

# Choreonoid入門

宮本 信彦

国立研究開発法人産業技術総合研究所  
インテリジェントシステム研究部門



# 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>



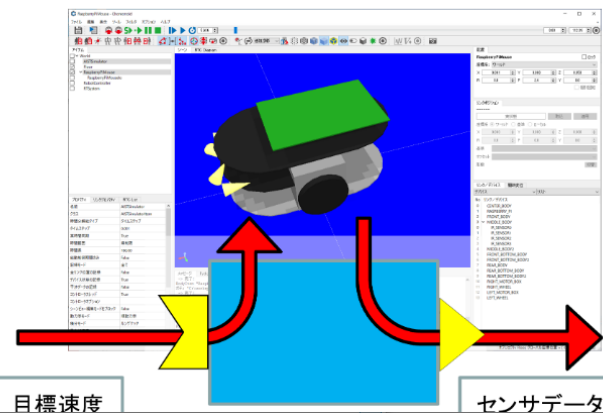
• [コンフィギュレーションパラメータの編集](#)

## はじめに

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です。拡張性が高く、物理エンジン、通信機能、スクリプティング機能、制御アルゴリズム等をC++プラグインとして追加できます。

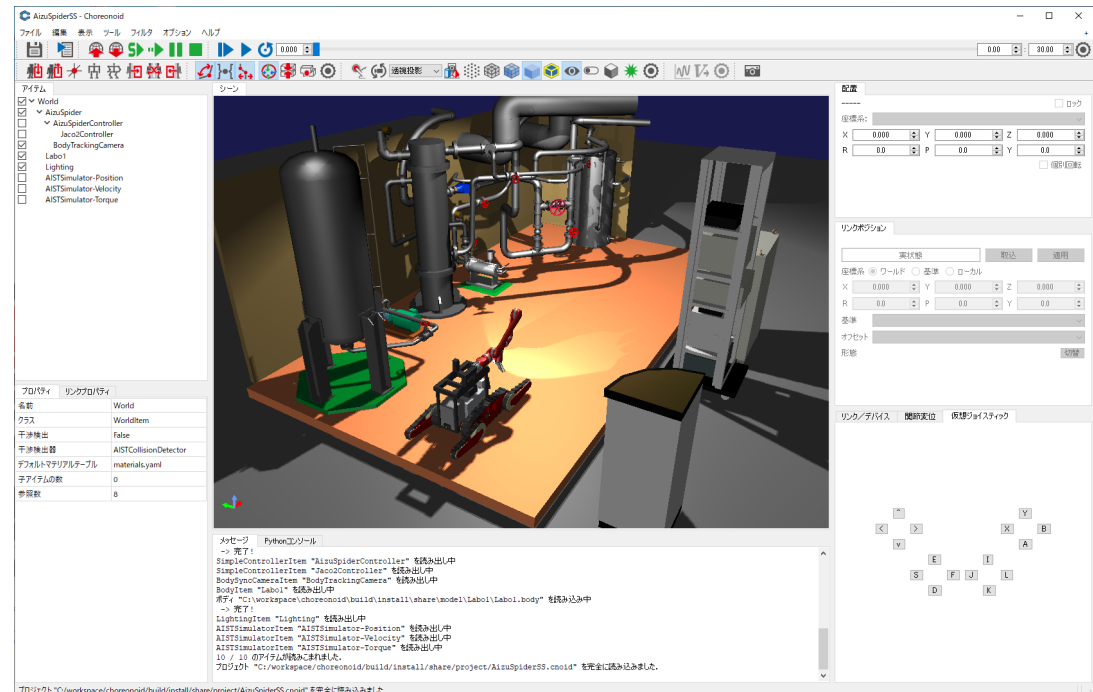
• [Choreonoid ホームページ](#)

このページでは、Choreonoidシミュレータ上の移動ロボットの入出力を行うRTCの作成手順を説明します。



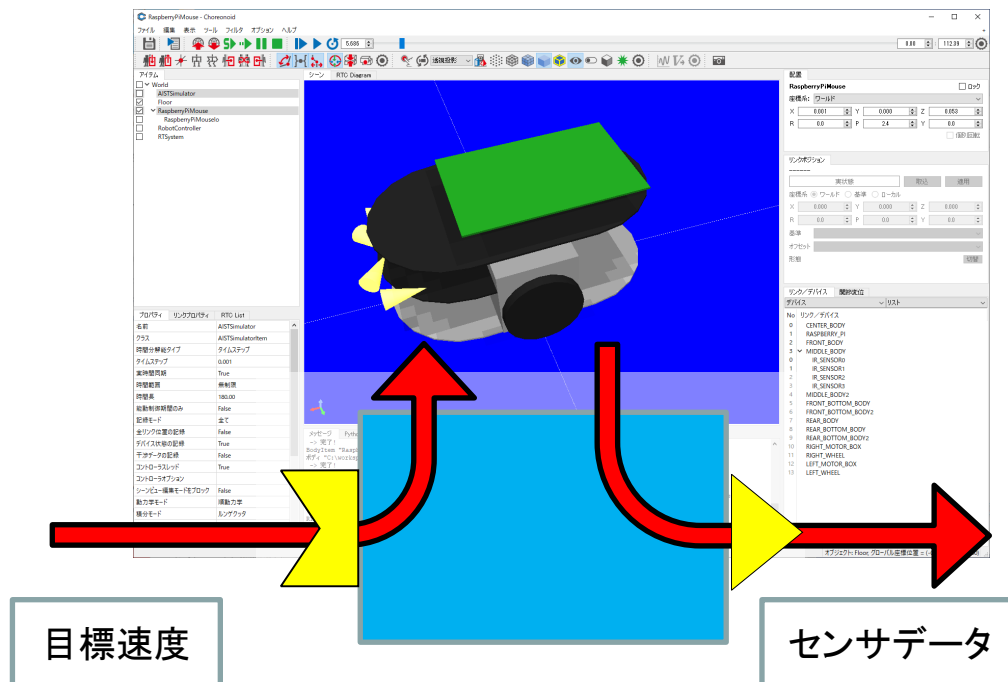
# Choreonoid

- 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、カセンサ、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

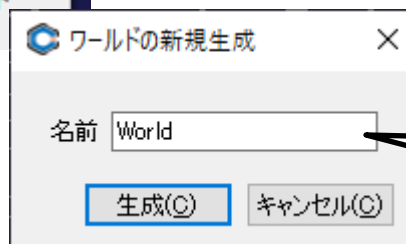
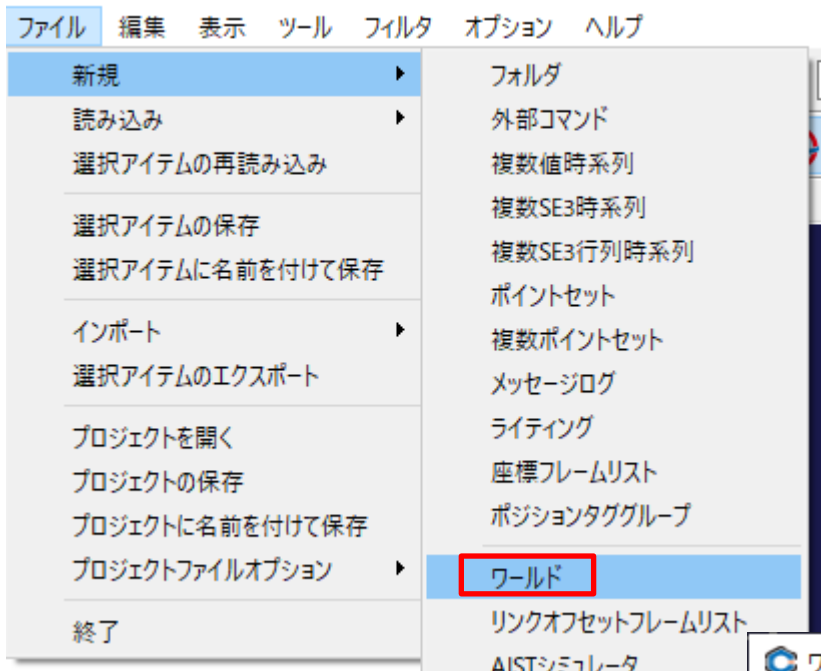
- Choreonoid上でシミュレーションの実行に必要なアイテムを追加することで、環境を構築する。
- 配布資料 (USBメモリ) のchoreonoidフォルダの **choreonoid.bat**を実行する
- 今回は以下のアイテムを追加する。
  - ワールドアイテム
  - シミュレータアイテム(AISTSimulator)
  - ボディアイテム(地面・Floor)
  - ボディアイテム(Raspberry Piマウス)
  - RTSystemアイテム
  - RTCアイテム(RobotController)
  - pyRTCアイテム(RaspberryPiMouselo)

