Choreonoid入門

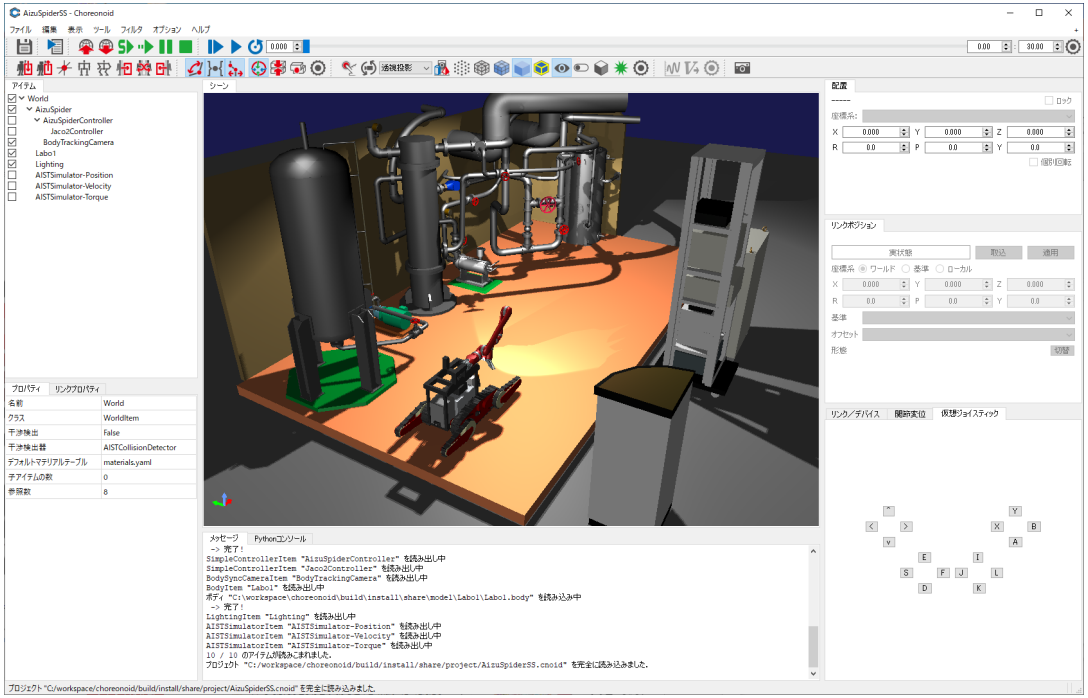
宮本 信彦
国立研究開発法人産業技術総合研究所
インダストリアルCPS研究センター
ソフトウェアプラットフォーム研究チーム

資料

- 配布資料の「 WEBpage 」のHTMLファイルを開く
  - Choreonoid入門 _ OpenRTM-aist.html
- もしくは以下のリンク
  - https://openrtm.org/openrtm/ja/node/7150

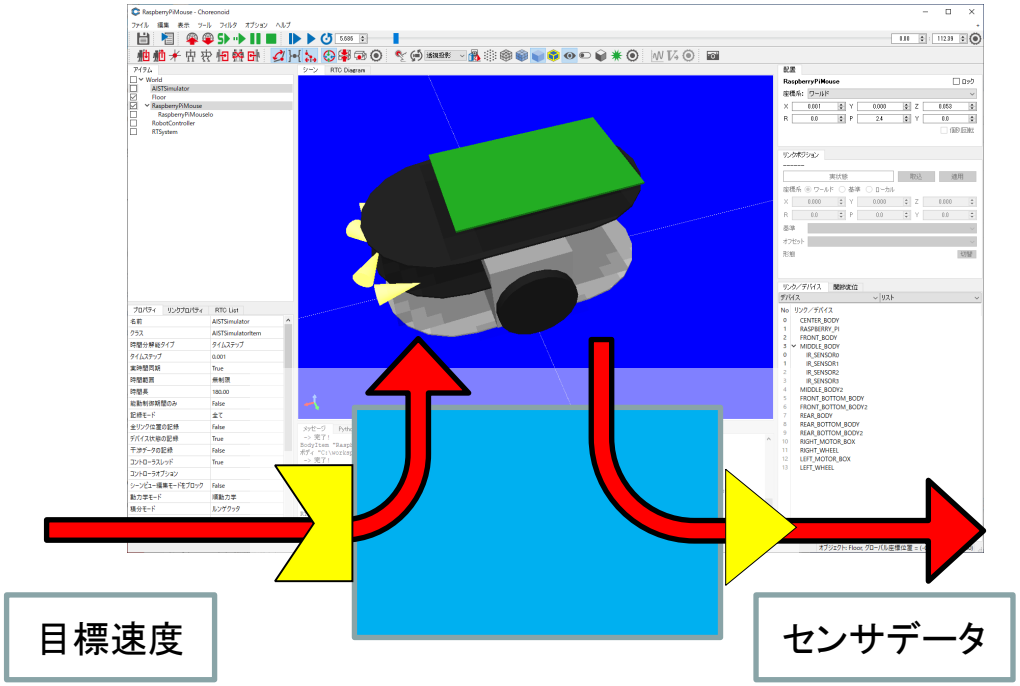
Choreonoid

- Choreonoidはオープンソースのロボット用シミュレーションソフトウェア
  - プラグインによる高い拡張性
  - 3DCGによるロボットモデルのアニメーション表示
  - 動力学シミュレーション
  - センサのシミュレーション(カメラ、レーザーレンジセンサ、力センサ、ジャイロセンサ、・・・)
  - ロボットの動作生成
  - ・・・・



Choreonoid OpenRTMプラグイン

- ChoreonoidとOpenRTM-aistを連携して、シミュレータ上のオブジェクトの入出力(制御指令やセンサ値の取得)をするRTCを開発可能にする拡張プラグイン



課題：Choreonoid上のRaspberry Piマウスを RobotControllerコンポーネントで操作するための入出力RTCの作成

シミュレーション環境構築

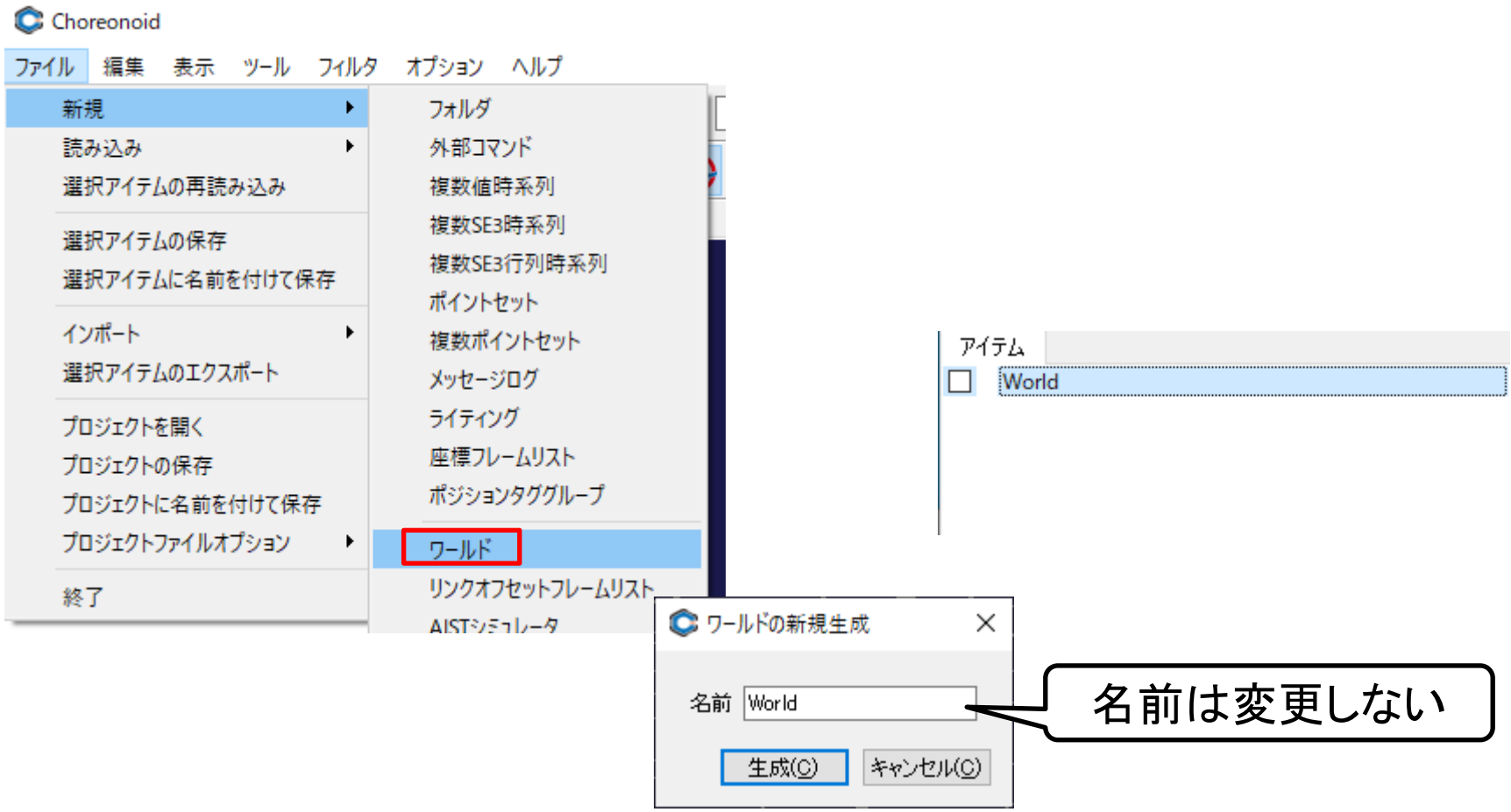
- Choreonoid上でシミュレーションの実行に必要なアイテムを追加することで、環境を構築する。

