箱庭とUnityですすめる オレオレロボットの動かしかた

森 崇
(永和システムマネジメント)

TOPPERS
Toyohashi OPen Platform for Embedded Real-time Systems

5月
8

箱庭チュートリアル会 #4 箱庭とUnityですすめるオレオレロボットの動かしかた

箱庭で１からロボットを動かしてみよう！

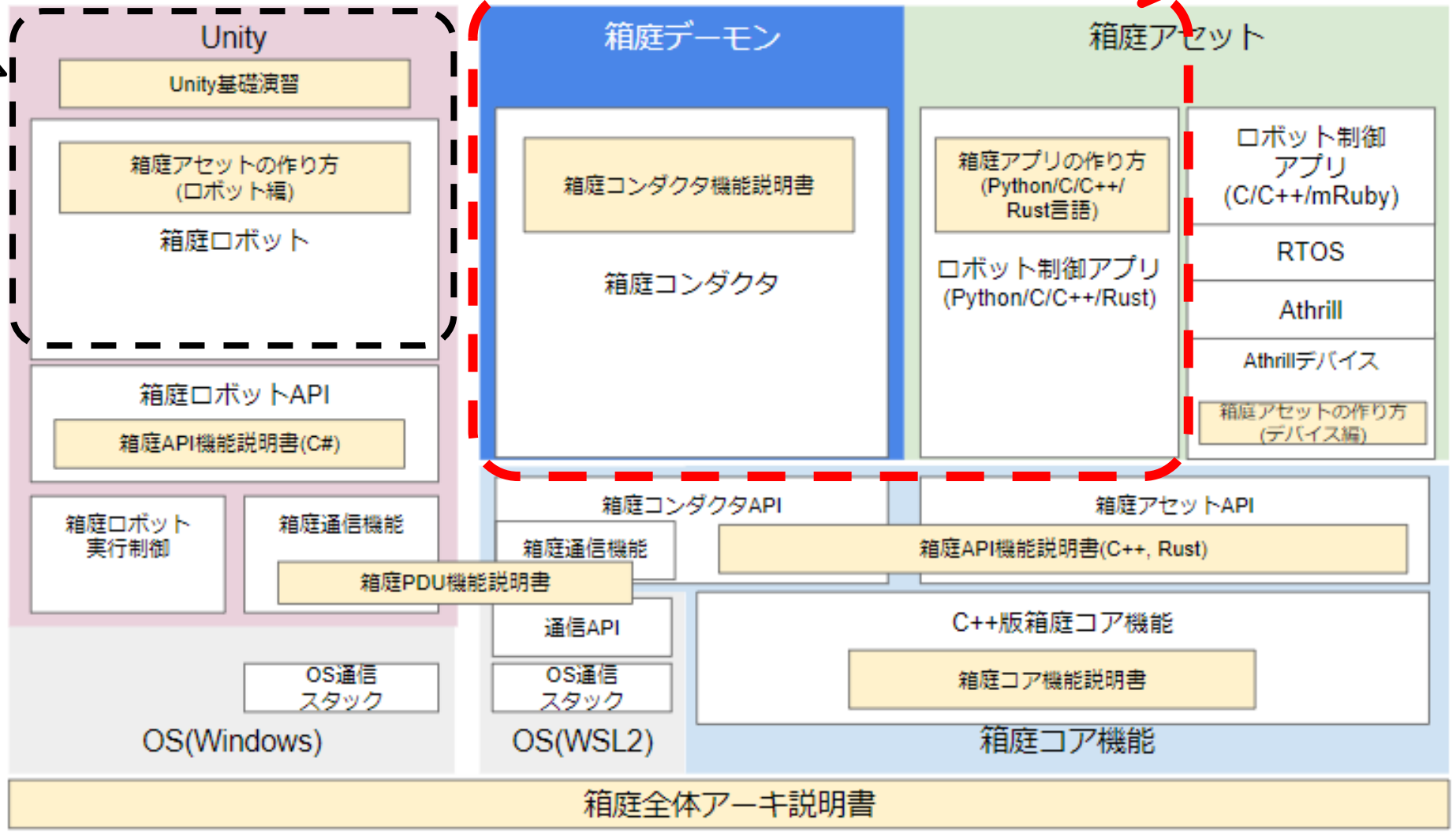
1. 前回の続き
    1. 箱庭ロボットの作り方
2. 箱庭ロボットの動かし方
    1. 箱庭強化学習をもう一度
    2. 組み立てたロボットを動かそう！
3. 箱庭のインテグ方法
    1. 箱庭の内部アーキテクチャ
    2. 箱庭のリポジトリ構成とビルド方法

TOPPERS
Toyohashi OPen Platform for Embedded Real-time Systems

現状の箱庭の全体像（Windows版）

今回はココ
(5/8)

前回
(4/17)



箱庭ロボットの動かし方

- まずは、箱庭環境を準備頂く必要があります。
- そのうえで、今の箱庭体験頂こうと思います。

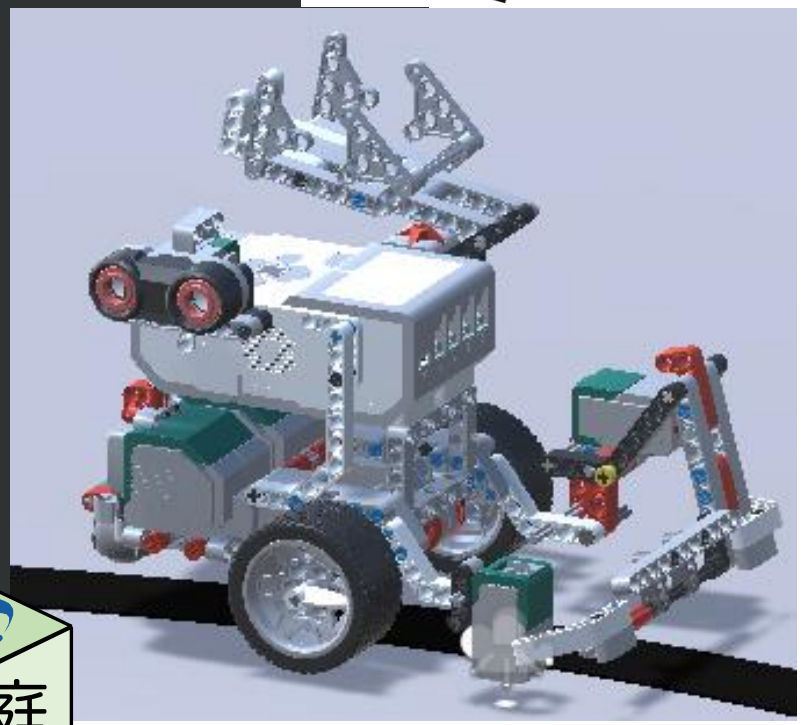
事前に準備頂きたいこと

- 以下のQiita記事を参照していただき、インストール／動作確認を事前にお願いいたします
  - Windows＋Unity＋Pythonで箱庭ロボットを強化学習できるようにするための手順書
  - Mac＋Unity＋Pythonで箱庭ロボットを強化学習できるようにするための手順書
  - Ubuntuでも箱庭で機械学習するやつを動かそう