名前	更新日時
bin	2022/09/30 10:56
include	2022/09/30 10:56
lib	2022/09/30 10:56
python-3.7.9-embed-amd64	2022/09/30 10:58
share	2022/09/30 10:58
choreonoid.bat	2022/05/24 20:08

# 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

Choreonoid

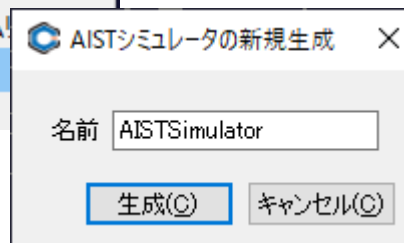
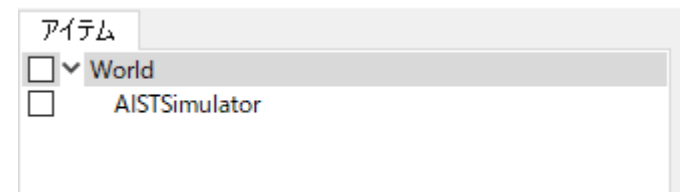
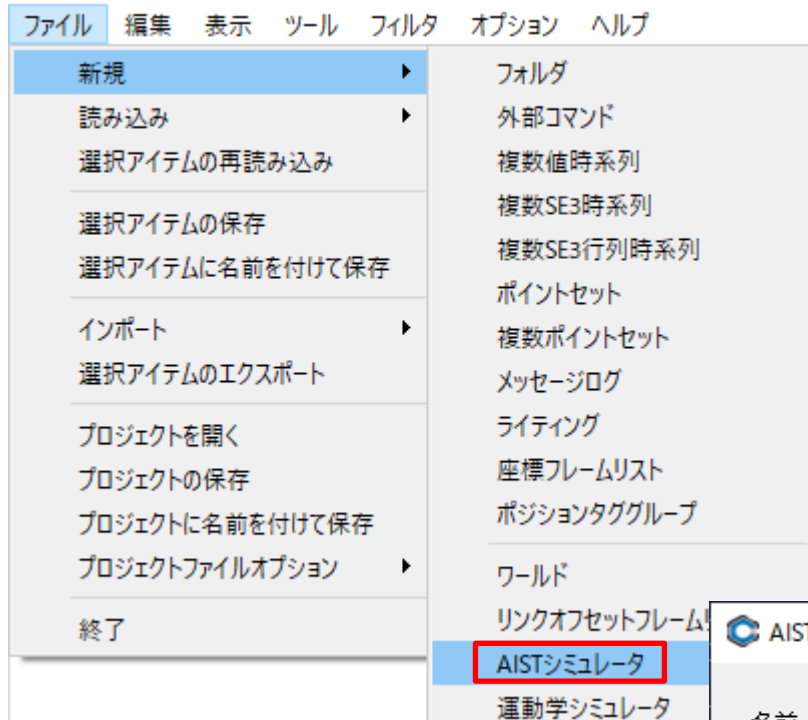


名前は変更しない

# シミュレータアイテム追加

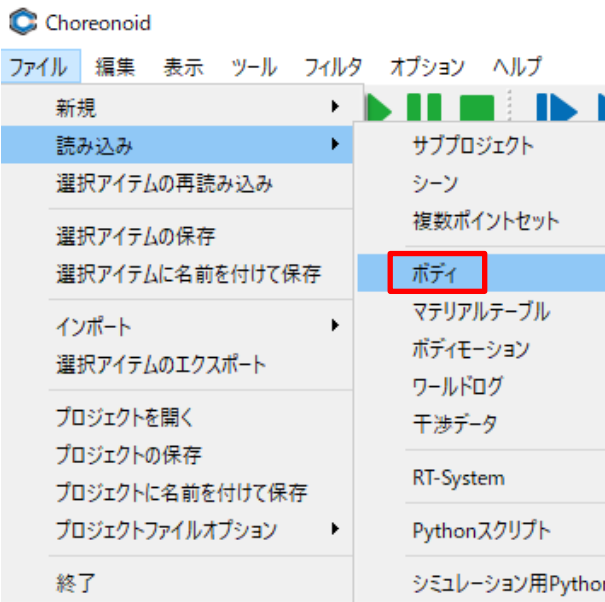
- Choreonoidは複数の物理シミュレータ(ODE、Bullet、PhysX)等から選択できる。
  - 今回はAISTシミュレータを選択する。

Choreonoid

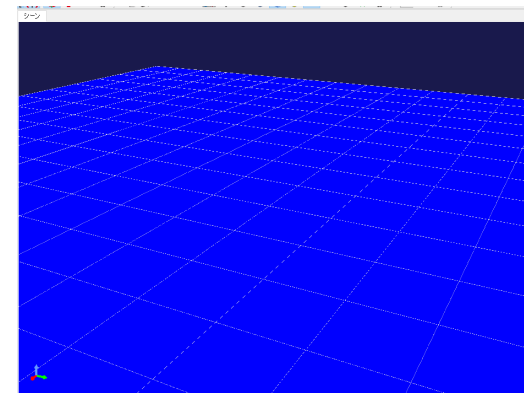
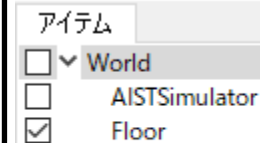
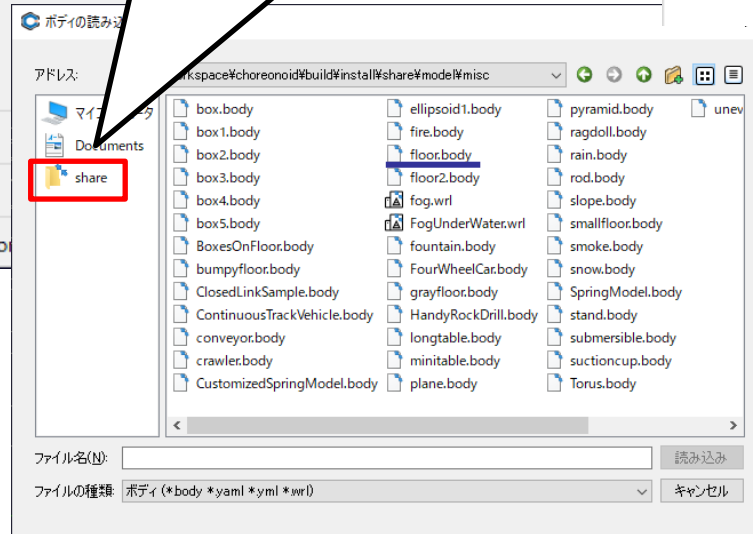


