2023年度
仮想シミュレーション環境
「箱庭」最新情報

森 崇
(永和システムマネジメント)

TOPPERS
Toyohashi OPen Platform
for Embedded Real-time Systems

自己紹介

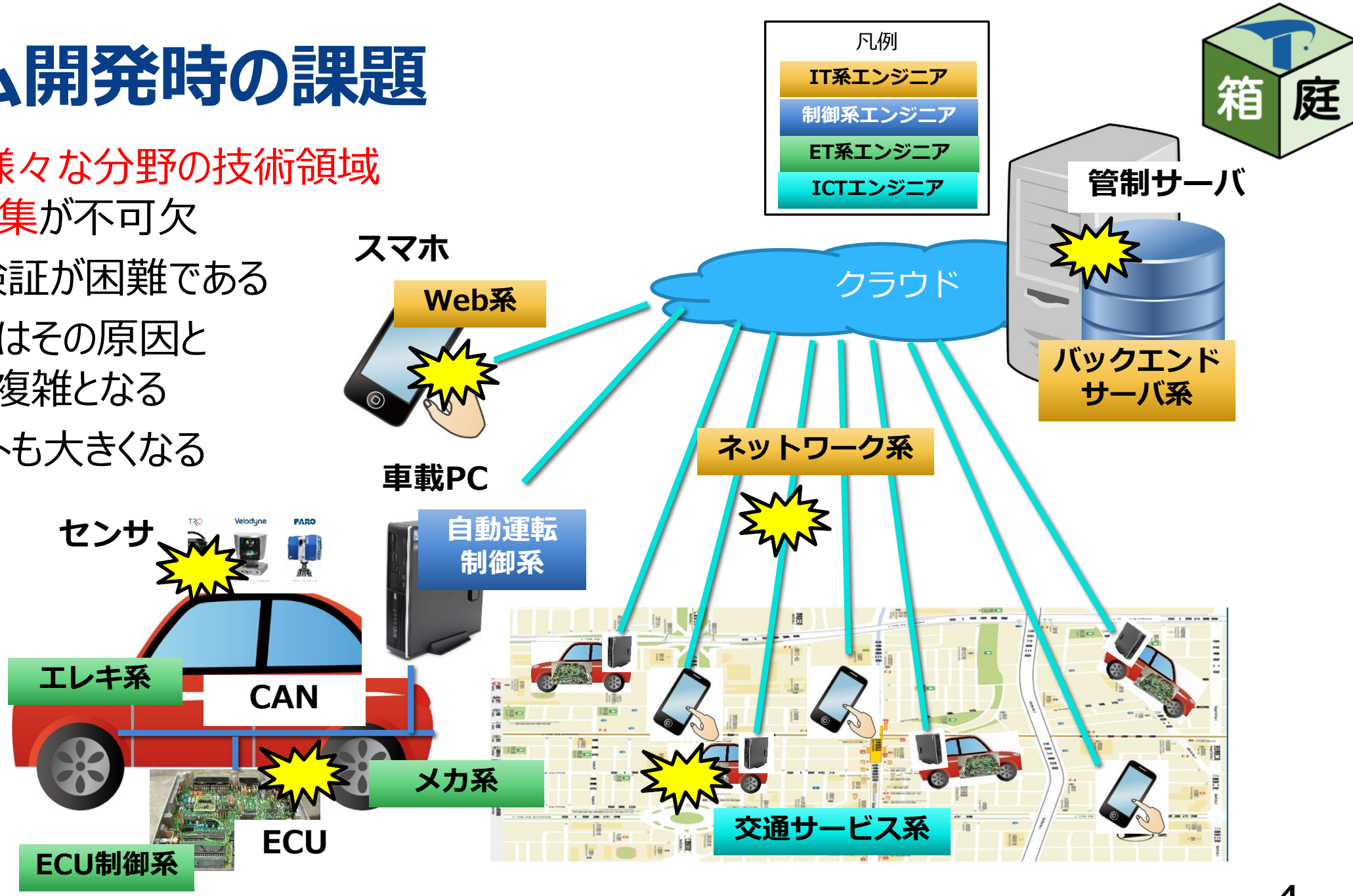
- 森　崇(もり　たかし)
- 職歴/専門分野
  - Linuxファイルシステム/カーネル周りのお仕事(10年)
  - 組込みRTOS/ROS周りのお仕事(10年)
  - 新規ビジネス向けWebシステム開発(２年目)
- TOPPERS/箱庭WG参加・開発(5年)
  - 組込みソフト/マイコンシミュレータ担当
  - 箱庭コア機能担当
  - 物理シミュレーション/ビジュアライズ
  - その他いろいろ

アジェンダ

1. 『箱庭』の狙いとコンセプト
2. 箱庭を実現する技術
3. 箱庭プロトタイプモデルとデモ
4. 全体アーキテクチャ
5. 今後の方向性／ロードマップ
6. 箱庭の最新情報

IoTシステム開発時の課題

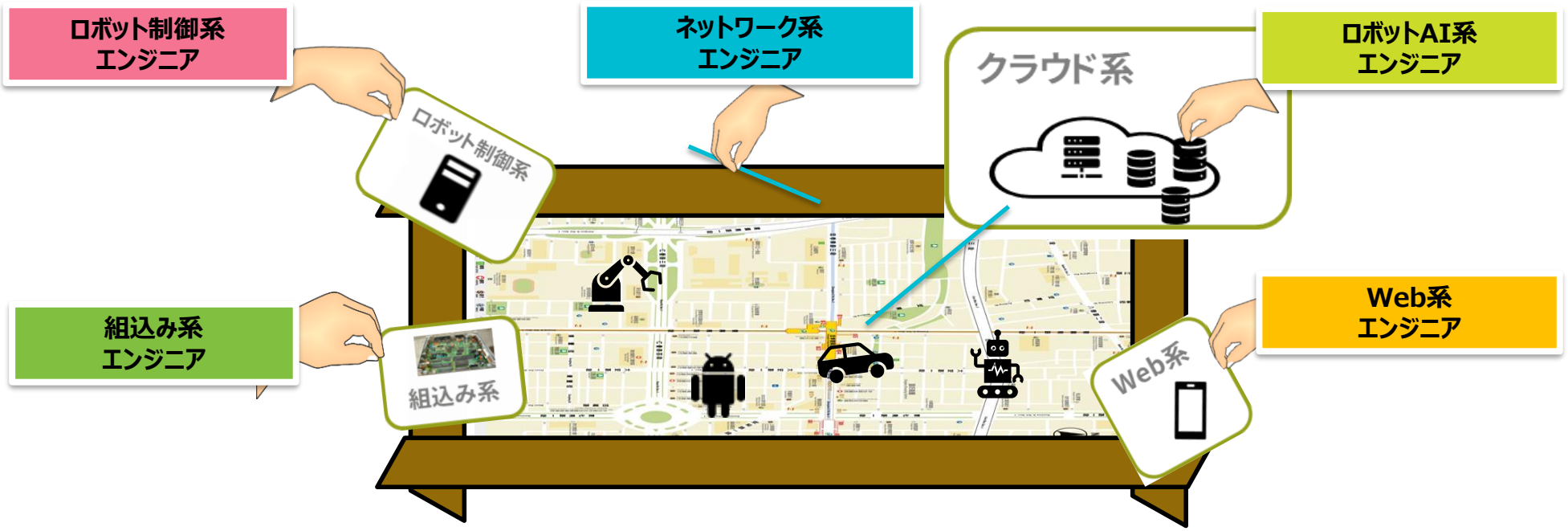
- IoT開発には様々な分野の技術領域＝技術者の結集が不可欠
- 結合テストや検証が困難である
- 問題発生時にはその原因と経路の調査が複雑となる
- 実証実験コストも大きくなる



TOPPERS
Toyohashi OPen Platform for Embedded Real-time Systems

『箱庭』の狙いとコンセプト

- 箱の中に，様々なモノをみんなの好みで配置して，いろいろ試せる！
  - 仮想環境上(箱庭)でIoT/ロボット・システムを開発する
- ⇒ 各分野のソフトウェアを持ち寄って，机上で全体結合＆実証実験！



箱庭を実現する技術

ロボット制御系

ROS
ROS1/ROS2

unity
UI/ビジュアライズ/物理演算エンジン

VR

Photon

クラウド系

docker

Robotics
Developers BOX

RDBOX
A Robotics Developers BOX

箱庭

Athrill
マイコン
シミュレータ

ROS
組込み向け
ROS軽量実行環境

RTOS

astah

TOPPERS
Toyohashi OPen Platform
for Embedded Real-time Systems

組込み系

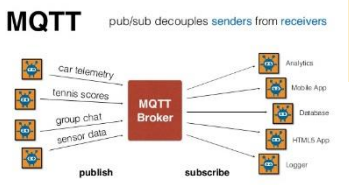
Protocol Buffers

OPEN API
INITIATIVE

GRPC

spring
Boot

Vue.js



Web/ネットワーク系

TOPPERS
Toyohashi OPen Platform
for Embedded Real-time Systems

アジェンダ

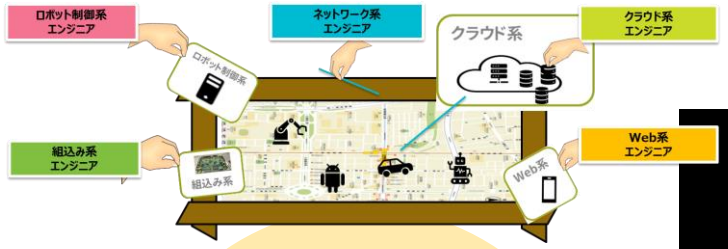
1. 『箱庭』の狙いとコンセプト
2. 箱庭を実現する技術
3. **箱庭プロトタイプモデルとデモ**
4. 全体アーキテクチャ
5. 今後の方向性／ロードマップ
6. 箱庭の最新情報

箱庭プロトタイプモデルとデモ

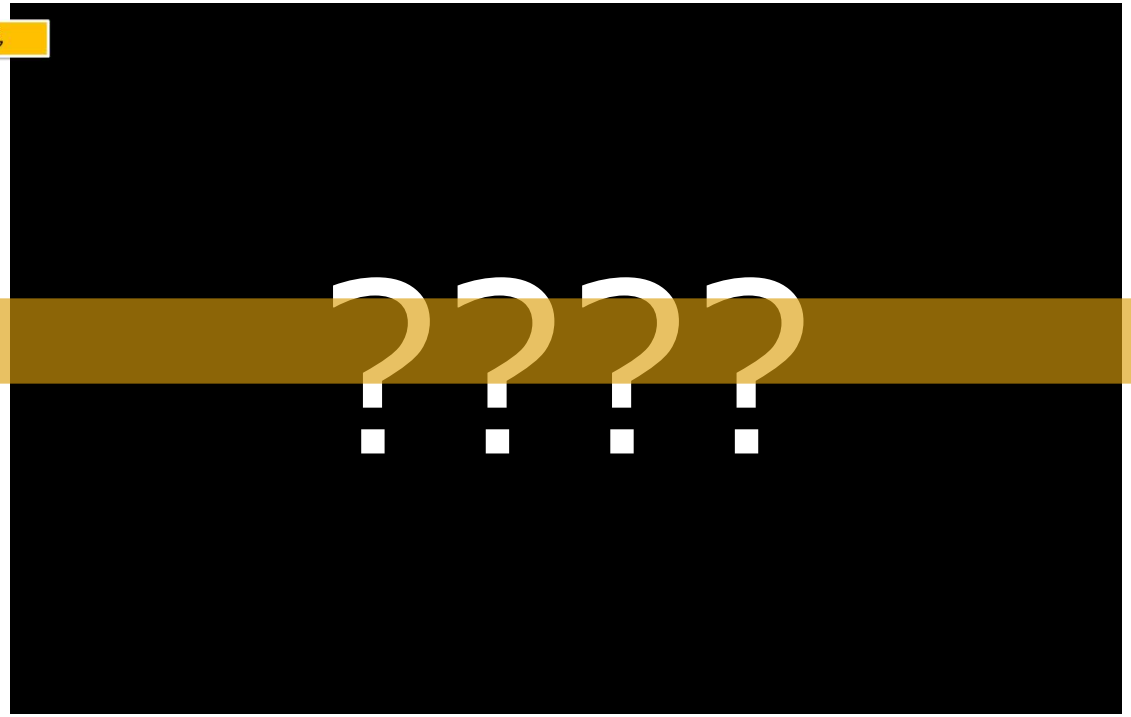
- プロトタイプモデルの狙い
- ３つのプロトタイプモデル
  - マイコン制御ロボットシミュレーション
  - ROS制御ロボットシミュレーション
  - 複数ロボットの連携制御シミュレーション

プロトタイプモデルの狙い

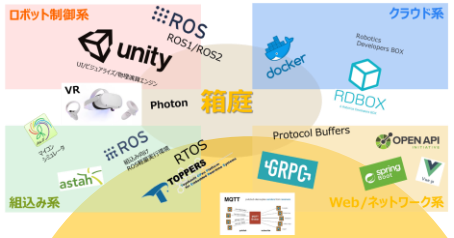
- 箱庭コンセプトの実現と技術研鑽！



箱庭
コンセプト

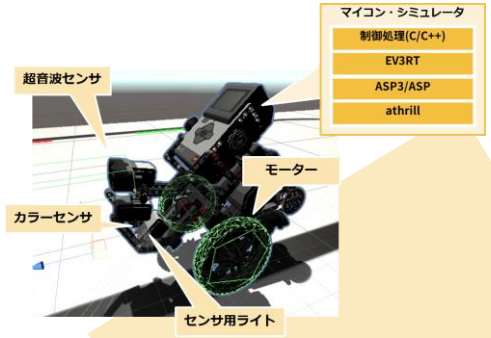


箱庭コンセプトを形あるものに！

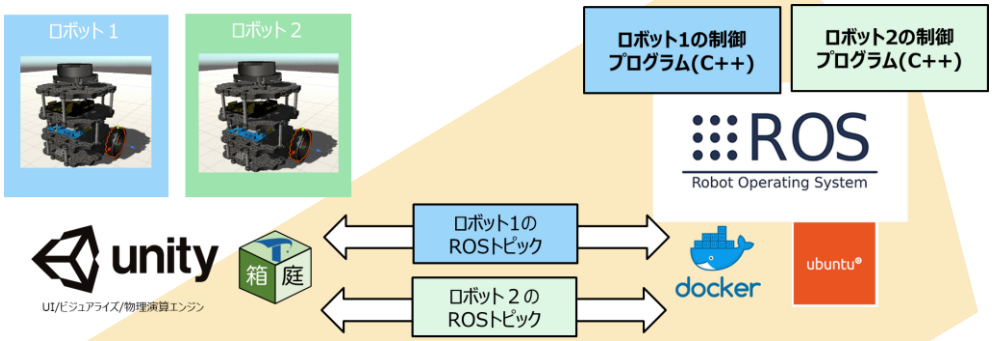


箱庭プロトタイプモデル

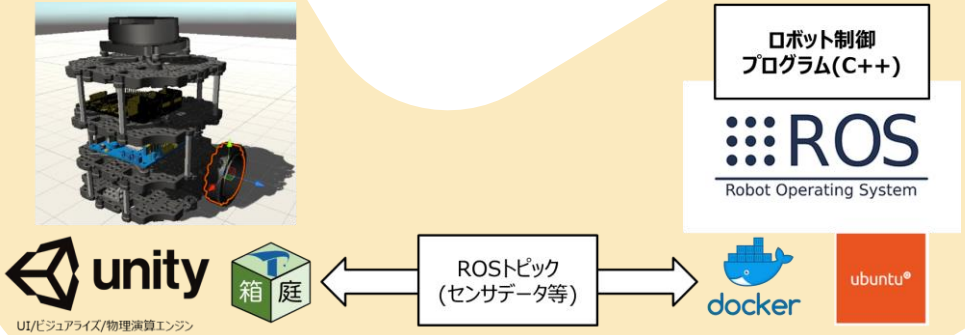
３つのプロトタイプモデル



マイコン制御ロボットシミュレーション



複数ロボットの連携制御シミュレーション



ROS制御ロボットシミュレーション

TOPPERS
Toyohashi OPen Platform
for Embedded Real-time Systems

マイコン制御ロボットシミュレーション

■狙い

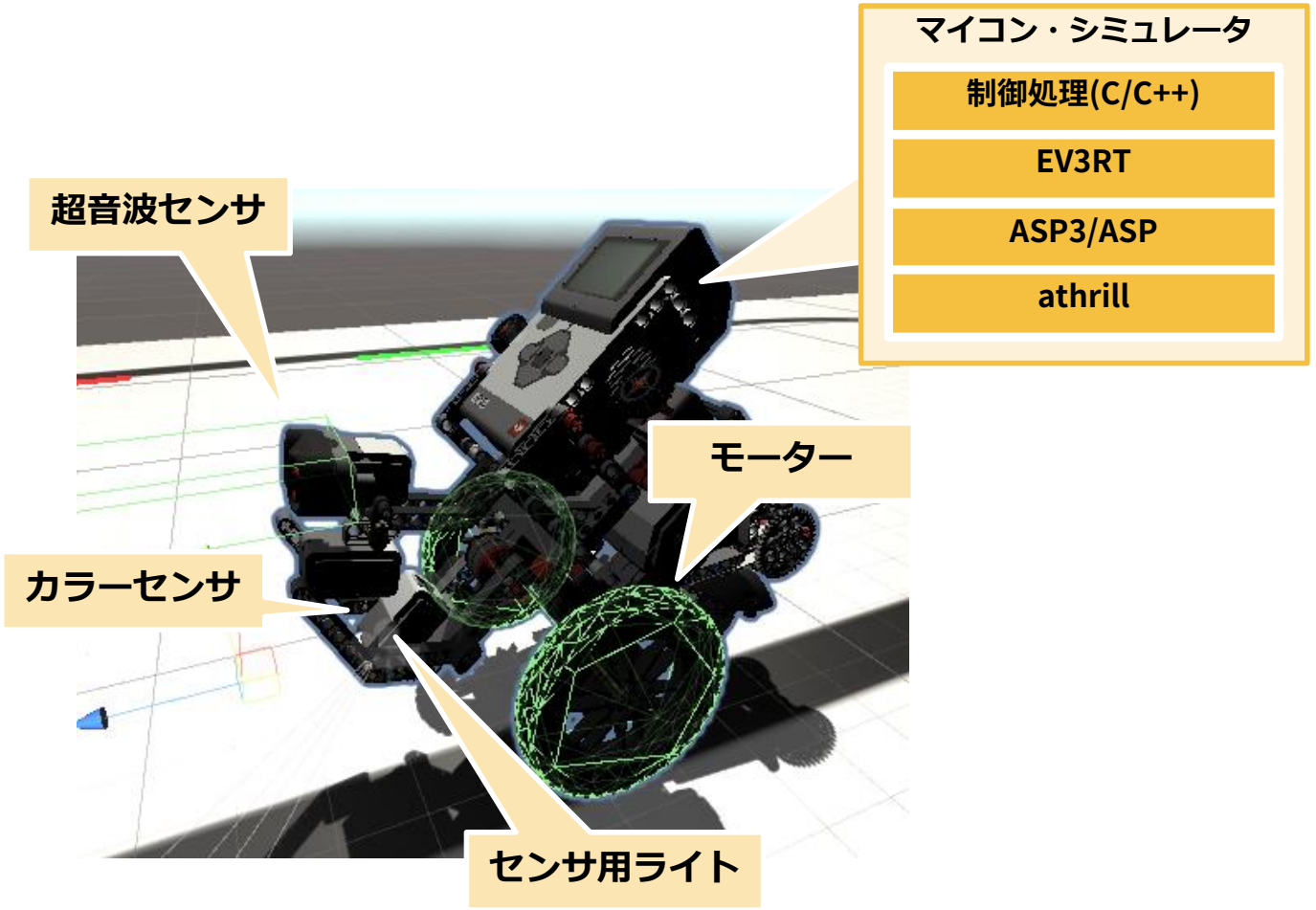
Unity上のロボット(1個)と仮想マイコンを連携させたシミュレーションの実現

■構成要素

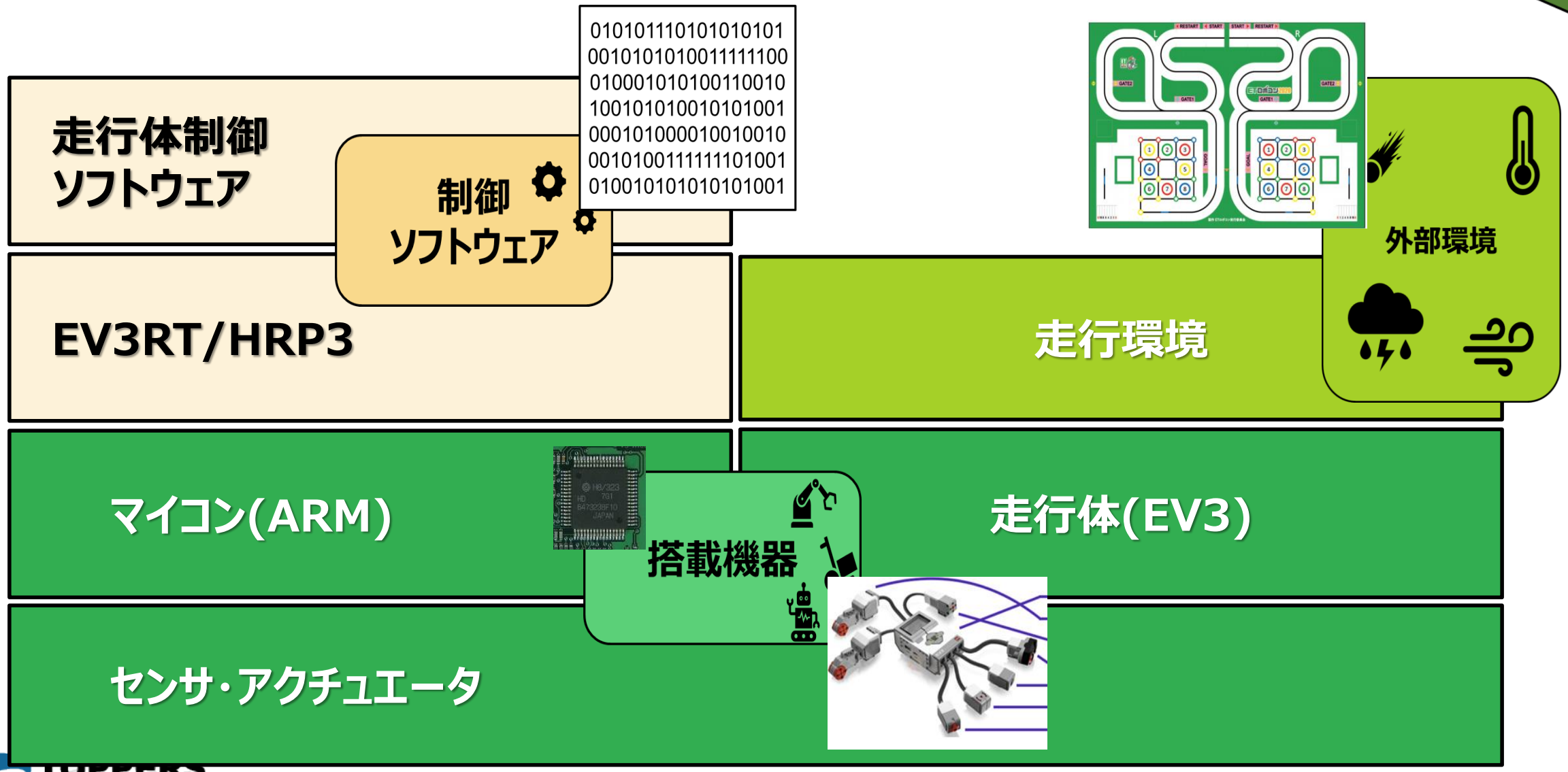
- ロボット本体
  - ・モータ/センサ等
- ロボットを制御するマイコン
  - ・RTOSおよびロボット制御が動きます
- 外部環境
  - ・走行コース/障害物等