箱庭強化学習では、なにができる？

- UnityとPython使って、ロボットの強化学習を手軽にためせます！

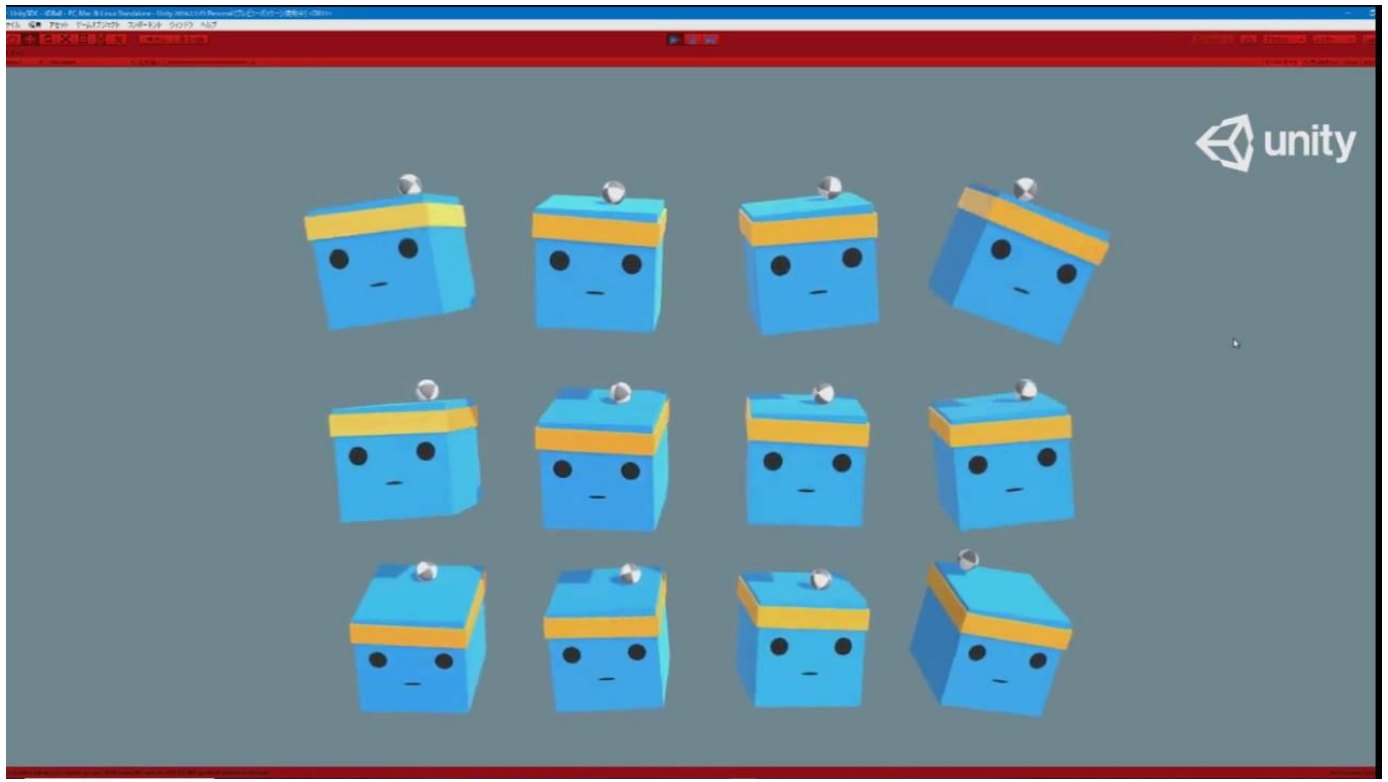

```
 ai model
l = qtable_model2.get_model(env.robo().num_states(), env.robo().num_actions())
l.load('./dev/ai/qtable_model.csv')

simulation
 = env.robo()
episode in range(100):
tal_time = 0
ne = False
state = 0
total_reward = 0
while not done and total_time < 4000:
    action = model.get_action(state)
    next_state, reward, done, _ = env.step(action)
    total_reward = total_reward + reward
    model.learn(state, action, reward, next_state)
    state = next_state
    total_time = total_time + 1
env.reset()
model.save('./dev/ai/qtable_model.csv')
```



TOPPERS
Toyohashi OPen Platform for Embedded Real-time Systems

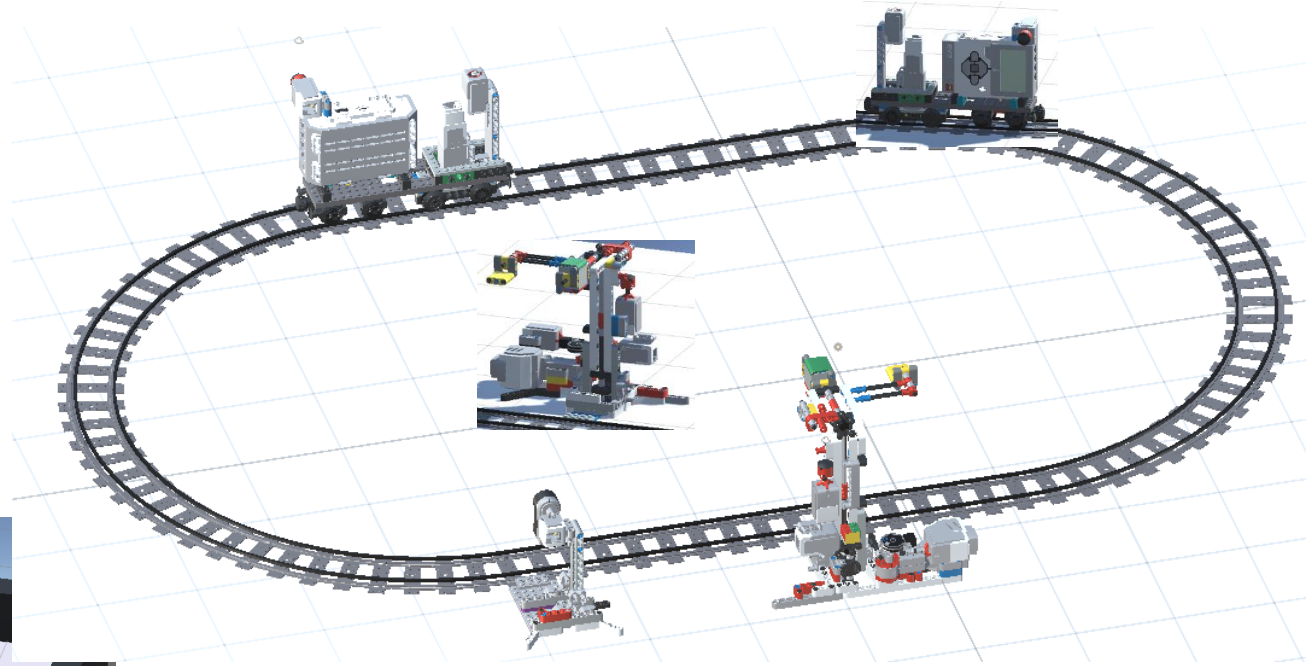
Unity MLAgents との違いはなに？

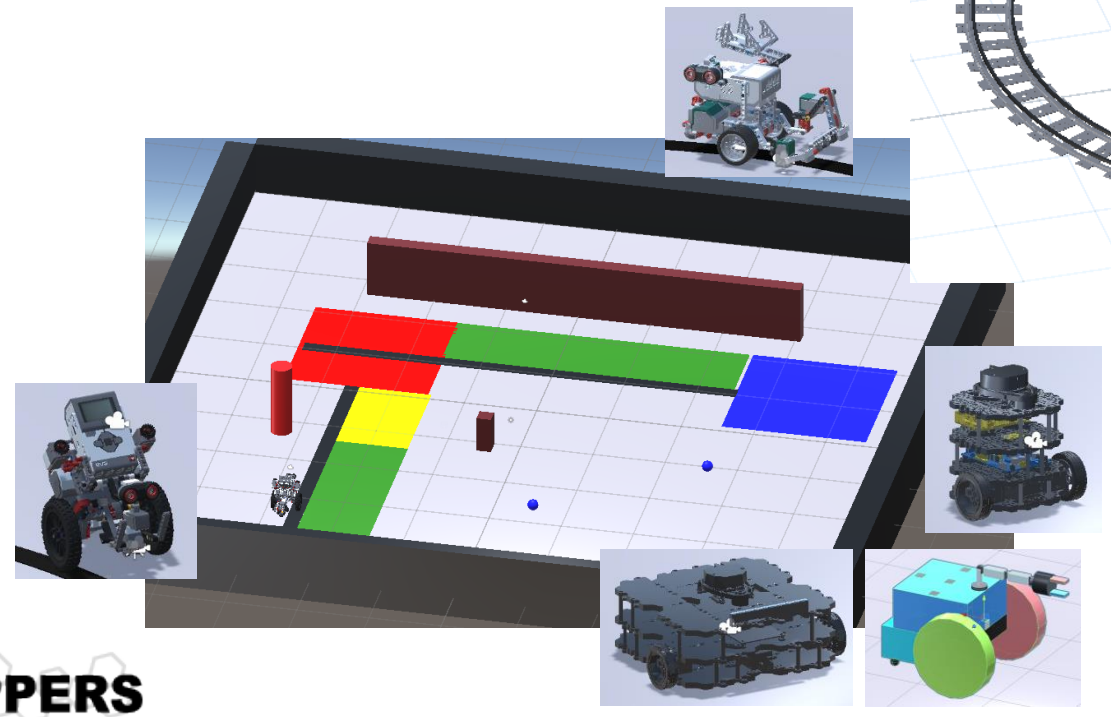
- 似ている点
  - UnityとPython使って、Unity上のゲームオブジェクトの機械学習が手軽にできる