- 配布資料のchoreonoidフォルダの**choreonoid.bat**を実行する

- 今回は以下のアイテムを追加する。
  - ワールドアイテム
  - シミュレータアイテム(AISTSimulator)
  - ボディアイテム(地面・Floor)
  - ボディアイテム(Raspberry Piマウス)
  - RTSystemアイテム
  - RTCアイテム(RobotController)
  - pyRTCアイテム(RaspberryPiMouseIo)

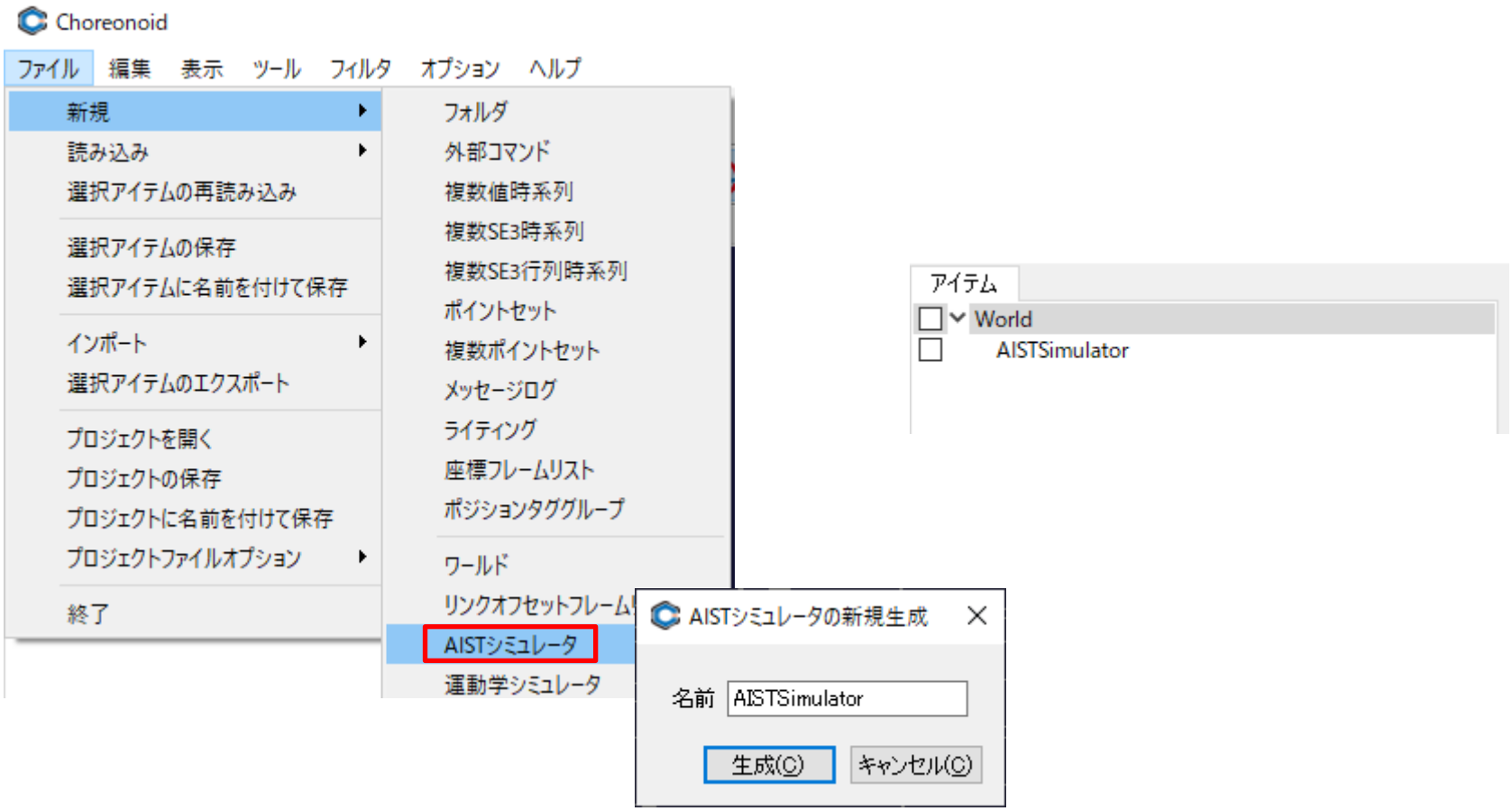
ワールドアイテム追加

- ワールド：仮想世界を表すアイテム
  - ファイル → 新規 → ワールド



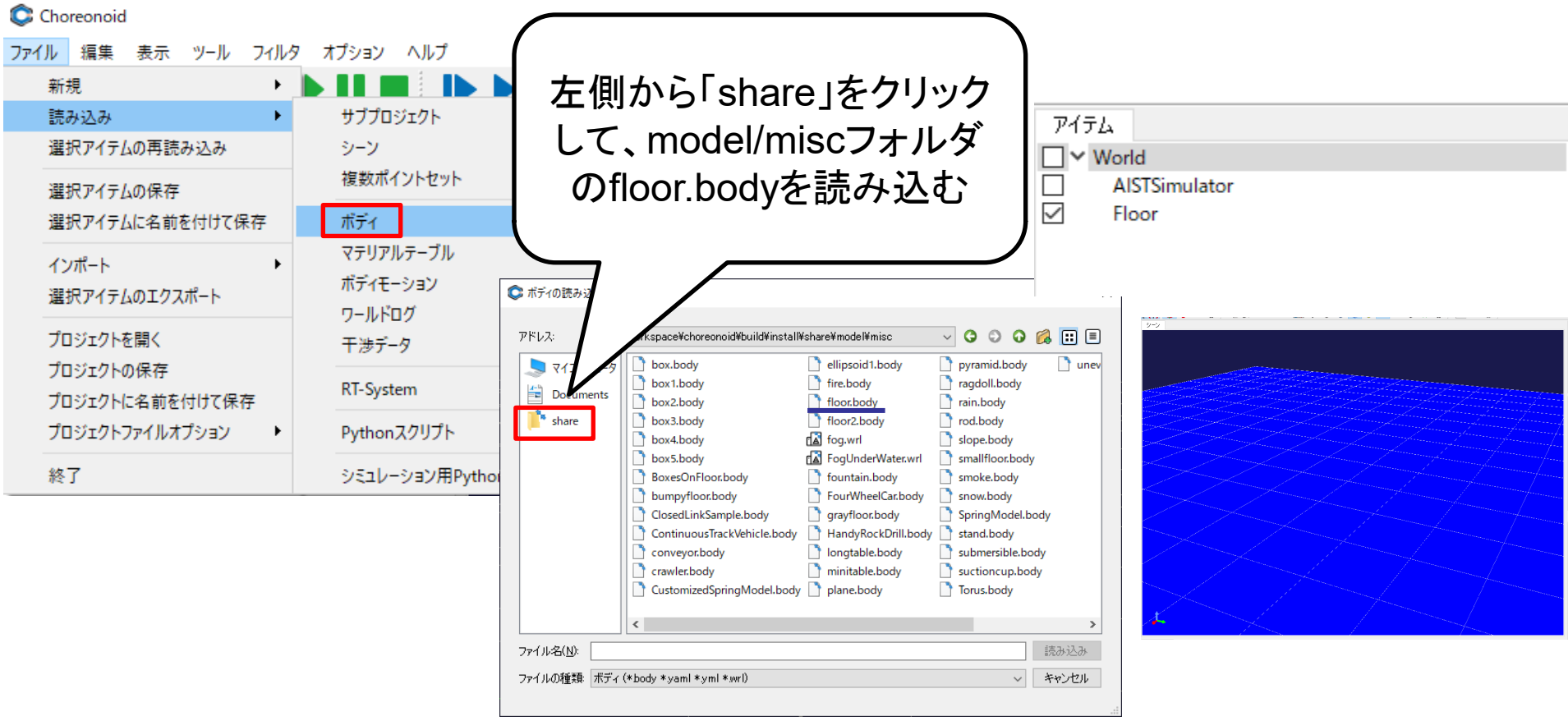
シミュレータアイテム追加

- Choreonoidは複数の物理シミュレータ(ODE、Bullet、PhysX)等から選択できる。