■広報活動

- ETロボコンユーザ層に箱庭を広める
- ロボット教育演習での活用

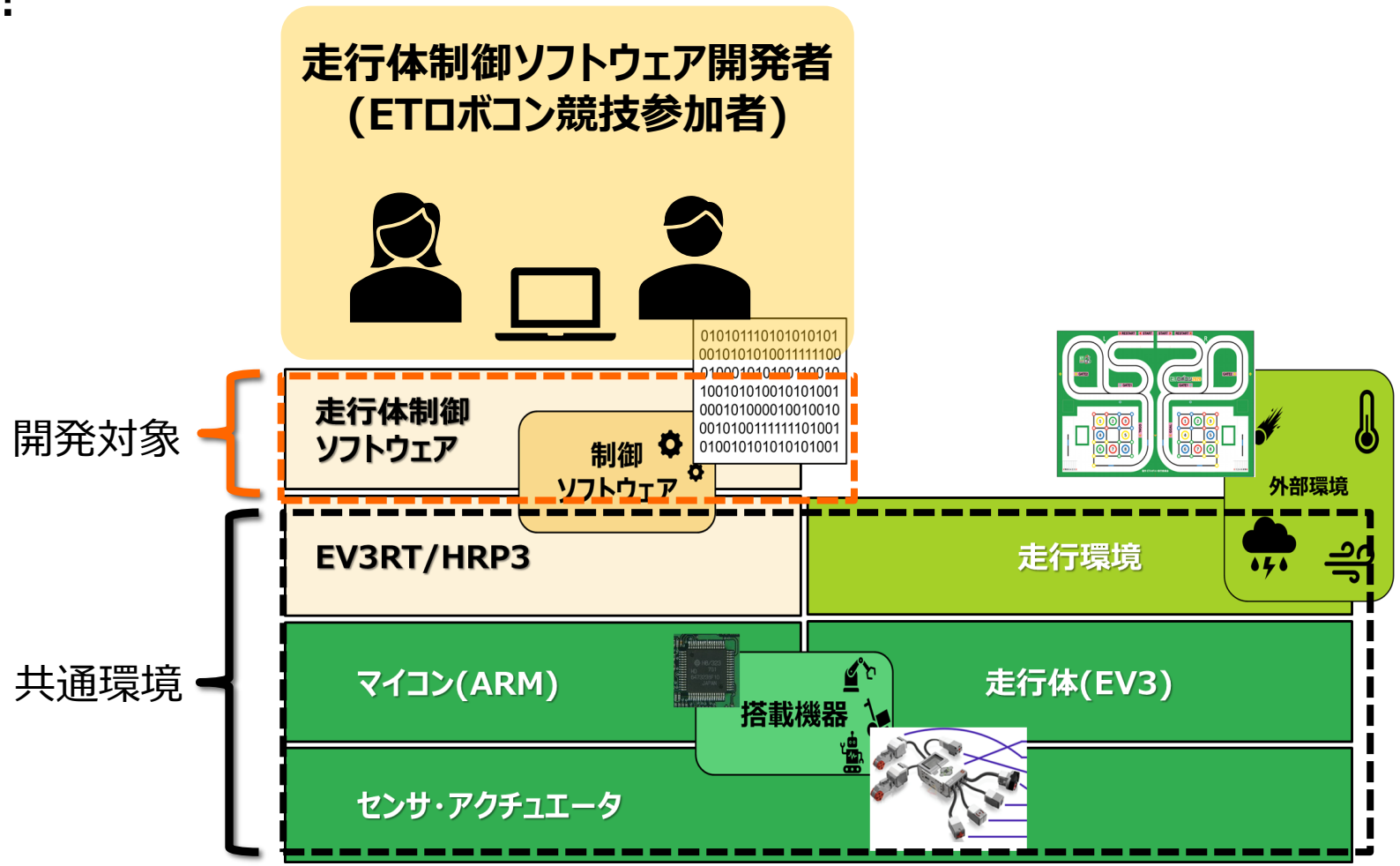


ETロボコンのアーキテクチャ



なにを目的としたシミュレータなのか(1/2)

- 誰をハッピーにしたいのか？



TOPPERS
Toyohashi OPen Platform
for Embedded Real-time Systems

なにを目的としたシミュレータなのか(2/2)

- どういう課題を解決したいのか？

Cost

- 実機/コース購入費用
  ⇒実機/コースの奪い合い！
- 実験するには広いスペースが必要
  ⇒実験場の準備/移動しないといけない！

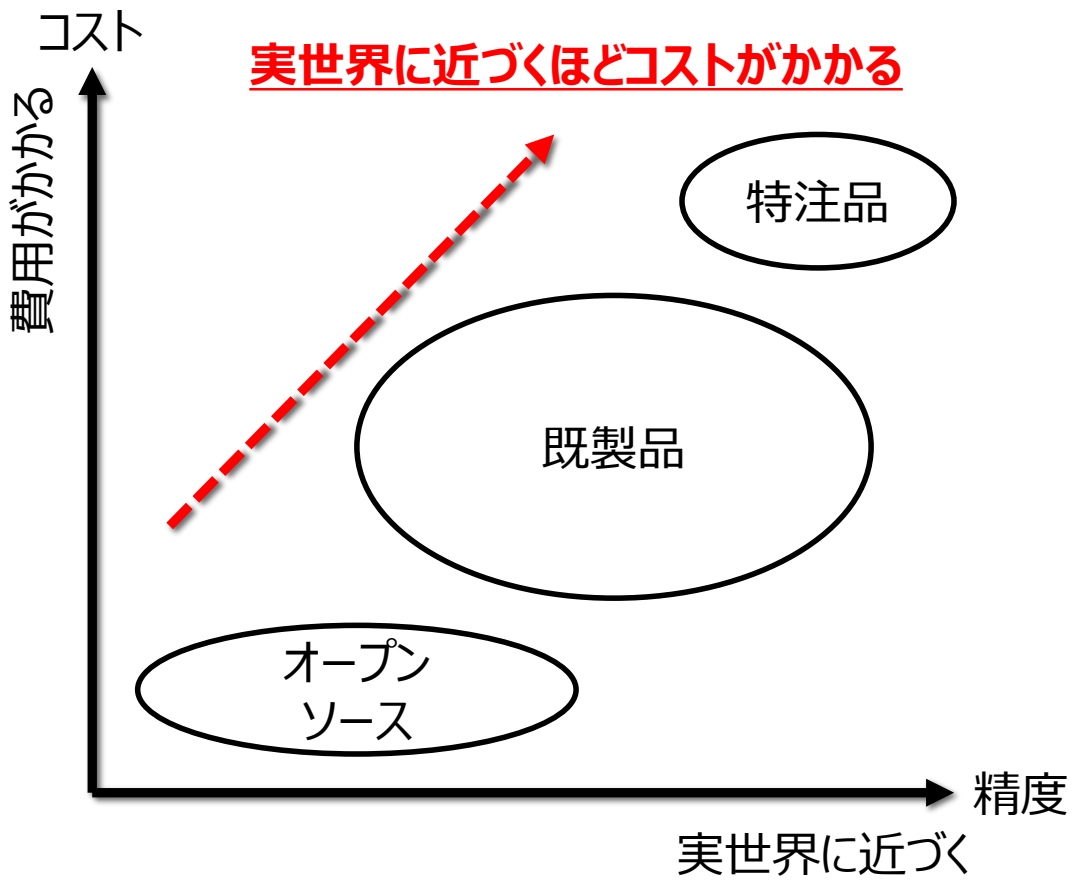
Quality

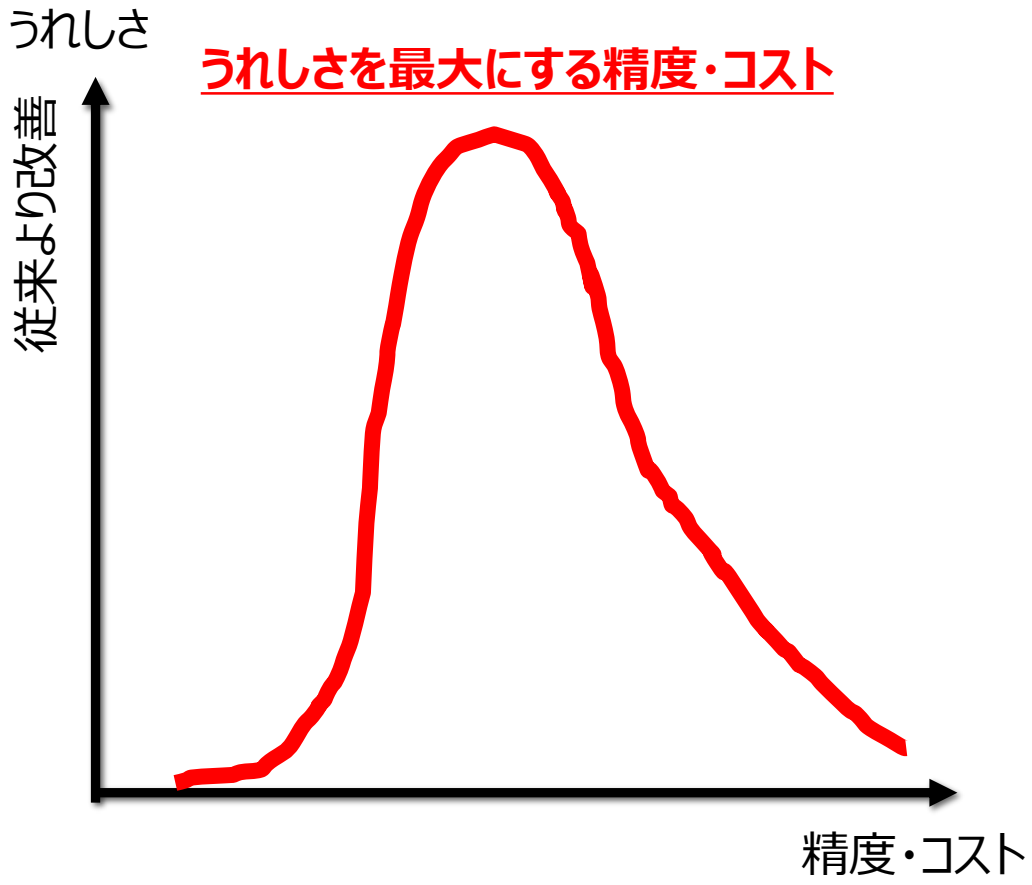
- ロボット制御/RTOSの知識不足
  ⇒初心者だと普通に走らせるまでが大変！
- 光センサの外乱/コースの微妙なズレ
  ⇒物理的な外乱系は狙ってテストできない！

Delivery

- デプロイ/デバッグの手間
  ⇒ひたすら時間をかけてやる！
- 特定シーン/条件のテストの手間
  ⇒ひたすら同じシーンを繰り返す/試行錯誤
- バッテリ切れ，実機故障
  ⇒そもそもテストできない！

精度とコストのバランス





コストを無視してシミュレータ精度を実世界に近づけるとハッピーになれるか？

TOPPERS
Toyohashi OPen Platform
for Embedded Real-time Systems

ETロボコンシミュレータの構造

[補足]
Unity側は箱庭成果物として存在しており,
独自進化を続けています(後述).

走行体制御ソフトウェア

箱庭要素技術

シミュレータ向け EV3RT/ASP3

マイコンシミュレータ Athrill(V850)

ETロボコン実行委員成果

走行環境

走行体(EV3)

物理エンジン unity

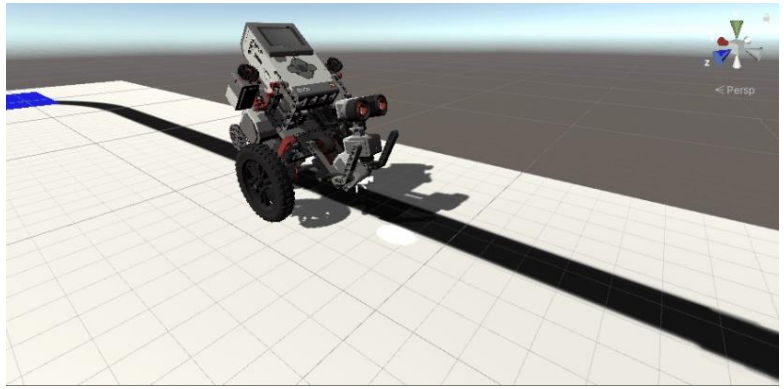
シミュレータ間通信(センサ・アクチュエータ)/時間同期機構

箱庭だと何がうれしいか？

実機なしで手軽にいろいろ試せます！

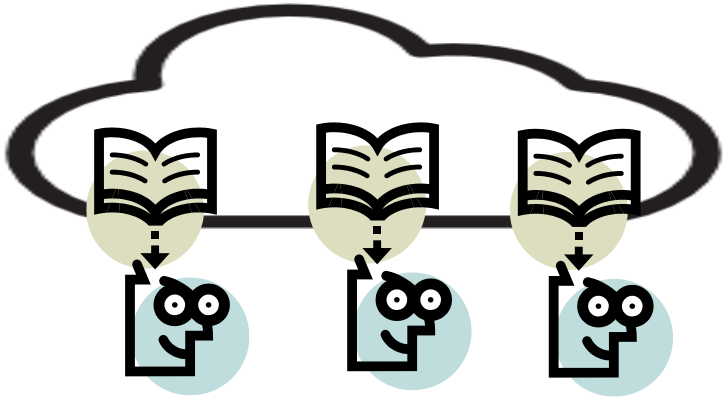
実機なしでロボット制御方法を学べます

ロボット用スペースを用意する必要がありません



思いついたアイディアをすぐに試せます

オンラインでロボット教育できます



EV3RTを使用してロボットの基本操作を学ぶ

アクチュエータ操作

HackEVを操作して，走る･曲がる･止まるを実現する

センサ操作

HackEVの様々なセンサの値を見てみよう

演習

簡単なロボット制御をやってみよう

HackEVを操作して，走る･曲がる･止まるを実現する

■プログラム構成
編集対象：app.c
タスク名　：main_task
実行周期：100msec

VScode で app.c を開き，main_taskを探してください

[コード説明]
アクチュエータ/センサの　ポート設定部分

```
312 void main_task(intptr_t unused) {
313     ev3_sensor_config(color_sensor, COLOR_SENSOR);
314     ev3_sensor_config(ultrasonic_sensor, ULTRASONIC_SENSOR);
315     ev3_sensor_config(touch_sensor0, TOUCH_SENSOR);
316     ev3_sensor_config(touch_sensor1, TOUCH_SENSOR);
317     ev3_motor_config(left_motor, LARGE_MOTOR);
318     ev3_motor_config(right_motor, LARGE_MOTOR);
319     ev3_motor_config(arm_motor, LARGE_MOTOR);
320
321     syslog(LOG_NOTICE, "#### motor control start");
322     while(1) {
323         /*
324          * ここに制御プログラムを入れてください
325          */
326         tslp_tsk(100000); /* 100msec */
327     }
328 }
```

[コード説明]
100msec周期の
制御処理実行部分
※ここに制御プログラム
※を追加しましょう

HackEVを操作して，走る･曲がる･止まるを実現する

■走る
[API]
ev3_motor_set_power
第一引数：モータ･ポート番号
第二引数：モータ･パワー

[対象モータ]
・左右のモータ
・アームのモータ

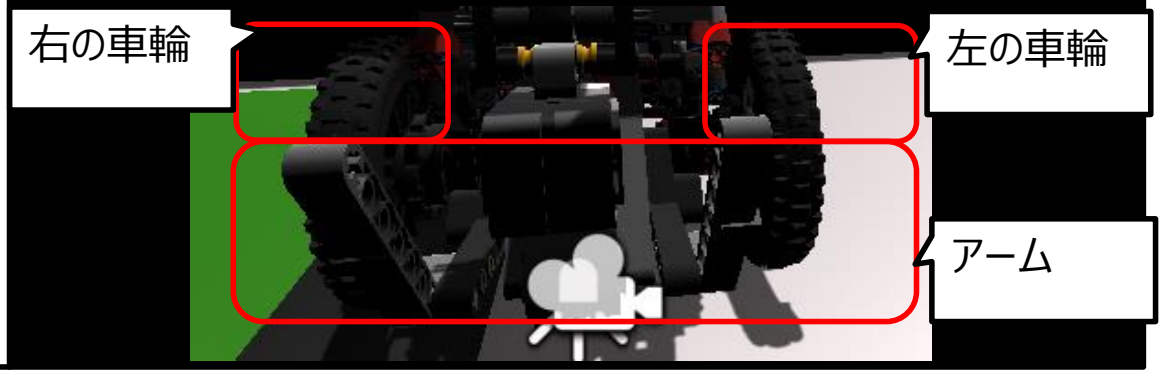
[詳細説明]
https://www.toppers.jp/ev3pf/EV3RT_C_API_Reference/group__ev3motor.html#gaf5c96b6a7e3fdc7e13ba3466359294a0

[補足]
ev3_motor_steerでも走るを実現できます

■左右モータ

[前進]
```
ev3_motor_set_power(left_motor,  5);
ev3_motor_set_power(right_motor, 5);
```
[後進]
```
ev3_motor_set_power(left_motor,  -5);
ev3_motor_set_power(right_motor, -5);
```
■アームモータ

[UP]
```
ev3_motor_set_power(arm_motor,  5);
```
[DOWN]
```
ev3_motor_set_power(arm_motor,  -5);
```



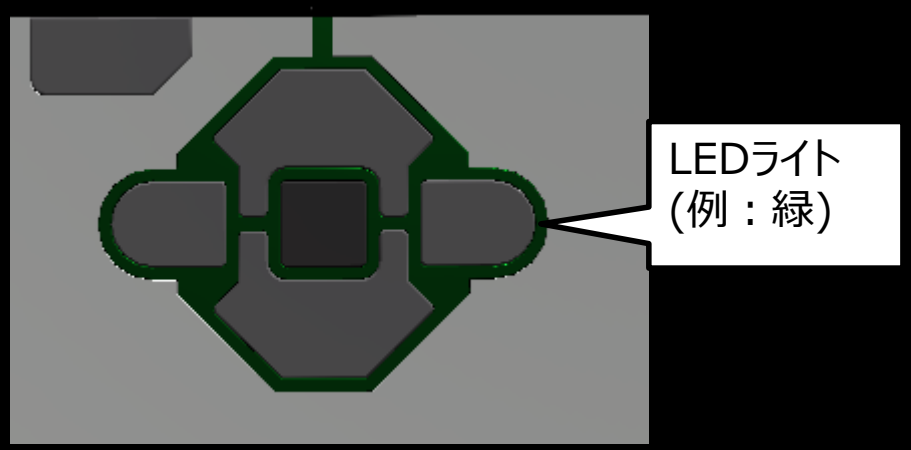
HackEVを操作して，走る･曲がる･止まるを実現する

■LEDライト
[API]
ev3_led_set_color
第一引数：LEDカラー

[対象]
・LED

[詳細説明]
https://www.toppers.jp/ev3pf/EV3RT_C_API_Reference/group__ev3led.html#gaa40eed1dd40c81573646f95ec8c427cb

