です)

Management Commands:

  compose*   Docker Compose (Docker Inc., v2.18.1)