# 地面追加

- 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



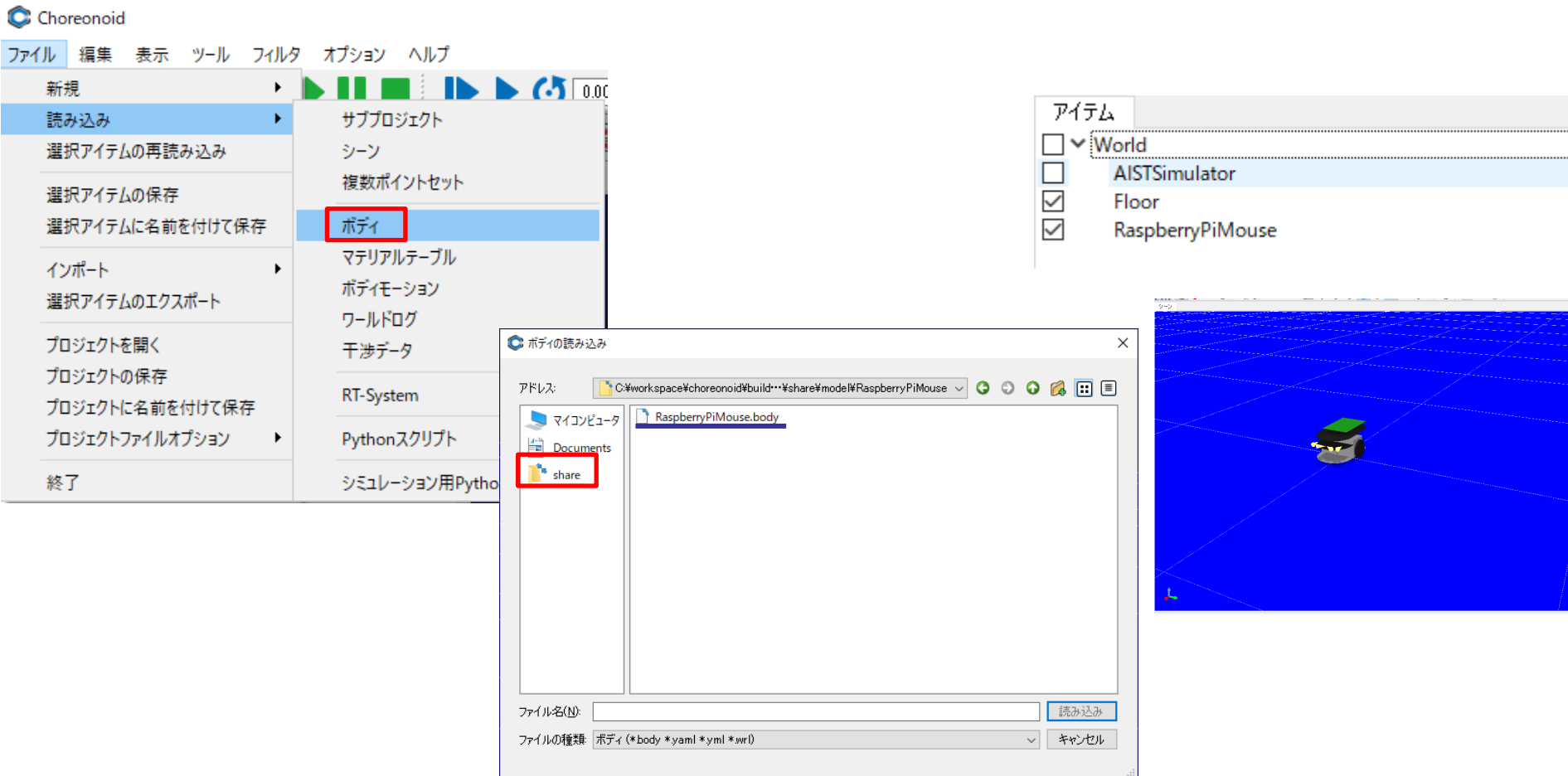
左側から「share」をクリックして、model/miscフォルダのfloor.bodyを読み込む





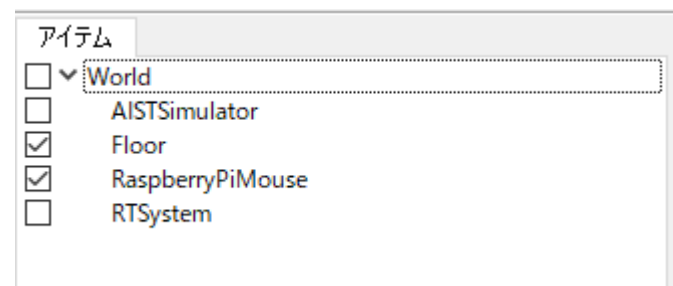
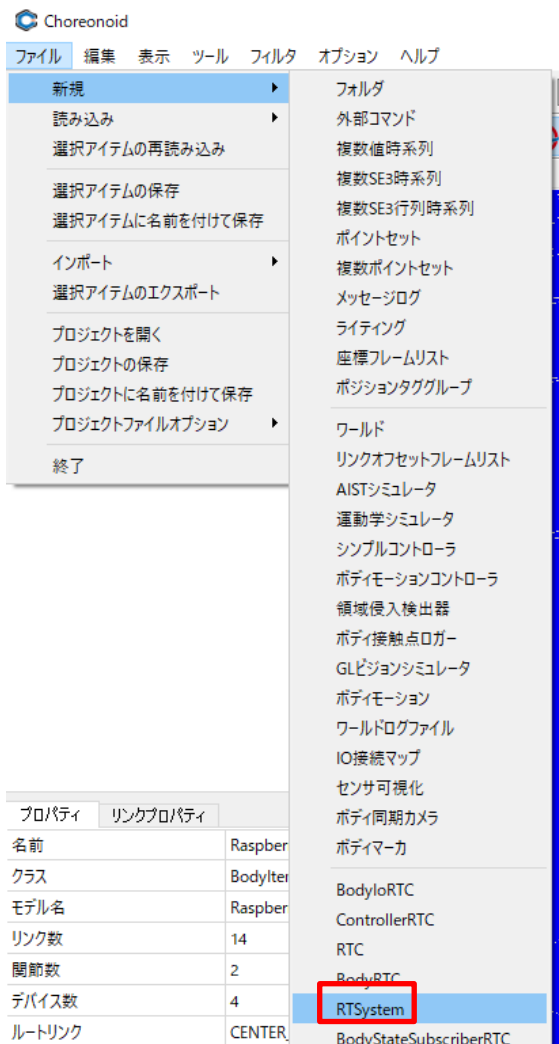
# Raspberry Piマウス追加