■**LEDライト**
[LEDオフ]
ev3_led_set_color(LED_OFF);
[赤]
ev3_led_set_color(LED_RED);
[緑]
ev3_led_set_color(LED_GREEN);
[オレンジ]
ev3_led_set_color(LED_ORANGE);



HackEVを操作して，走る･曲がる･止まるを実現する

■走る
１０秒前進
１０秒後進
１０秒アームUP
１０秒アームDOWN

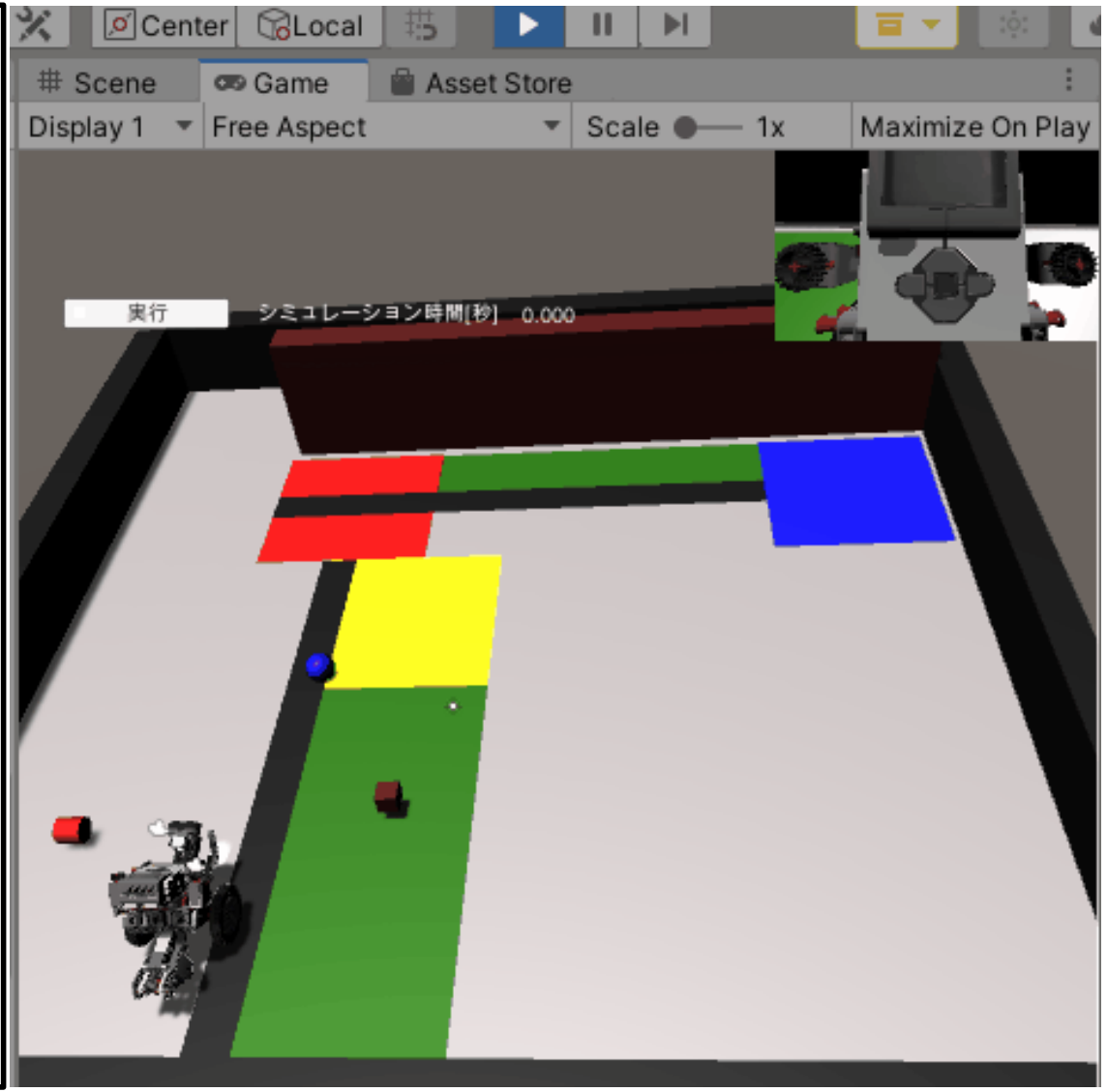
■プログラム例

```
int count = 0;
ev3_motor_stop(left_motor, true);
ev3_motor_stop(right_motor, true);
while(1) {
    /*
     * ここに制御プログラムを入れてください
     */
    if (count <= 100) {
        ev3_motor_set_power(left_motor, 3);
        ev3_motor_set_power(right_motor, 3);
    }
    else if ((count > 100) && (count <= 200)) {
        ev3_motor_set_power(left_motor, -3);
        ev3_motor_set_power(right_motor, -3);
    }
    else if ((count > 200) && (count <= 300)) {
        ev3_motor_set_power(left_motor, 0);
        ev3_motor_set_power(right_motor, 0);
        ev3_motor_set_power(arm_motor, 1);
    }
    else if ((count > 300) && (count <= 400)) {
        ev3_motor_set_power(left_motor, 0);
        ev3_motor_set_power(right_motor, 0);
        ev3_motor_set_power(arm_motor, -1);
    }
    else {
        /* nothing to do */
    }
    count++;
    tslp_tsk(100000); /* 100msec */
}
```

動作イメージ(モータ)

■動作イメージ
　１０秒前進
　１０秒後進
　１０秒アームUP
　１０秒アームDOWN

[補足]
Unityのシミュレーション精度が粗い(1msecより大きい)と，モータ停止させた後も若干動いてしまう場合があります



ロボット教育演習向けコンテンツ

アクチュエータ操作

HackEVを操作して，走る･曲がる･止まるを実現する

↓

センサ操作

HackEVの様々なセンサの値を見てみよう

↓

演習

簡単なロボット制御をやってみよう

HackEVの様々なセンサの値を見てみよう

■カラーセンサ

[API]

ev3_color_sensor_get_color

第一引数：センサ・ポート番号
戻り値：識別したカラー※
※無色，黒，青，緑，黄，赤，白，茶

ev3_color_sensor_get_reflect

第一引数：センサ・ポート番号
戻り値：反射光の強さ(0～100)

[対象]

・カラーセンサ

[詳細説明]

https://www.toppers.jp/ev3pf/EV3RT_C_API_Reference/group__ev3sensor.html#ga776b35d98fd73550278e27ee39a3d3aa

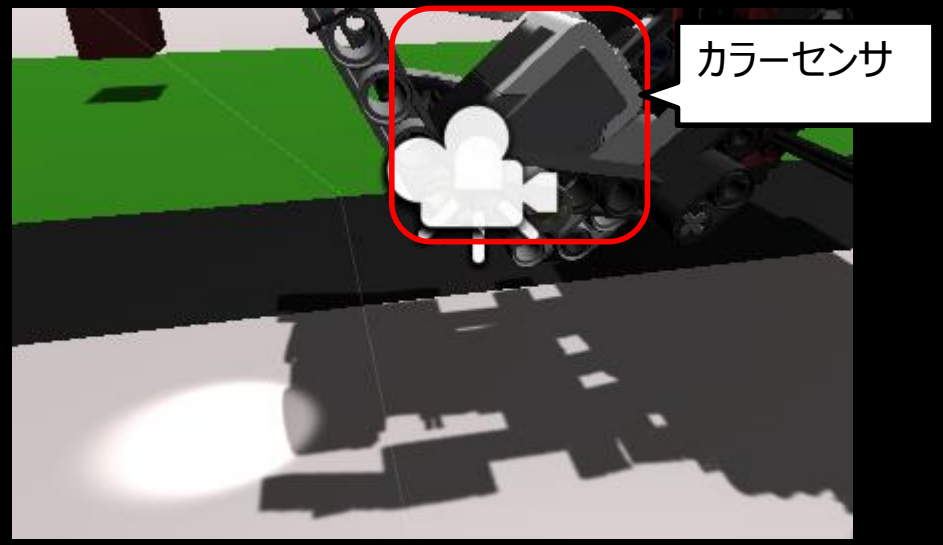
■カラーセンサ

[カラー]

```
ev3_color_sensor_get_color(color_sensor);
```

[反射光]

```
ev3_color_sensor_get_reflect (color_sensor);
```



センサ値のデバッグ出力例：

```
syslog(LOG_NOTICE,
  "color_value=%d reflect_value=%d",
   color_value, reflect_value);
```

HackEVの様々なセンサの値を見てみよう

■超音波センサ

[API]

ev3_ultrasonic_sensor_get_distance

第一引数：センサ・ポート番号

戻り値：距離(単位はセンチメートル)

[対象]

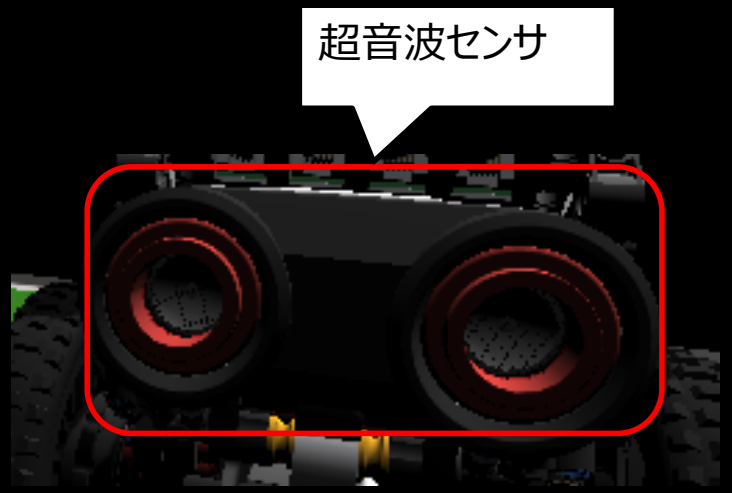
・超音波センサ

[詳細説明]

https://www.toppers.jp/ev3pf/EV3RT_C_API_Reference/group__ev3sensor.html#ga956536aaab7677c2b512b2d3bd4e498f

■超音波センサ

ev3_ultrasonic_sensor_get_distance (ultrasonic_sensor);



センサ値のデバッグ出力例：

```
syslog(LOG_NOTICE,
  “ultrasonic_value=%d", ultrasonic_value);
```

HackEVの様々なセンサの値を見てみよう

■タッチセンサ

[API]

ev3_touch_sensor_is_pressed

第一引数：センサ・ポート番号

戻り値：接触有無

[対象]

・タッチセンサ

[詳細説明]

https://www.toppers.jp/ev3pf/EV3RT_C_API_Reference/group__ev3sensor.html#ga0bd7ec7b7ac2fbaaa6cbe46eff3a829d

■タッチセンサ

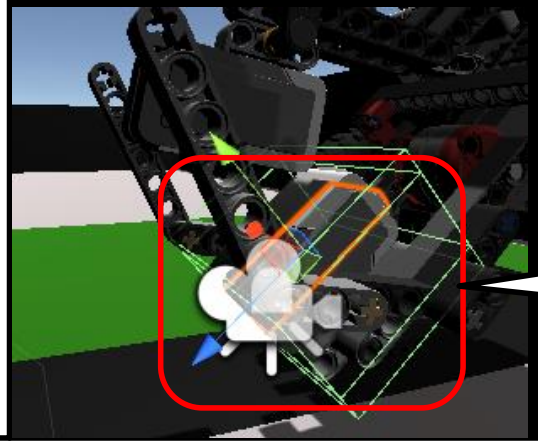
[左肩側のタッチセンサ]

ev3_touch_sensor_is_pressed (touch_sensor0);

[アーム側のタッチセンサ]

ev3_touch_sensor_is_pressed (touch_sensor1);

左側のタッチセンサ
クリック操作を検出



アーム側のタッチセンサ
障害物と接触すると検出

ロボット教育演習向けコンテンツ

アクチュエータ操作

HackEVを操作して，走る･曲がる･止まるを実現する

↓

センサ操作

HackEVの様々なセンサの値を見てみよう

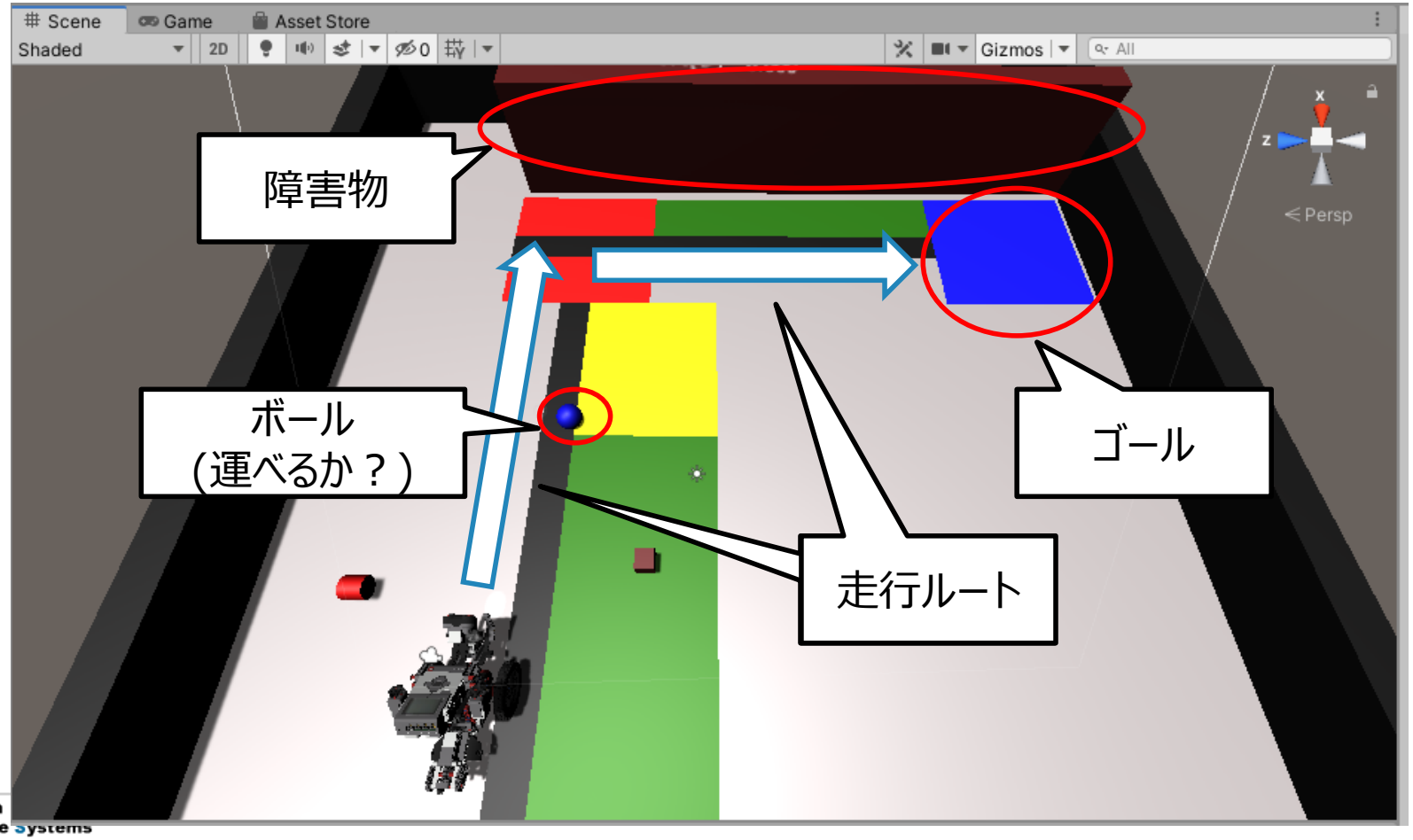
↓

演習

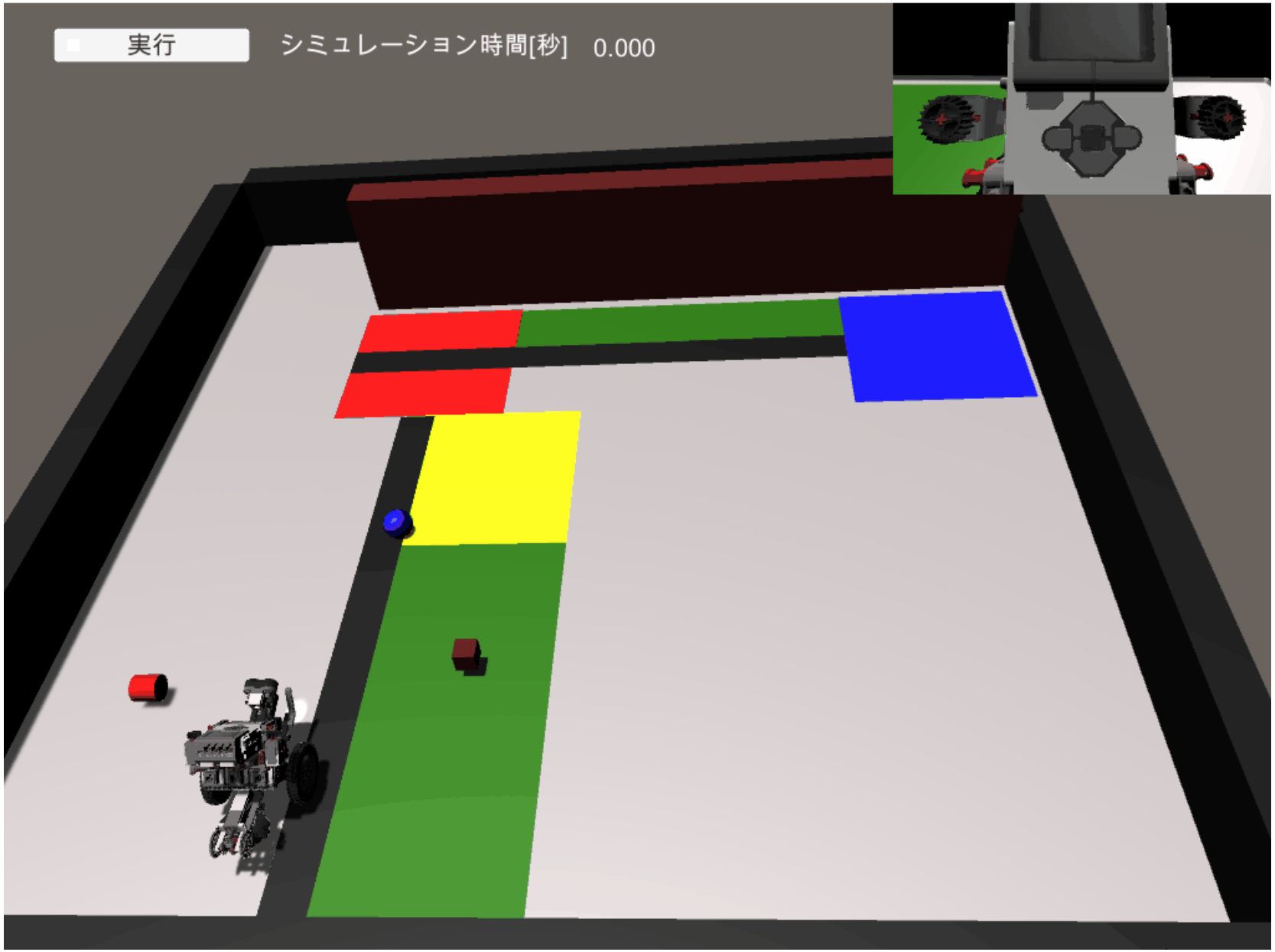
簡単なロボット制御をやってみよう

簡単なロボット制御をやってみよう

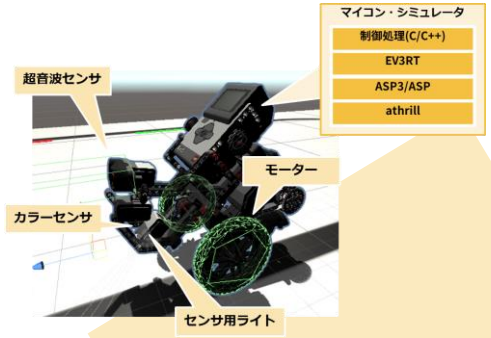
フロアの色，障害物までの距離を各種センサで検出して，ゴールまでたどり着こう.
[可能なら]
・ロボットの内部状態でLEDカラーで変化させてみよう
・地面に落ちているボールを運んでみよう