  - 今回はAISTシミュレータを選択する。



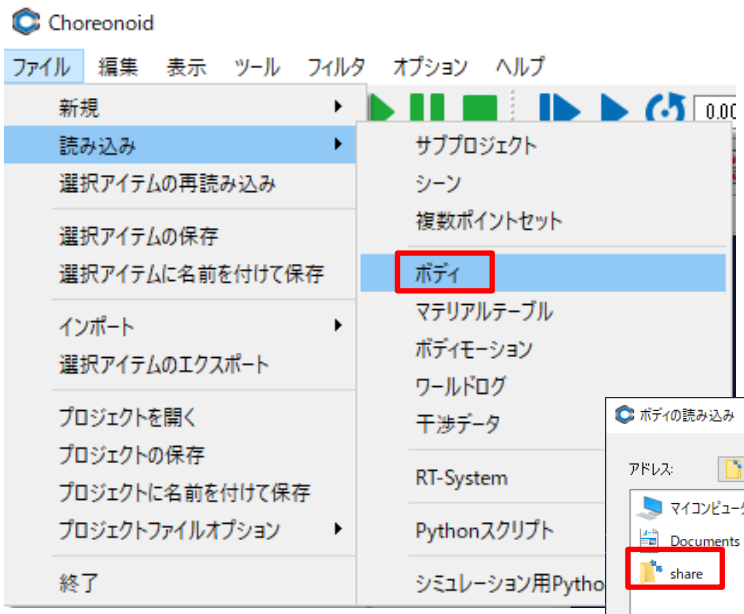
地面追加

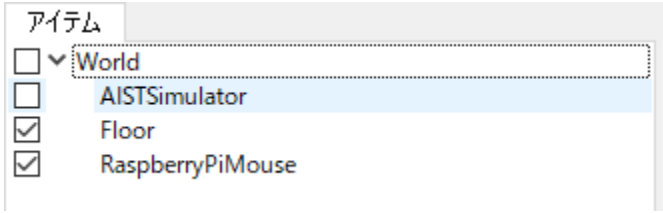
- 環境に地面を表現するボディアイテムを追加する
  - ファイル → 読み込み → ボディ
  - share/model/misc/floor.bodyを読み込む

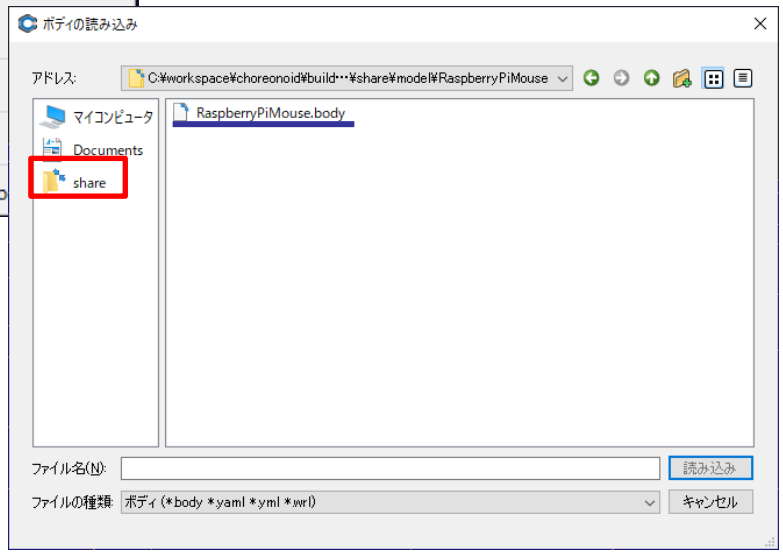


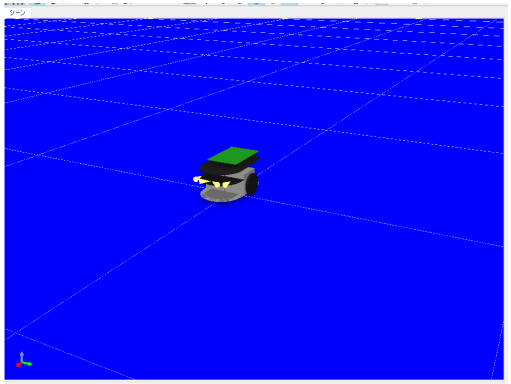
Raspberry Piマウス追加

- Raspberry Piマウスを表現するボディアイテムを追加する
  - share/model/RaspberryPiMouse/RaspberryPiMouse.bodyを読み込む



