

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>



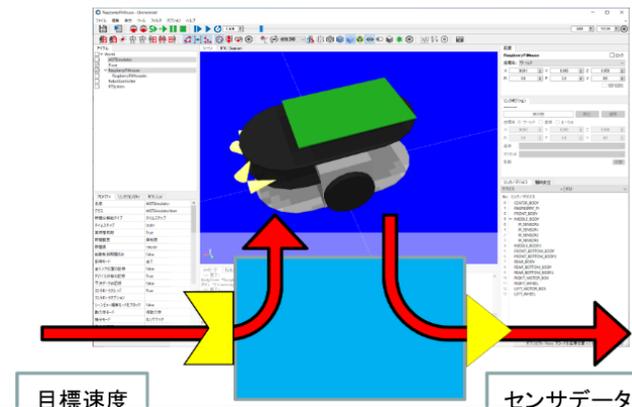
• [コンフィギュレーションパラメータの編集](#)

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。

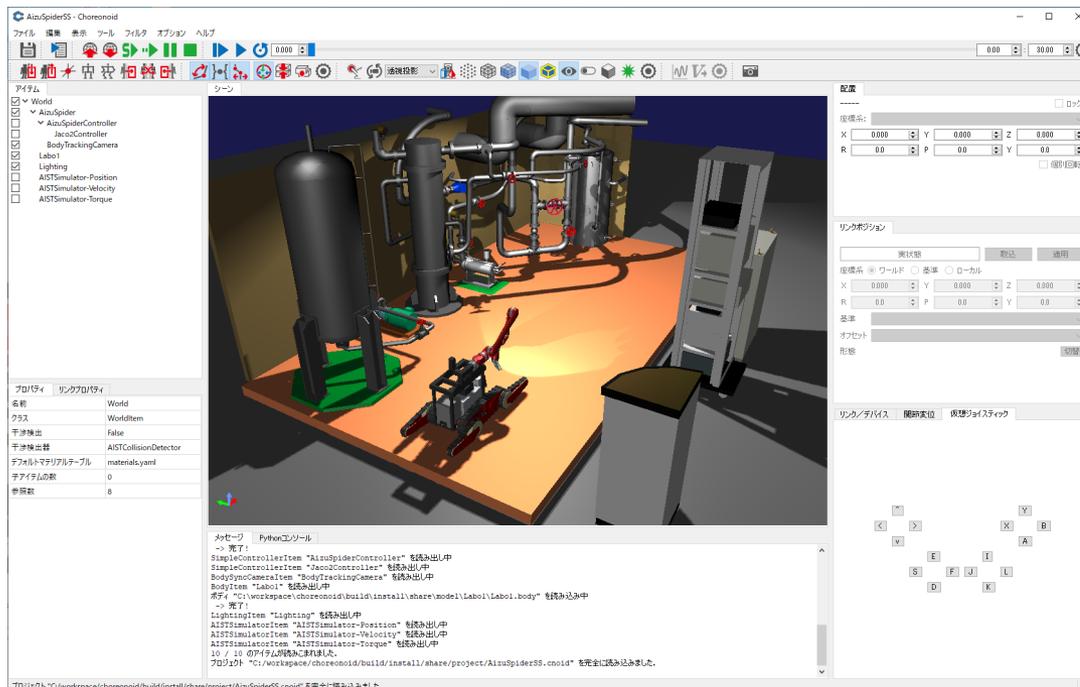
• [Choreonoid ホームページ](#)

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。



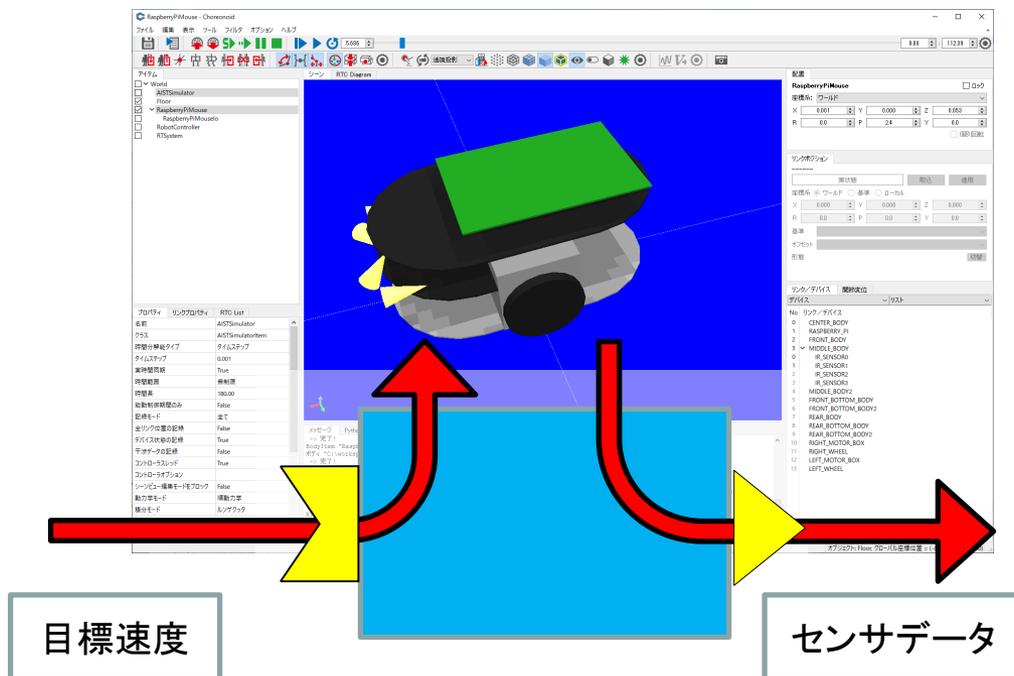
Choreonoid

- 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上と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