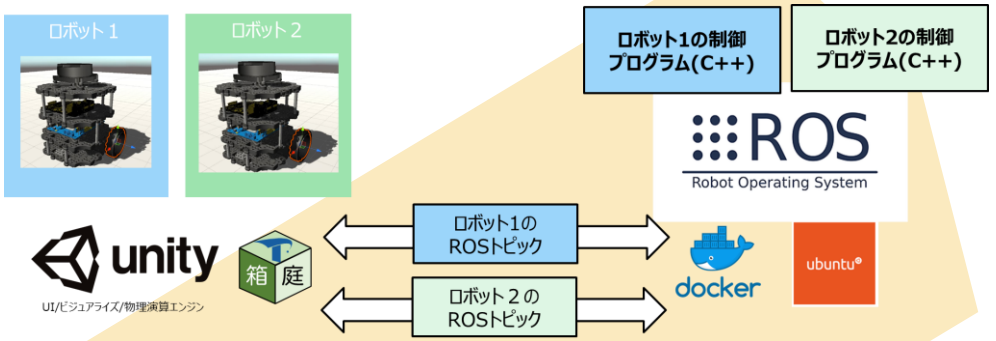
実行例



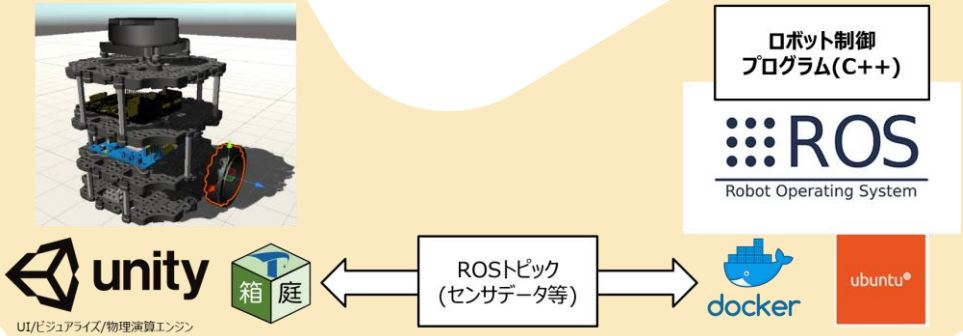
３つのプロトタイプモデル



マイコン制御ロボットシミュレーション



複数ロボットの連携制御シミュレーション



ROS制御ロボットシミュレーション

TOPPERS
Toyohashi OPen Platform
for Embedded Real-time Systems

ROS制御ロボットシミュレーション

■狙い

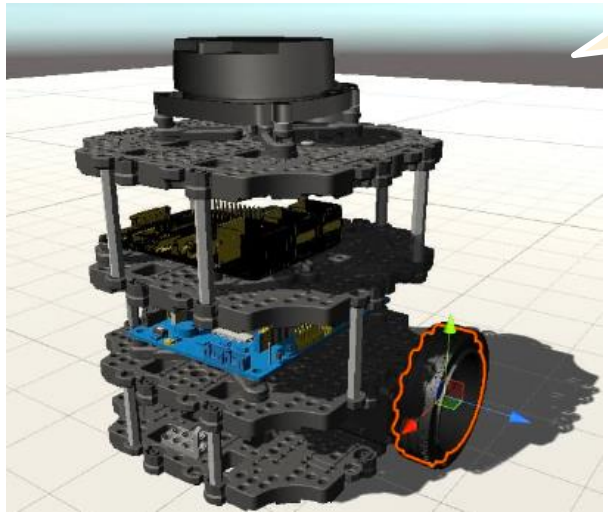
Unity上のロボット(1個)とROS2を連携させたシミュレーションの実現

■構成要素

- ロボット本体
  - ・ROSでよく利用されているTB3
- ロボットを制御するROSプログラム
  - ・ROS2でロボット制御
- 外部環境
  - ・走行コース/障害物等

■広報活動

- ROSユーザ層に箱庭を広める

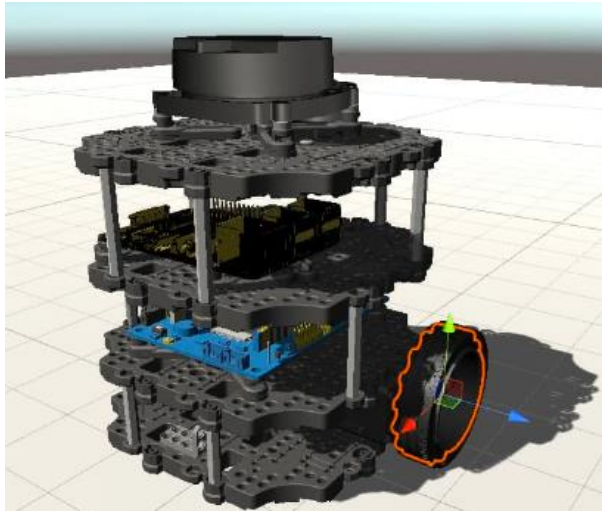


ROS2プログラム

- 制御処理(C/C++)
- ROS2 API
- Ubuntu
- Docker

ROS制御ロボットシミュレーションの構造

- Unity上の１台のTurtlebot3をROS2で動かします



ロボット制御
プログラム(C++)

ROS
Robot Operating System

unity
UI/ビジュアライズ/物理演算エンジン

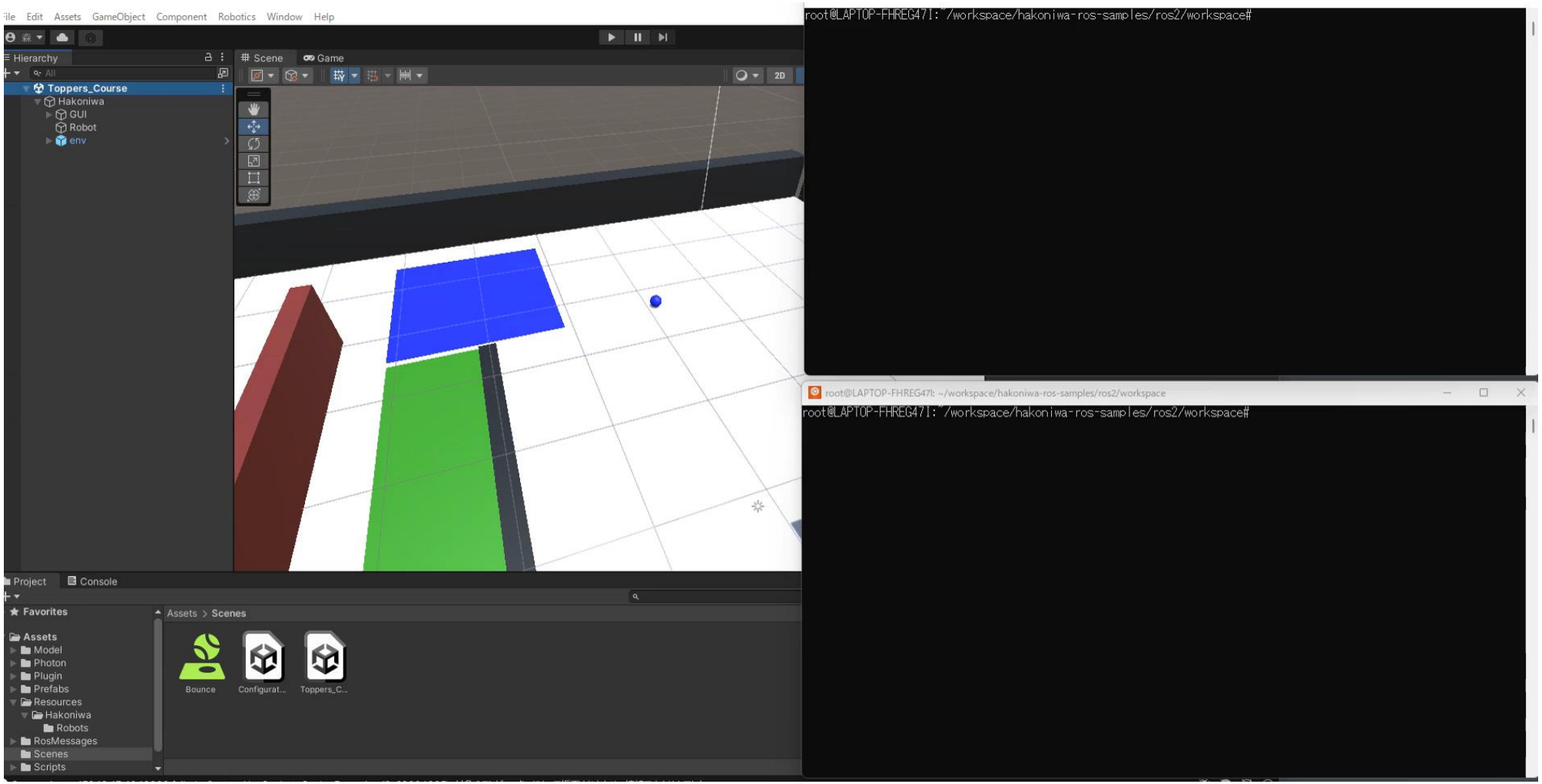
箱庭

ROSトピック
(センサデータ等)

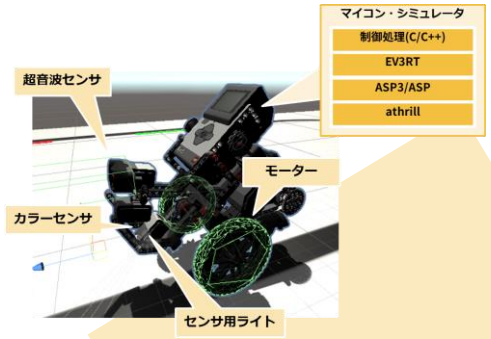
docker

ubuntu®

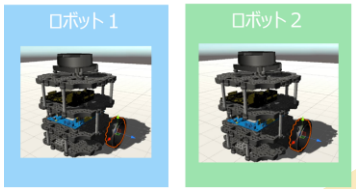
1個のロボットを動かす



３つのプロトタイプモデル



マイコン制御ロボットシミュレーション



ロボット1の制御プログラム(C++)

ロボット2の制御プログラム(C++)

ROS
Robot Operating System

unity
UI/ビジュアライズ/物理演算エンジン

箱庭

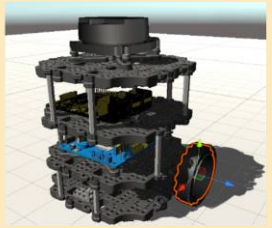
ロボット１の ROSトピック

ロボット２の ROSトピック

docker

ubuntu

複数ロボットの連携制御シミュレーション



ロボット制御プログラム(C++)

ROS
Robot Operating System

unity
UI/ビジュアライズ/物理演算エンジン

箱庭

ROSトピック (センサデータ等)

docker

ubuntu

ROS制御ロボットシミュレーション

TOPPERS
Toyohashi OPen Platform for Embedded Real-time Systems

複数ロボットの連携制御シミュレーション

■狙い

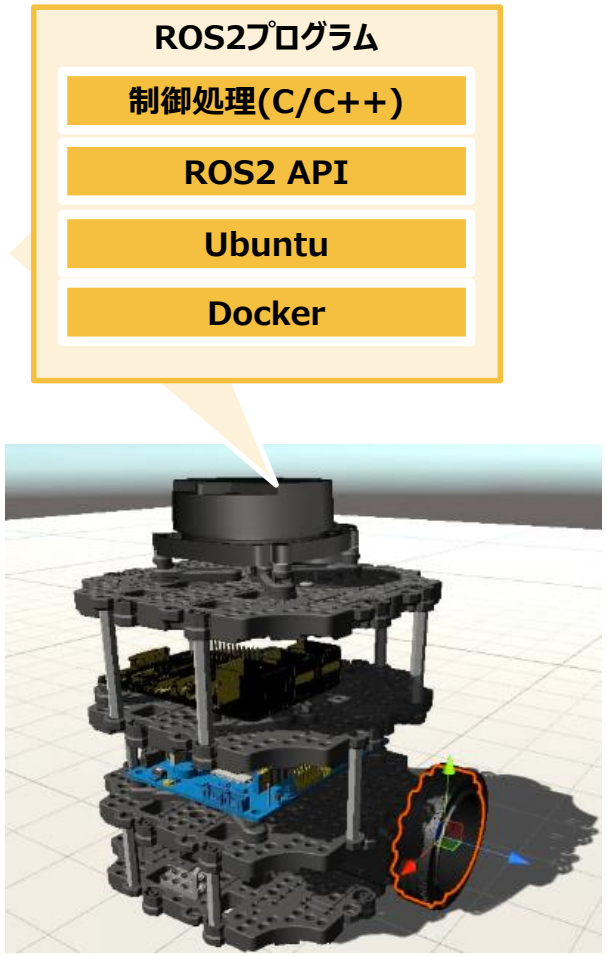
Unity上のロボット(複数)とROS2を連携させたシミュレーションの実現

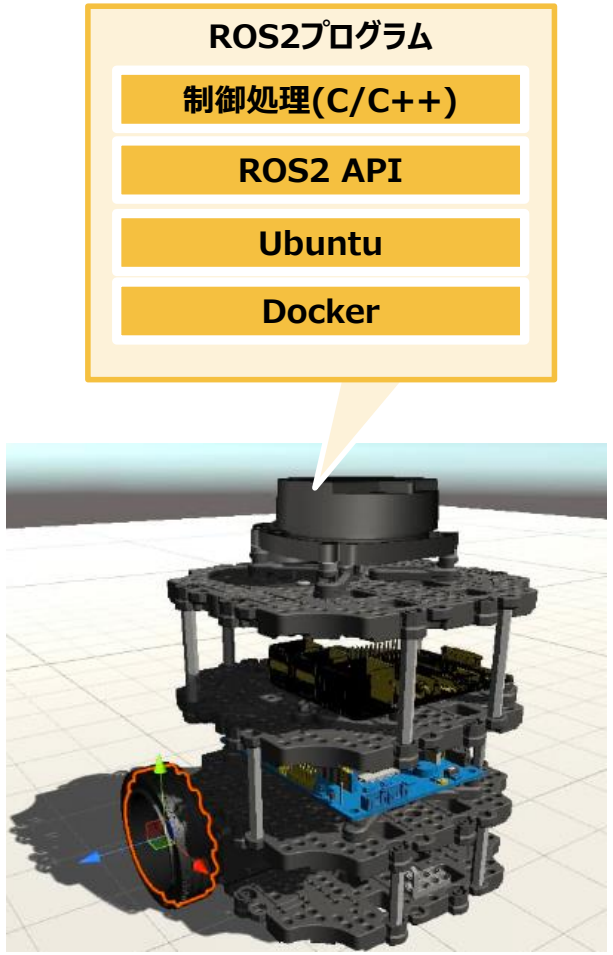
■構成要素

- ロボット本体
  - ・ROSでよく利用されているTB3
- ロボットを制御するROSプログラム
  - ・ROS2でロボット制御
- 外部環境
  - ・走行コース/障害物等

■広報活動

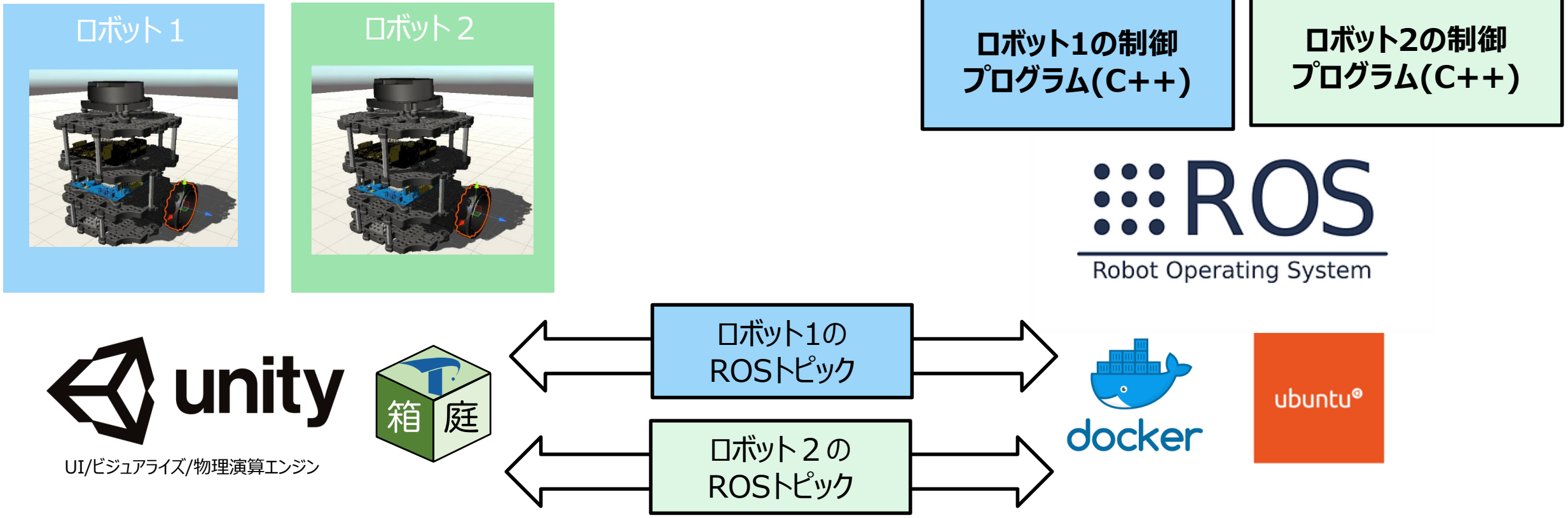
- ROSユーザ層に箱庭を広める





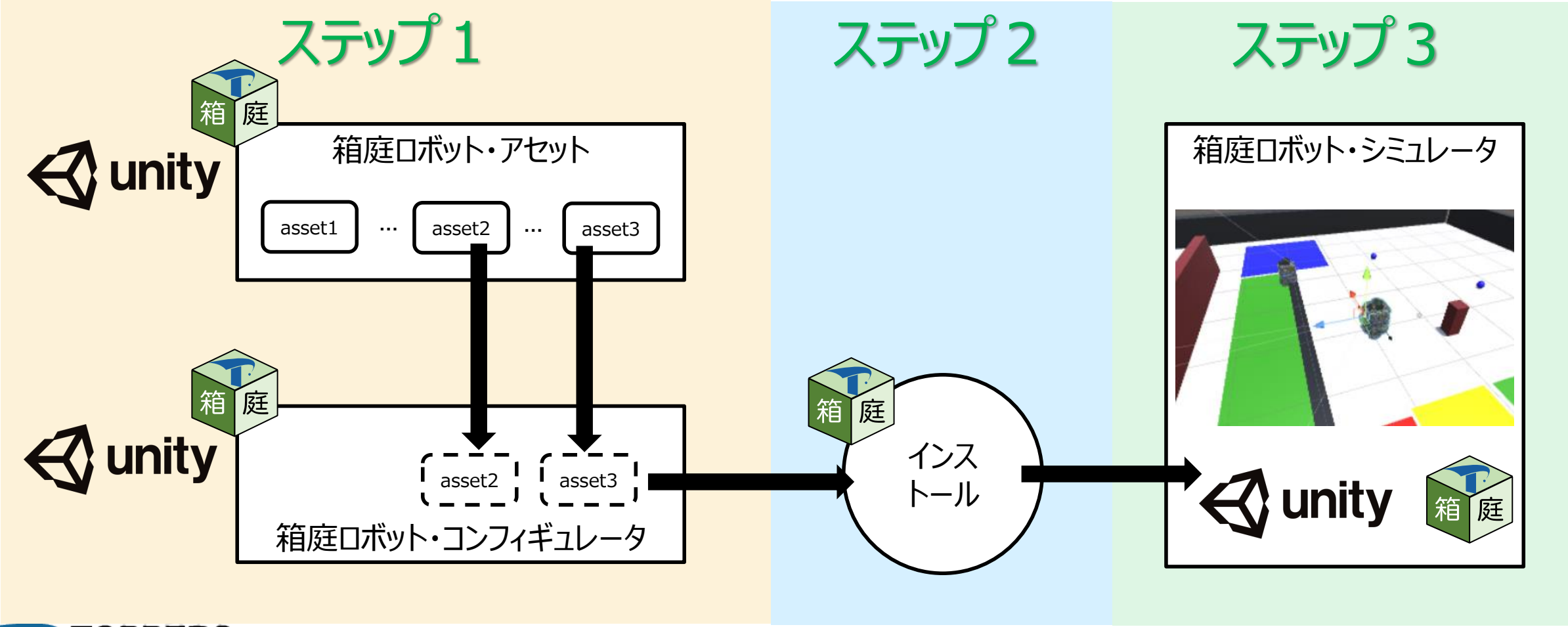
複数のロボットを動かす

- Unity上の２台のTurtlebot3をROS2で動かします(何台でもいけます)