Unity MLAgents との違いはなに？

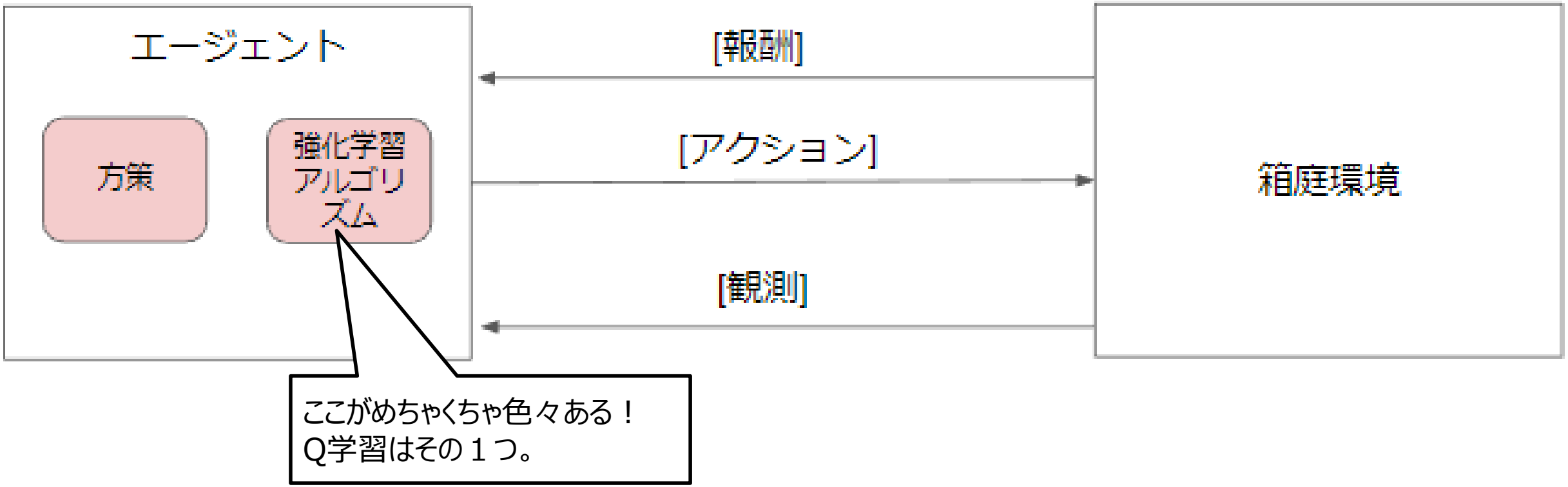
- 箱庭のうれしさ
  - 箱庭のロボットを試せる！
  - 箱庭のロボット環境も作れる！
  - ホンモノでも試せる！



