TOPPERS
Toyohashi OPen Platform for Embedded Real-time Systems

箱庭

単体ロボット向けプロトタイプの導入手順と使用方法の紹介

2020/07/18

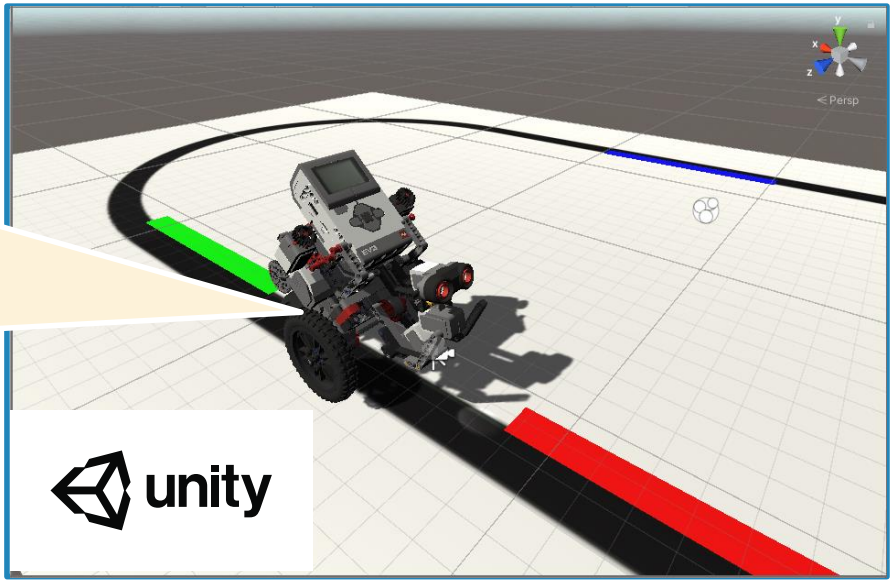
辻　悠斗（永和システムマネジメント）
森　　崇（永和システムマネジメント）

はじめに

単体ロボット(ETロボコン)向けシミュレータの構成

マイコン・シミュレータ

- 制御処理(C/C++)
- EV3RT
- ASP3/ASP
- athrill



ETロボコンを題材として構築

技術研鑽視点での狙い：

・物理シミュレータとマイコンシミュレータ間の連携方法の検討

・異なるシミュレータ間の時間同期の検討

その他の狙い：

・ETロボコンユーザ層に箱庭を広める（広報活動）

Unityパッケージの設計と作成にあたっては，
宝塚大学 東京メディア芸術学部 吉岡章夫准教授
および学部生の杉崎涼志さん，木村明美さん，
千葉純平さんにご協力いただきました。

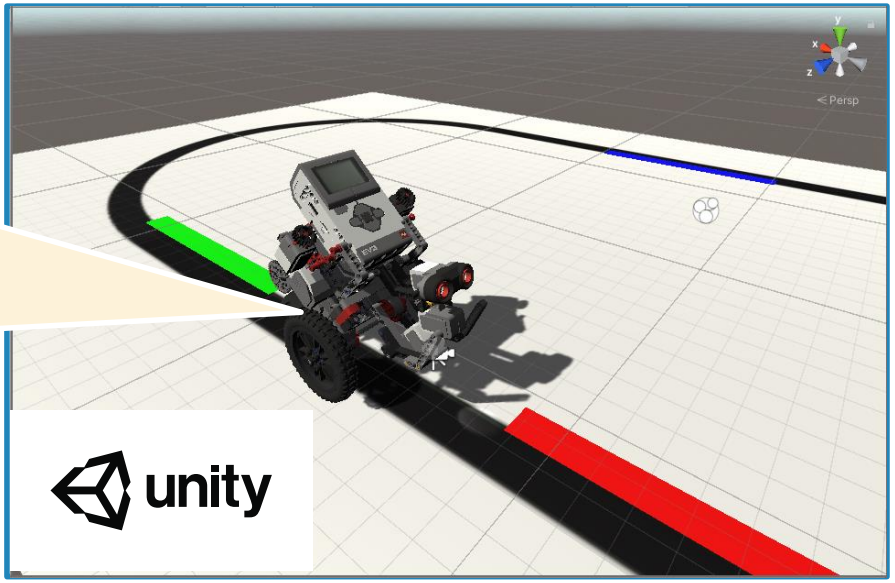
HackEVのUnityアセットは，ETロボコン実行委員会より提供いただいたデータを基に作成しています．
実行委員会の皆さまに深く感謝いたします．
ただし本アセットはETロボコンの本番環境とは異なりますので，大会に参加予定の方はご注意ください．
また，本アセットは，個人利用または教育利用に限定してご利用ください．