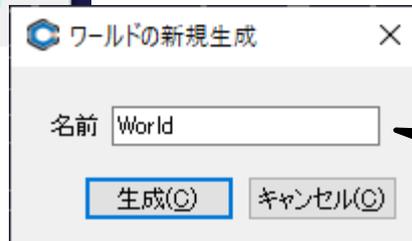
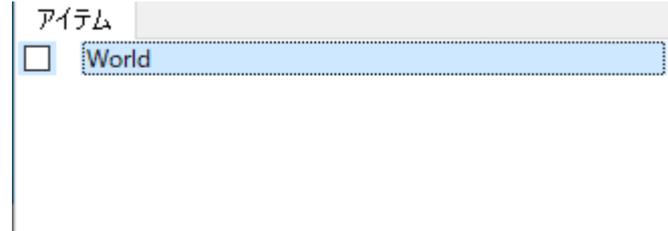
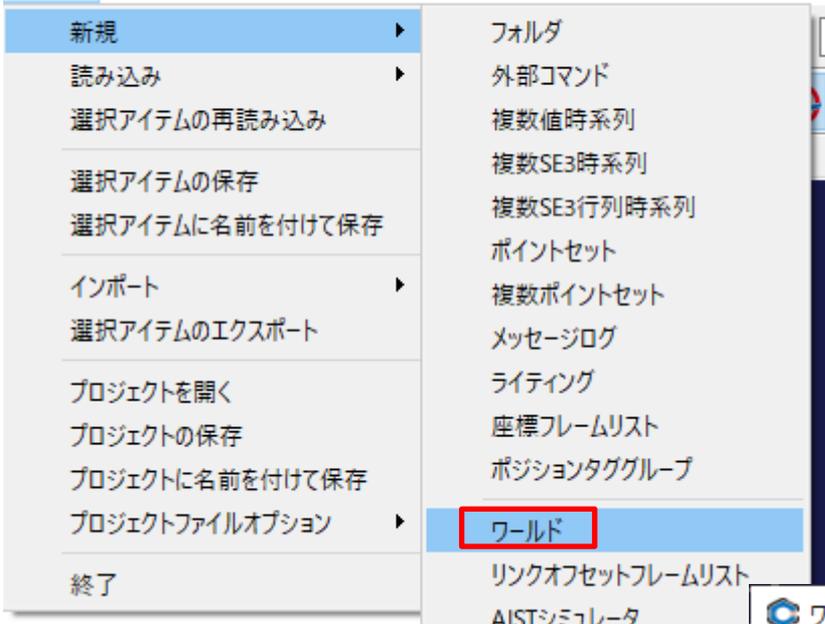
- Choreonoid上でシミュレーションの実行に必要なアイテムを追加することで、環境を構築する。
- 配布資料のchoreonoidフォルダの**choreonoid.bat**を実行する
- 今回は以下のアイテムを追加する。
 - ワールドアイテム(World)
 - シミュレータ(AIST Simulator)
 - 地面(Floor)
 - Raspberry Piマウス
 - RTSystem
 - RTC(RobotController)
 - RTC(RaspberryPiMouse)

ワールド追加

- 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

Chreonoid

ファイル 編集 表示 ツール フィルタ オプション ヘルプ

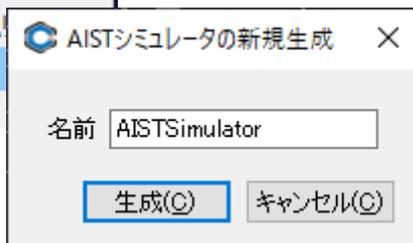
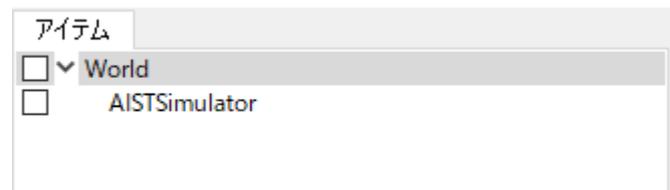
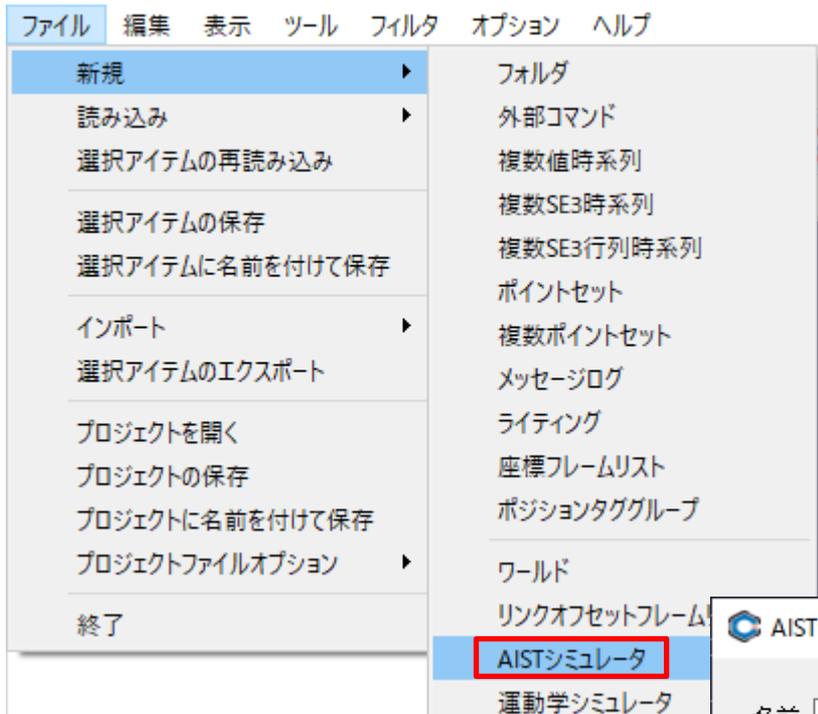