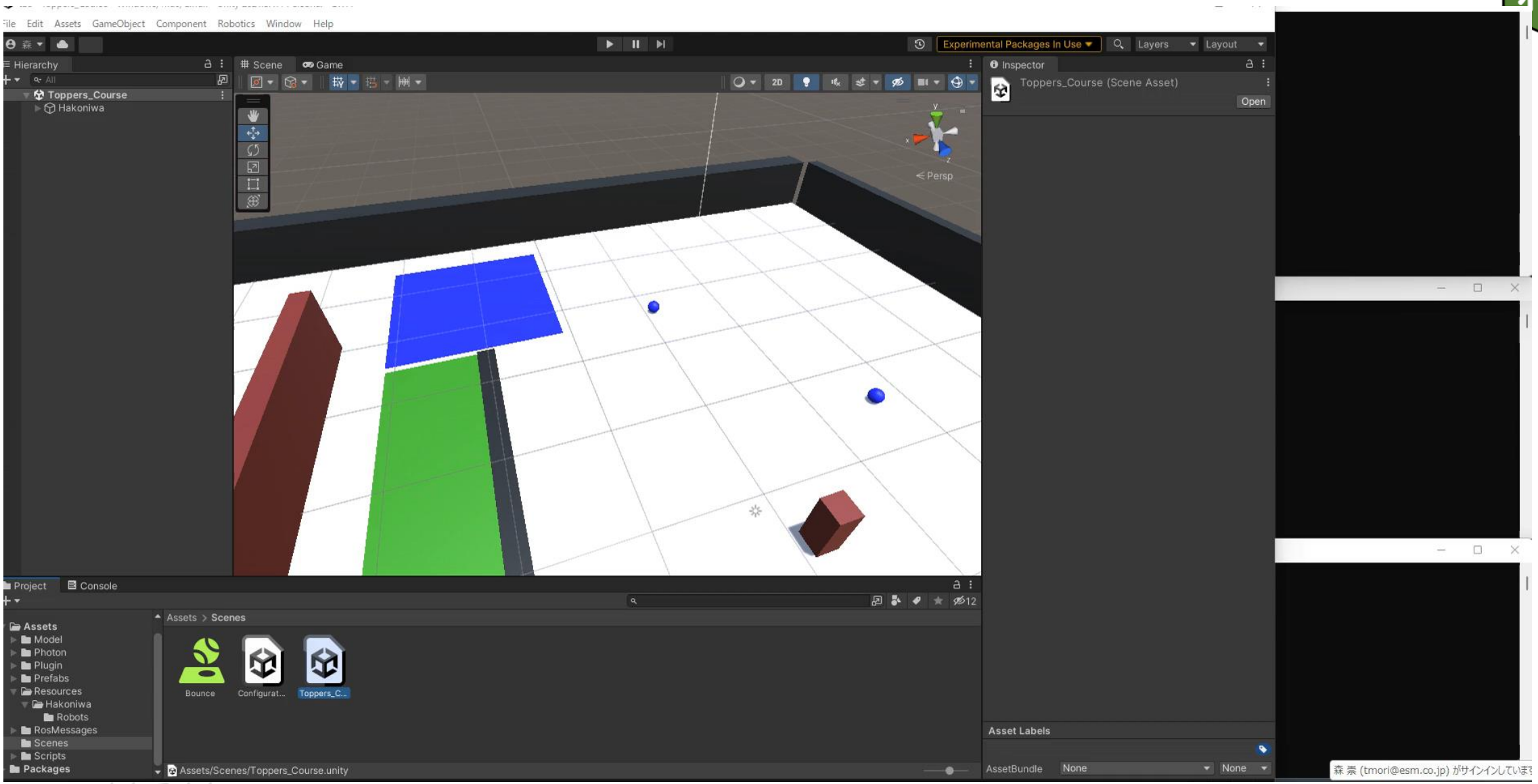


複数のロボットを動かすための手順

- たったの３ステップで，複数のロボット配置してシミュレーション実行できます！



複数のロボットを動かす

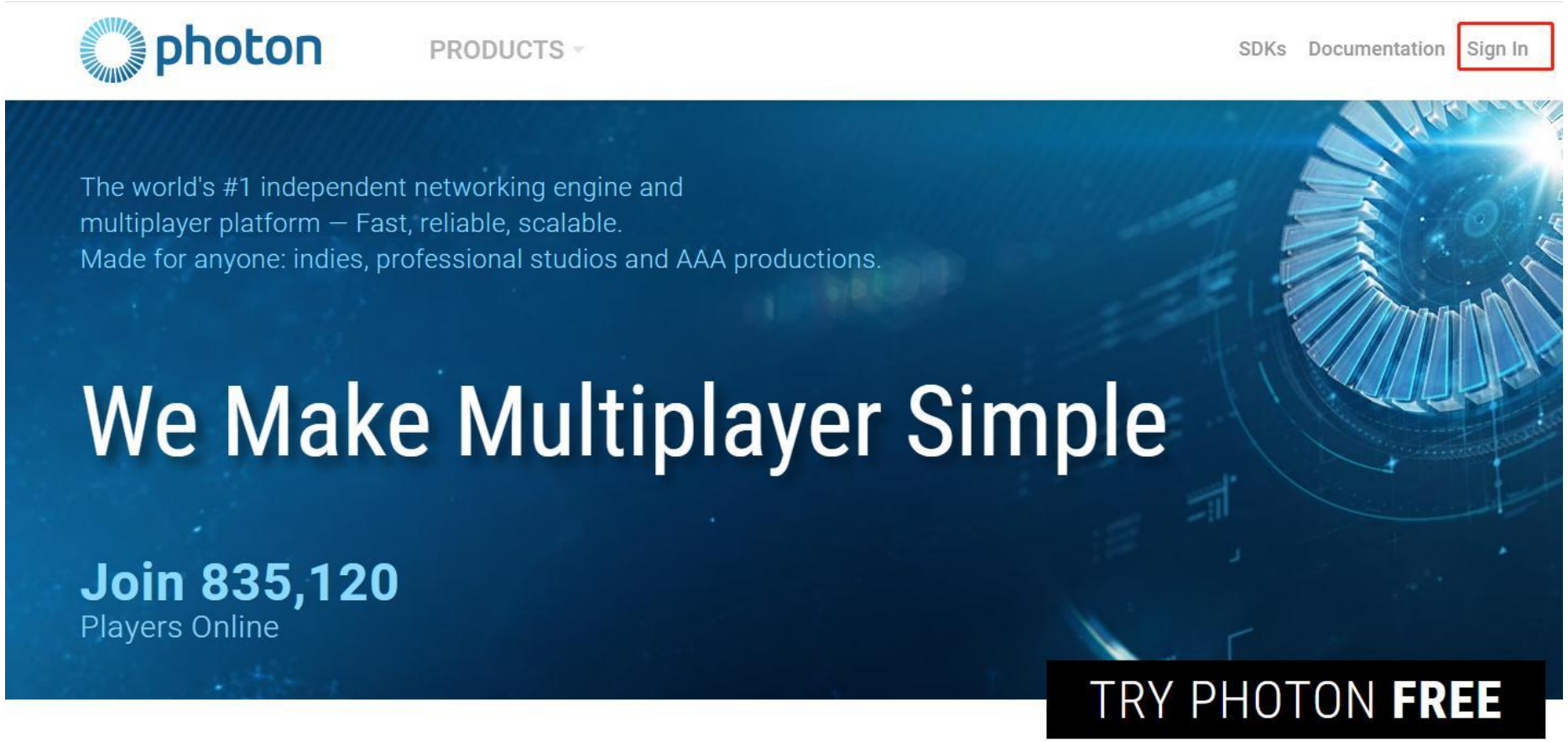


複数ロボットの連携制御シミュレーションの発展型

- Photonでロボット・マルチプレイする
- Oculus Quest2で箱庭VR空間にダイブする！

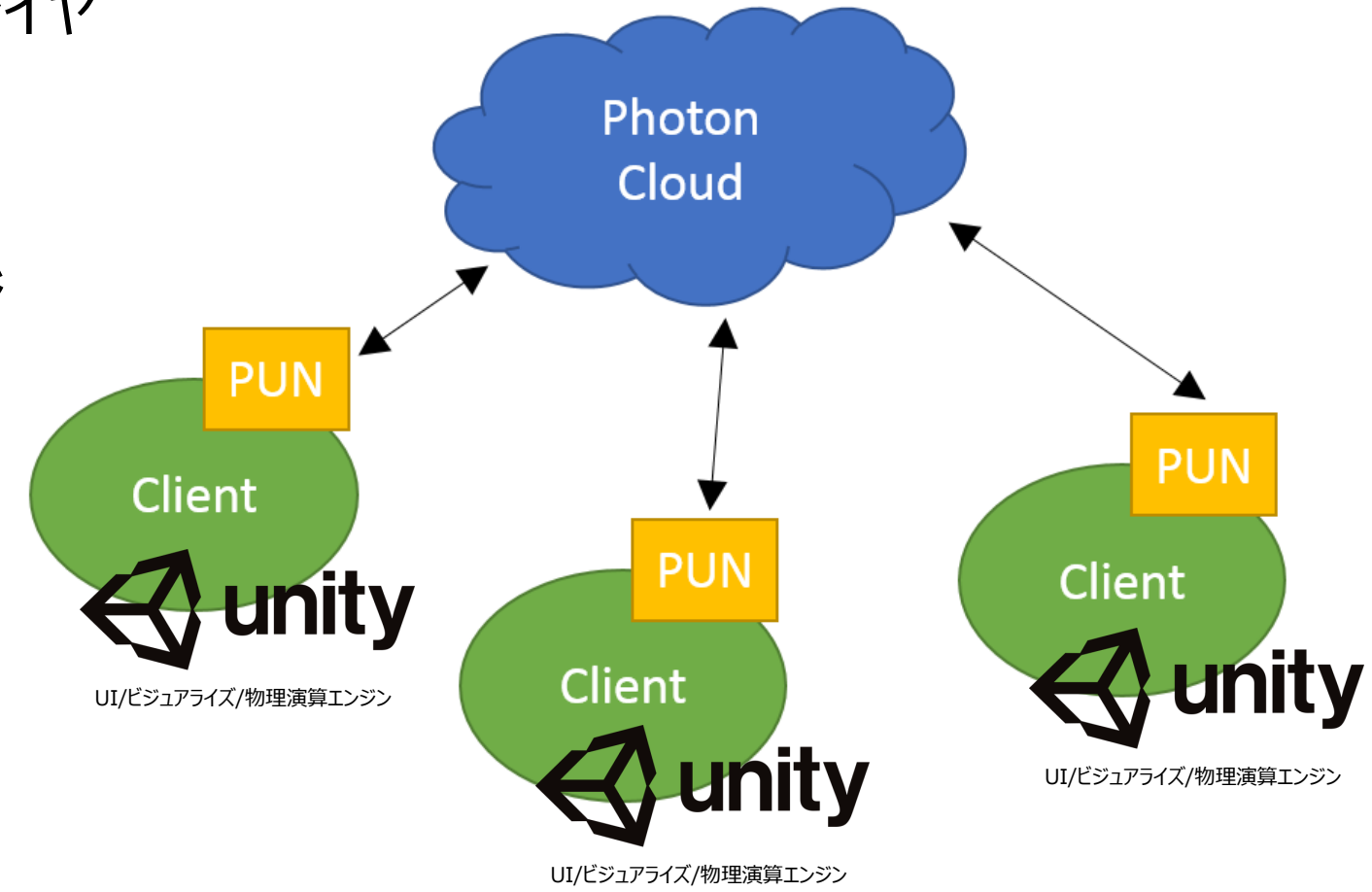
Photon ってなに？

100%フリーでマルチプレイゲームを実現できる！(MAX:20ユーザ)



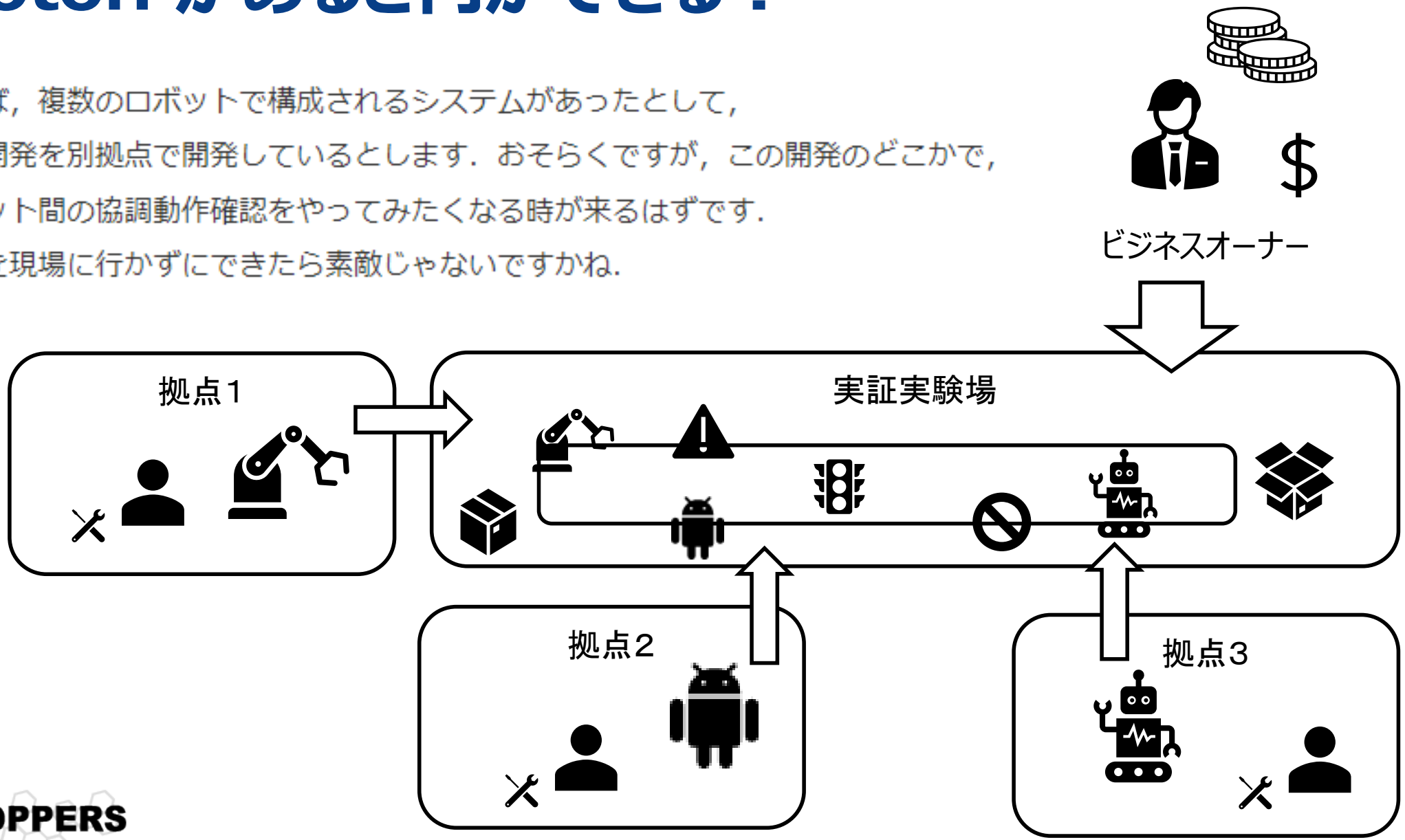
Photon アーキテクチャ

- Client
  - マルチプレイゲームに参加するプレイヤ
- PUN
  - Photon Unity Networking
  - マルチプレイ対応Unityパッケージ
- Photon Cloud
  - マルチプレイゲームを実現する
  - クラウドサービス
    - Client上のプレイヤを
    - クラウド上で同期する



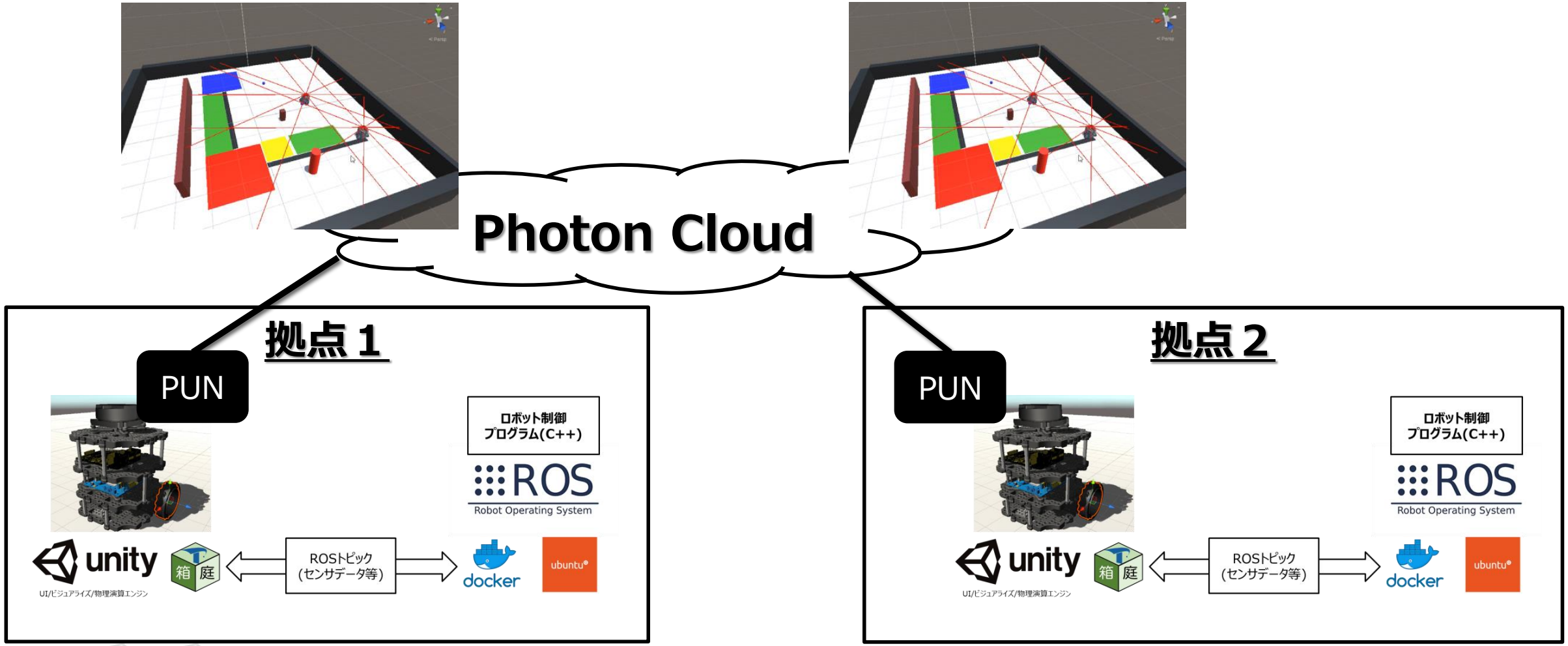
Photon があると何ができる？

例えば，複数のロボットで構成されるシステムがあったとして，
その開発を別拠点で開発しているとします．おそらくですが，この開発のどこかで，
ロボット間の協調動作確認をやってみたくなる時が来るはずです．
これを現場に行かずにできたら素敵じゃないですかね．