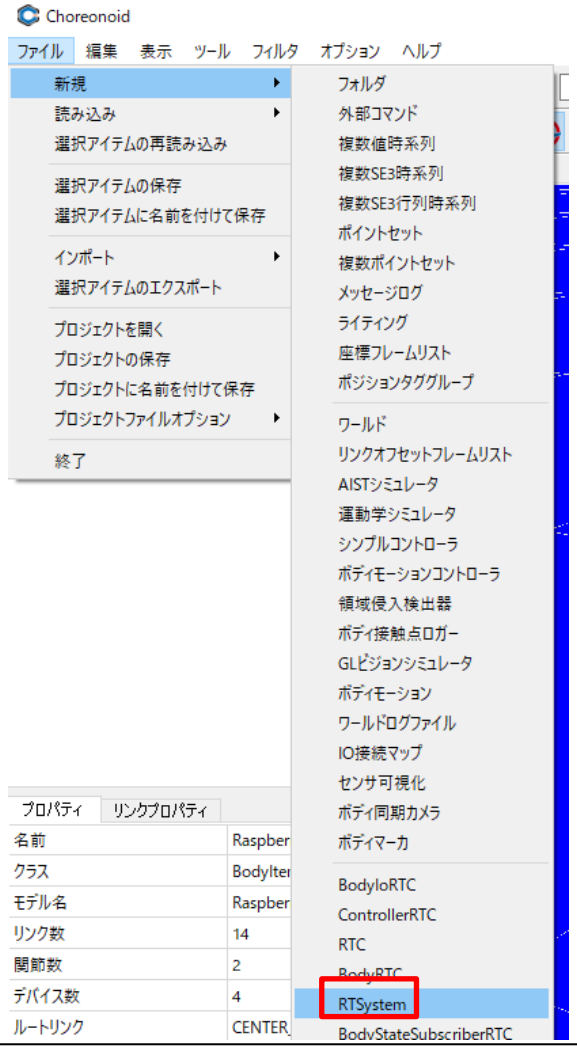


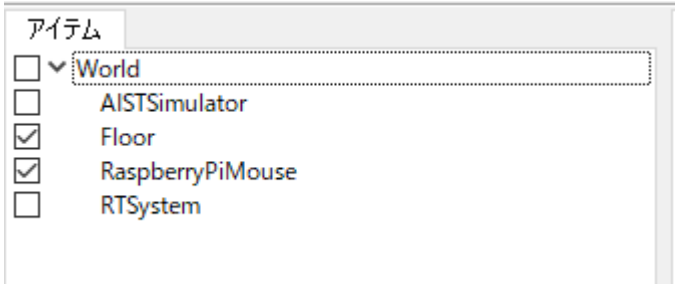




RTSystem追加

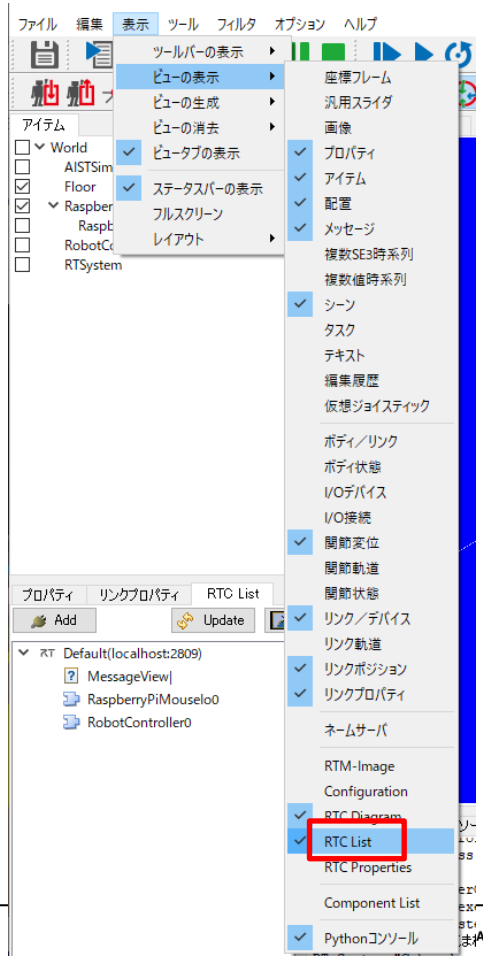
- Choreonoid上でRTシステムエディタの一部機能を使用できる

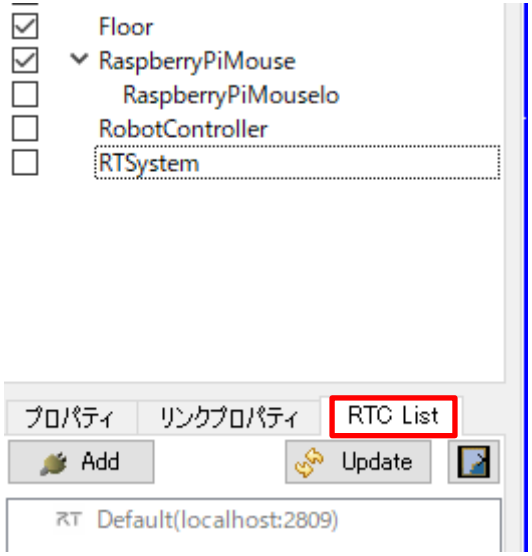




ネームサーバー、システムエディタ表示

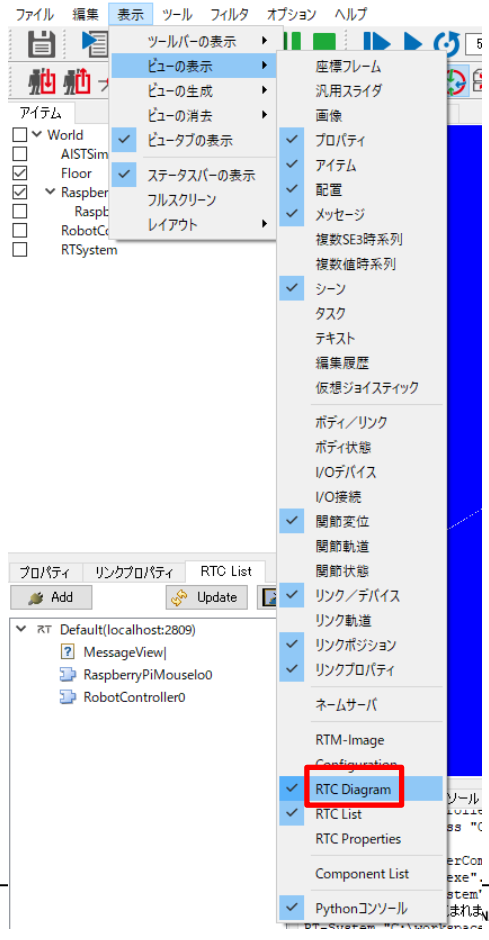
- 初期状態ではネームサーバー、システムエディタが非表示のため設定を変更する
  - 表示 → ビューの表示 → RTC List

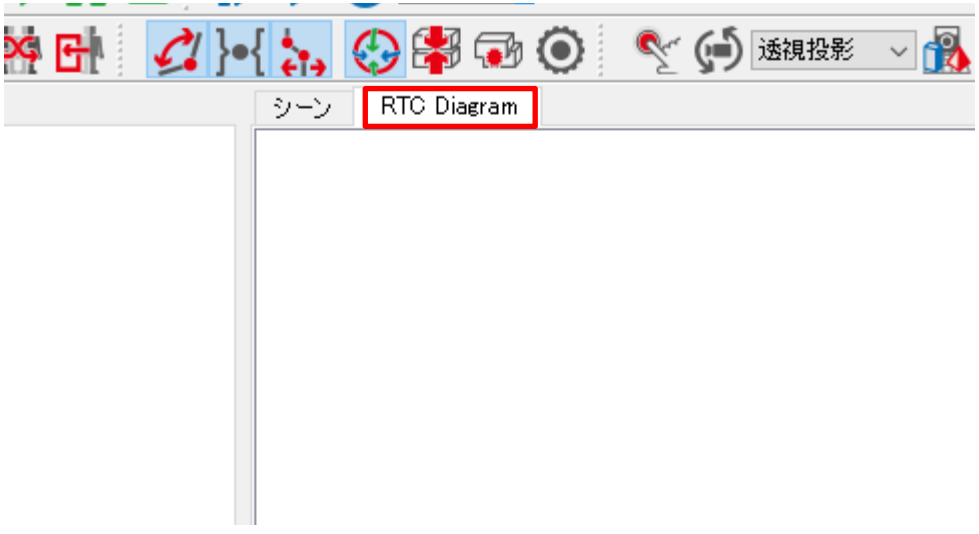




ネームサーバー、システムエディタ表示

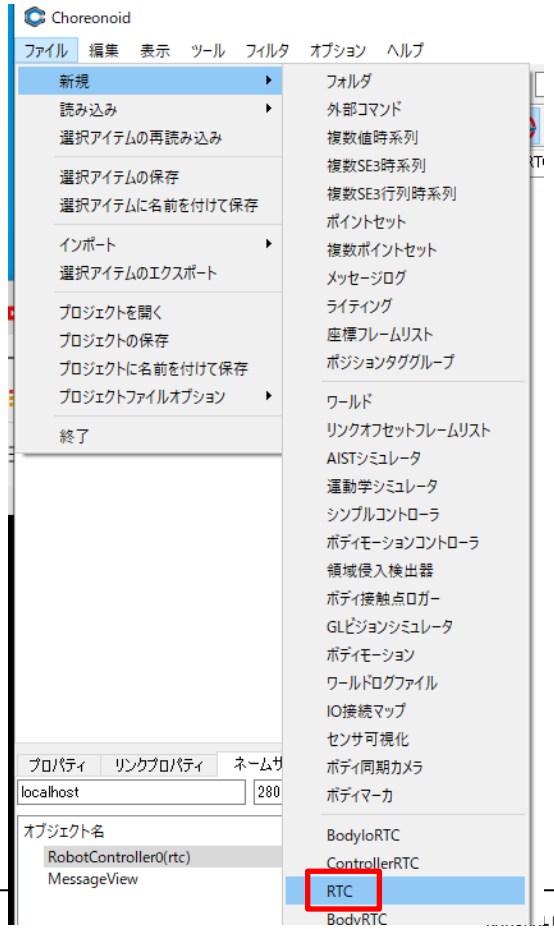
- 初期状態ではネームサーバー、システムエディタが非表示のため設定を変更する
  - 表示 → ビューの表示 → RTC Diagram