名前は変更しない

シミュレータ追加

- 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

Choreonoid

ファイル 編集 表示 ツール フィルタ オプション ヘルプ

新規 ▶

読み込み ▶

選択アイテムの再読み込み

選択アイテムの保存

選択アイテムに名前を付けて保存

インポート ▶

選択アイテムのエクスポート

プロジェクトを開く

プロジェクトの保存

プロジェクトに名前を付けて保存

プロジェクトファイルオプション ▶

終了

サブプロジェクト

シーン

複数ポイントセット

ボディ

マテリアルテーブル

ボディモーション

ワールドログ

干渉データ

RT-System

Pythonスクリプト

シミュレーション用Python

アイテム

World

AISTSimulator

Floor

ボディの読み込み

アドレス: C:\w... choreonoid\build\instal\share\model\misc

share

box.body

box1.body

box2.body

box3.body

box4.body

box5.body

BoxesOnFloor.body

bumpyfloor.body

ClosedLinkSample.body

ContinuousTrackVehicle.body

conveyor.body

crawler.body

CustomizedSpringModel.body

ellipsoid1.body

fire.body

floor.body

floor2.body

fog.wrl

FogUnderWater.wrl

fountain.body

FourWheelCar.body

grayfloor.body

HandyRockDrill.body

longtable.body

minitable.body

plane.body

pyramid.body

ragdoll.body

rain.body

rod.body

slope.body

smallfloor.body

smoke.body

snow.body

SpringModel.body

stand.body

submersible.body

suctioncup.body

Torus.body

unev

ファイル名(N):

読み込み

ファイルの種類: ボディ (*body *yaml *yml *wrl)

キャンセル

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

The screenshot illustrates the steps to add a Raspberry Pi mouse body item in the Choreonoid environment:

- File Menu:** The 'File' menu is open, and the 'ボディ' (Body) option is selected and highlighted with a red box.
- File Selection:** A dialog box titled 'ボディの読み込み' (Load Body) is shown. The address bar contains the path `C:\workspace\choreonoid\build\share\model\RaspberryPiMouse`. The 'share' folder is selected in the left pane, and the file `RaspberryPiMouse.body` is selected in the right pane.
- Item List:** The 'アイテム' (Items) panel on the right shows the hierarchy: World (expanded), AISTSimulator, Floor, and RaspberryPiMouse (checked).
- 3D Viewport:** The main 3D view shows a blue grid floor with a small green and black Raspberry Pi mouse model placed on it.

RTSystem追加

- Choreonoid上にRTシステムエディタの一部機能を使用できる

Choreonoid

ファイル 編集 表示 ツール フィルタ オプション ヘルプ

- 新規
 - フォルダ
 - 外部コマンド
 - 複数値時系列
 - 複数SE3時系列
 - 複数SE3行列時系列
 - ポイントセット
 - 複数ポイントセット
 - メッセージログ
 - ライティング
 - 座標フレームリスト
 - ポジションタグループ
- 読み込み
- 選択アイテムの再読み込み
- 選択アイテムの保存
- 選択アイテムに名前を付けて保存
- インポート
 - ワールド
 - リンクオフセットフレームリスト
 - AISTシミュレータ
 - 運動学シミュレータ
 - シンプルコントローラ
 - ボディモーションコントローラ
 - 領域侵入検出器
 - ボディ接触点ロガー
 - GLビジョンシミュレータ
 - ボディモーション
 - ワールドログファイル
 - IO接続マップ
 - センサ可視化
 - ボディ同期カメラ
 - ボディマーカ
- 選択アイテムのエクスポート
- プロジェクトを開く
- プロジェクトの保存
- プロジェクトに名前を付けて保存
- プロジェクトファイルオプション
 - BodyRTC
 - ControllerRTC
 - RTC
 - BodyRTC
 - RTSystem**
 - BodyStateSubscriberRTC
- 終了

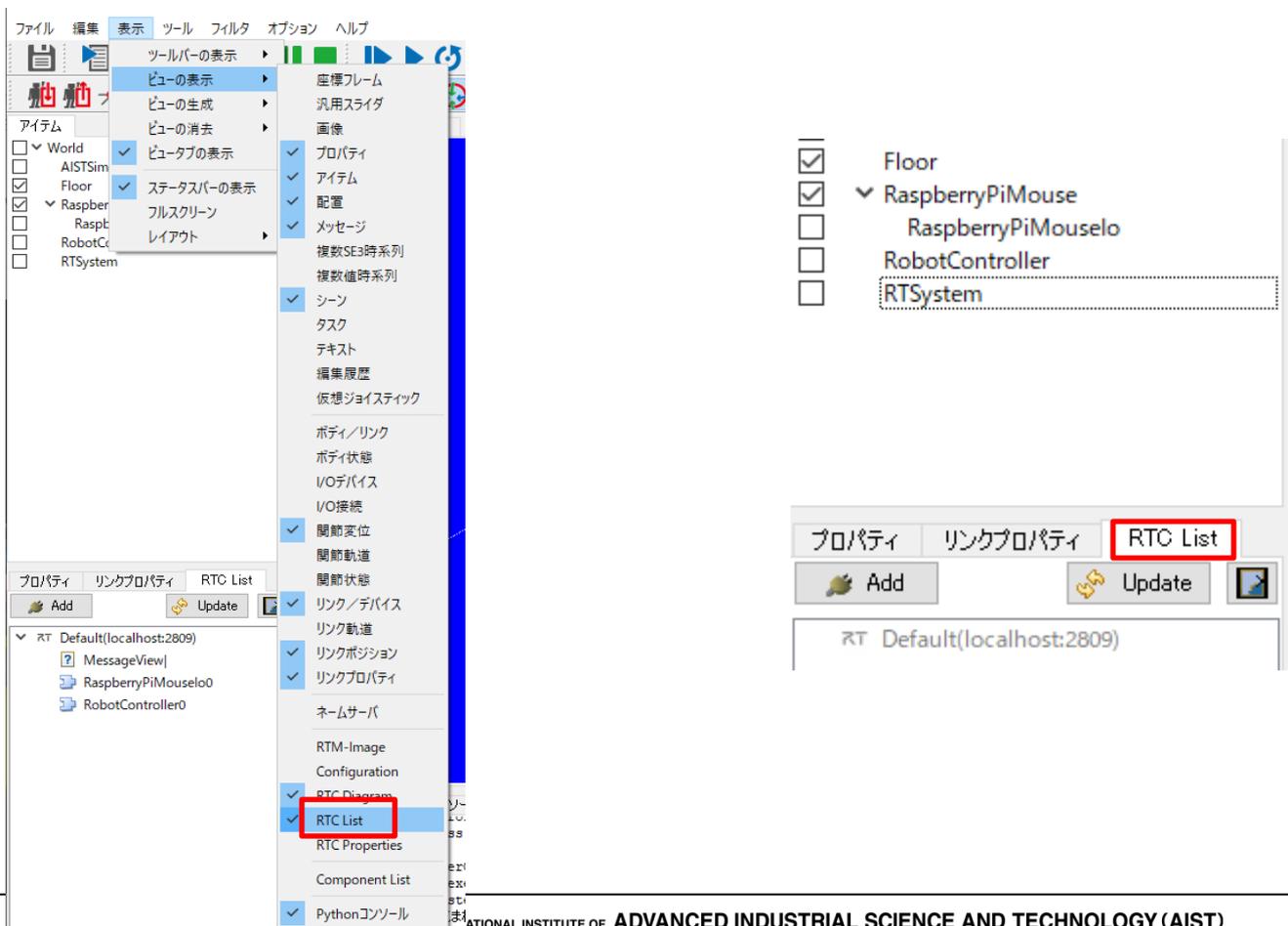
プロパティ	リンクプロパティ
名前	Raspber
クラス	BodyIter
モデル名	Raspber
リンク数	14
関節数	2
デバイス数	4
ルートリンク	CENTER