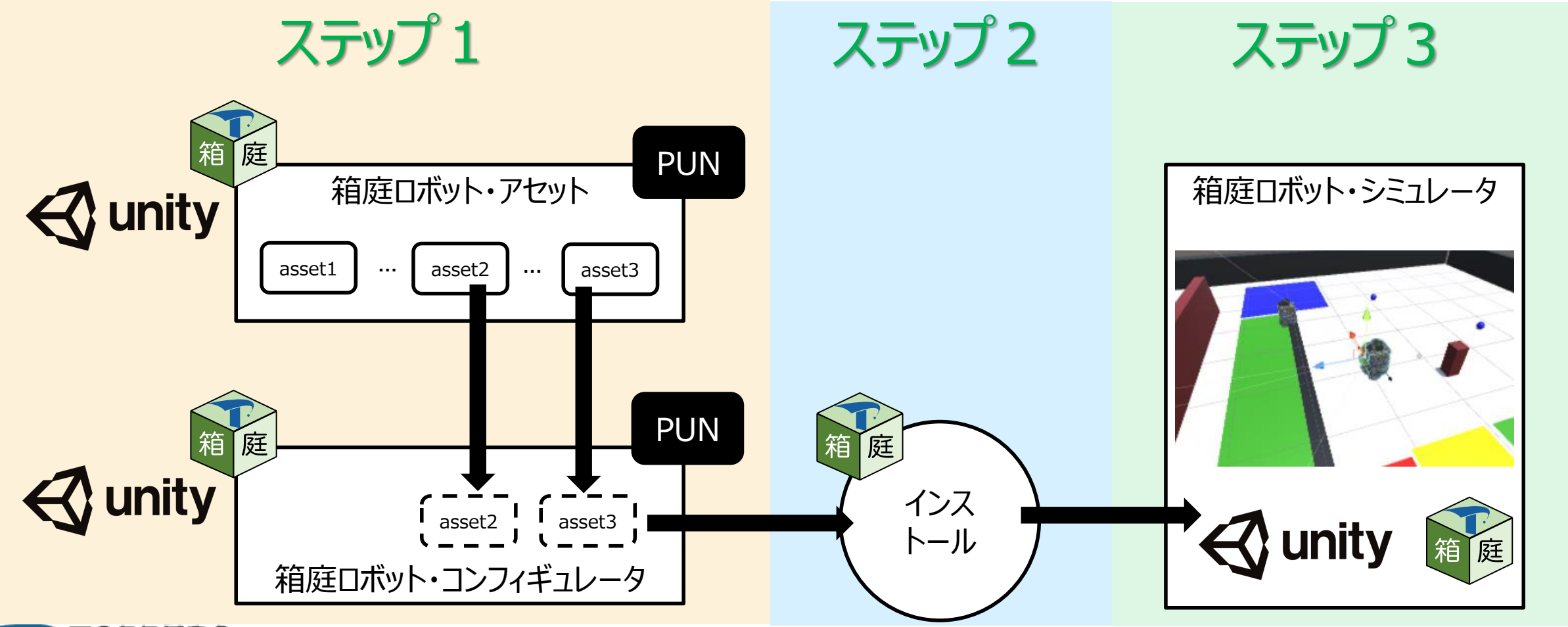
Photonでロボット・マルチプレイする

- 複数拠点にいる Turtlebot3 を Photon Cloud で同期します

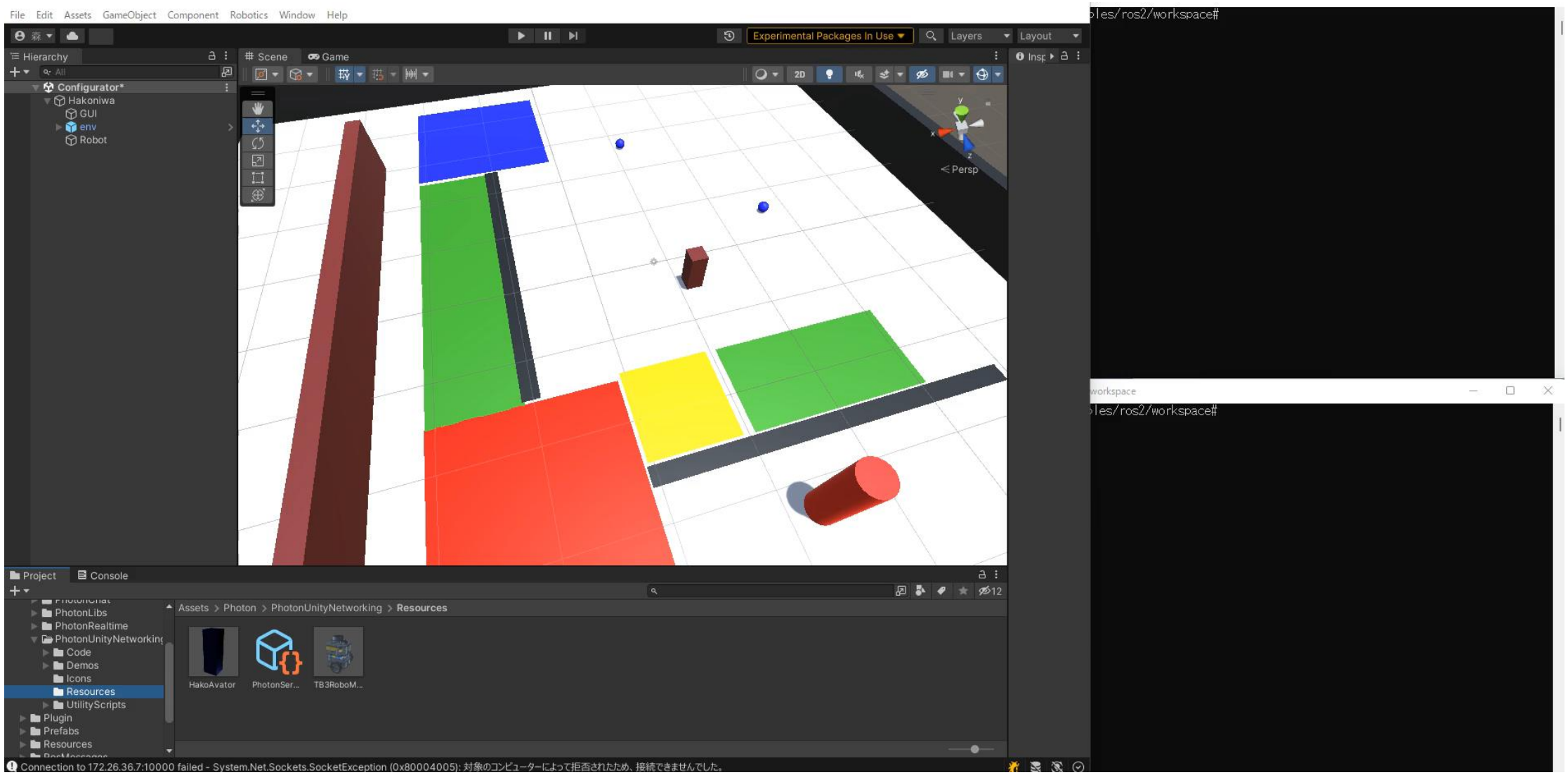


Photonでロボット・マルチプレイするための手順

- 複数のロボットの場合と同じです
  - Photon用のアセットを利用して，箱庭コードをGenerateする点が異なります



Photonでロボット・マルチプレイする



複数ロボットの連携制御シミュレーションの発展型

- Photonでロボット・マルチプレイする
- **Oculus Quest2で箱庭VR空間にダイブする！**

VRってなに？

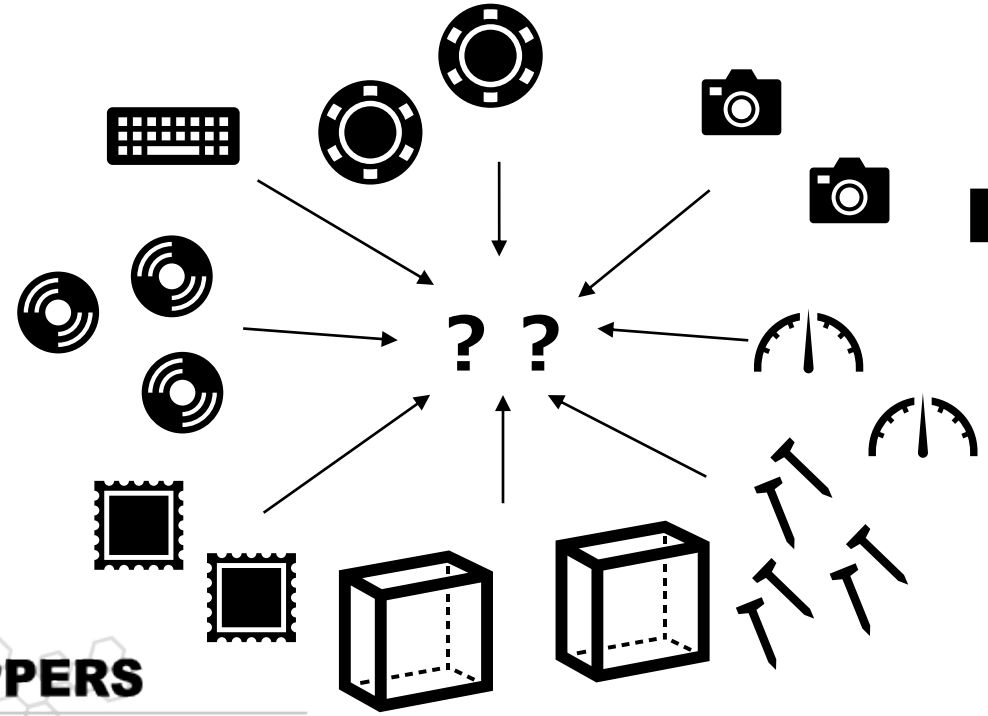
言わずと知れたバーチャルリアリティ(VR)
ゲームの世界では手軽に楽しめるようになった！

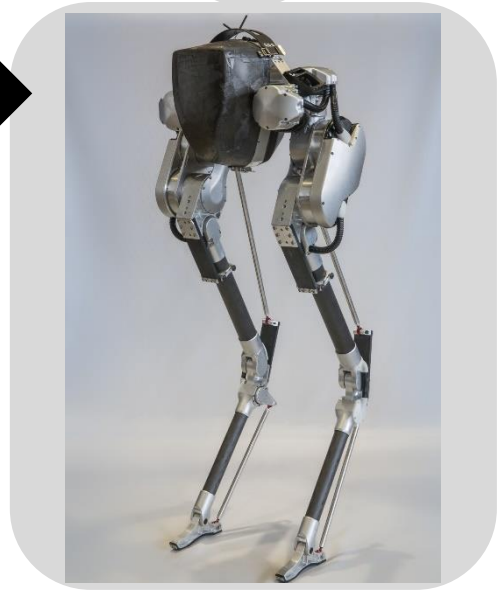
Oculus Quest2

3D空間にダイブして，臨場感のあるVR体験

VR(Oculus Quest2) があると何ができる？

例えば，ロボット開発において，分業・部品開発をやっていくと，
全体象が見えなくなると思います．全体としてこういうロボット作っていて，
この部品が動くとロボットはこう動きます．的なのをパワポとかで説明して
伝わるでしょうか．たぶんですが，現場で本物を見た方が早いですよね．
こういうのをVR空間上で手軽にやれたら素敵じゃないですかね．

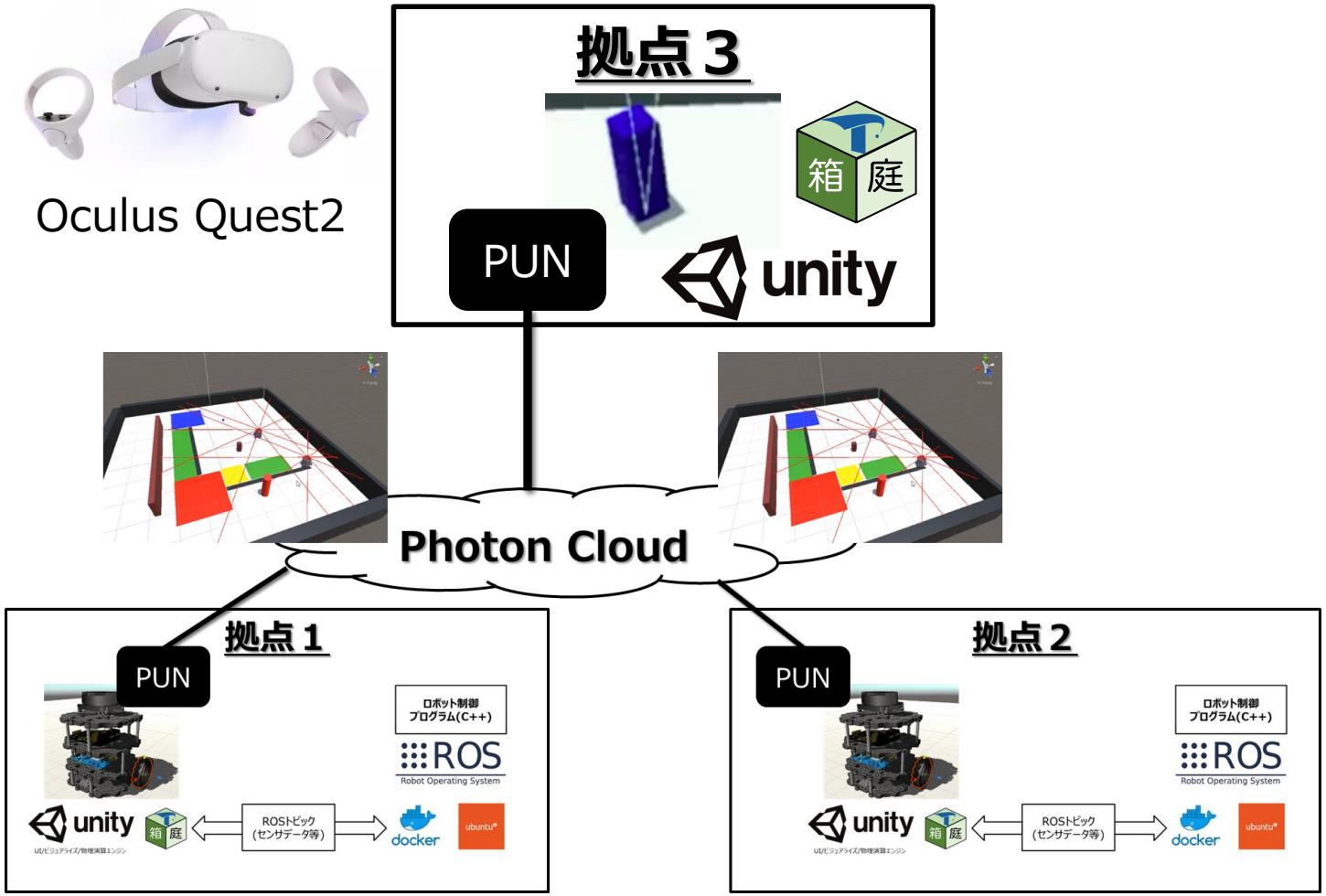




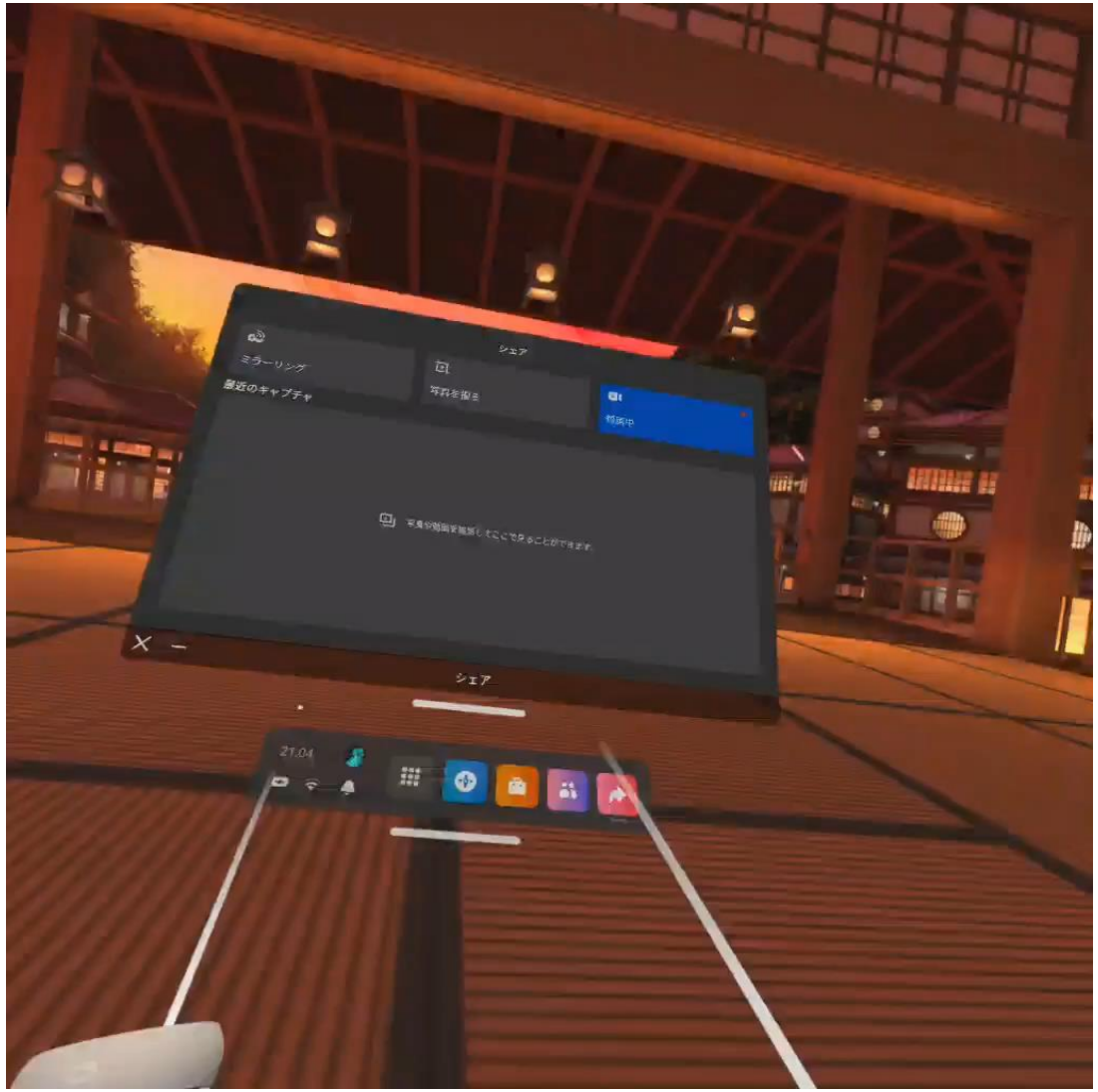
TOPPERS
Toyohashi OPen Platform for Embedded Real-time Systems

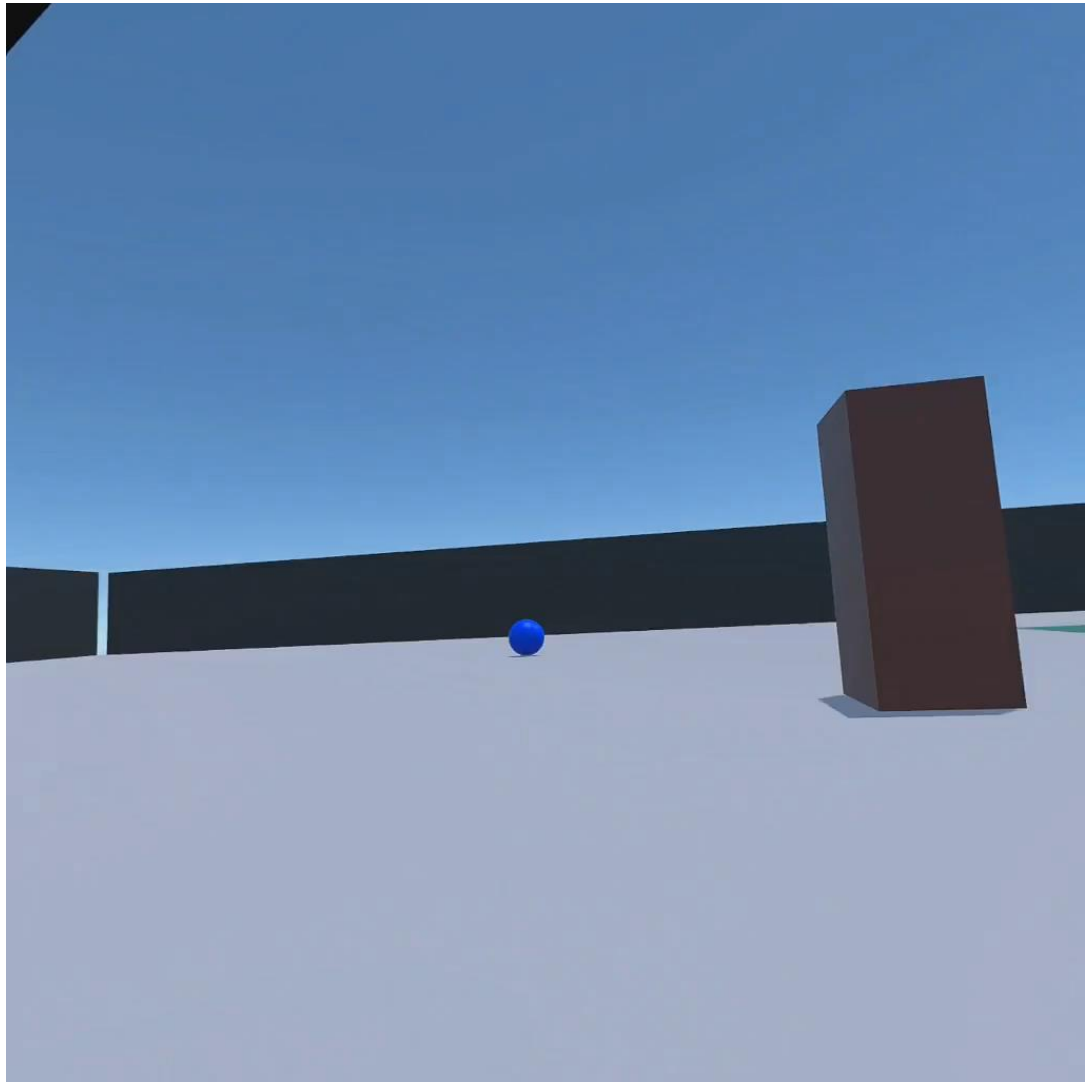
Oculus Quest2で箱庭VR空間にダイブする！

- 箱庭アバターを使って，Photon Cloudにログインして，VR体験します！



Oculus Quest2で箱庭VR空間にダイブする！





アジェンダ

1. 『箱庭』の狙いとコンセプト
2. 箱庭を実現する技術
3. 箱庭プロトタイプモデルとデモ
4. **全体アーキテクチャ**
5. 今後の方向性／ロードマップ
6. 箱庭の最新情報