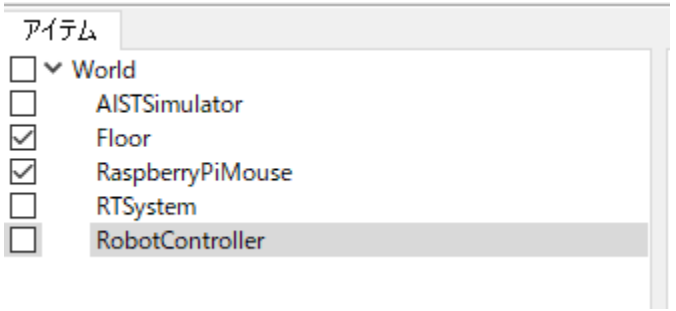


RTCアイテム

- RTCアイテムを追加して、ChoreonoidがRobotControllerコンポーネントを起動するように設定する
  - ファイル → 新規 → RTC



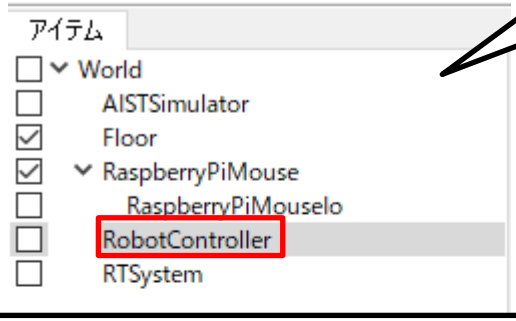
名前を「RobotController」に変更する



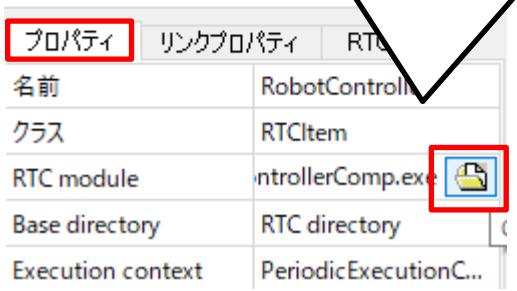
RobotControllerコンポーネントの設定

- RobotControllerアイテムでRobotControllerComp.exeを設定する

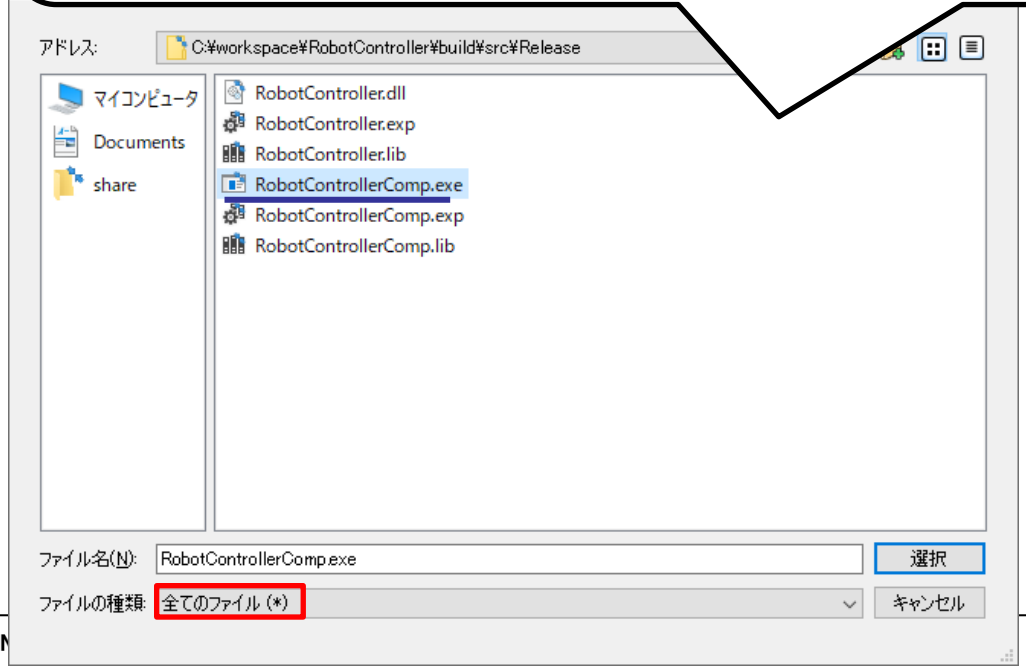
1. アイテムから「RobotController」を選択する。



2. 下の「プロパティ」で「RTC Module」を設定する

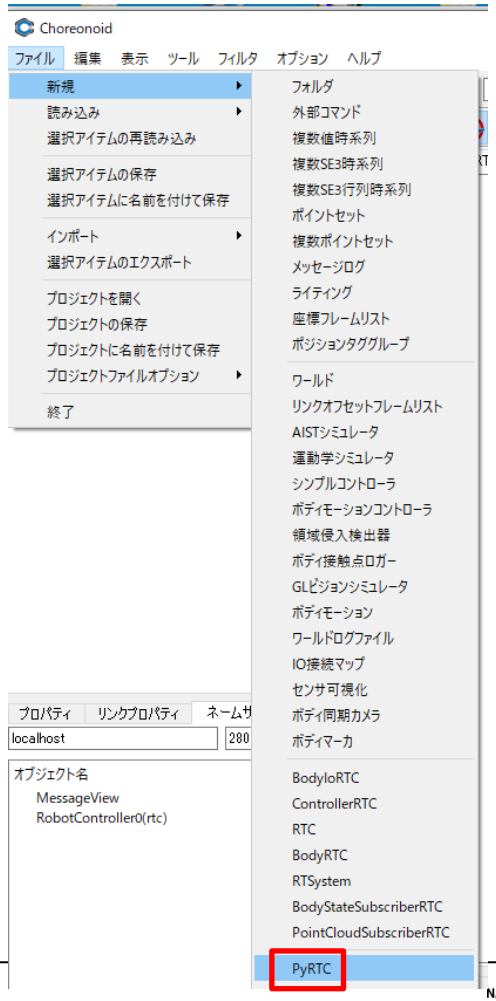


3. ファイル選択で前の実習で作成した「RobotControllerComp.exe」を選択する。
※初期状態ではexeファイルが表示されないので、「ファイルの種類」を「すべてのファイル(*)」に変更する

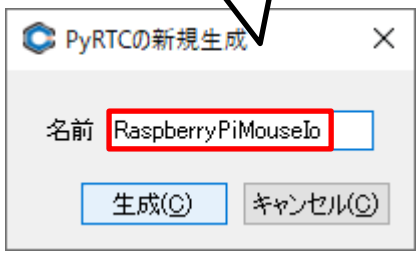


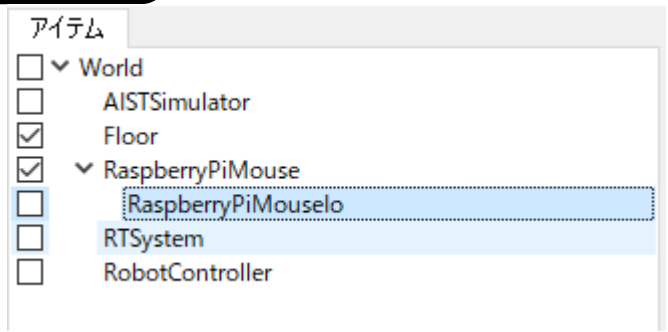
RaspberryPiMouseIoコンポーネント追加

- シミュレータ上のRaspberryPiマウスの入出力RTCを追加する
  - ファイル → 新規 → PyRTC



名前を「RaspberryPiMouseIo」に変更する





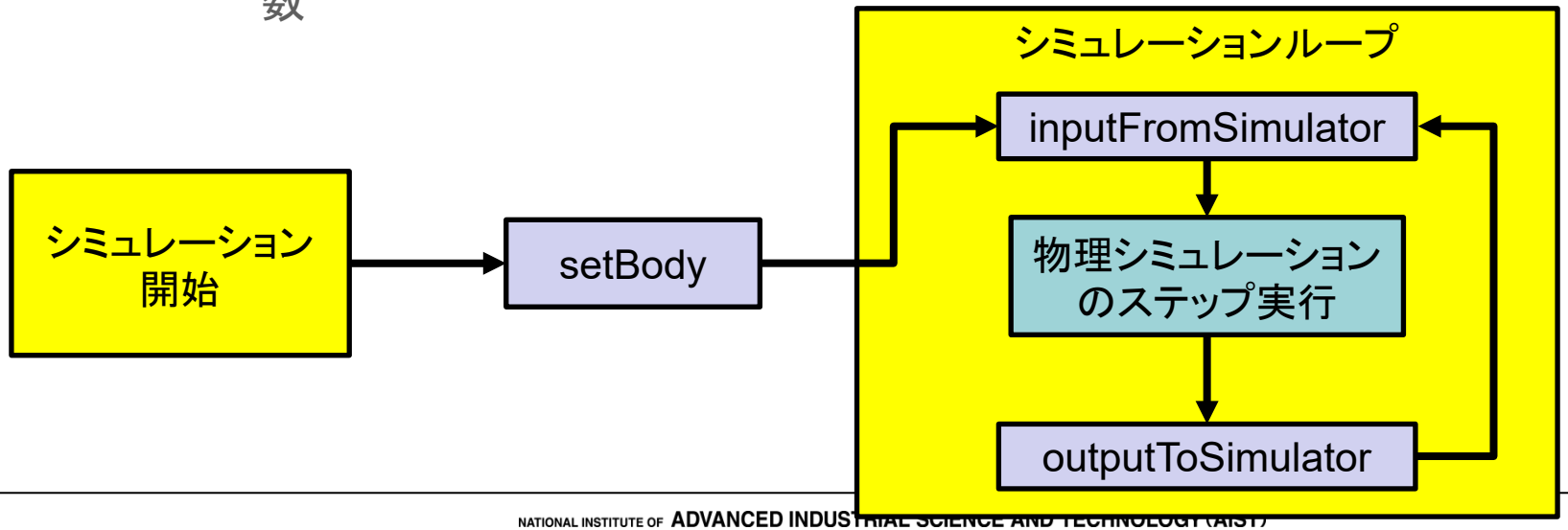
RaspberryPiMouseIoコンポーネント作成

- ※ここからはRTC Builderで作業します
- RTC Builderで以下のRTコンポーネントを作成する
  - コード生成すると**RaspberryPiMouseIo.py**が作成される

基本	
コンポーネント名	RaspberryPiMouseIo
言語	Python
アクティビティ	
なし	
データポート(OutPort)	
なし	
データポート(InPort)	
ポート名	velocity
データ型	RTC::TimedVelocity2D
コンフィギュレーション	
なし	