アイテム

- World
- AISTSimulator
- Floor
- RaspberryPiMouse
- RTSystem

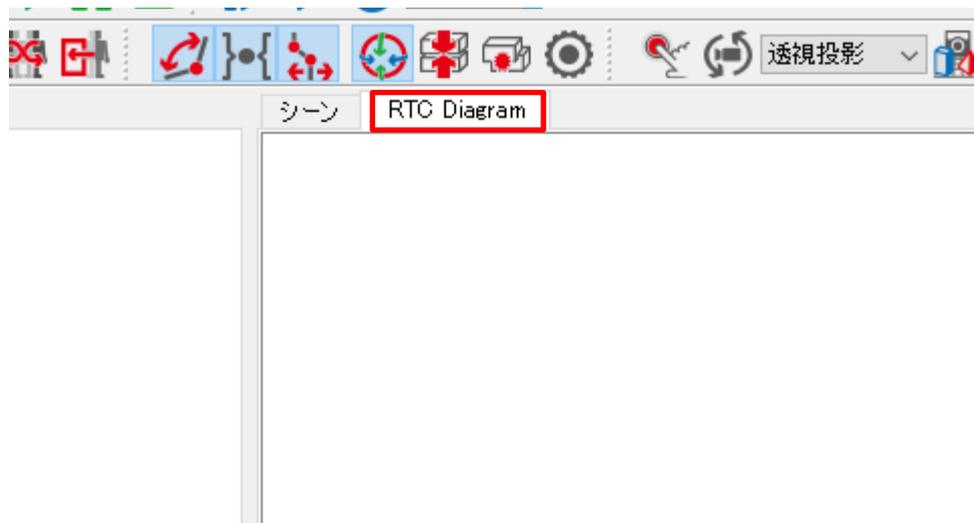
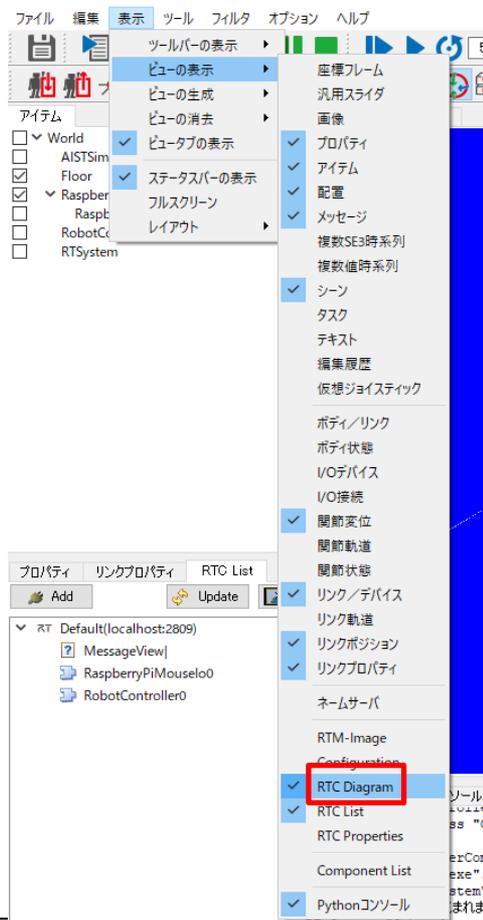
ネームサーバー、システムエディタ表示

- 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

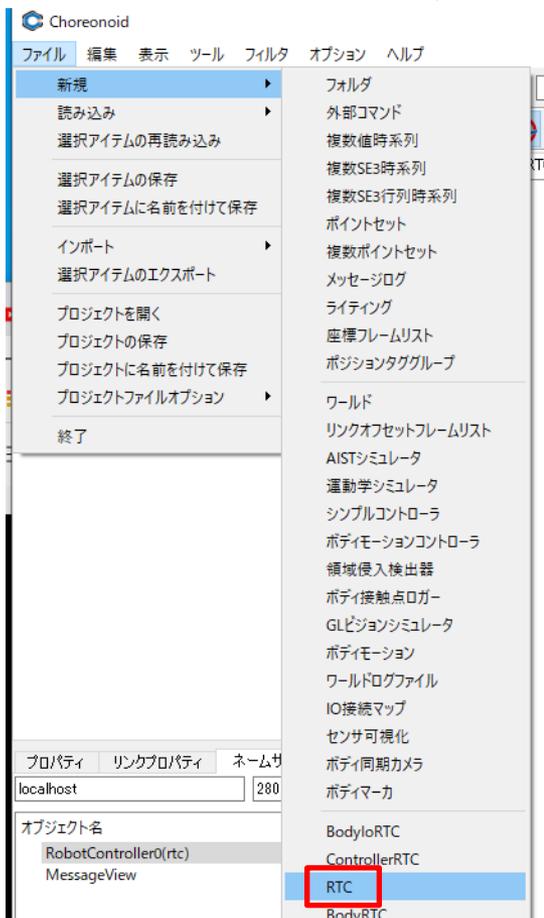
- 初期状態ではネームサーバー、システムエディタが非表示のため設定を変更する
 - 表示 → ビューの表示 → **RTC Diagram**