- 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



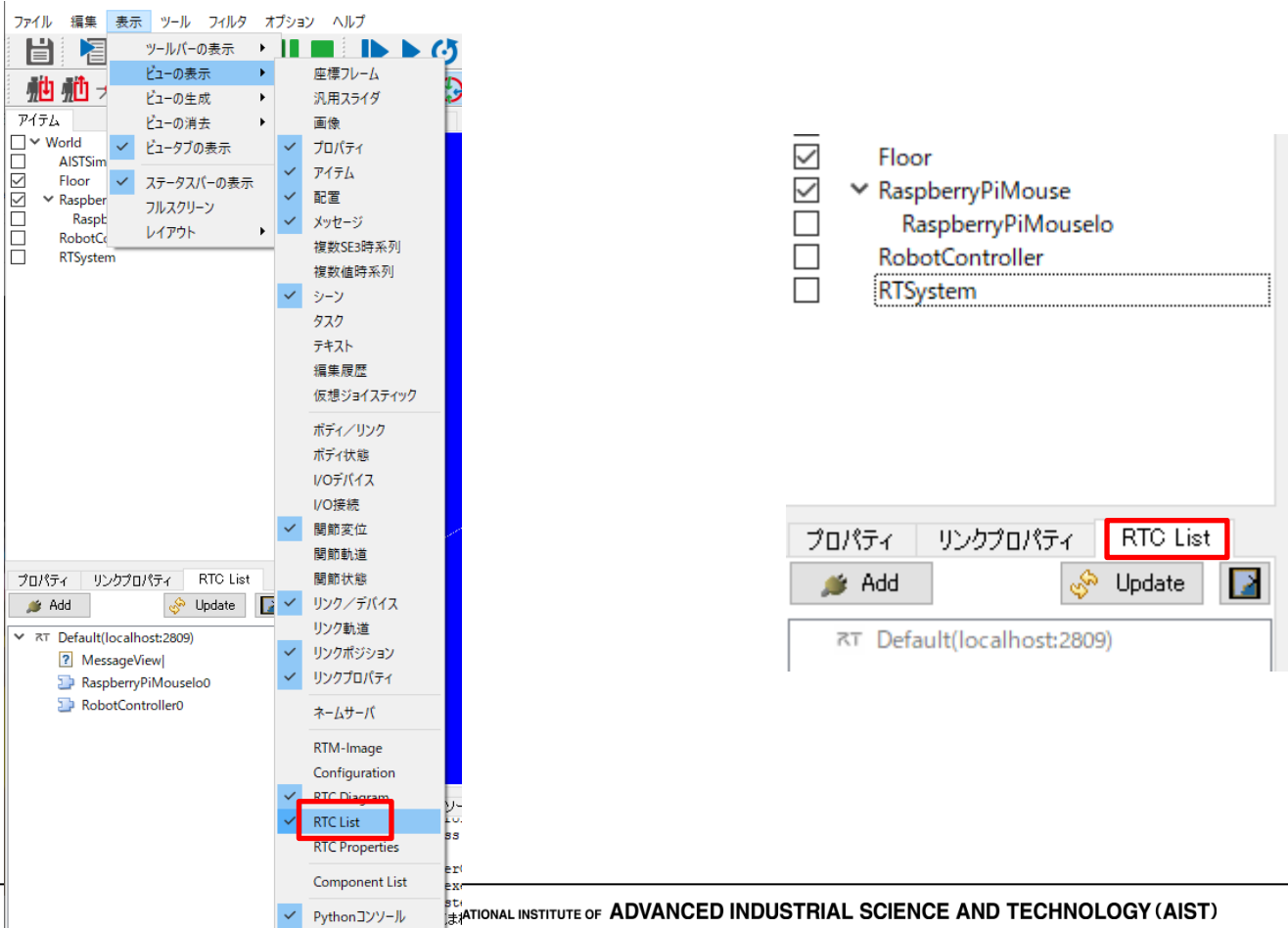
# RTSystem追加

- Choreonoid上でRTシステムエディタの一部機能を使用できる



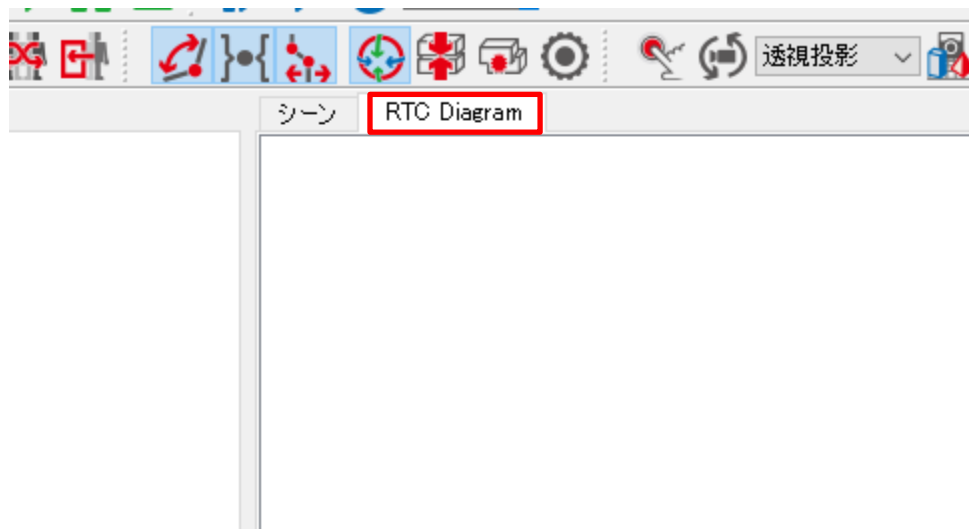
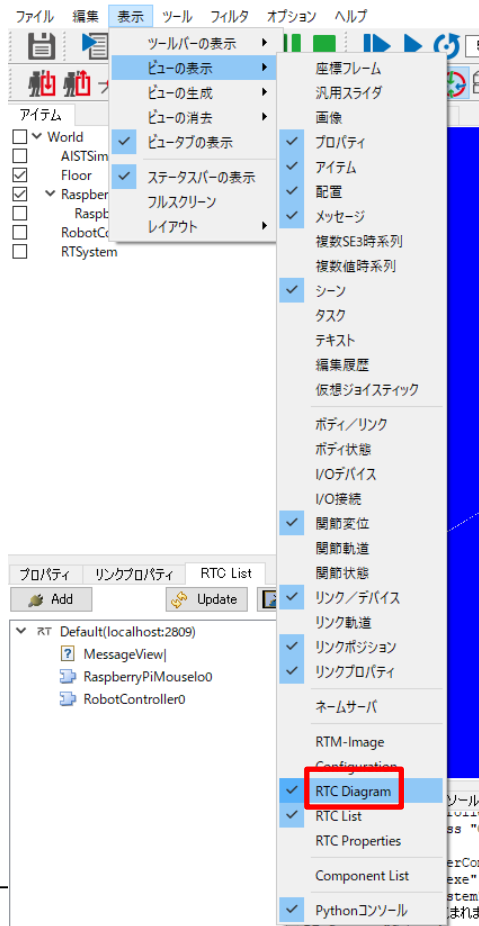
# ネームサーバー、システムエディタ表示

- 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

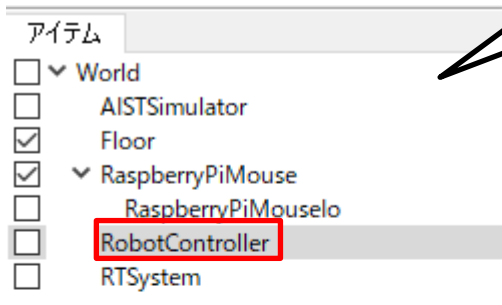
- RTCアイテムを追加して、ChoreonoidがRobotControllerコンポーネントを起動するように設定する
  - ファイル → 新規 → RTC