RaspberryPiMouseIo.pyの編集

- RaspberryPiMouseIoクラスに以下のメンバ関数を追加する
  - **setBody**関数
    - RTC側でChoreonoidのBodyオブジェクトの参照を取得する関数
    - 取得したBodyオブジェクトからLinkオブジェクトを取得することで、対象のJointの入出力ができる
  - **outputToSimulator**関数
    - シミュレータ上のオブジェクトから取得したデータをOutPortから出力する処理を行う関数
  - **inputFromSimulator**関数
    - InPortの入力データをシミュレータ上のオブジェクトに入力する処理を行う関数



RaspberryPiMouseIo.pyの編集

onRateChanged関数の下あたりに追加する

```
    # def onRateChanged(self, ec_id):
    #
    #   return RTC.RTC_OK

    def setBody(self, body):
        self.ioBody = body
        self.wheelR = self.ioBody.link("RIGHT_WHEEL")
        self.wheelL = self.ioBody.link("LEFT_WHEEL")

    def outputToSimulator(self):
        pass
```

※インデントに注意

※インデントに注意

Bodyオブジェクトから
「RIGHT_WHEEL」、
「LEFT_WHEEL」のリンクを取得する

outputToSimulator関数は、
今回は何の処理もしない

RaspberryPiMouseIoクラスのメンバ関数として追加するため、
インデントには注意する。
(上の「# def onRateChanged～」の前のインデントと同じにする)

RaspberryPiMouselo.pyの編集

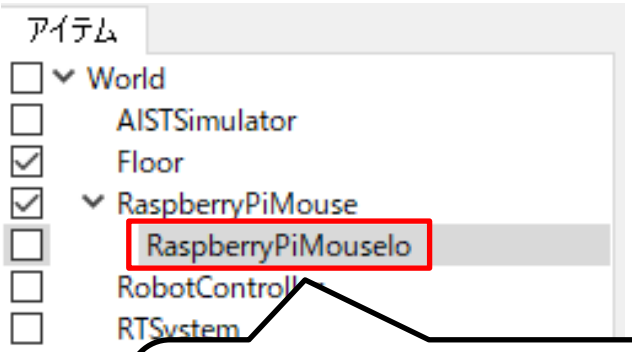
※インデントに注意

```
    def inputFromSimulator(self):
        if self._velocityIn.isNew():
            data = self._velocityIn.read()
            vx = data.data.vx
            va = data.data.va

            wheel_distance = 0.0425
            wheel_radius = 0.04
            rms = (vx + va*wheel_distance)/wheel_radius
            lms = (vx - va*wheel_distance)/wheel_radius

            self.wheelR.dq = rms
            self.wheelL.dq = lms
```

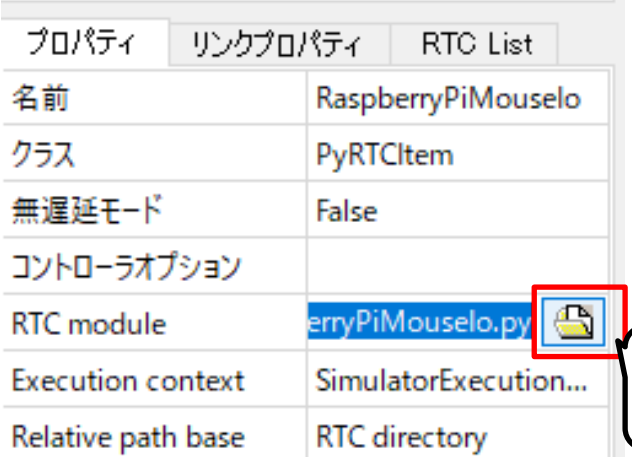
InPortで受信したTimedVeclocity2D型のデータを車輪の回転速度に変換

「RIGHT_WHEEL」、「LEFT_WHEEL」のリンクオブジェクトの「dq」にJointの回転速度を設定する

RaspberryPiMouseloコンポーネントの設定

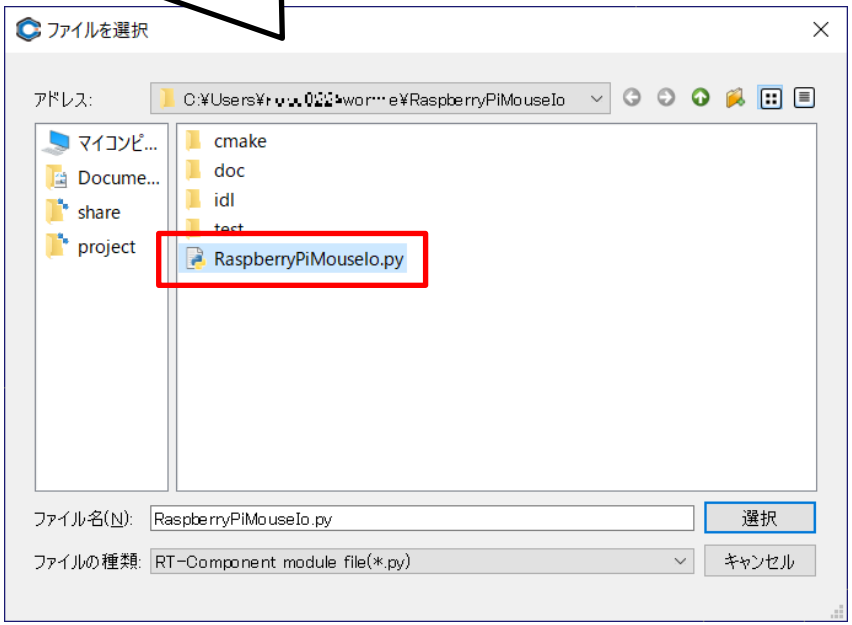


1. Choreonoidのアイテムビューで RaspberryPiMouseloを選択する



2. RTC moduleを設定する

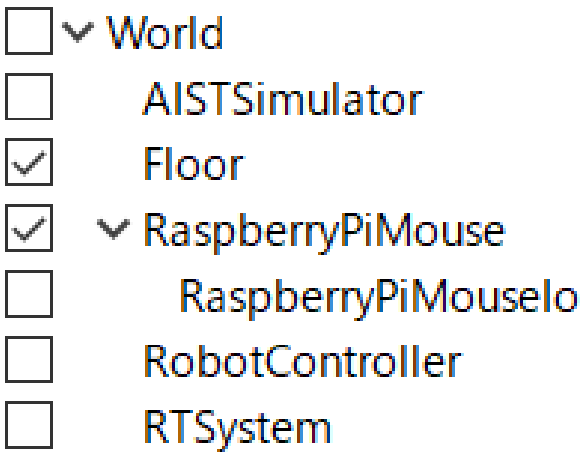
3. 先ほど編集した「RaspberryPiMouselo.py」を選択する



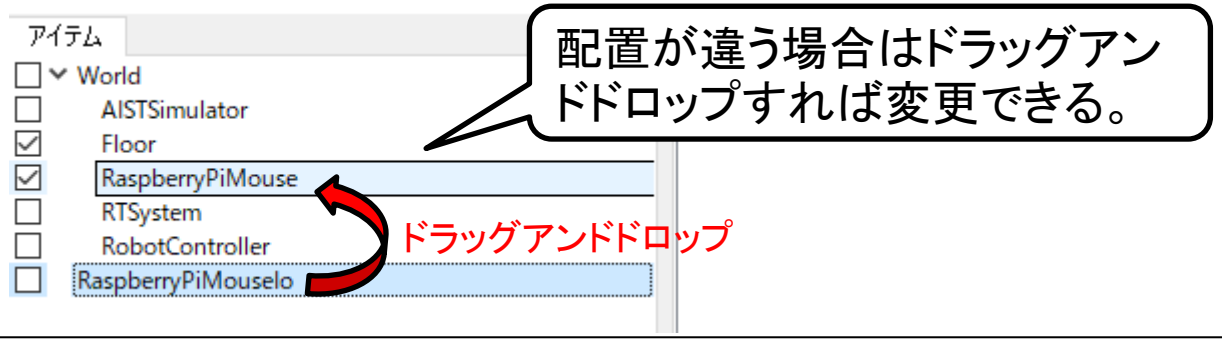
※ RaspberryPiMouselo.pyを更新した場合は、再度この作業を行う事で再読み込みする