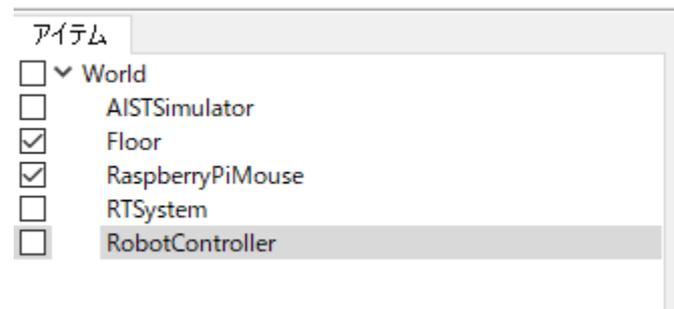
RobotControllerコンポーネント追加

- RobotControllerコンポーネントをChoreonoidが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

アイテム

- World
- AISTSimulator
- Floor
- RaspberryPiMouse
 - RaspberryPiMouseelo
 - RobotController**
 - RTSystem

プロパティ リンクプロパティ RTC List

名前	RobotController
クラス	RTCItem
RTC module	ntrollerComp.exe
Base directory	RTC directory
Execution context	PeriodicExecutionC...

アイテムから「RobotController」を選択後、下の「プロパティ」で「RTC Module」を設定する

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、
 「ファイルの種類」を「すべてのファイル(*)」に変更する

ファイルを選択

アドレス: C:\workspace\RobotController\build\src\Release

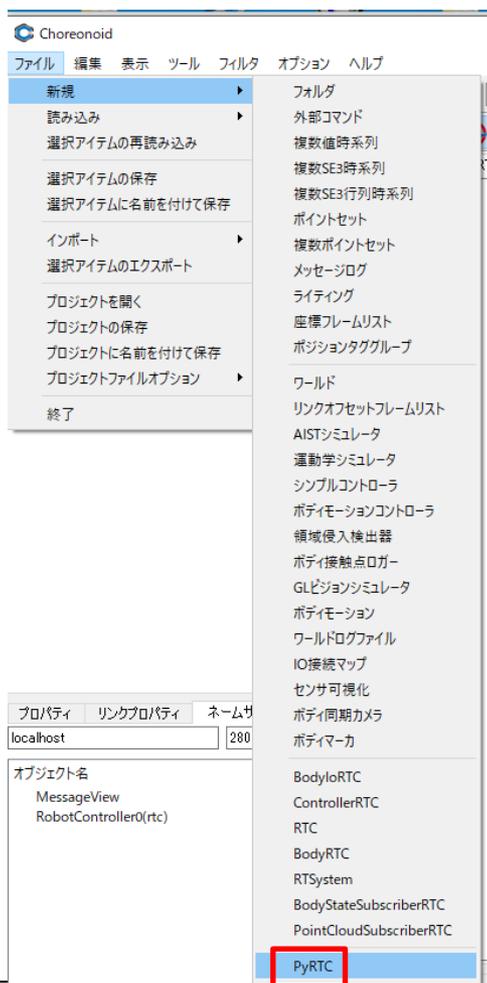
- マイコンピュータ
- Documents
- share
- RobotController.dll
- RobotController.exp
- RobotController.lib
- RobotControllerComp.exe**
- RobotControllerComp.exp
- RobotControllerComp.lib

ファイル名(N): RobotControllerComp.exe 選択

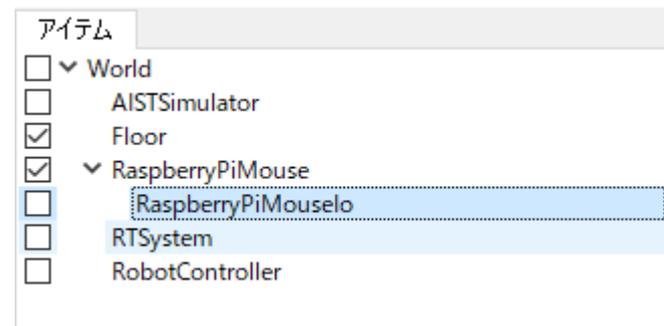
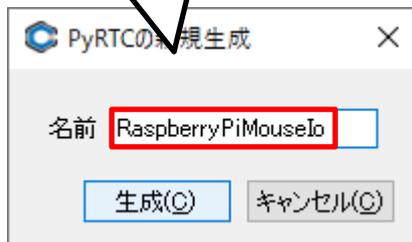
ファイルの種類: **すべてのファイル(*)** キャンセル

RaspberryPiMouseの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「RaspberryPiMouse0」に変更する