# RobotControllerコンポーネントの設定

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

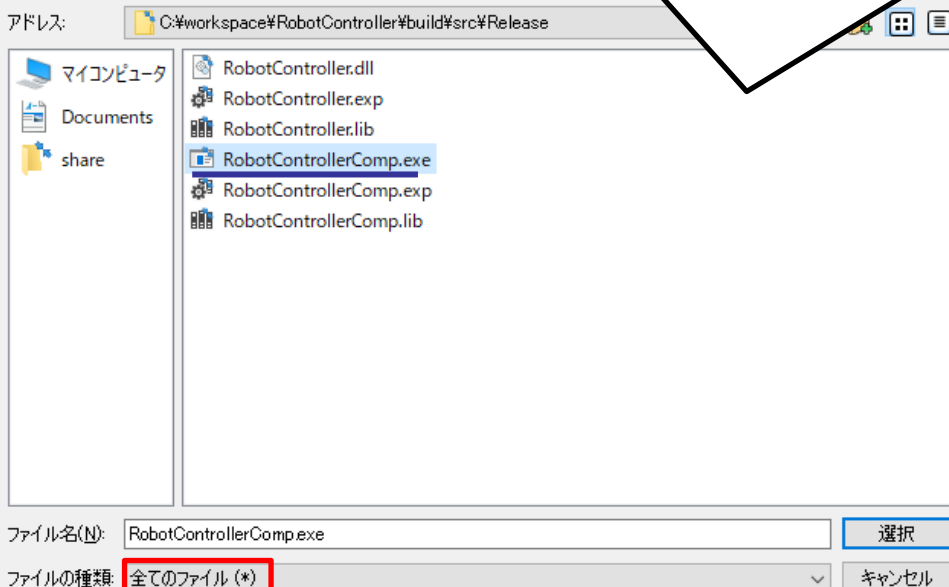
1. アイテムから「RobotController」を選択する。



2. 下の「プロパティ」で「RTC Module」を設定する

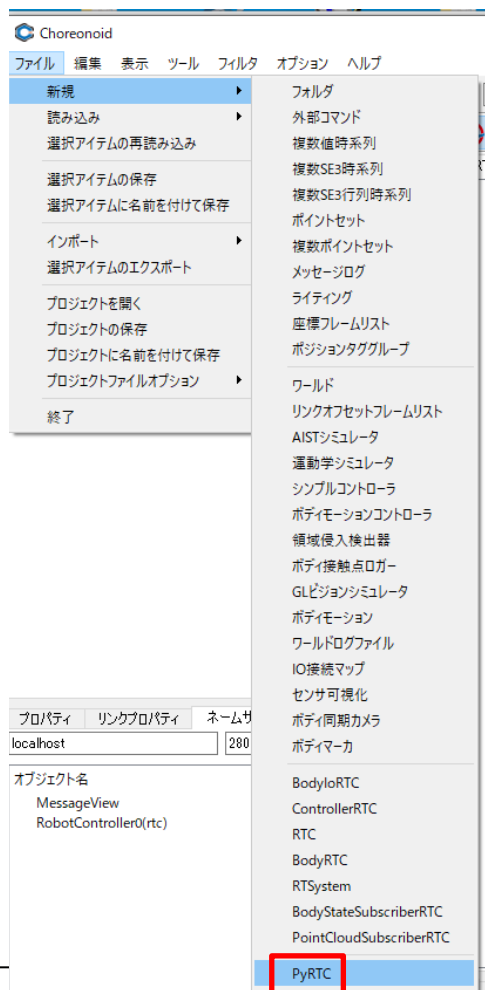


3. ファイル選択で前の実習で作成した「RobotControllerComp.exe」を選択する。  
※初期状態ではexeファイルが表示されないので、「ファイルの種類」を「すべてのファイル(\*)」に変更する

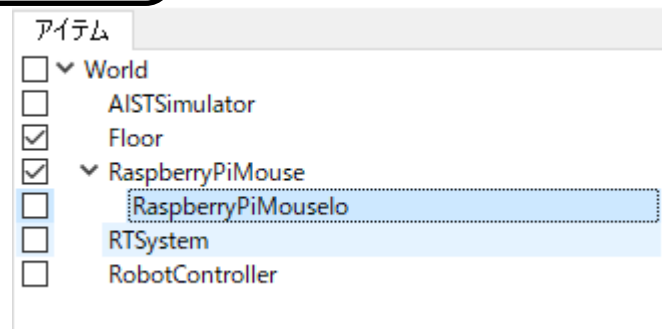
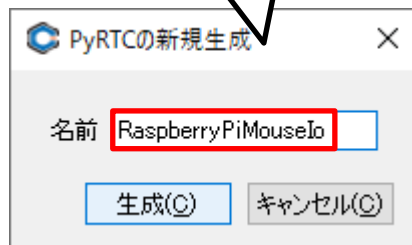


# RaspberryPiMouseLoコンポーネント追加

- 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「RaspberryPiMouseLo」  
に変更する



# RaspberryPiMouseloコンポーネント作成

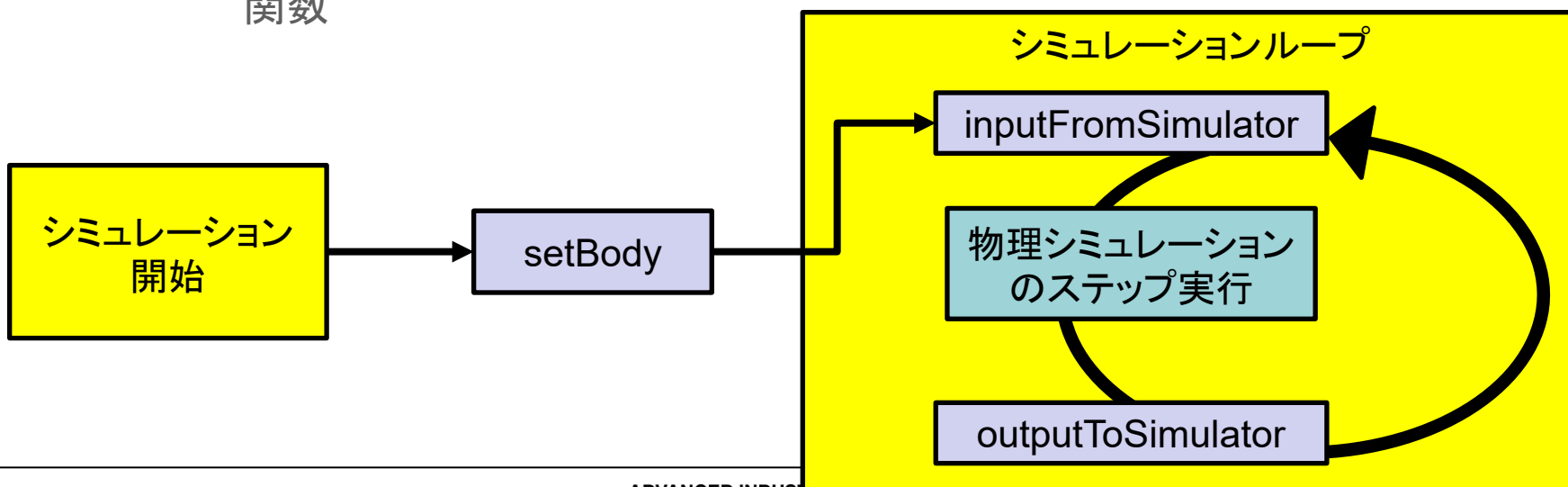
- ※ここからはRTC Builderで作業します
- RTC Builderで以下のRTコンポーネントを作成する
  - コード生成すると**RaspberryPiMouselo.py**が作成される

基本	
コンポーネント名	RaspberryPiMouselo
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	