アイテムの位置関係

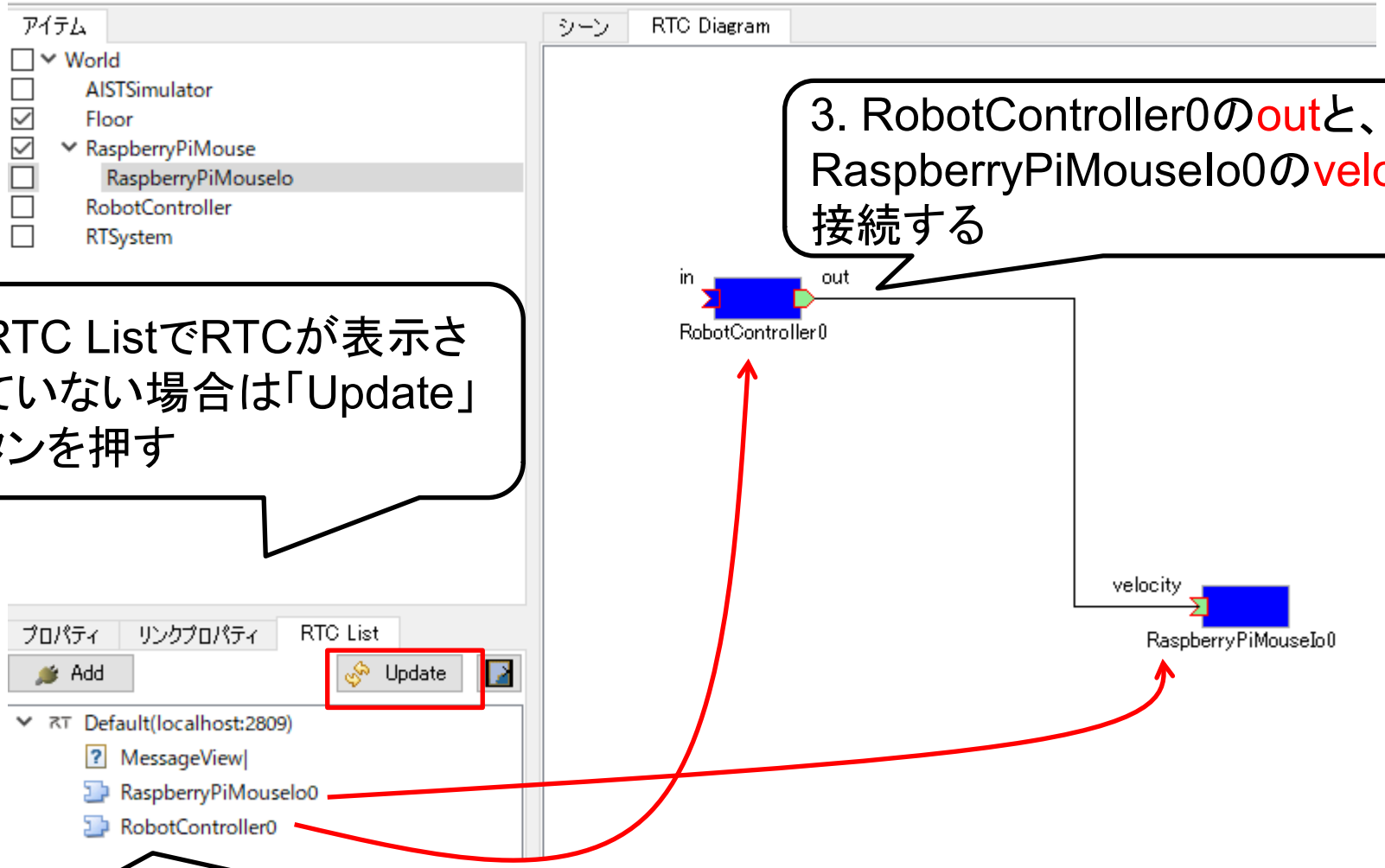
- 全てのアイテムをWorldの子アイテムとして配置
- 「RaspberryPiMouseIo」は「RaspberryPiMouse」の子アイテムとして配置
  - 配置が違う場合はドラッグアンドドロップして移動する



World
|- AISTSimulator
|- Floor
|- RaspberryPiMouse
| |- RaspberryPiMouseIo
|- RobotController
|- RTSystem

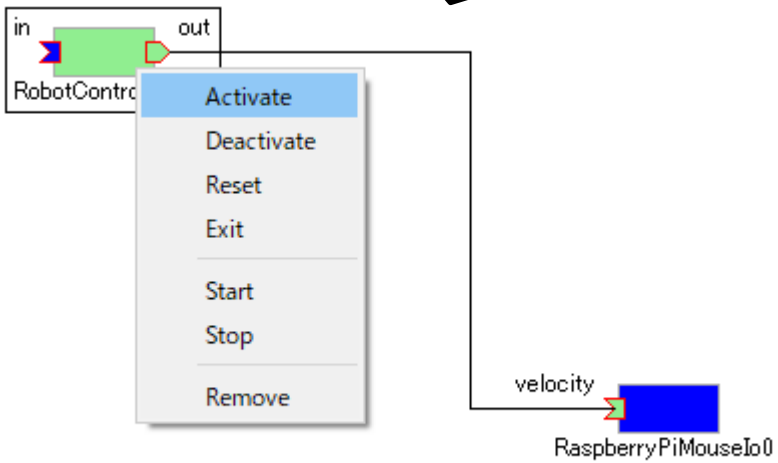


ポート接続

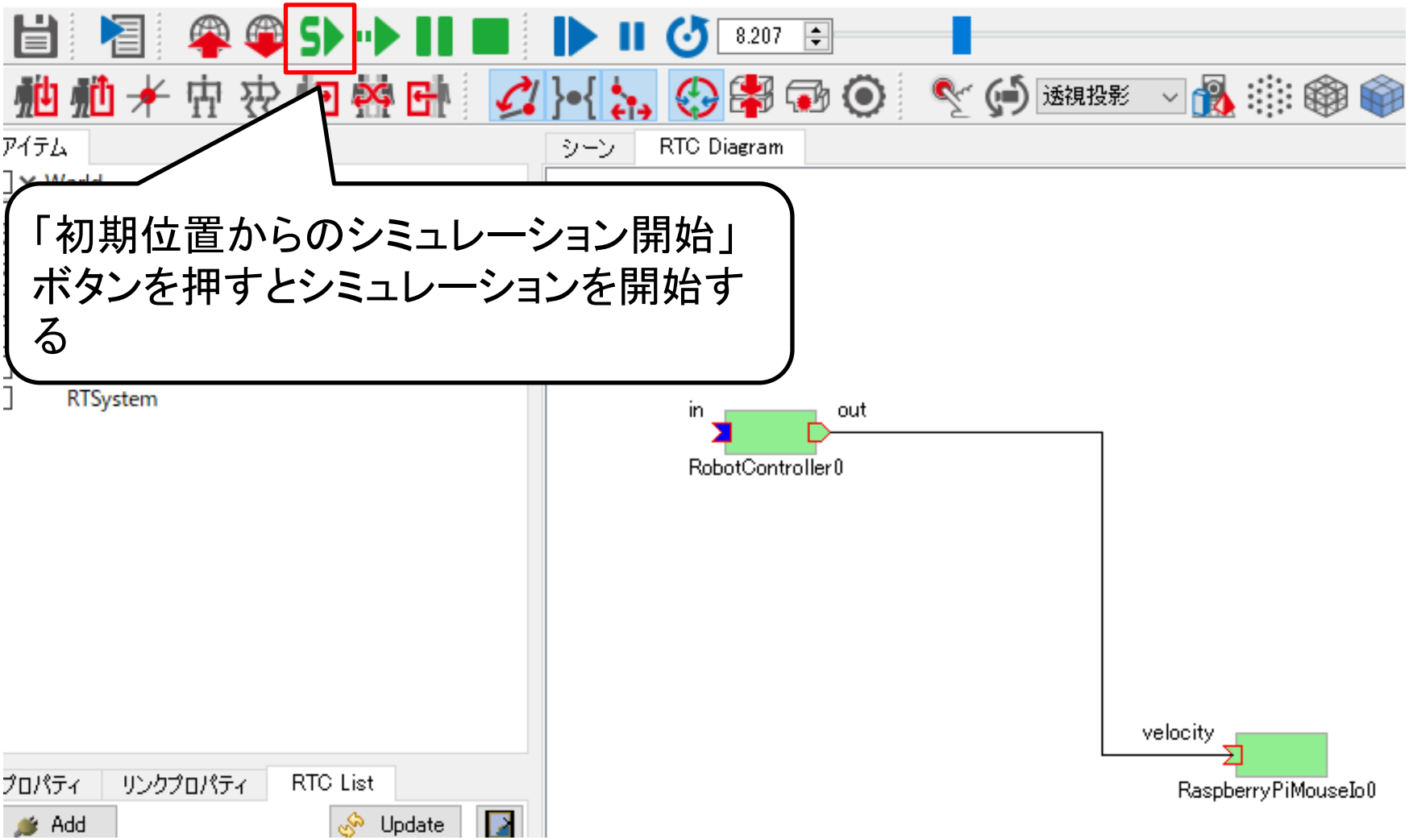

アイテム
World
AISTSimulator
Floor
RaspberryPiMouse
RaspberryPiMouseIo
RobotController
RTSystem
シーン
RTC Diagram
3. RobotController0のoutと、RaspberryPiMouseIo0のvelocityを接続する
in
out
RobotController0
velocity
RaspberryPiMouseIo0
1. RTC ListでRTCが表示されていない場合は「Update」ボタンを押す
プロパティ
リンクプロパティ
RTC List
Add
Update
Default(localhost:2809)
MessageView|
RaspberryPiMouseIo0
RobotController0
2. RTC DiagramにRTCをドラッグアンドドロップする