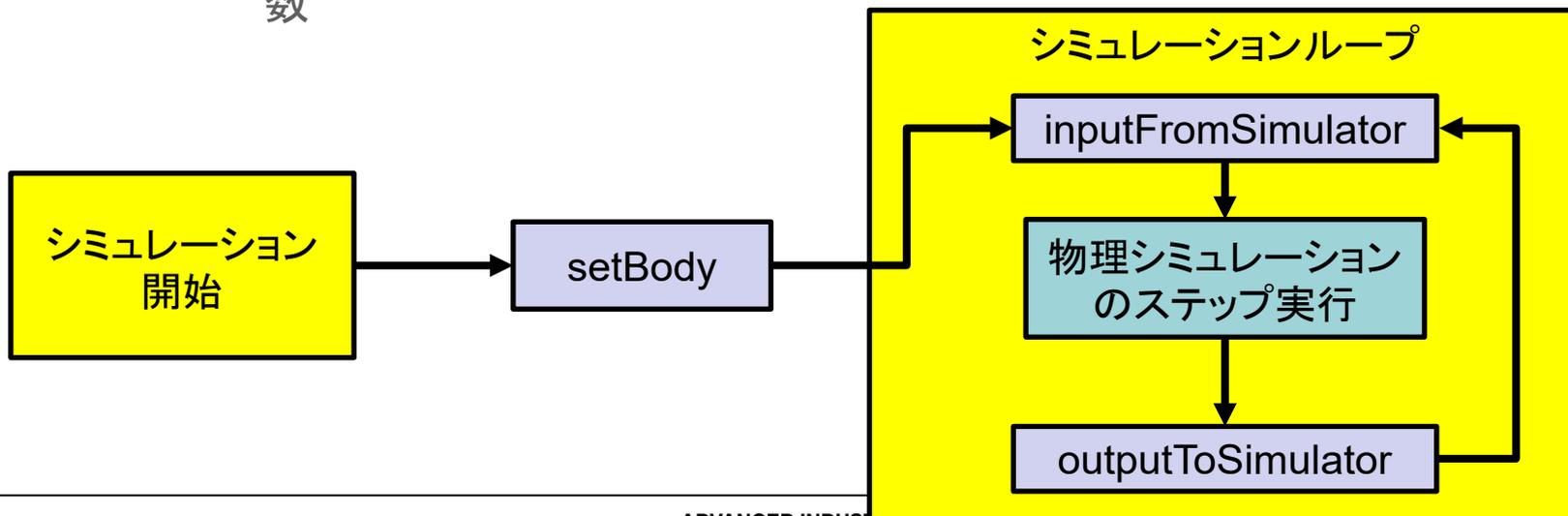
RaspberryPiMouseIoコンポーネント作成

- RTC Builderで以下のRTコンポーネントを作成する

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	

RaspberryPiMouse0.pyの編集

- RaspberryPiMouse0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



RaspberryPiMouse0.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")
```

※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

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

※インデントに注意

outputToSimulator関数は、
今回は何の処理もしない

RaspberryPiMouse0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)

RaspberryPiMouseIlo.pyの編集

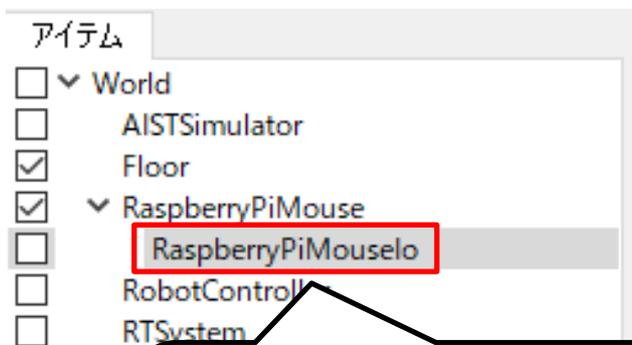
```
def inputFromSimulator(self):  
    ※インデントに注意 if self._velocityIn.isNew():  
        data = self._velocityIn.read()  
        vx = data.data.vx  
        va = data.data.va  
  
        wheel_distance = 0.0425  
        wheel_radius = 0.04  
        rms = (vx + va*wheel_distance)/wheel_radius  
        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

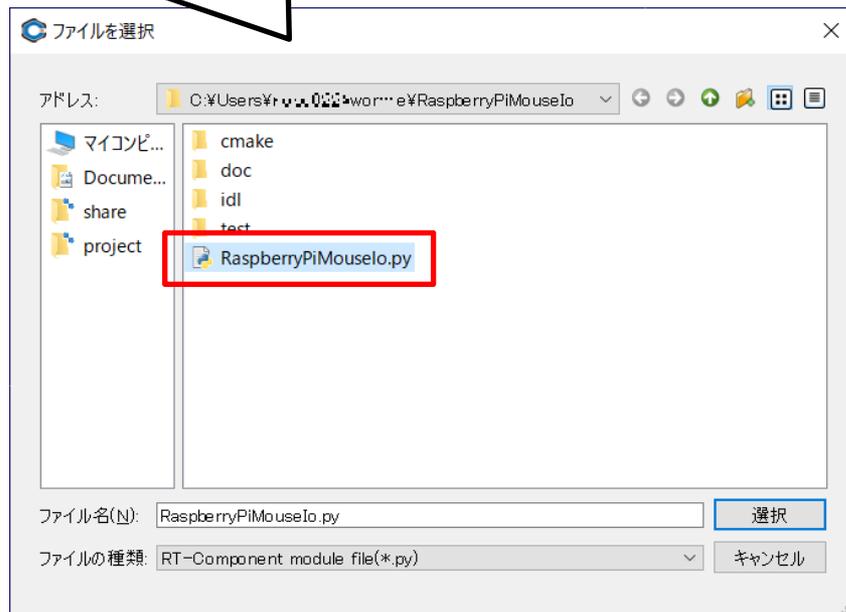
「RIGHT_WHEEL」、「LEFT_WHEEL」のリンクオブジェクトの「dq」にJoint
の回転速度を設定する

RaspberryPiMouseIoコンポーネントの設定

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



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



プロパティ	リンクプロパティ	RTC List
名前	RaspberryPiMouseIo	
クラス	PyRTCItem	
無遅延モード	False	
コントローラオプション		
RTC module	RaspberryPiMouseIo.py 	
Execution context	SimulatorExecution...	
Relative path base	RTC directory	

2. RTC moduleを設定する

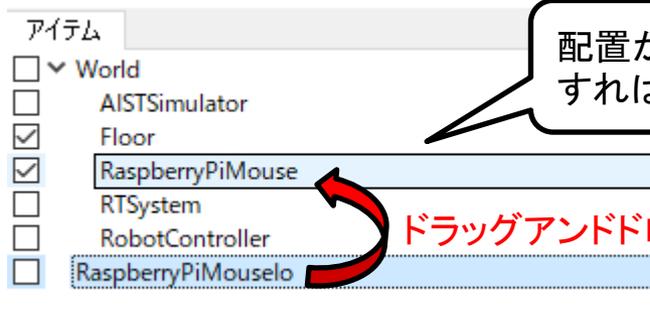
※ RaspberryPiMouseIo.pyを更新した場合は、再度この作業を行う事で再読み込みする