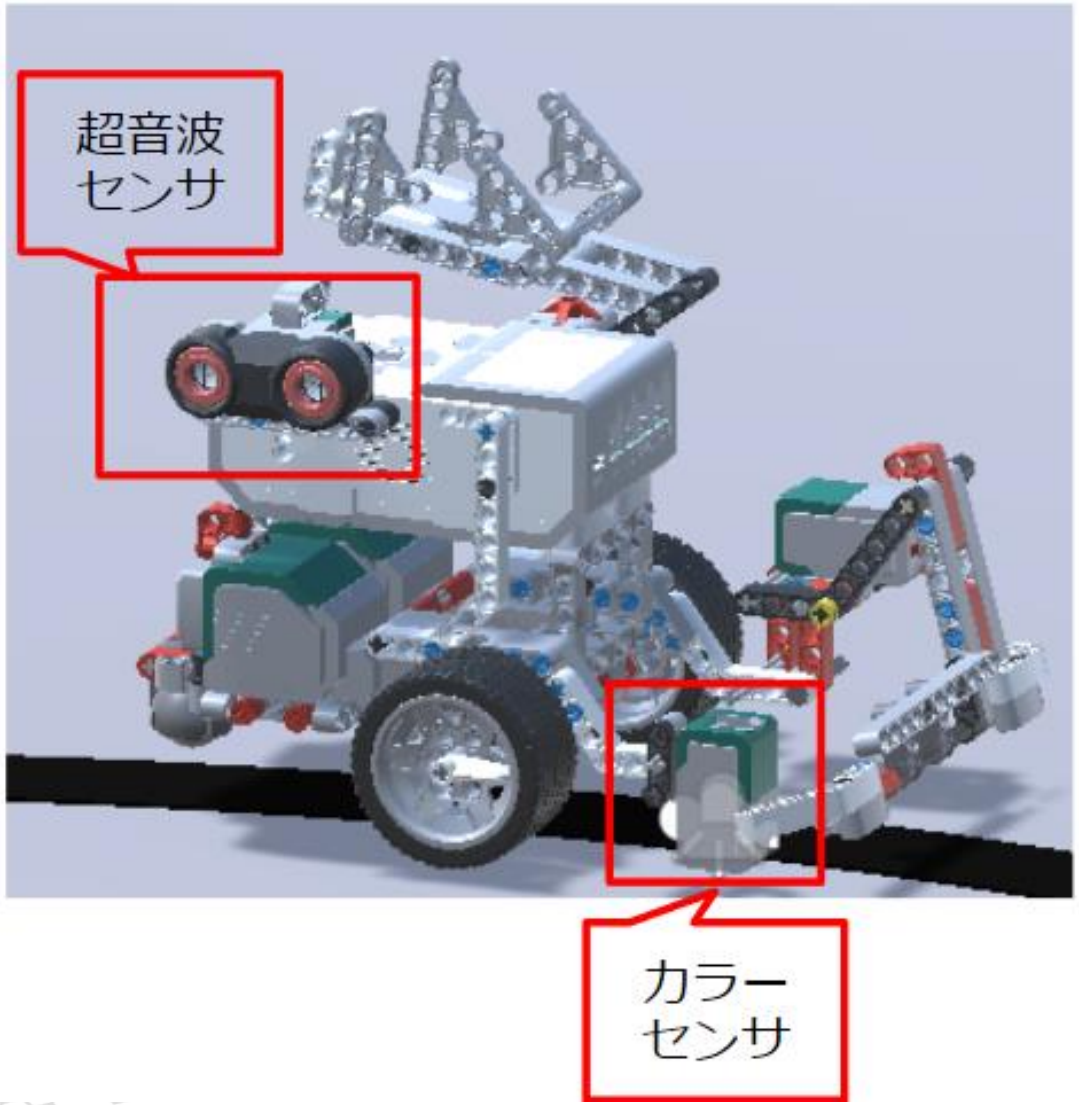


強化学習のアーキテクチャ

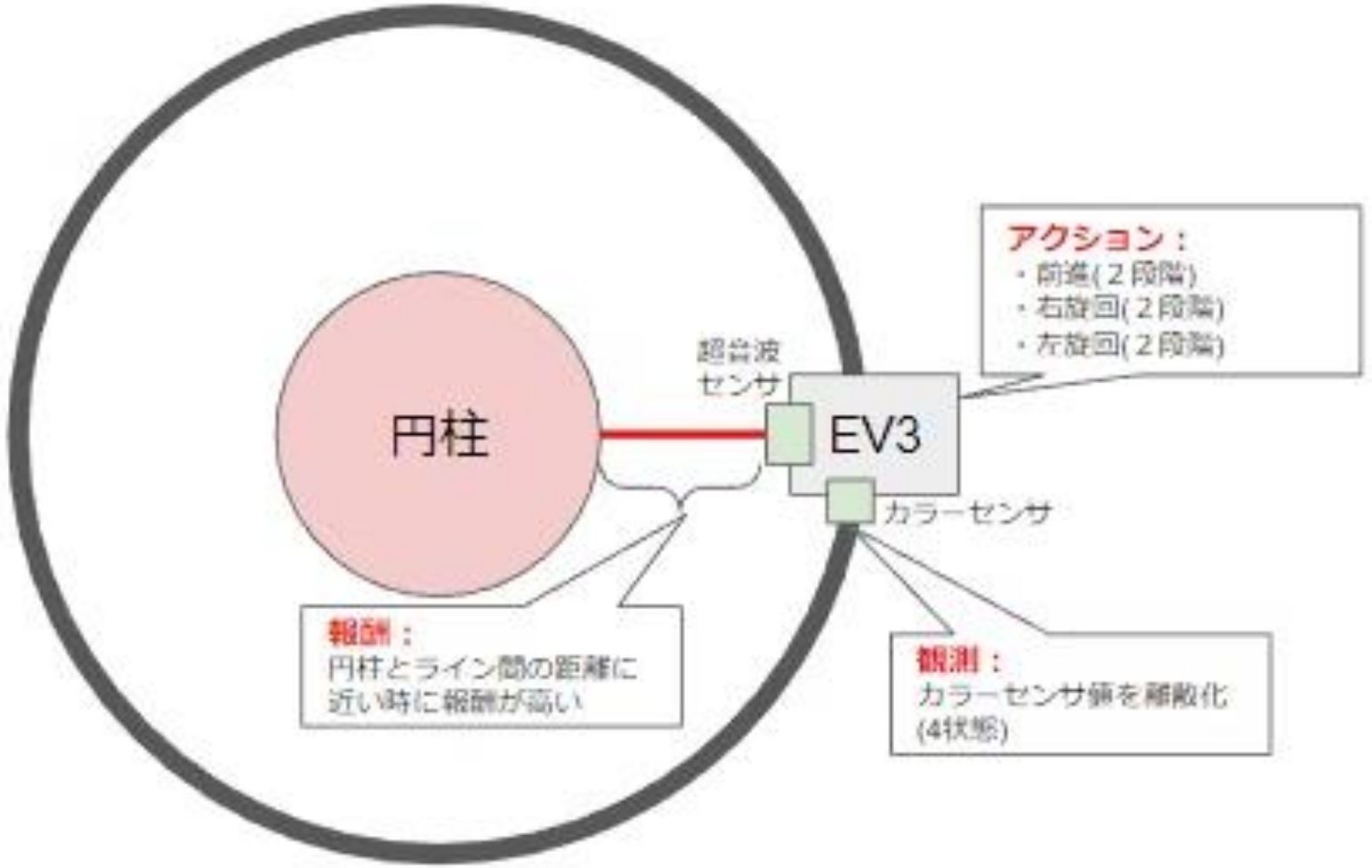
- こんな感じ。



強化学習で使用するロボット



強化学習デモ環境の説明



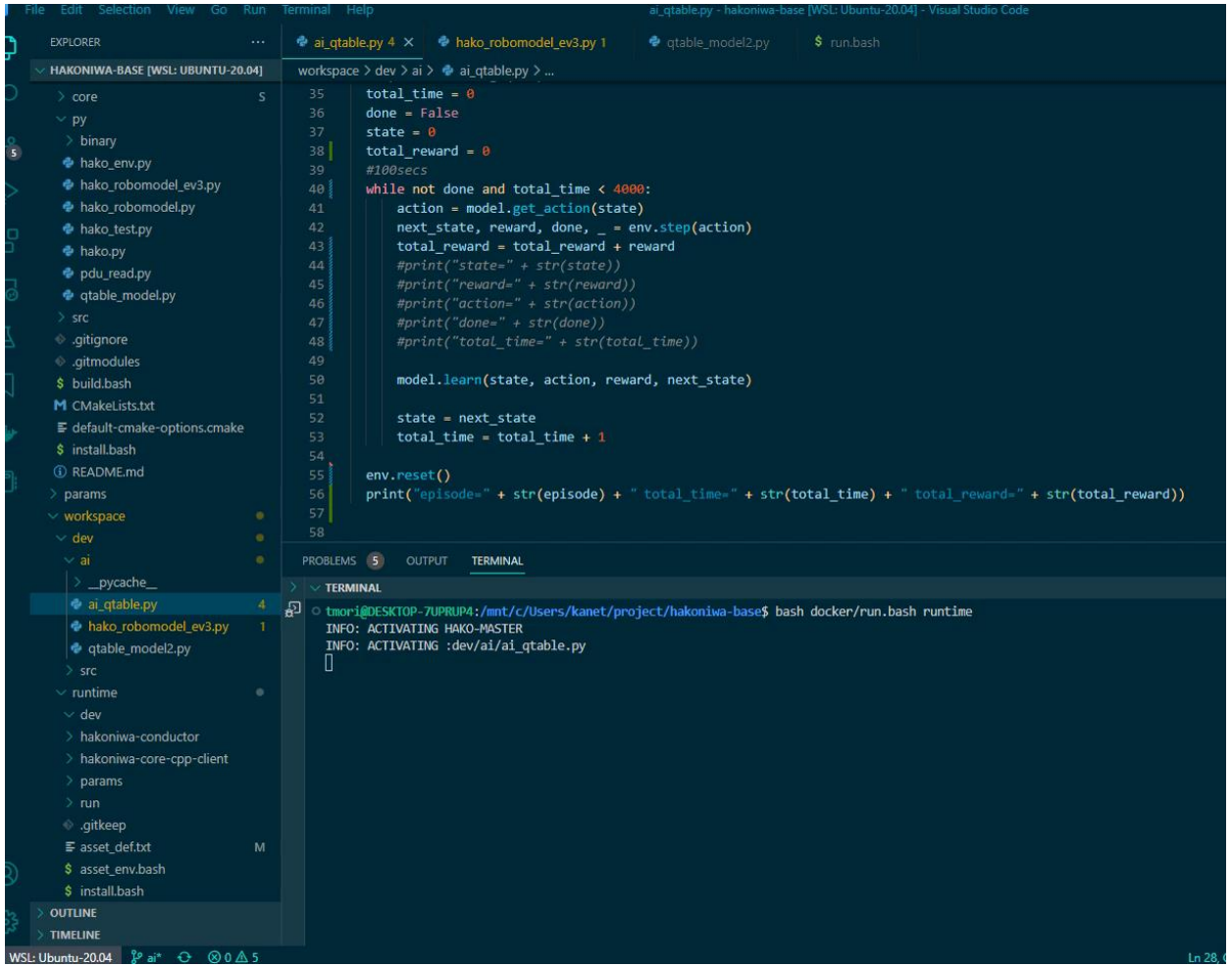
Qテーブル設計(ロボット視点)

状態	前進		右旋回		左旋回	
	Power=50	Power=20	Power=50	Power=20	Power=50	Power=20
黒	??	??	??	??	??	??
黒に近い	??	??	??	??	??	??
白に近い	??	??	??	??	??	??
白	??	??	??	??	??	??

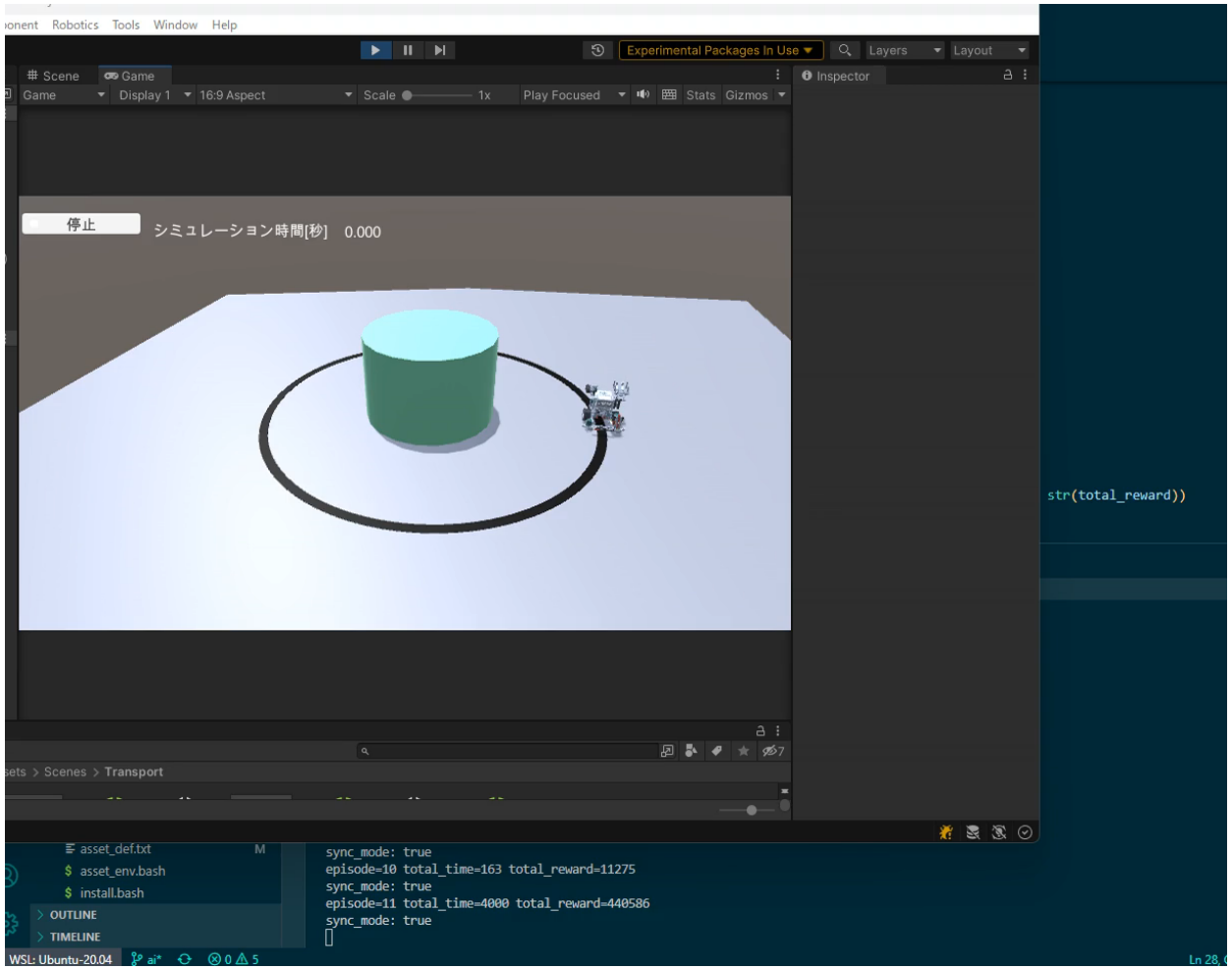
https://github.com/toppers/hakoniwa-base/blob/ai/workspace/dev/ai/qtable_model.csv