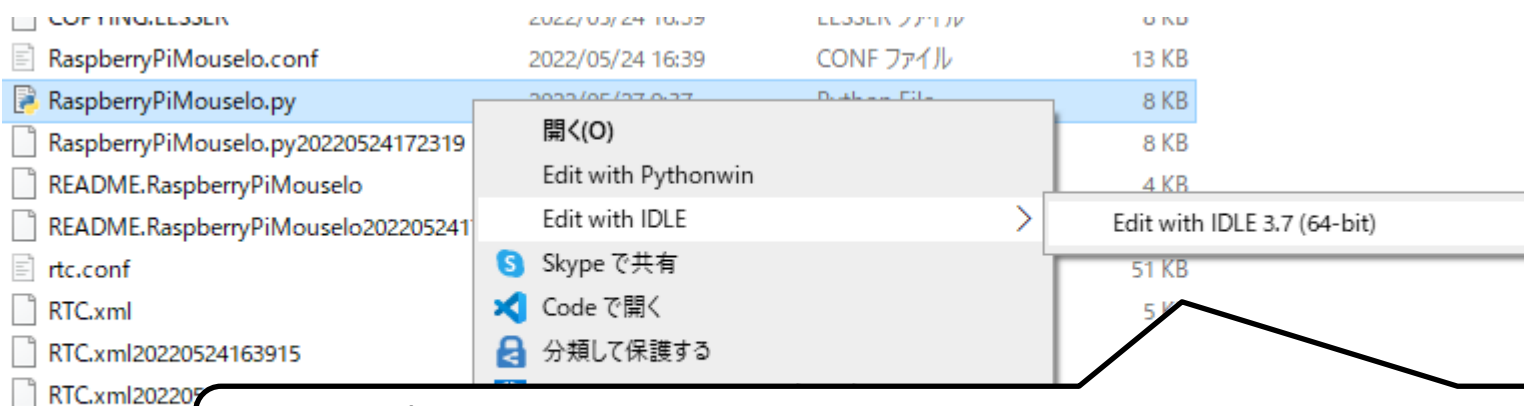
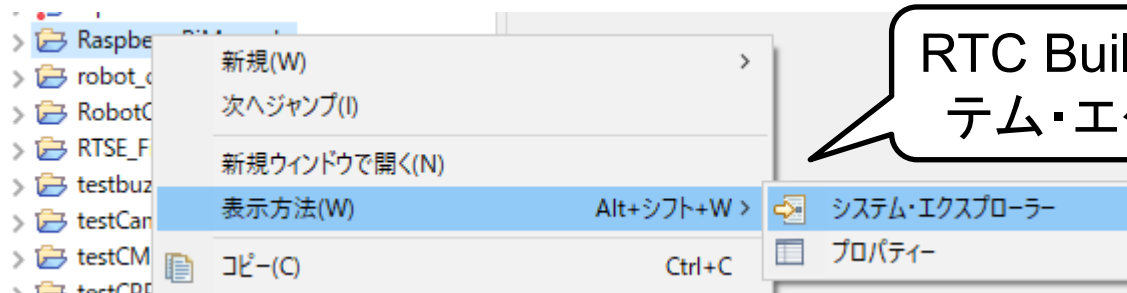
# RaspberryPiMouseLo.pyの編集

- RaspberryPiMouseLo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



# RaspberryPiMouseilo.pyの編集

- RaspberryPiMouseilo.pyをIDLEで編集する



# RaspberryPiMouselo.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
```

Bodyオブジェクトから  
「RIGHT\_WHEEL」、  
「LEFT\_WHEEL」のリンクを取得する

※インデントに注意

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

※インデントに注意

```
def outputToSimulator(self):  
    pass
```

outputToSimulator関数は、  
今回は何の処理もしない

RaspberryPiMouselo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)

# RaspberryPiMouseIo.pyの編集

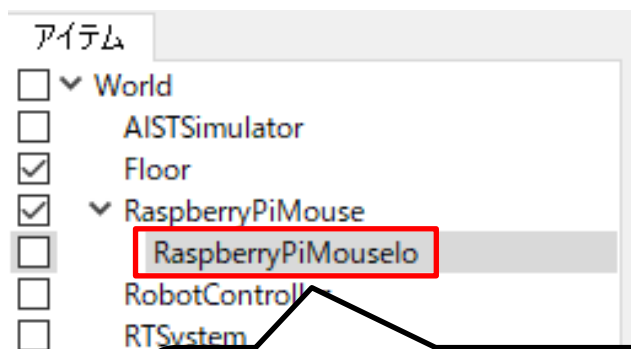
※インデントに注意

```
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

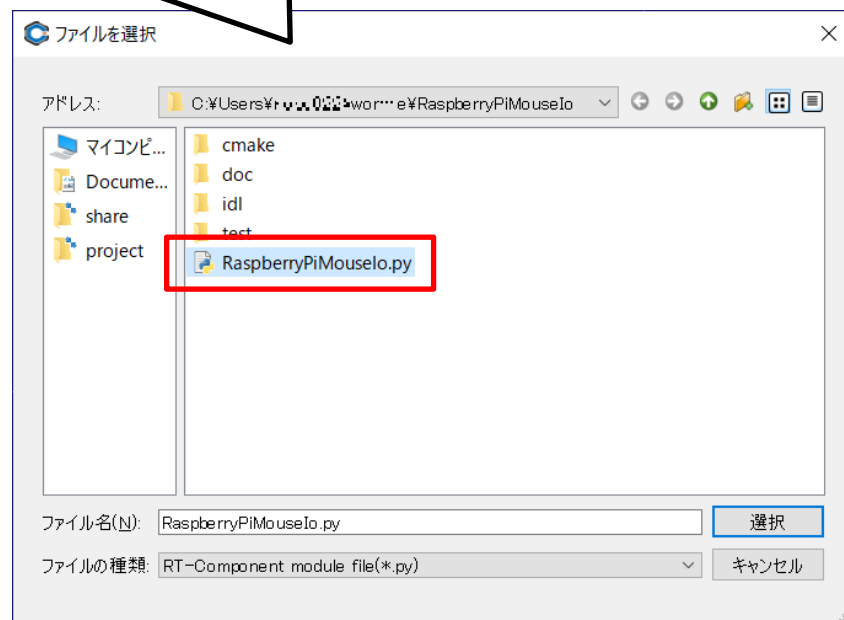
「RIGHT\_WHEEL」、「LEFT\_WHEEL」のリンクオブジェクトの「dq」にJoint  
の回転速度を設定する

# RaspberryPiMouseIoコンポーネントの設定

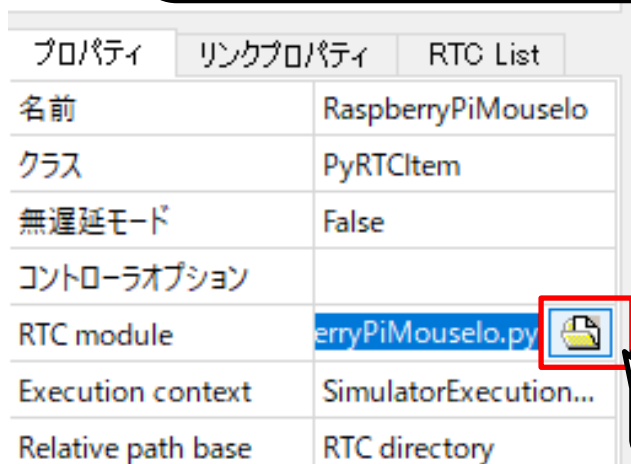


1. ChoreonoidのアイテムビューでRaspberryPiMouseIoを選択する

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



※ RaspberryPiMouseIo.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



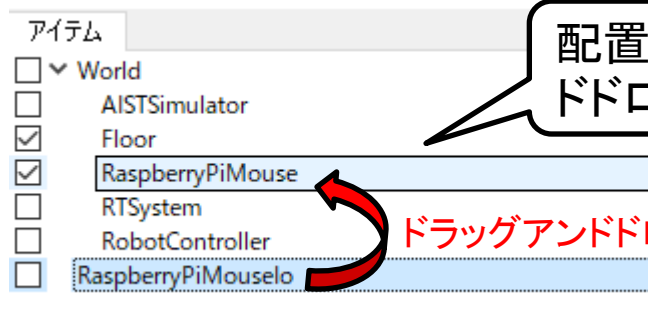
2. RTC moduleを設定する

## 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
- 「RaspberryPiMouselo」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
- ☐ RaspberryPiMouselo
- ☐ RobotController
- ☐ RTSystem