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
- 「RaspberryPiMouseelo」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
- RaspberryPiMouseelo
- RobotController
- RTSystem

- World
- |- AISTSimulator
- |- Floor
- |- RaspberryPiMouse
- |- RaspberryPiMouseelo
- |- 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

The screenshot shows the RT Middleware interface. On the left, the 'アイテム' (Items) list includes World, AISTSimulator, Floor, RaspberryPiMouse (expanded to show RaspberryPiMouseIo0), RobotController, and RTSystem. Below this is the 'RTC List' panel with an 'Update' button highlighted in red. The main 'RTC Diagram' shows a blue block labeled 'RobotController0' with 'in' and 'out' ports. A red arrow connects the 'out' port to the 'velocity' port of a blue block labeled 'RaspberryPiMouseIo0'. Red arrows also point from the 'Update' button to the 'RobotController0' and 'RaspberryPiMouseIo0' blocks in the diagram.

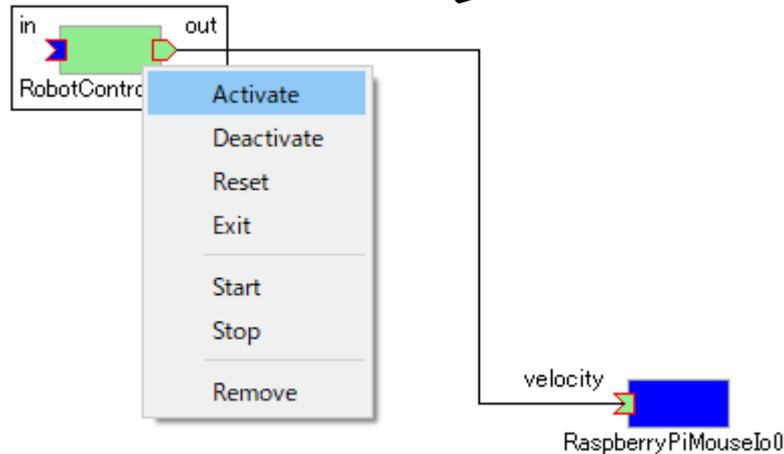
1. RTC ListでRTCが表示されていない場合は「Update」ボタンを押す

3. RobotController0のoutと、RaspberryPiMouseIo0のvelocityを接続する

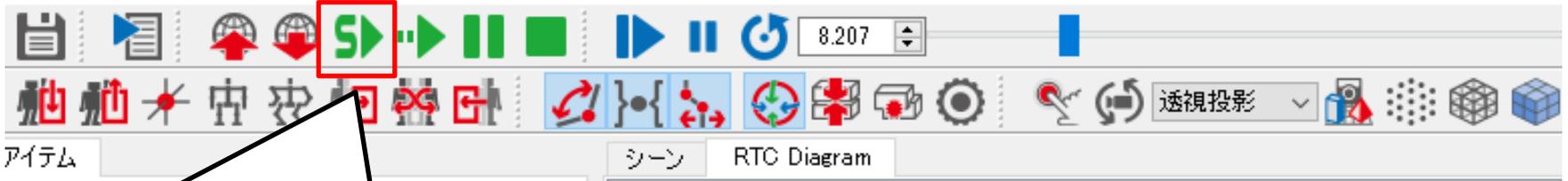
2. RTC DiagramにRTCをドラッグアンドドロップする

RobotControllerコンポーネントのアクティブ化

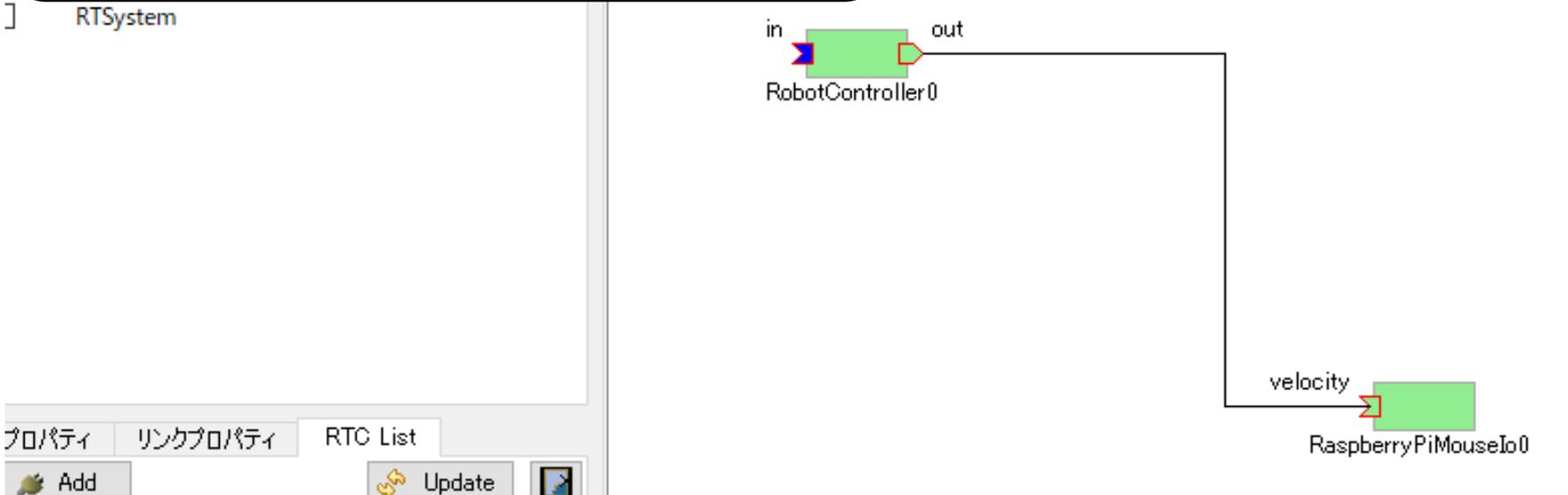
exeファイルを指定したRTCはシミュレーション開始時に自動でアクティブ化されないので手で操作する。
RobotController0を右クリックして「Activate」を選択する。