強化学習デモ

未学習状態：



学習済み状態：



5月
8

箱庭チュートリアル会 #4 箱庭とUnityですすめるオレオレロボットの動かしかた

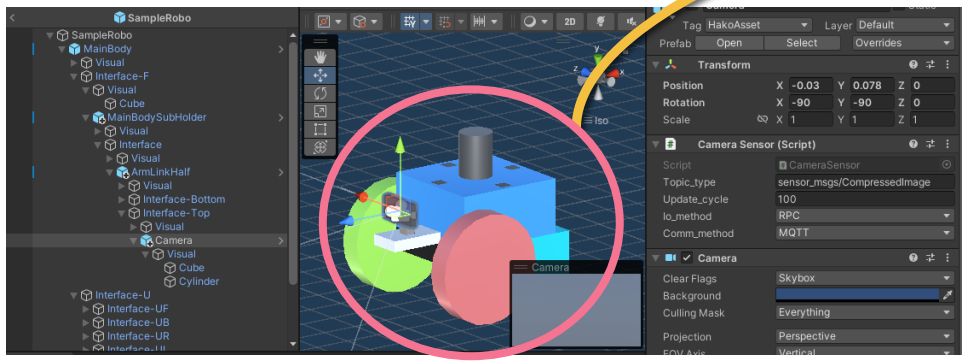
箱庭で１からロボットを動かしてみよう！

1. 前回の続き
    1. 箱庭ロボットの作り方
2. 箱庭ロボットの動かし方
    1. 箱庭強化学習をもう一度
    2. **組み立てたロボットを動かそう！**
3. 箱庭のインテグ方法
    1. 箱庭の内部アーキテクチャ
    2. 箱庭のリポジトリ構成とビルド方法

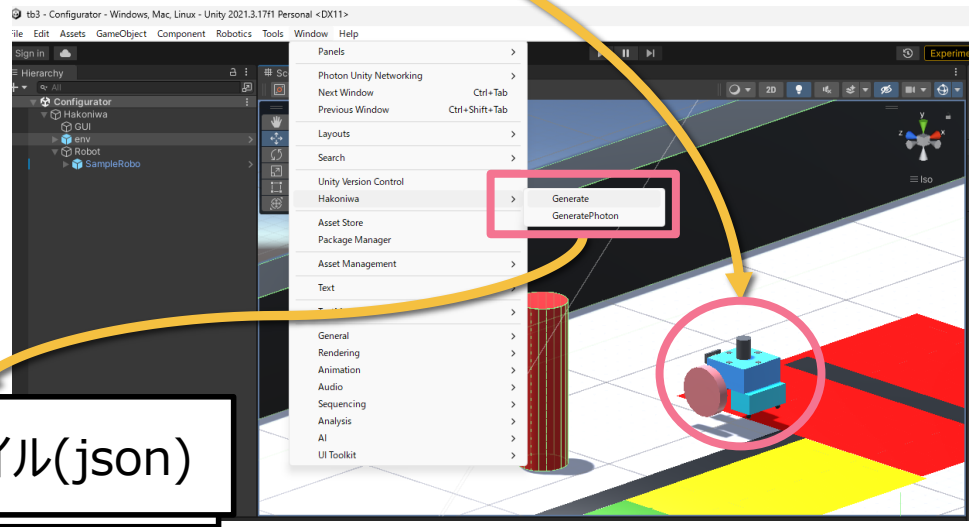
TOPPERS
Toyohashi OPen Platform for Embedded Real-time Systems

組み立てたロボットを動かそう！

①ロボットを組み立てる



②ロボットを配置する



コンフィグファイル(json)

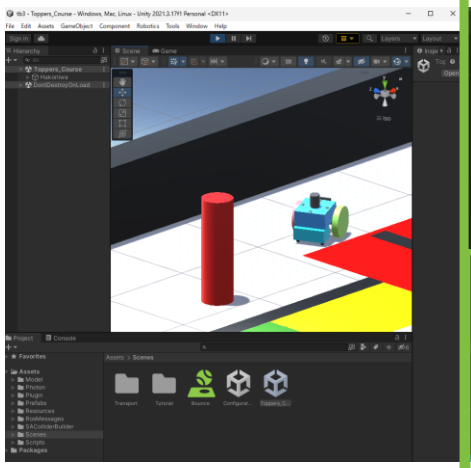
③Pythonでロボット制御プログラムを作成する

ロボット制御データ

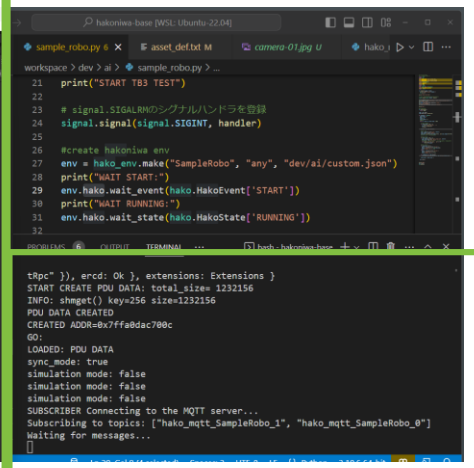
```
for episode in range(100):
  total_time = 0
  done = False
  state = 0
  total_reward = 0
  while not done and total_time < 4000:
      action = model.get_action(state)
      next_state, reward, done, _ = env.step(action)
      total_reward = total_reward + reward
      model.learn(state, action, reward, next_state)
      state = next_state
      total_time = total_time + 1
  env.reset()
  model.save('./dev/ai/qtable_model.csv')
```