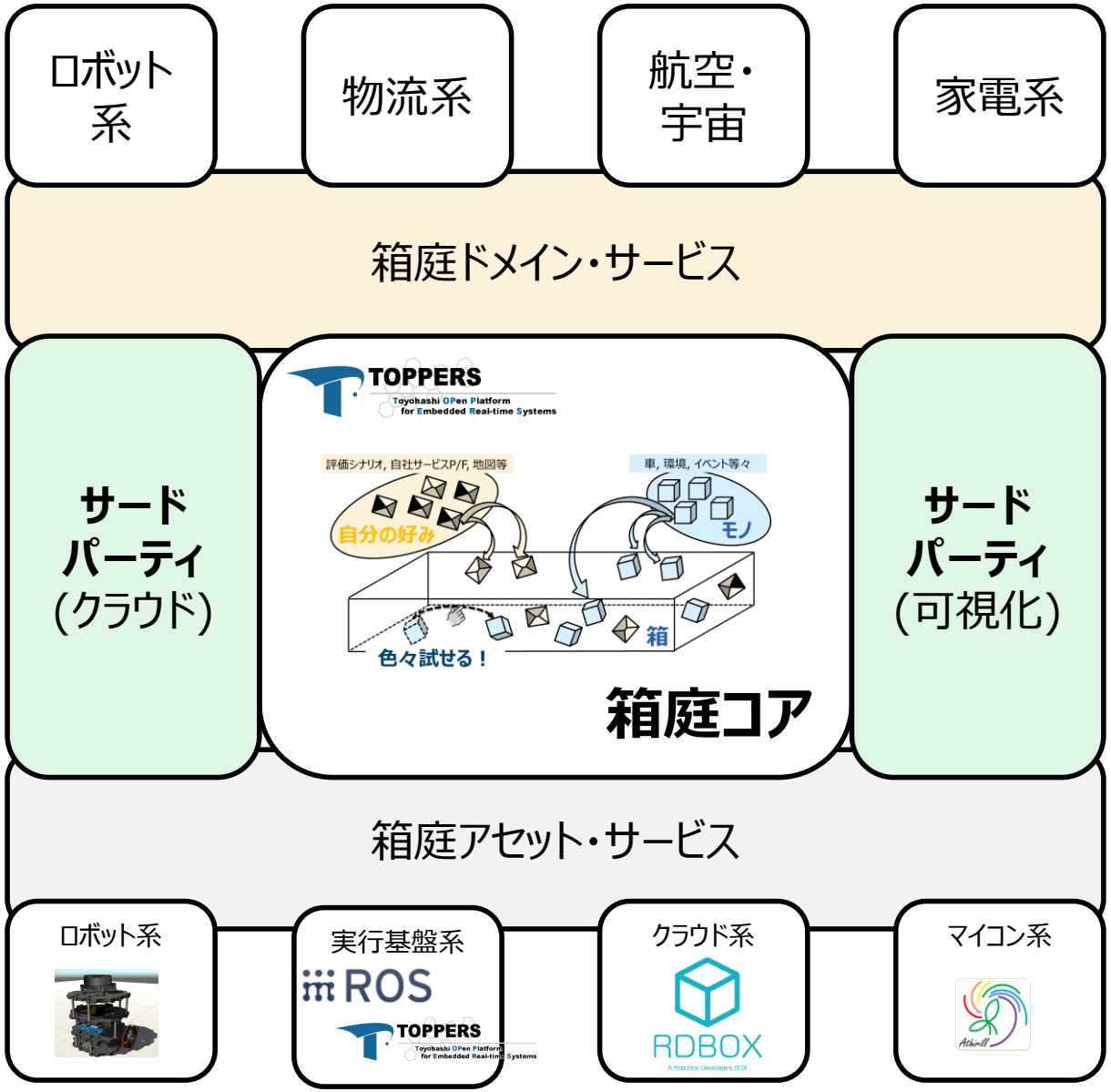
箱庭の目指すところとアーキテクチャ

■箱庭ドメイン・サービス
　様々な分野への適応を目指す

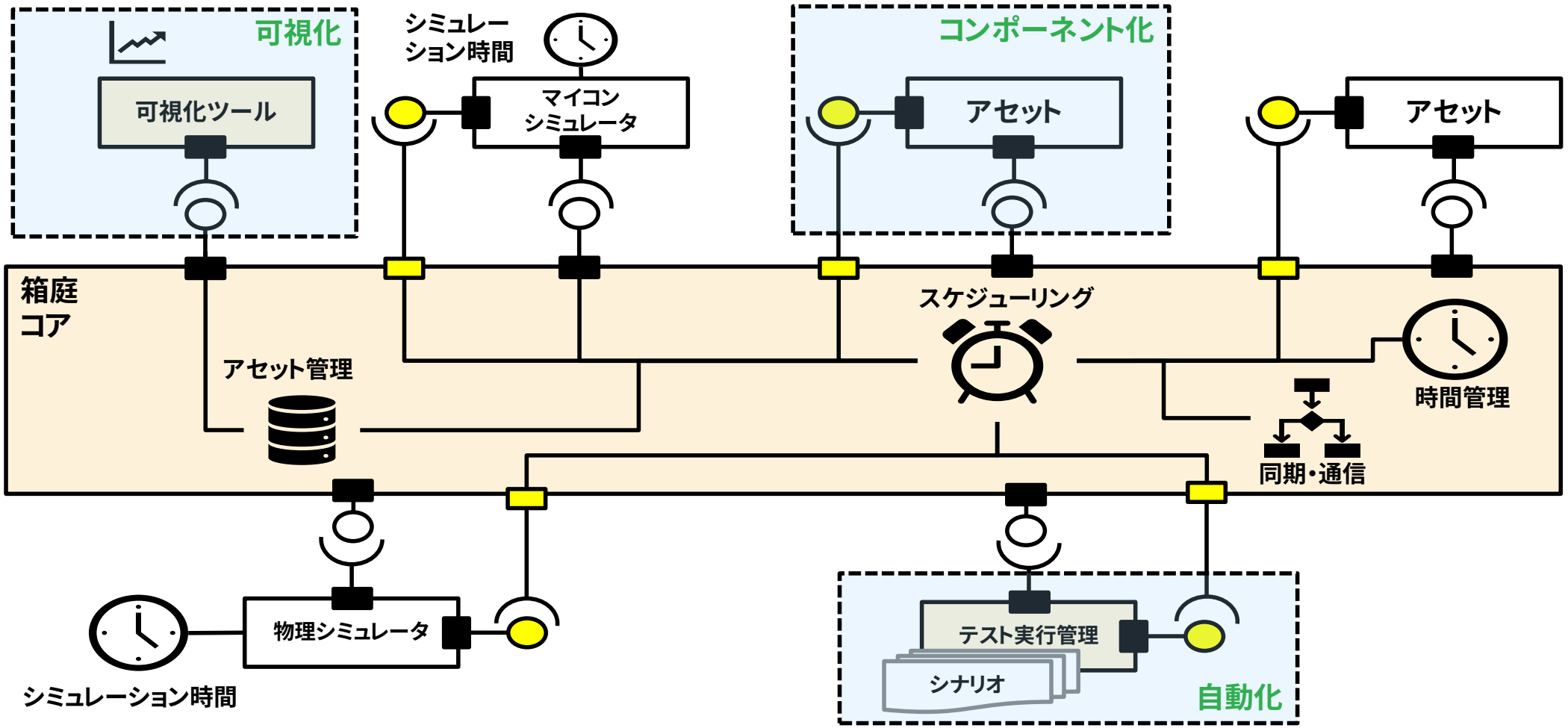
■箱庭コア
　箱庭固有のシミュレーション技術をコア技術化

■サードパーティ
　既存のサードパーティ製で出来ていることは積極利用

■箱庭アセット・サービス
　シミュレーション内の登場物を箱庭アセット化し，アセット数拡充を目指す



Hakoniwa Engine:
複雑なシステムを仮想環境で動作させる仕組み



アジェンダ

1. 『箱庭』の狙いとコンセプト
2. 箱庭を実現する技術
3. 箱庭プロトタイプモデルとデモ
4. 全体アーキテクチャと現在取り組んでいる内容
5. **今後の方向性／ロードマップ**
6. 箱庭の最新情報

今後の方向性／ロードマップ

2019−2021

箱庭プロトタイプ
モデル開発

結合環境構築
(箱庭ベース環境)

2022−2024

箱庭チュートリアル

組込み系エンジニア

ロボット制御系エンジニア

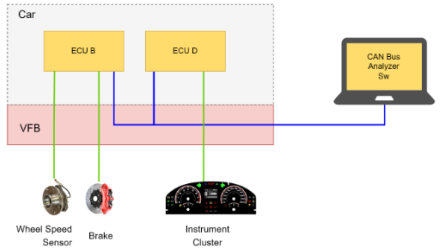
ロボットAI系エンジニア

箱庭ユーザを
広める活動

箱庭チュートリアル開発
&&広報活動

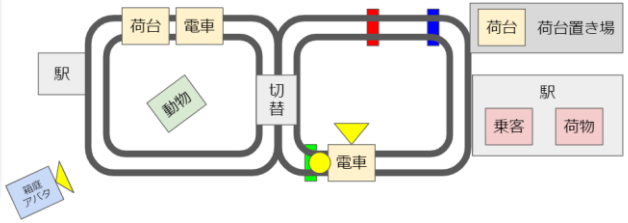
組込み系エンジニア

ECU/AUTOSAR開発入門シミュレーション環境



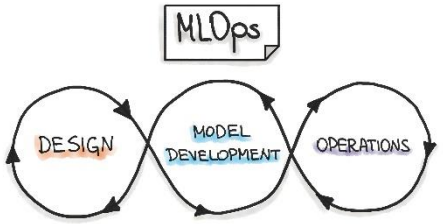
ロボット制御系エンジニア

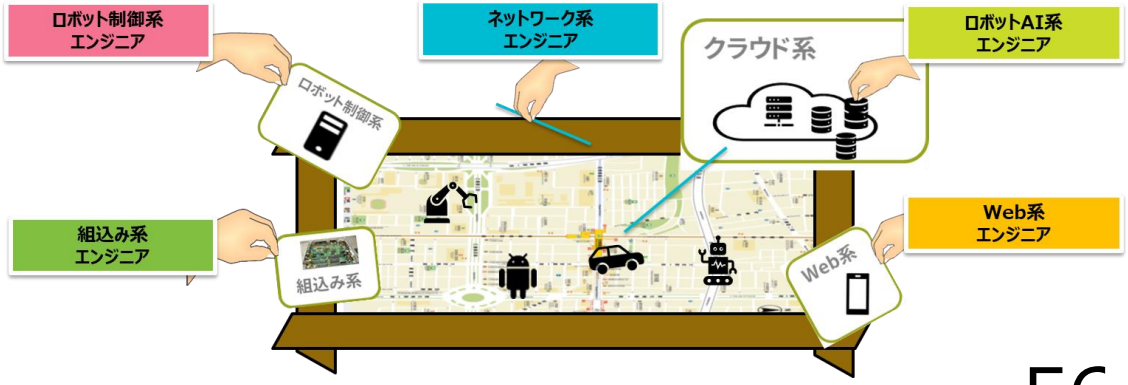
ロボット制御モデリング/プログラミング入門シミュレーション環境



ロボットAI系エンジニア

ロボット制御学習サイクル自動化入門シミュレーション環境





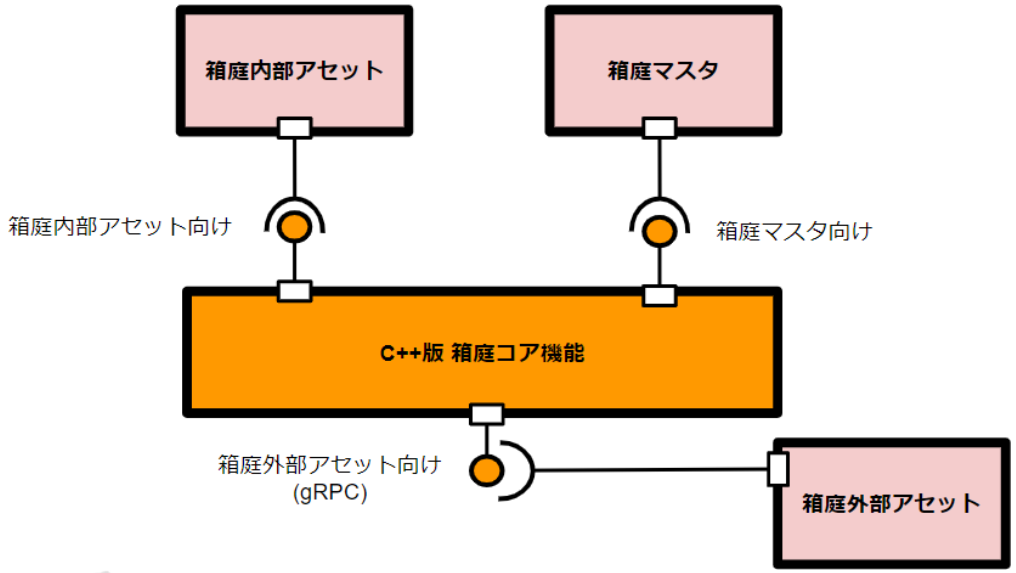
TOPPERS
Toyohashi OPen Platform
for Embedded Real-time Systems

箱庭の最新情報

- 最新の箱庭コア機能
- OpenEL連携
- 箱庭強化学習
- 箱庭とVR連携の可能性について

最新の箱庭コア機能

- 2022年～箱庭コア機能を再設計はじめました
- 狙い：
  - 箱庭コア機能をUnityから分離
  - 様々なシミュレータとの連携を視野に入れる



2022年度

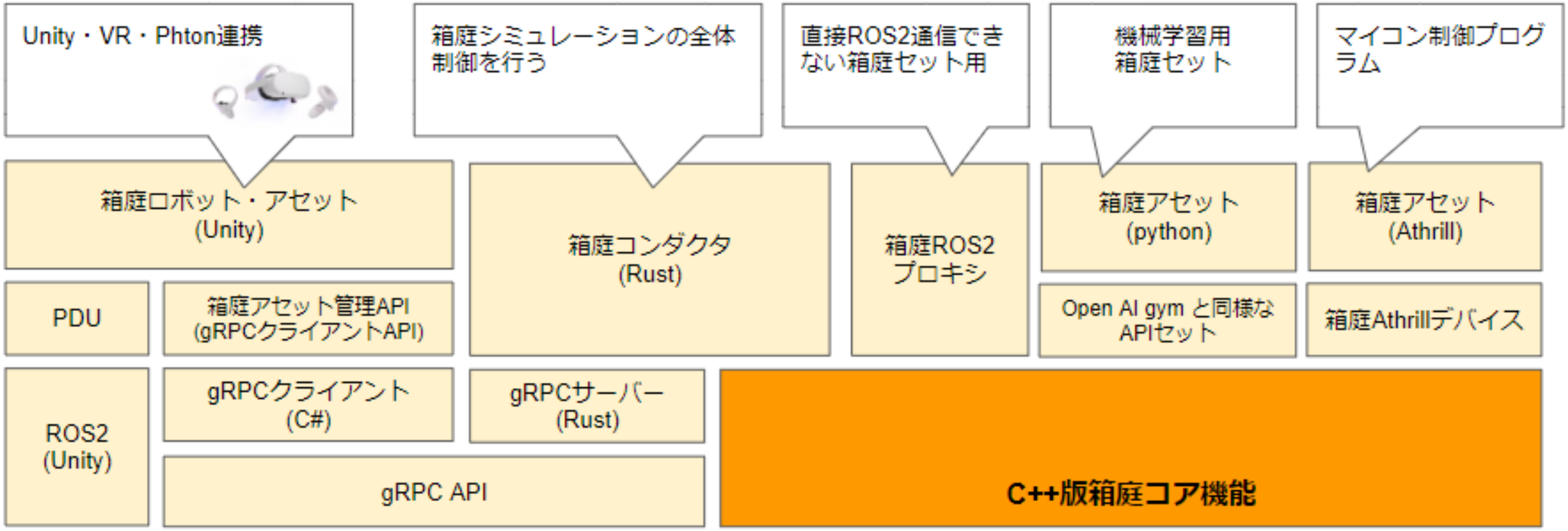
第12回TOPPERS活用アイデア・アプリケーション開発コンテストの審査結果は、以下の通りです。 受賞された皆様、おめでとうございます。