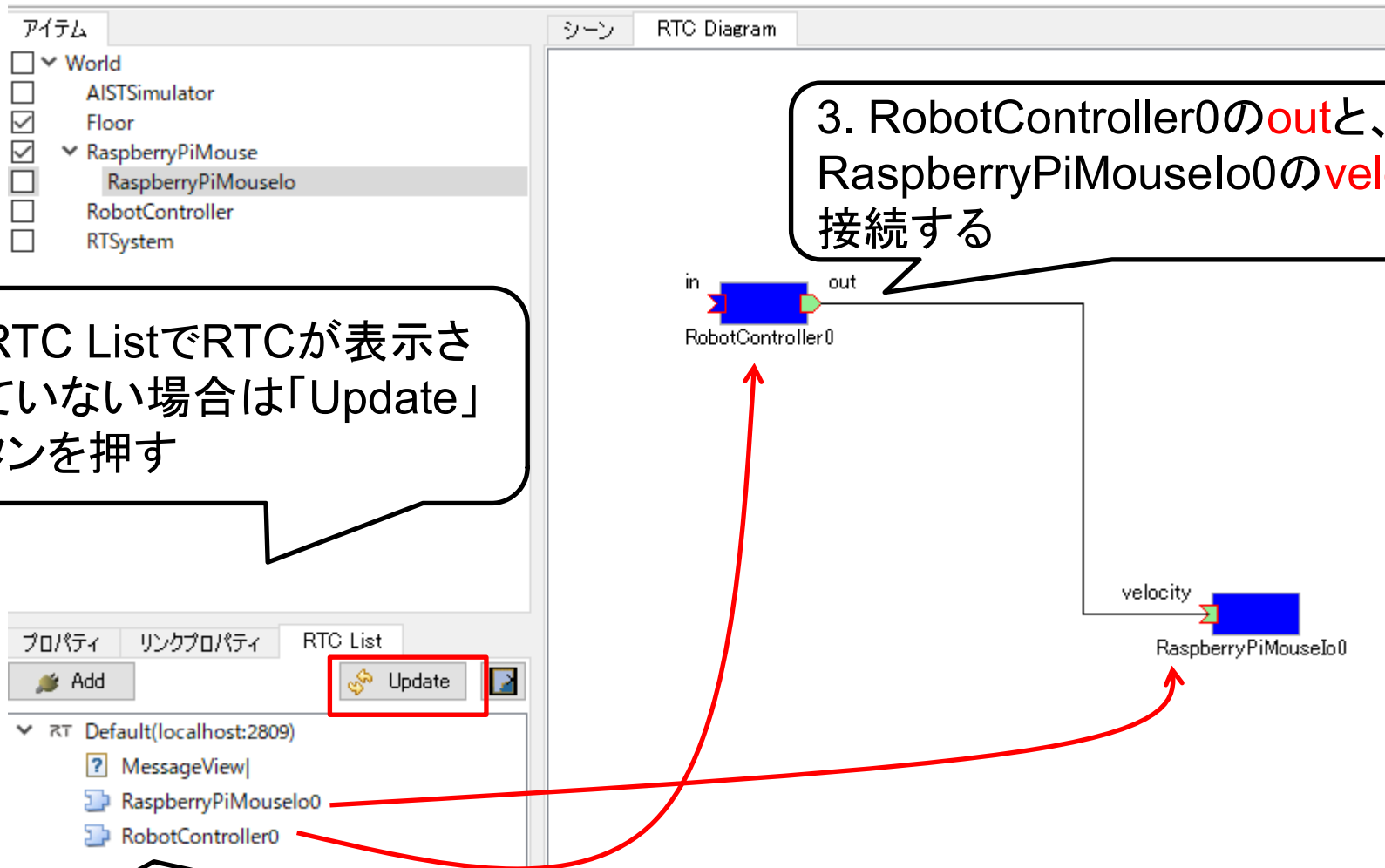
- World
- |- AISTSimulator
  - |- Floor
  - |- RaspberryPiMouse
    - |- RaspberryPiMouselo
  - |- RobotController
  - |- RTSystem