④箱庭シミュレーションを実行する

Unity

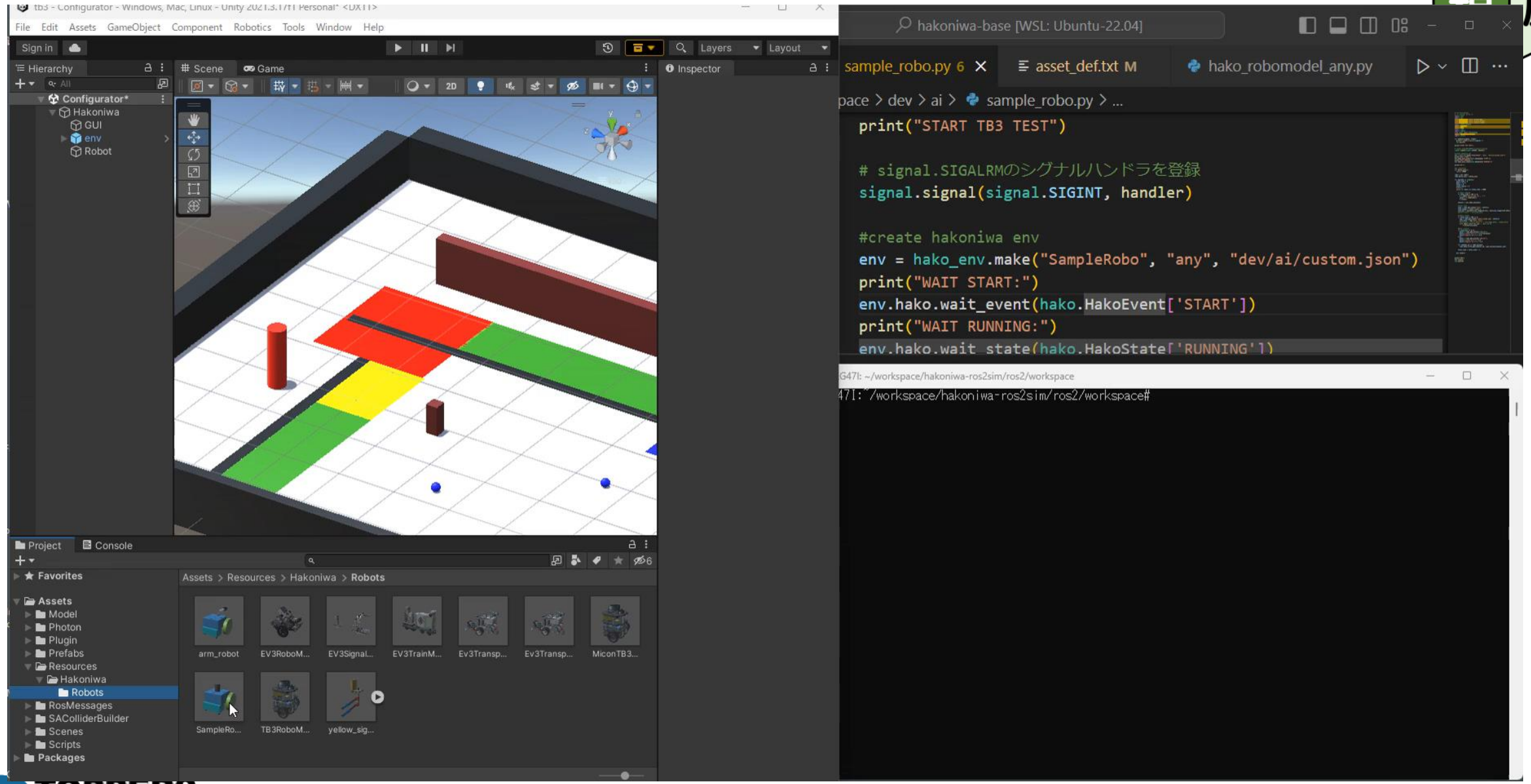


Python

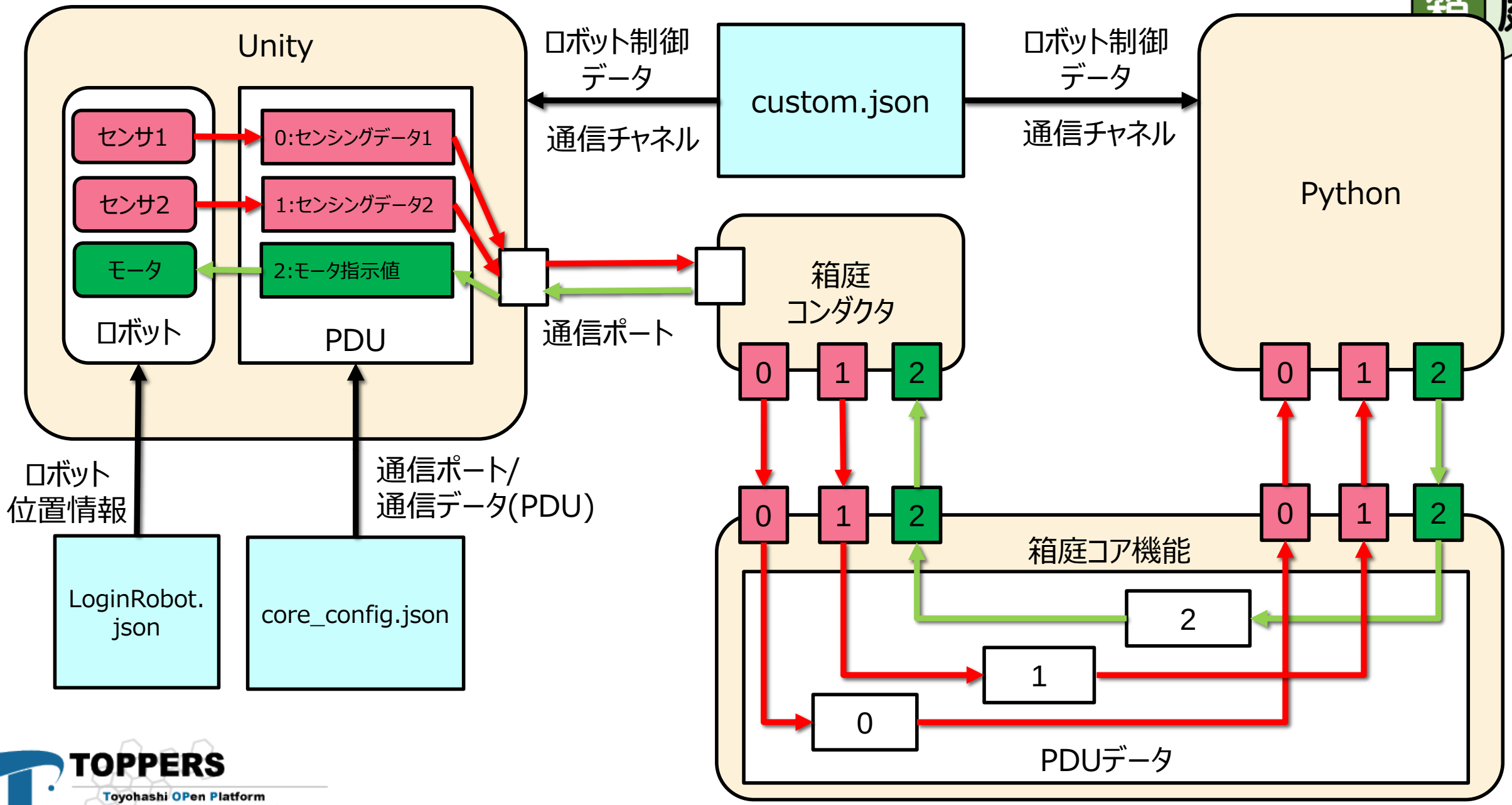


箱庭コンダクタ

操作方法


File Edit Assets GameObject Component Robotics Tools Window Help
Hierarchy
Configurator*
Hakoniwa
GUI
env
Robot
Scene
Game
Inspector
Project
Console
Favorites
Assets
Model
Photon
Plugin
Prefabs
Resources
Hakoniwa
Robots
RosMessages
SAColliderBuilder
Scenes
Scripts
Packages
Assets > Resources > Hakoniwa > Robots
arm_robot
EV3RoboM...
EV3Signal...
EV3TrainM...
Ev3Transp...
Ev3Transp...
MiconTB3...
SampleRo...
TB3RoboM...
yellow_sig...
hakoniwa-base [WSL: Ubuntu-22.04]
sample_robo.py 6
asset_def.txt M
hako_robomodel_any.py
pace > dev > ai > sample_robo.py > ...
print("START TB3 TEST")
# signal.SIGALRMのシグナルハンドラを登録
signal.signal(signal.SIGINT, handler)
#create hakoniwa env
env = hako_env.make("SampleRobo", "any", "dev/ai/custom.json")
print("WAIT START:")
env.hako.wait_event(hako.HakoEvent['START'])
print("WAIT RUNNING:")
env.hako.wait_state(hako.HakoState['RUNNING'])
/workspace/hakoniwa-ros2sim/ros2/workspace#

箱庭のコンフィグファイルを利用するひと



コンフィグファイルの種類

- LoginRobot.json（hakoniwa-ros2sim¥settings¥tb3）
  - ロボットの位置情報
- custom.json（hakoniwa-ros2sim¥settings¥tb3）
  - ロボットのI/O方法
  - 各I/Oで使うデータと型
  - 補足：以下にも配置されます
    - hakoniwa-base¥workspace¥dev¥ai
- core_config.json（hakoniwa-ros2sim¥ros2¥unity¥tb3）
  - 通信ポートの定義
  - 箱庭通信用のPDUデータ定義
  - ロボットとPDUデータの関係定義

5月
8

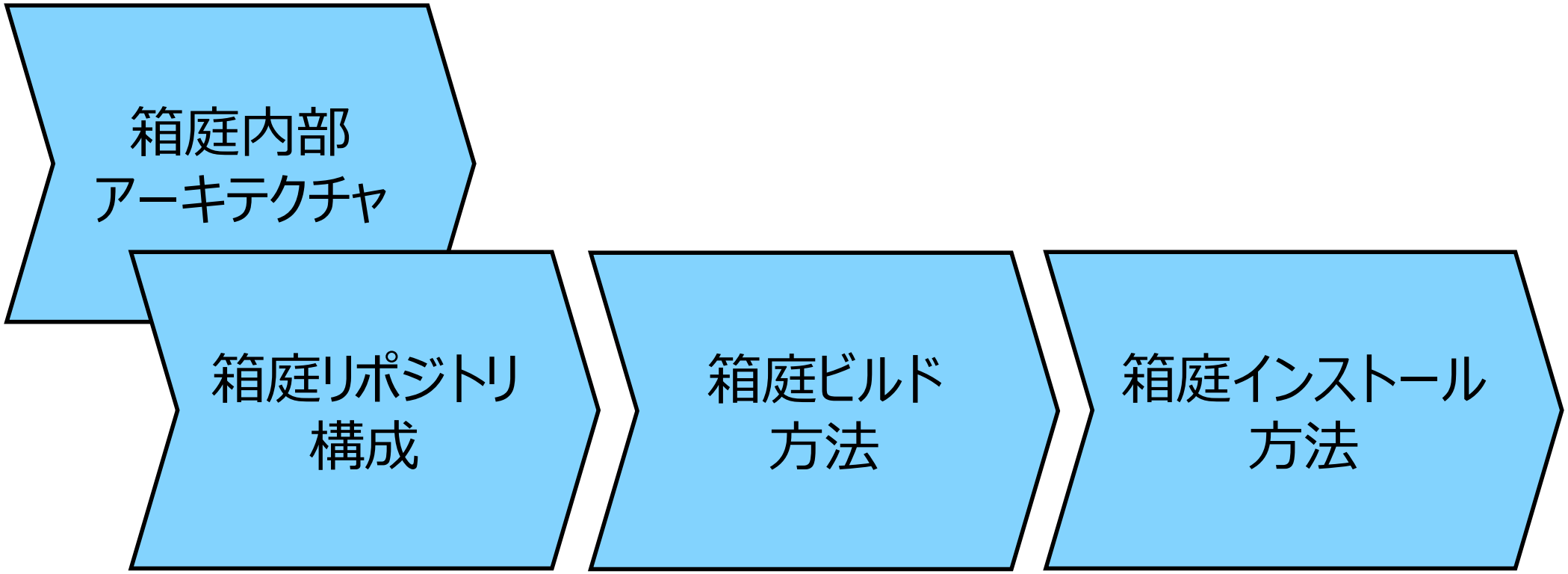
箱庭チュートリアル会 #4 箱庭とUnityですすめるオレオレロボットの動かしかた