- アプリケーション開発部門

金賞
C++版箱庭コア機能 森 崇((株)永和システムマネジメント) コンテスト応募資料 リポジトリ（コア機能） リポジトリ（コマンド類）

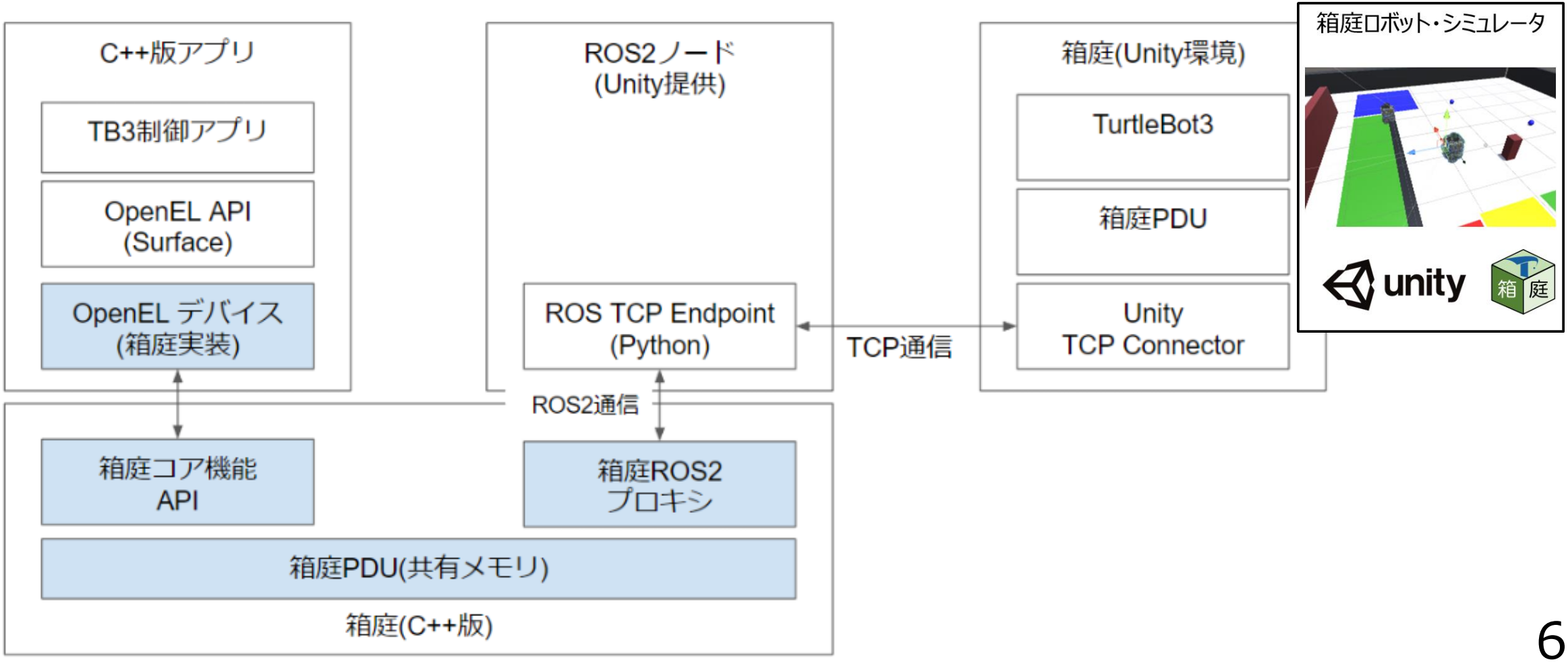
箱庭コア機能の内部アーキテクチャ

- 箱庭コア機能を介して、Unity/ROS/マイコンシミュレータなどと連携します！



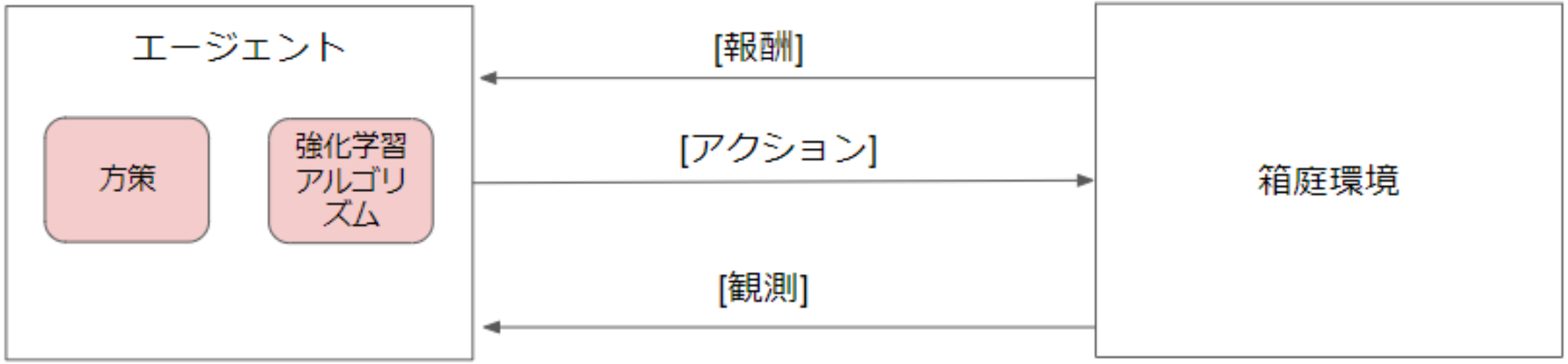
OpenEL連携

- OpenELデバイスを使用して箱庭上でロボット制御シミュレーションを実施



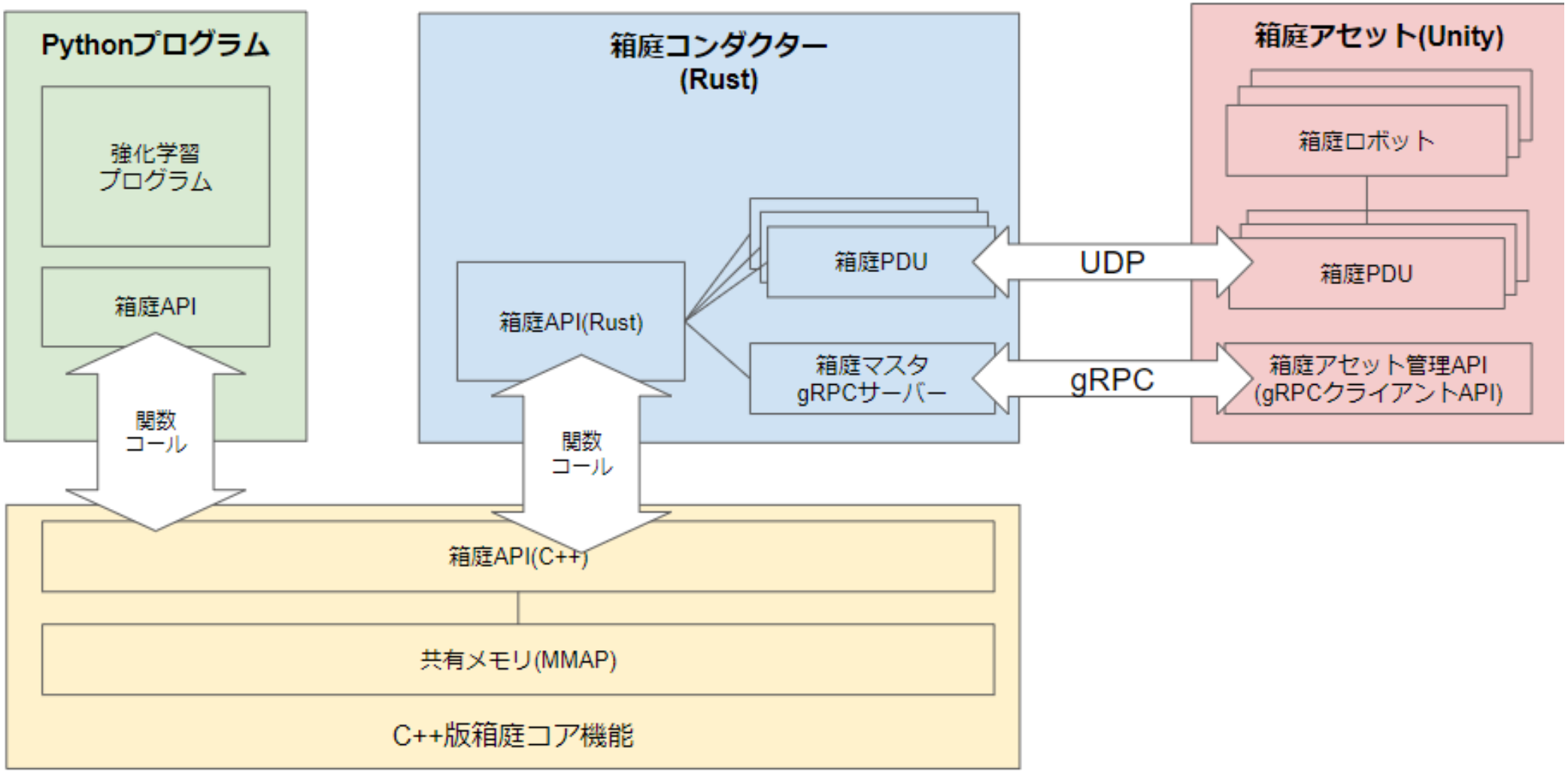
箱庭強化学習

- AI技術者をターゲットにした箱庭シミュレーション環境を提供開始！

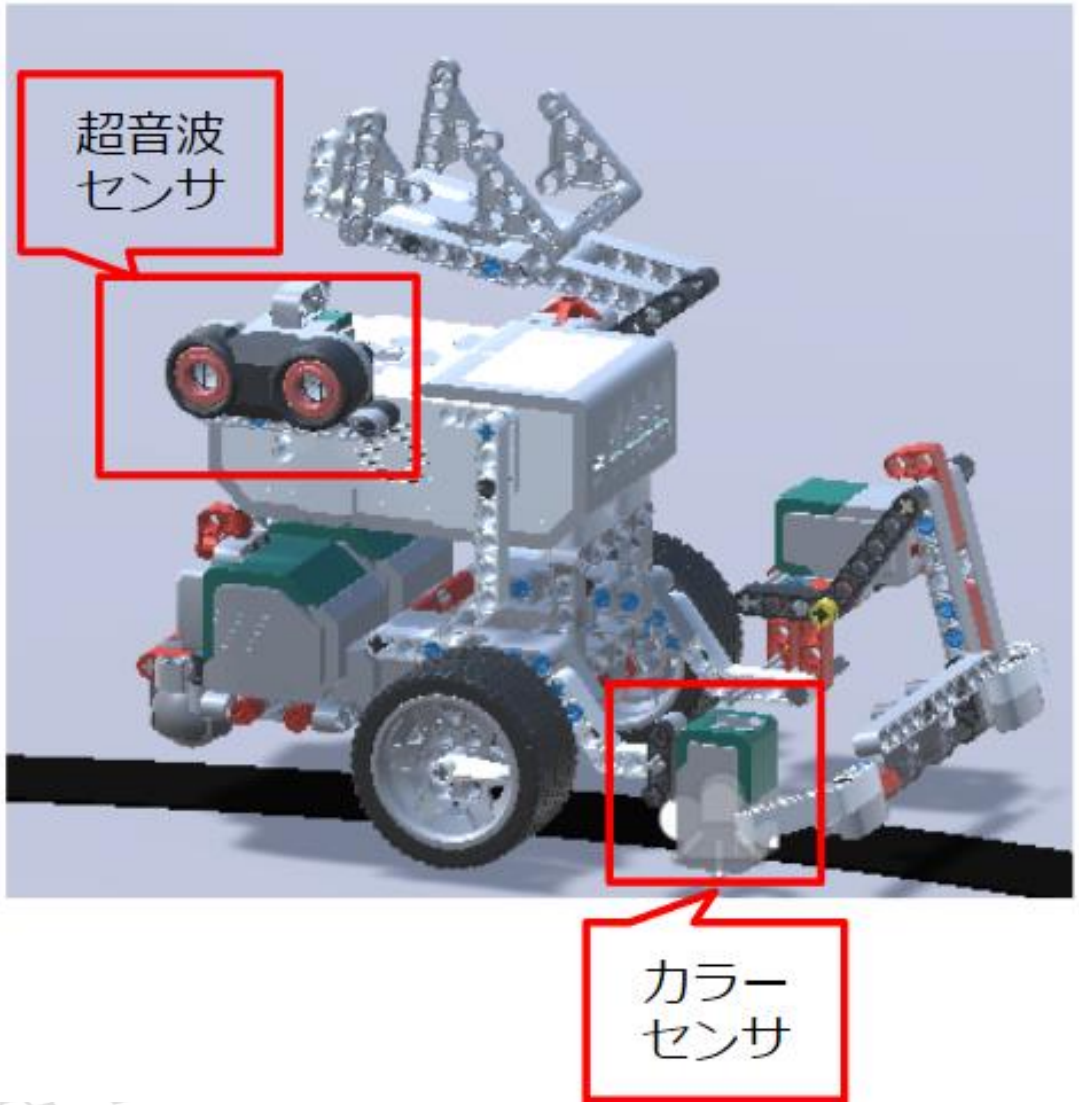


Unity + Python + 箱庭でロボットを強化学習させてみよう！

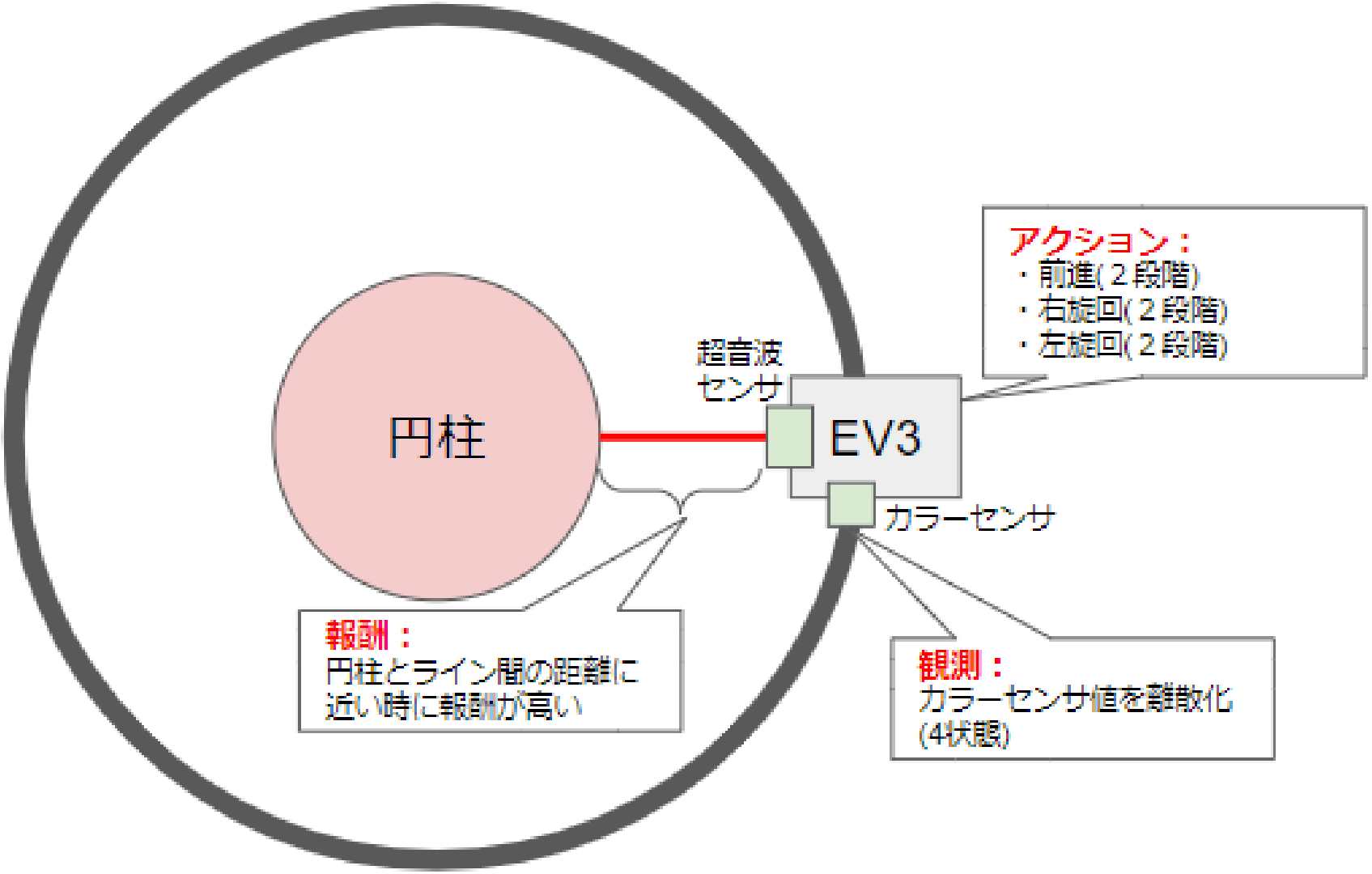
箱庭強化学習の内部アーキテクチャ



強化学習で使用するロボット

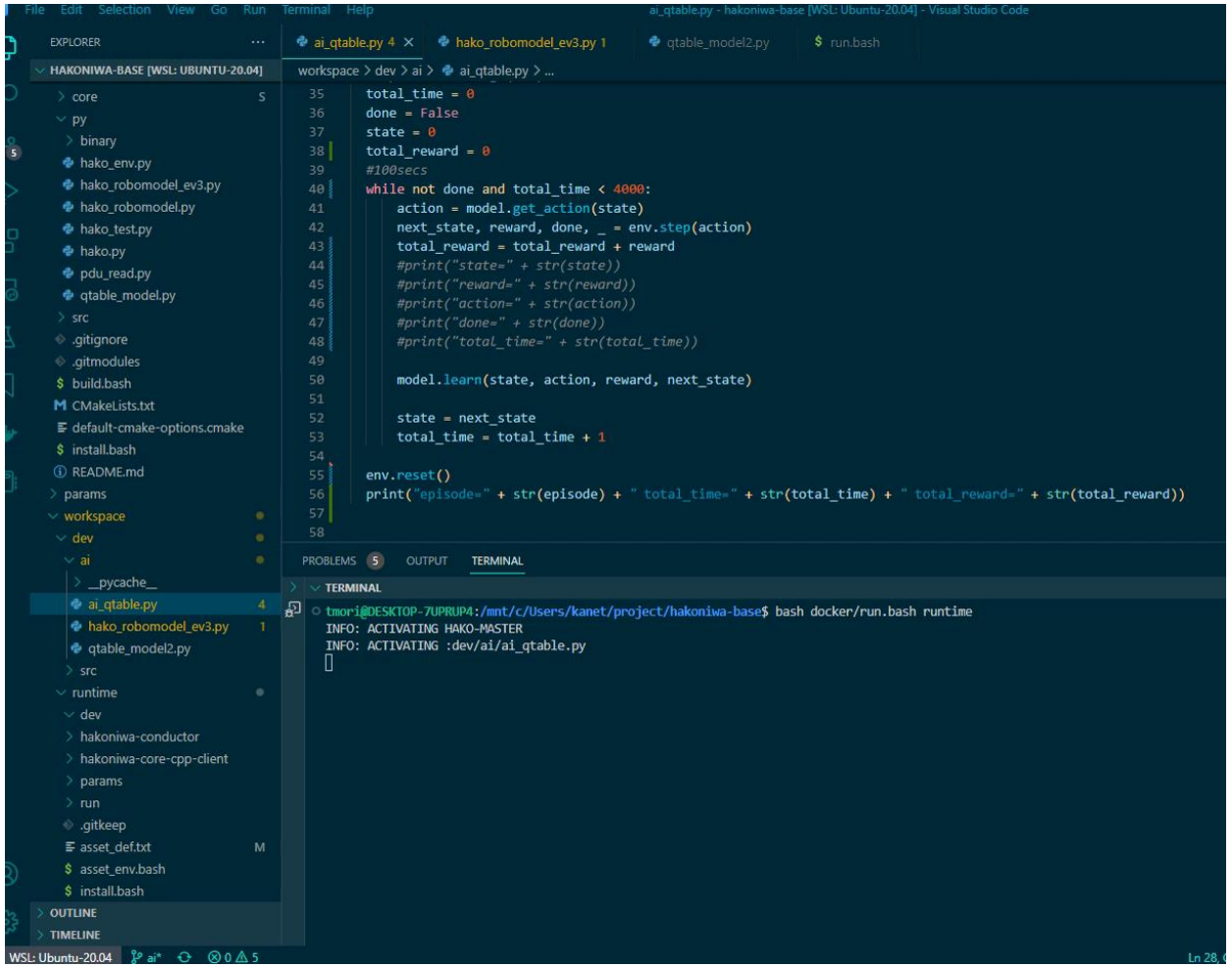


強化学習の環境

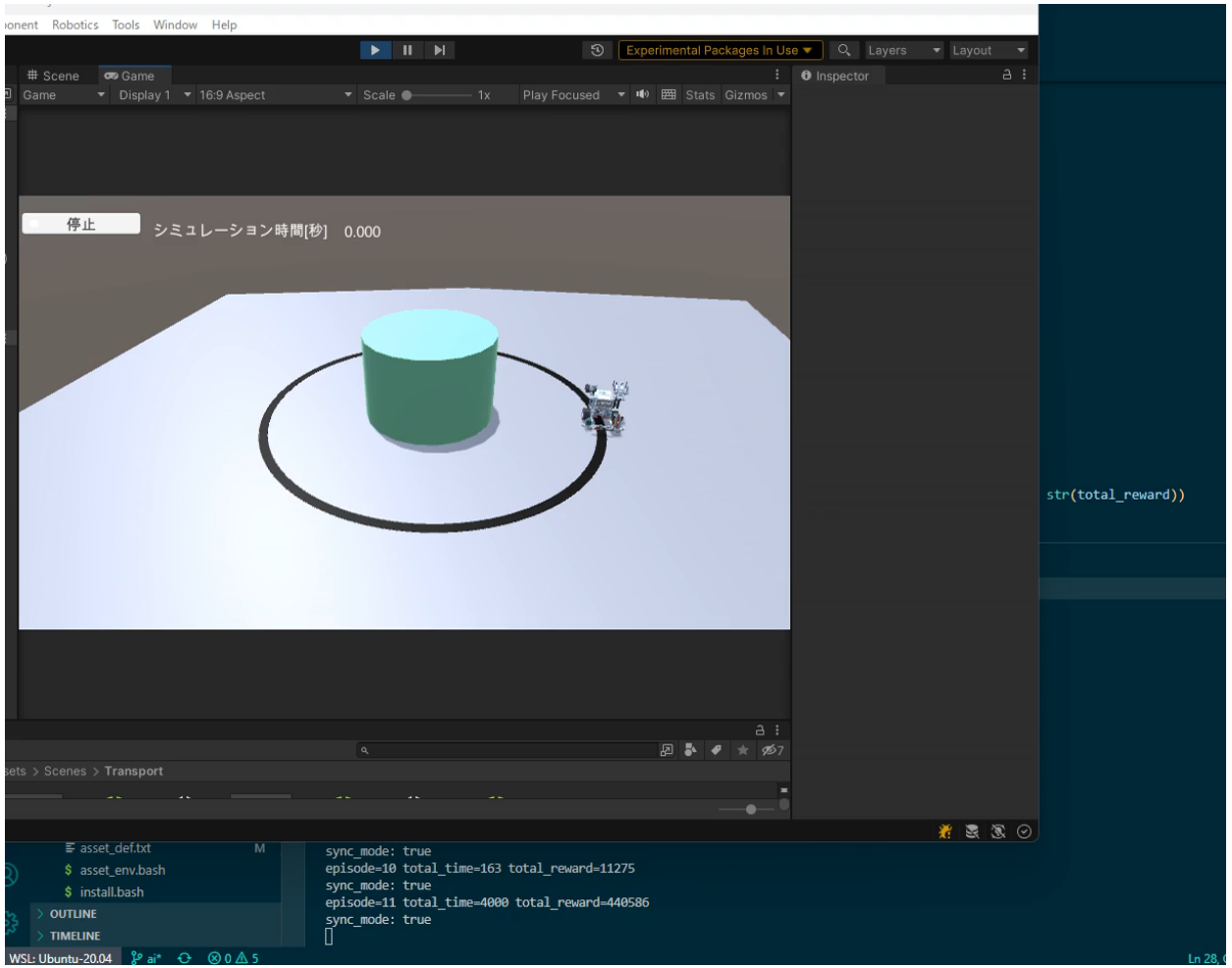

円柱
超音波
センサ
EV3
カラーセンサ
アクション：
・前進(２段階)
・右旋回(２段階)
・左旋回(２段階)
報酬：
円柱とライン間の距離に
近い時に報酬が高い
観測：
カラーセンサ値を離散化
(4状態)

強化学習デモ

未学習状態：



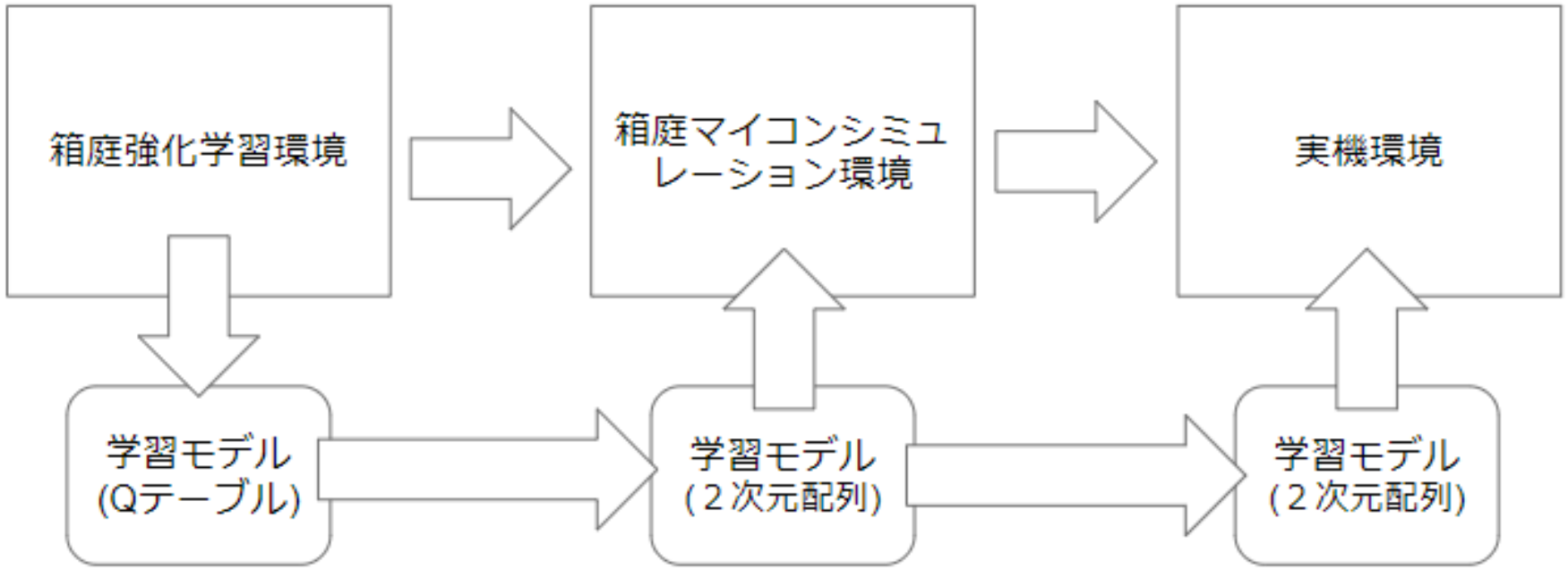
学習済み状態：



TOPPERS
Toyohashi OPen Platform for Embedded Real-time Systems

これからのAI研究はこうなる？

- 先に仮想空間で学習させてから、現実世界で最終調整がデフォルトになる
- 例：　インテグレーションフロー



箱庭とVR連携の可能性について

リアル世界にはまだない新しい体験

箱庭・VR体験からリアル世界を作る！

リアルな世界

AI・強化学習

箱庭・アセット

箱庭空間