配置が違う場合はドラッグアンドドロップすれば変更できる。

ドラッグアンドドロップ

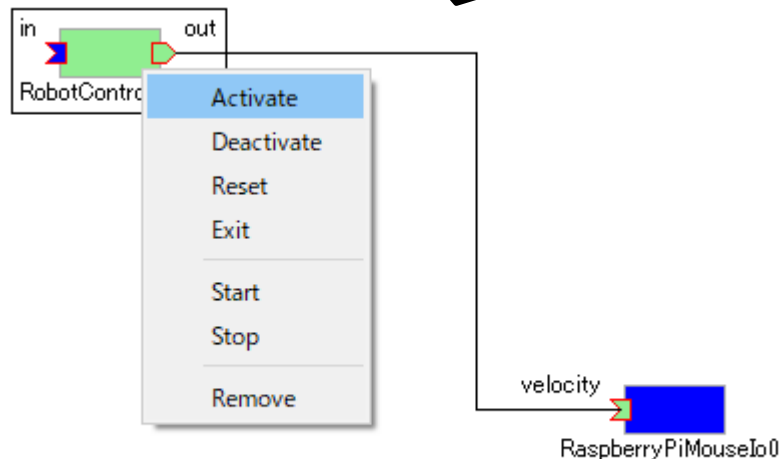
# ポート接続



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。



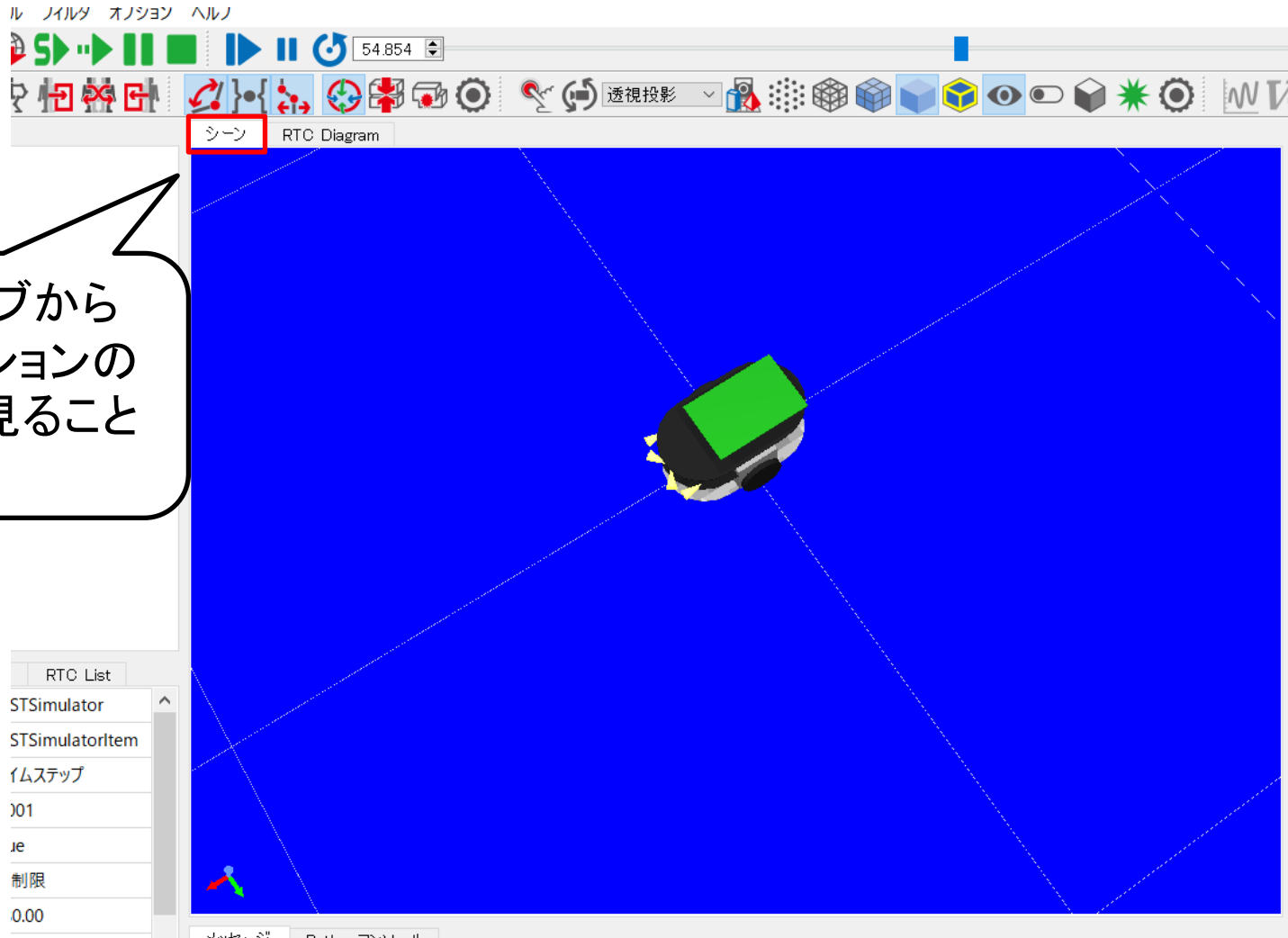


# シミュレーション開始

The screenshot displays the RT Middleware simulation environment. The top toolbar contains various icons for file operations, simulation control, and visualization. The 'Start' button, represented by a green play icon, is highlighted with a red square. A callout box points to this button with the text: 「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. Below the toolbar, the 'RTC Diagram' tab is active, showing a block diagram. The diagram includes a block labeled 'RobotController0' with 'in' and 'out' ports. A line connects the 'out' port of 'RobotController0' to the 'velocity' input of a block labeled 'RaspberryPiMouseIo0'. The bottom panel shows tabs for 'プロパティ' (Properties), 'リンクプロパティ' (Link Properties), and 'RTC List', with 'Add' and 'Update' buttons.

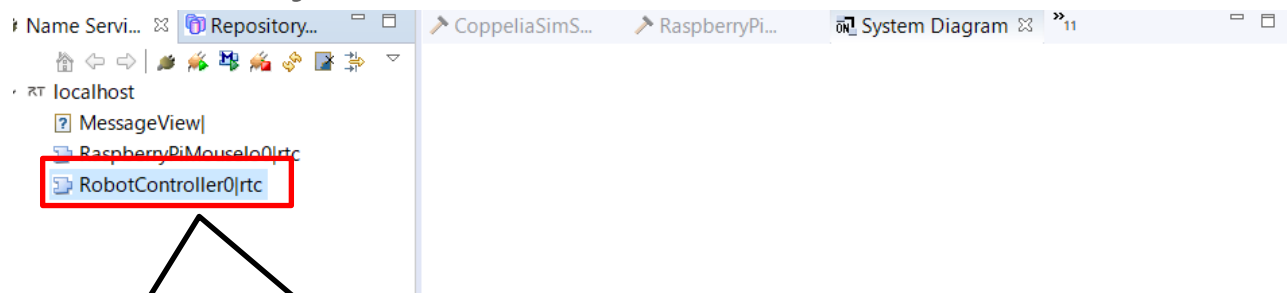
「初期位置からのシミュレーション開始」  
ボタンを押すとシミュレーションを開始する

# シミュレーション開始



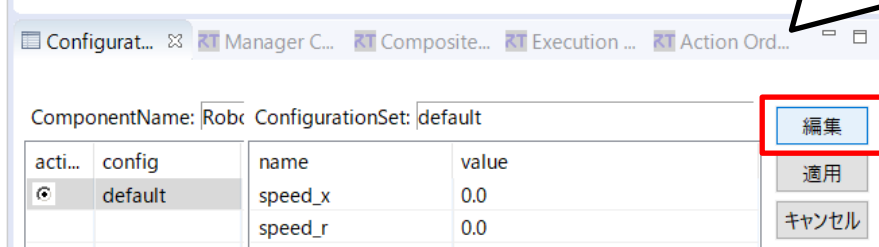
# コンフィギュレーションパラメータの編集

- 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。

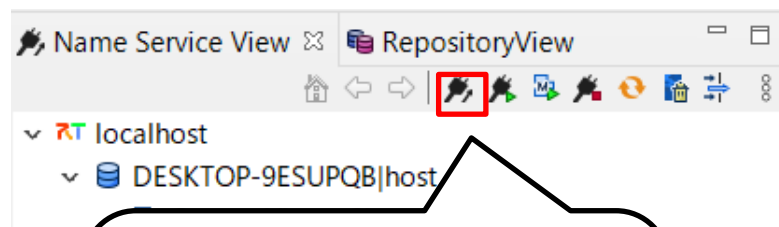
2. コンフィギュレーションビューから「編集」ボタンを押して、パラメータを変更する。



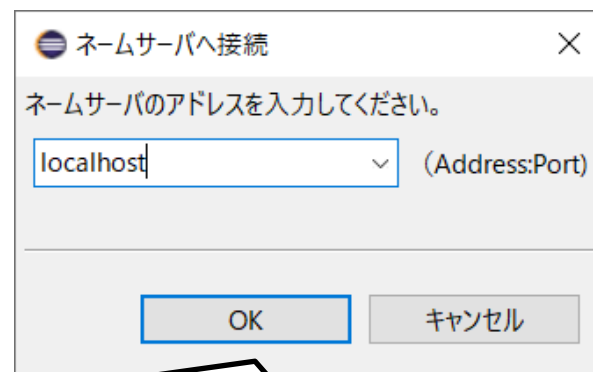
コンフィギュレーションパラメータを変更することで、Choreonoid上の Raspberry Piマウスが移動すれば課題達成です

# コンフィギュレーションパラメータの編集

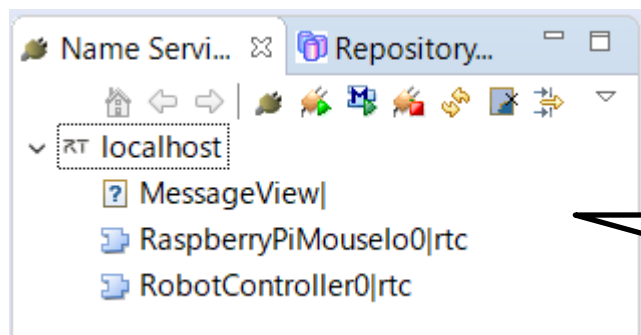
- ネームサービスビューにネームサーバが無い(localhostが非表示の場合)、以下の作業でネームサーバに接続してください



1. ネームサーバへの接続ボタンを押す



2. 「localhost」と入力してOKをクリックする



3. RTCが表示されたか確認する