箱庭で１からロボットを動かしてみよう！

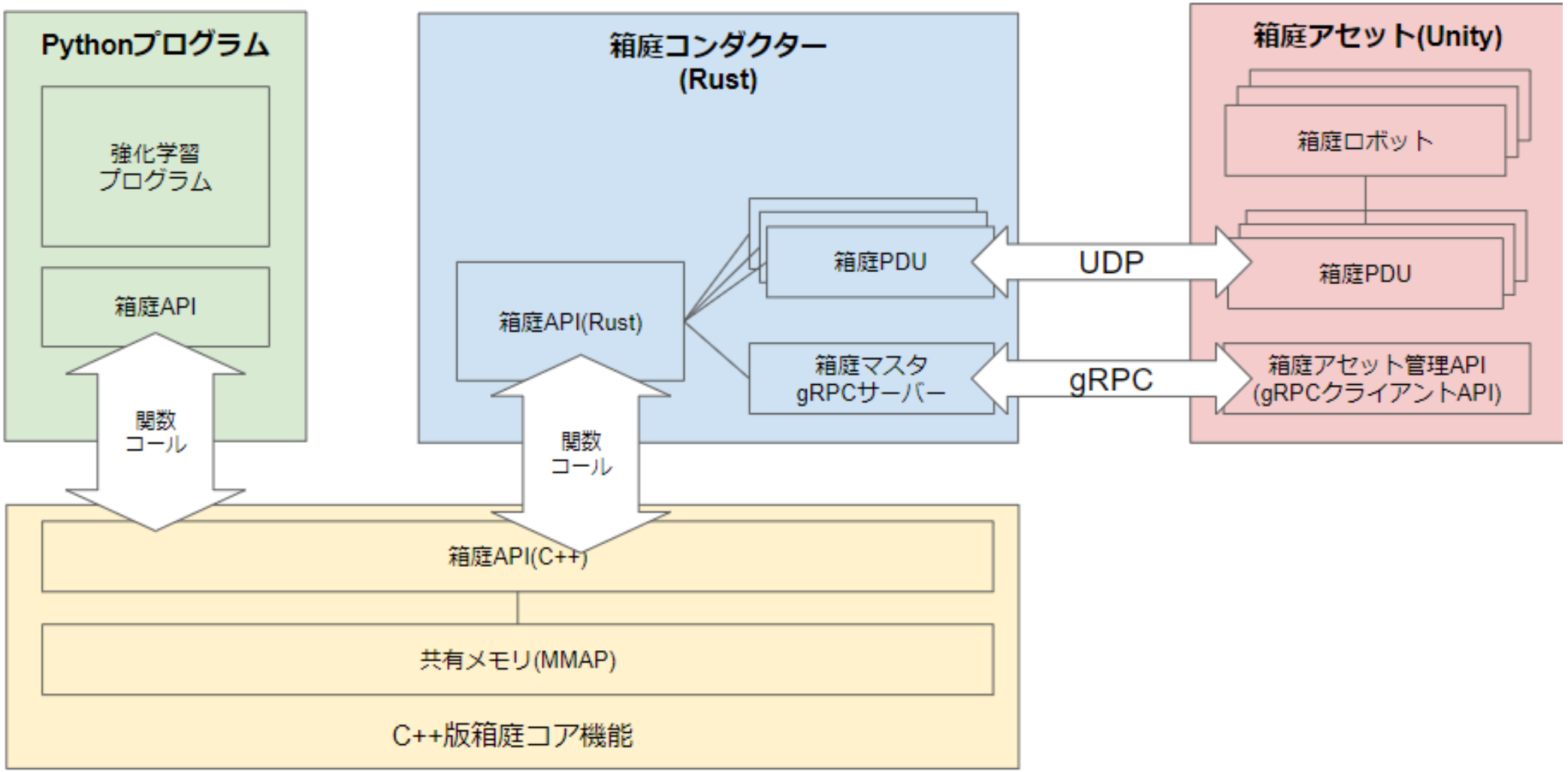
1. 前回の続き
    1. 箱庭ロボットの作り方
2. 箱庭ロボットの動かし方
    1. 箱庭強化学習をもう一度
    2. 組み立てたロボットを動かそう！
3. **箱庭のインテグ方法**
    1. 箱庭の内部アーキテクチャ
    2. 箱庭のリポジトリ構成とビルド方法