RobotControllerコンポーネントのアクティブ化

exeファイルを指定したRTCはシミュレーション開始時に自動でアクティブ化されないので手動で操作する。
RobotController0を右クリックして「Activate」を選択する。

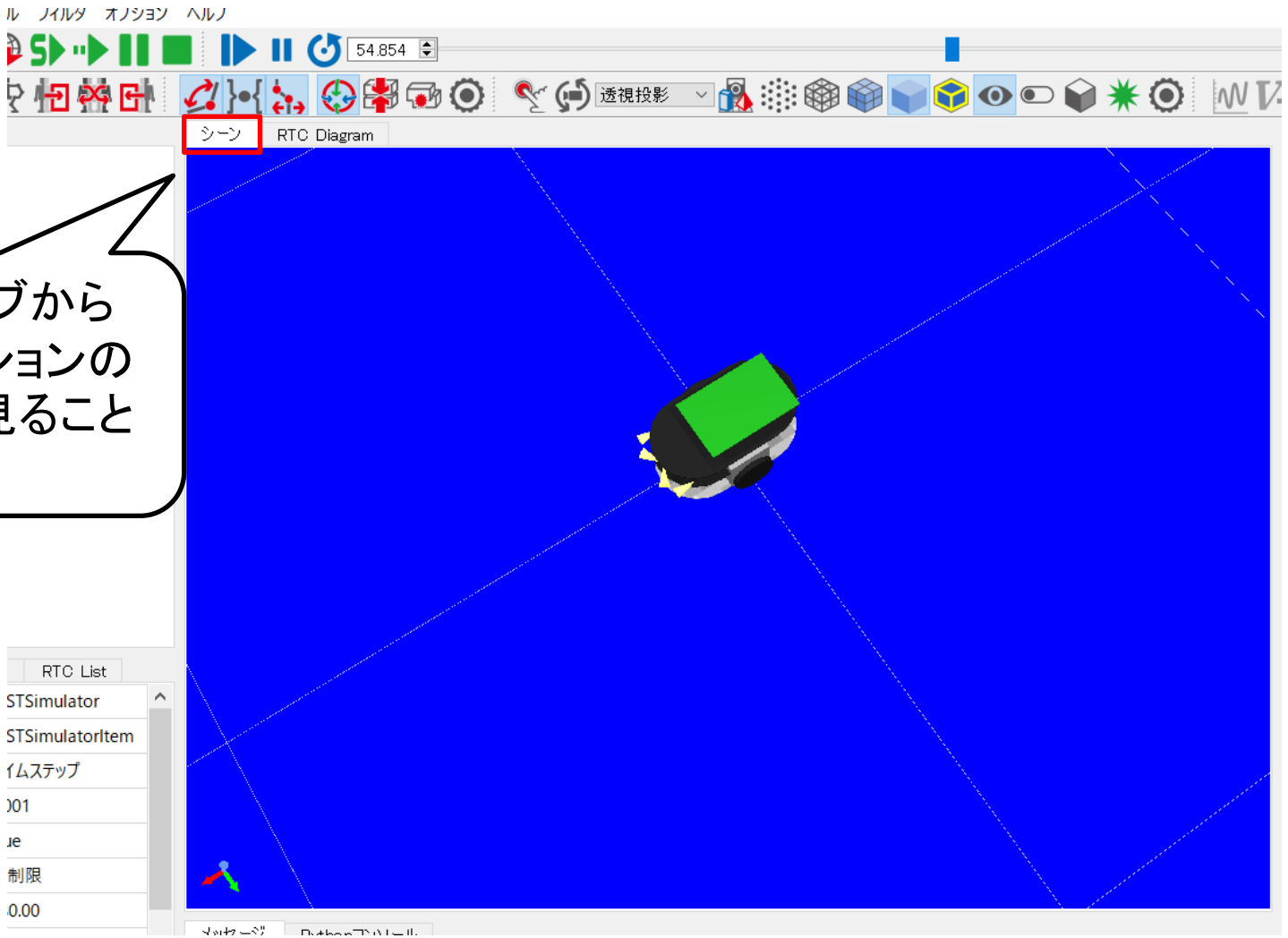


シミュレーション開始



「初期位置からのシミュレーション開始」ボタンを押すとシミュレーションを開始する

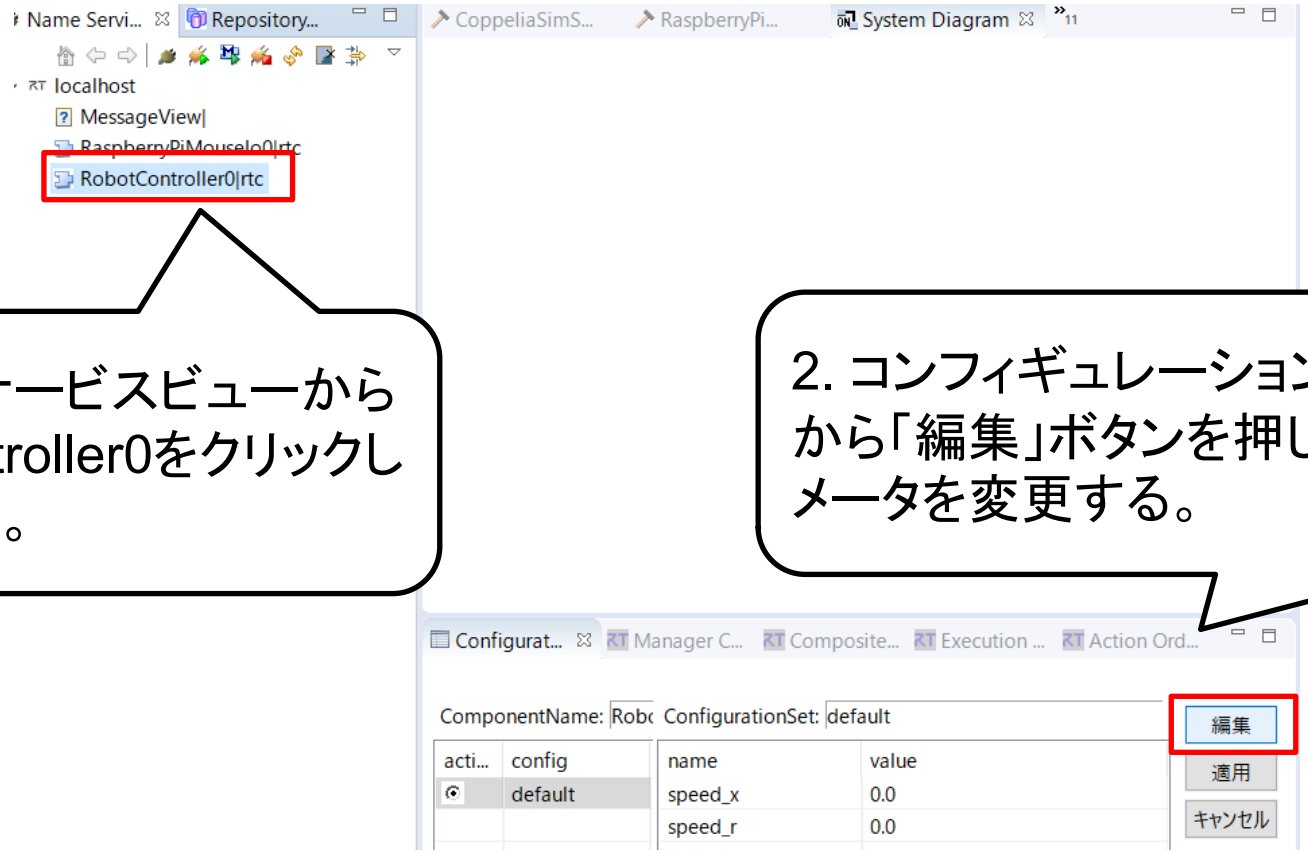
シミュレーション開始



「シーン」タブから
シミュレーションの
3D表示を見ること
ができる

コンフィギュレーションパラメータの編集

- Choreonoidからはコンフィギュレーションパラメータの編集ができないため、RT System Editorを起動してください



1. ネームサービスビューからRobotController0をクリックして選択する。

2. コンフィギュレーションビューから「編集」ボタンを押して、パラメータを変更する。

コンフィギュレーションパラメータを変更することで、Choreonoid上のRaspberry Piマウスが移動すれば課題達成です

コンフィギュレーションパラメータの編集

- ネームサービスビューにネームサーバが無い(localhostが非表示の場合)、以下の作業でネームサーバに接続してください