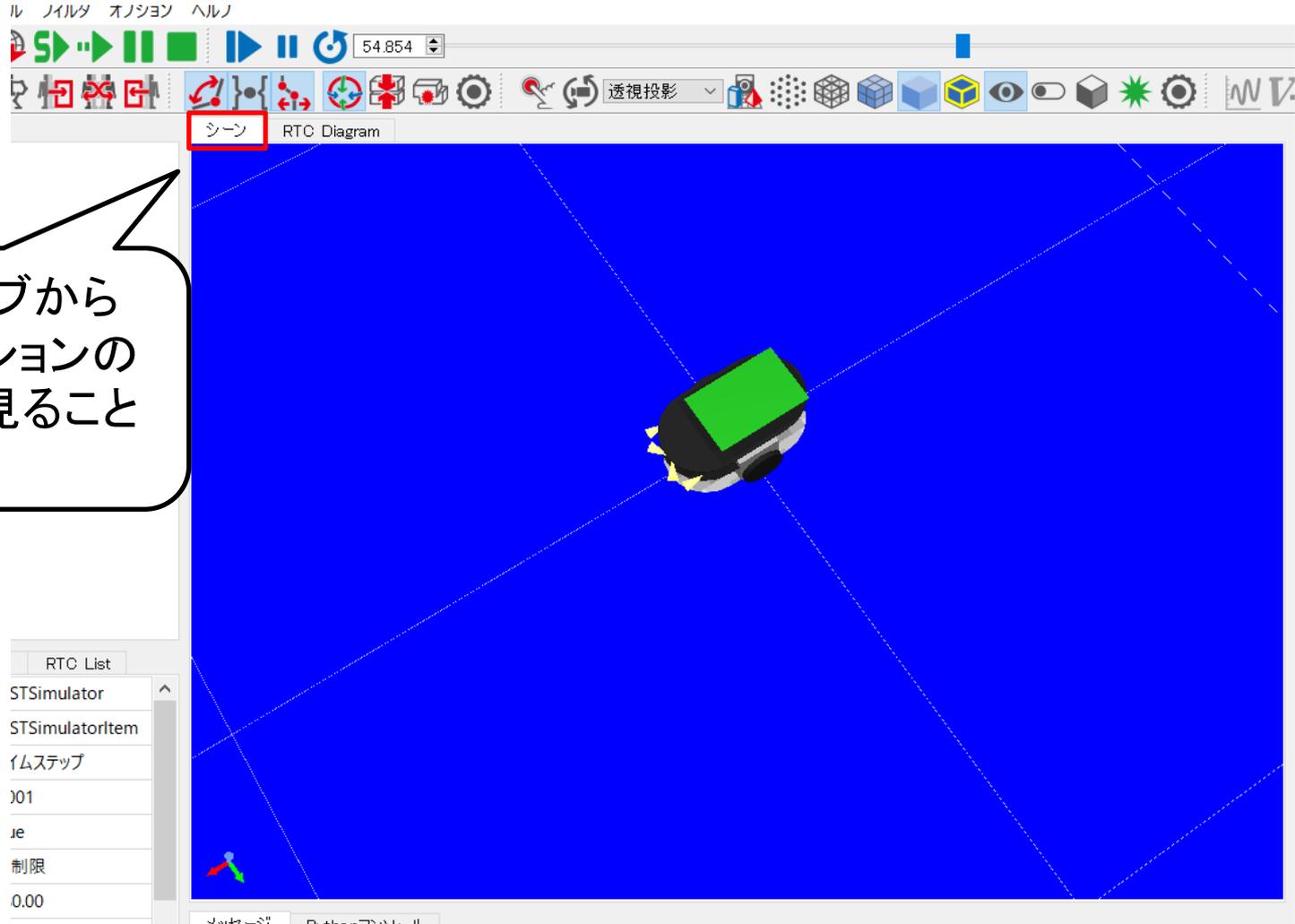
シミュレーション開始



「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



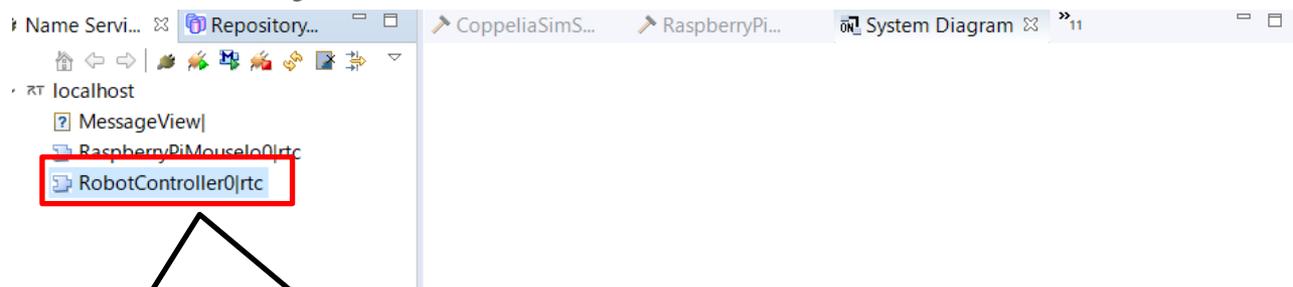
シミュレーション開始



「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

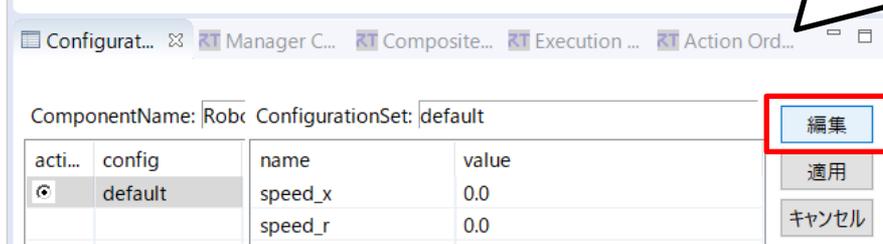
コンフィギュレーションパラメータの編集

- 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