TOPPERS
Toyohashi OPen Platform for Embedded Real-time Systems

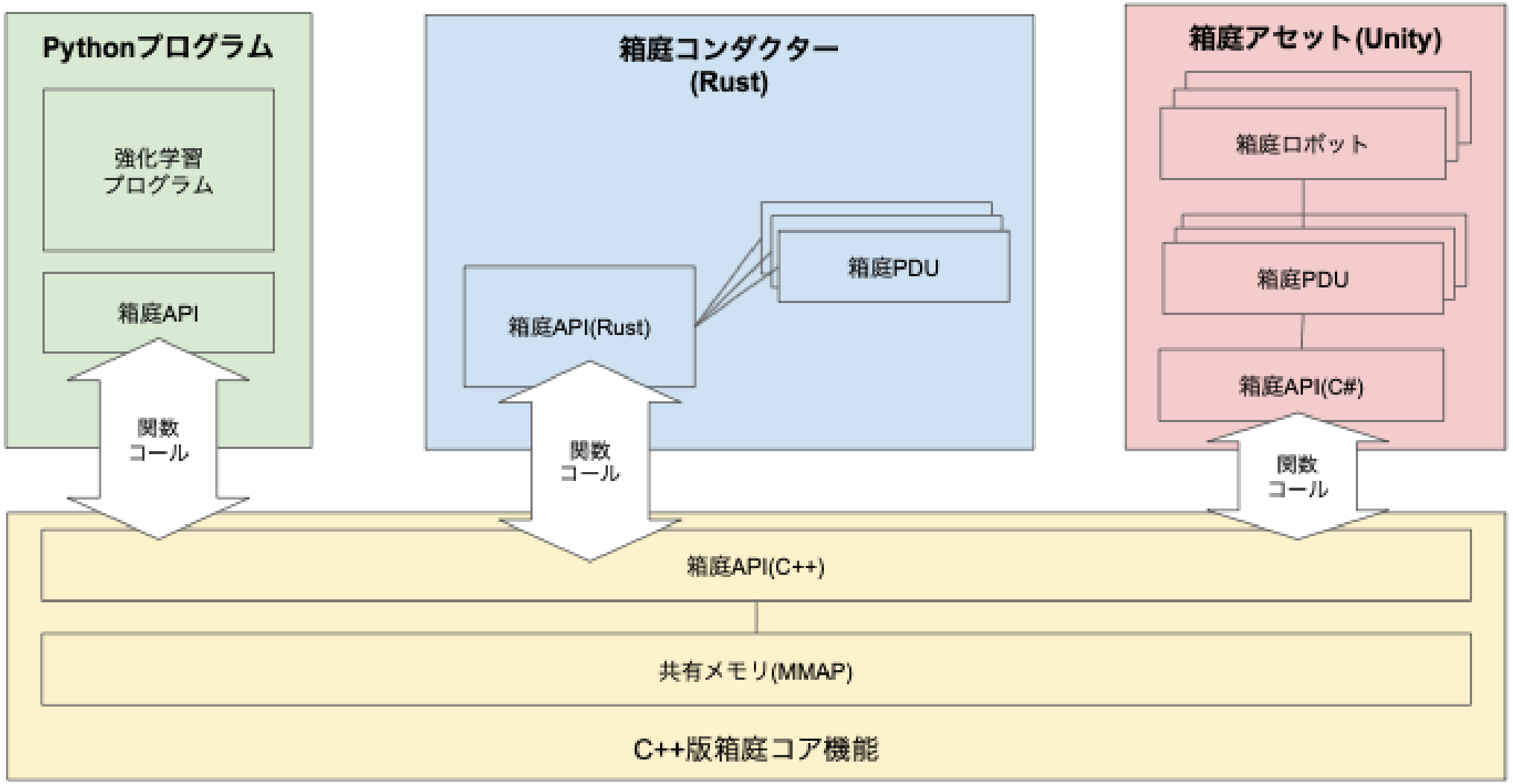
箱庭のインテグ方法



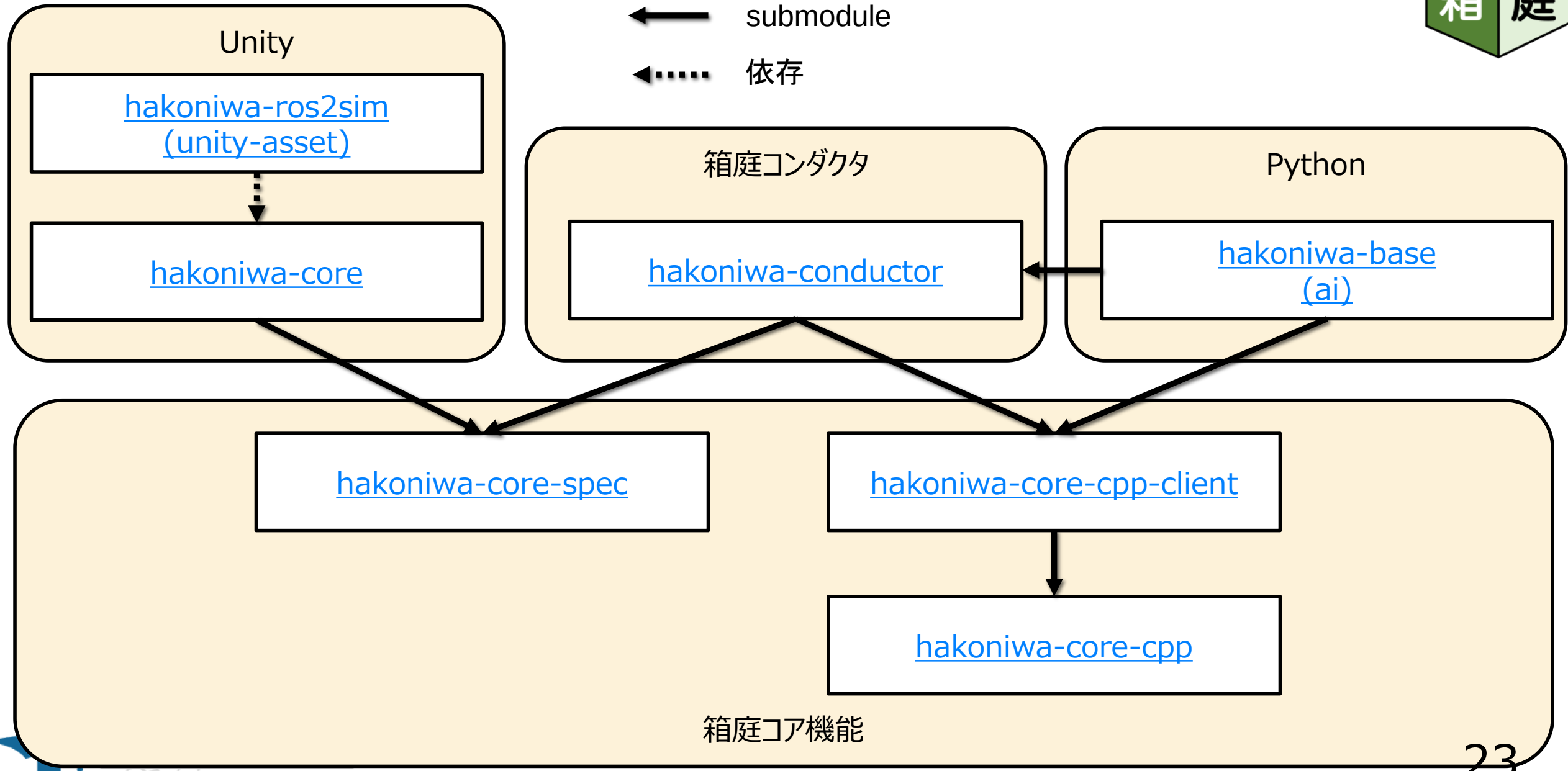
箱庭の内部アーキテクチャ(Windows)



箱庭の内部アーキテクチャ(Mac/Linux)



箱庭リポジトリ構成


submodule
依存
Unity
hakoniwa-ros2sim
(unity-asset)
hakoniwa-core
箱庭コンダクタ
hakoniwa-conductor
Python
hakoniwa-base
(ai)
hakoniwa-core-spec
hakoniwa-core-cpp-client
hakoniwa-core-cpp
箱庭コア機能

箱庭ビルド&インストール方法（Unity側）

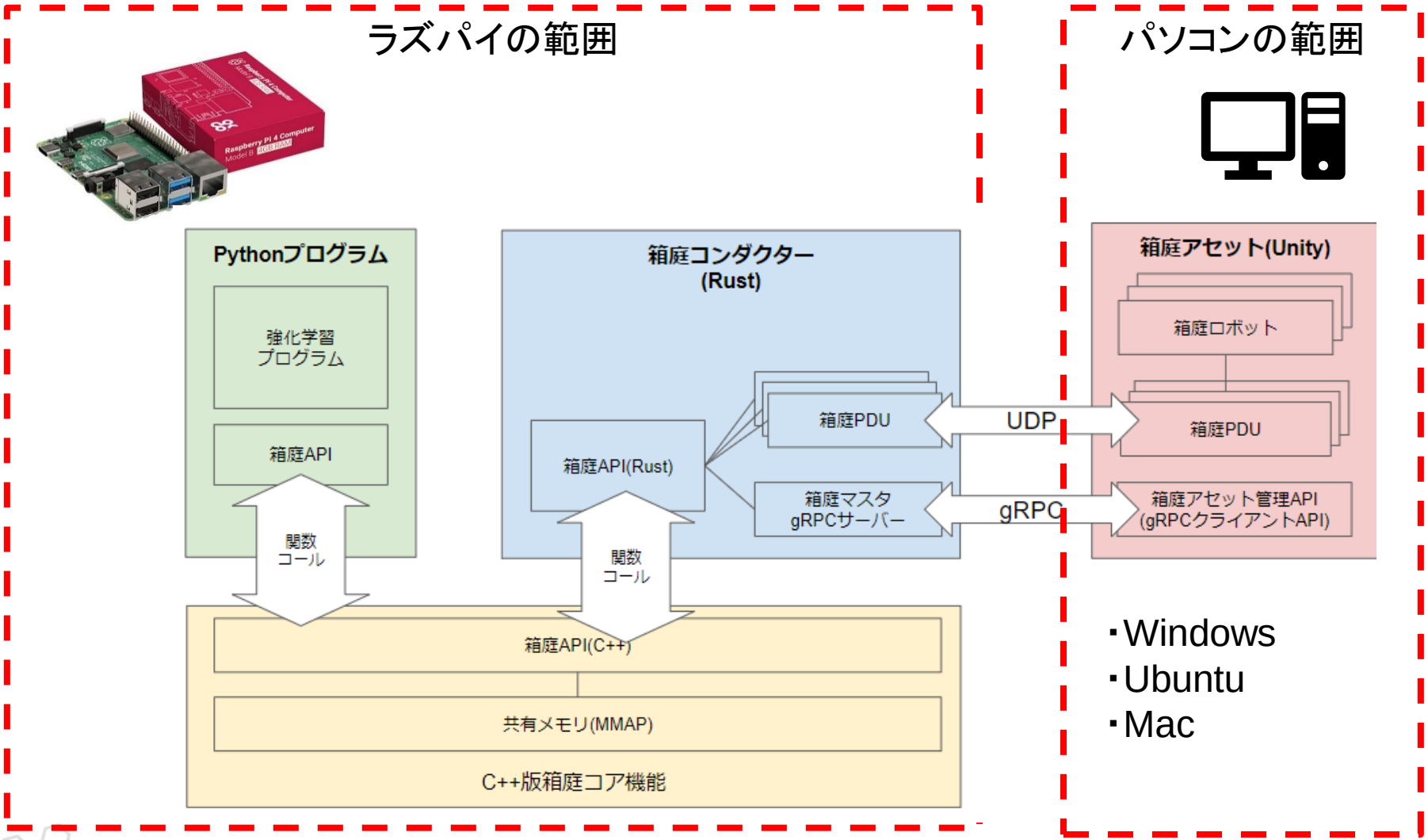
1. hakoniwa-coreをクローン
2. DLLのビルド
   - Windowsの場合
     - Visual Studio をインストール
       - 以下をダブルクリック
         - https://github.com/toppers/hakoniwa-core/blob/main/impl/asset/server/csharp/HakoniwaCore/Hakoniwa.csproj
     - Visual Studioが起動
     - Visual Studioのメニューからビルド実行
       - dllファイルが、binのフォルダのとこにできます。
         - hakoniwa-core¥impl¥asset¥server¥csharp¥HakoniwaCore¥bin¥Debug¥netstandard2.0
   - Linuxの場合
     - bash ./impl/asset/server/csharp/HakoniwaCore/build_by_docker.bash Rebuild
3. インストール
   - hakoniwa.dllを、hakoniwa-ros2simの以下のディレクトリにコピーします。
     - hakoniwa-ros2sim¥ros2¥unity¥tb3¥Assets¥Plugin

箱庭ビルド&インストール方法（Unity以外）

- hakoniwa-core-cpp-client
  - bash build.bash
    - 共有ライブラリができます(libshakoc.so)。
  - bash install.bash
    - /usr/local/lib/hakoniwaに libshakoc.soをコピーします。
- hakoniwa-conductor
  - cd main
  - bash build.bash
  - bash install.bash
    - /usr/local/bin/hakoniwaに箱庭コンダクタのバイナリをコピーします。

ラズパイ向けの手順と課題（まだ試してません・・・）

- アーキテクチャ



ラズパイでのビルドおよびインストール方法

1. 事前に以下をインストール
   - git, gcc, make, build-essential, protobuf-compiler, …
   - rust
   - Python3

   ※参考：https://github.com/toppers/hakoniwa-base/blob/ai/docker/template/runtime/ai/Dockerfile
2. hakoniwa-base(aiブランチ) を recursive で clone
3. 以下をビルド & インストール
   1. hakoniwa-core-cpp-client
   2. hakoniwa-conductor

【ラズパイ対応での心配ごと】
- rustがインストール可能かどうか
- rust用のgRPCライブラリをインストール可能かどうか
  - tonic

パソコン側の設定

- いつも通り、Unityのコンフィグファイルを生成します。
- core_config.jsonのパラメータを変更します。
  - core_ipaddr
    - IPアドレスをラズパイのIPアドレスに設定します。
  - asset_ipaddr
    - パソコンのイーサーネットのIPアドレスを設定します。

おまけ

- ChatGPTと箱庭をつなげてみました。