はじめに

単体ロボット(ETロボコン)向けシミュレータの構成

マイコン・シミュレータ

- 制御処理(C/C++)
- EV3RT
- ASP3/ASP
- athrill



ETロボコンを題材として構築

今からこのプロトタイプの
環境構築手順を紹介します

Unityパッケージの設計と作成にあたっては，
宝塚大学 東京メディア芸術学部 吉岡章夫准教授
および学部生の杉﨑涼志さん，木村明美さん，
千葉純平さんにご協力いただきました。

HackEVのUnityアセットは，ETロボコン実行委員会より提供いただいたデータを基に作成しています．
実行委員会の皆さまに深く感謝いたします．
ただし本アセットはETロボコンの本番環境とは異なりますので，大会に参加予定の方はご注意ください．
また，本アセットは，個人利用または教育利用に限定してご利用ください．

はじめに

- 紹介するプロトタイプの構成

使用プラットフォーム
  - Windows(WSL)，Linux，Mac

ターゲットCPU
  - V850，ARM

Unityとの通信方式
  - UDP，MMAP

今回は以下の構成での
環境構築手順を紹介します
・Windows(WSL)
・V850
・MMAP

MMAPはUDPよりもシミュレーション精度が高いという理由で，
今回は採用しました

- 導入手順でのつまづきポイントも紹介します
  - Webサイト上ではお伝えできていない細かい注意点についても今回お伝えできたら良いと思っています

! つまづきポイント

TOPPERS
Toyohashi OPen Platform for Embedded Real-time Systems

箱庭

アジェンダ

1. 単体ロボット向けシミュレータの導入手順

   こちらに記載の導入手順を紹介します

   https://toppers.github.io/hakoniwa/single-robot-setup/single-robot-setup-index/

2. 単体ロボット向けシミュレータの使用方法・デモ

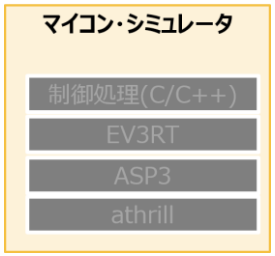
   基本的にはこちらに記載の使用方法を紹介します

   https://toppers.github.io/hakoniwa/single-robot-usage/single-robot-usage-index/

単体ロボット向けシミュレータの導入手順

Windows　V850を使用する場合の手順です

1. WSLのインストール
2. Rubyのインストール
3. athrill2のインストール
4. V850用クロスコンパイラのインストール
5. 箱庭用EV3RT環境のダウンロード
6. Unityのインストール・初期設定



1.WSLのインストール

- WSL1をインストールします
- Ubuntu 18.04 LTSをMicrosoft Storeからインストールします
- ! インストール直後に，sudo apt updateを行っておきます
  - 後ほどgccやmakeをインストールするのに必要となります

2.Rubyのインストール

※今回紹介するプロトタイプはカーネルがASP3ですので，
Rubyの使用が前提となります

$ sudo apt install ruby

3. athrill2のインストール

- athrillはターゲットCPUに依存する実装とそうでない実装で分けられています

1. athrill(ターゲット非依存部)のチェックアウト

   $ git clone https://github.com/toppers/athrill.git

2. athrill(ターゲット依存部)のチェックアウト

   $ git clone https://github.com/toppers/athrill-target-v850e2m.git

3. athrill2のインストール

- athrillをビルドするために，gccとmakeをインストールします

```
$ sudo apt install gcc
$ sudo apt install make
```

- athrillをビルドします

```
$ make timer32=true clean
$ make timer32=true
```

! 必ず，timer32オプションをつけること！
つけないと制御アプリが上手く動作しません

- athrillを使用するために環境変数を設定します

例：export PATH=<athrill配置フォルダパス>/athrill/bin/linux:${PATH}
.bashrcなどに登録しておきます

4.V850用クロスコンパイラのインストール

https://github.com/toppers/athrill-gcc-v850e2m/releases/tag/v1.1

4.V850用クロスコンパイラのインストール

- ダウンロードしたコンパイラを解凍し，コピーします

! 一度解凍したパッケージ内にさらに圧縮ファイルがあります

```
$ tar xzvf athrill-gcc-package.tar.gz
$ cd athrill-gcc-package/
$ tar xzvf athrill-gcc.tar.gz
$ sudo mv usr/local/athrill-gcc /usr/local
```

5.箱庭用EV3RT環境のダウンロード

cloneする場所は，以下のように athrillと同じフォルダ階層で実施してください.

```
|---athril
|---athrill-target-v850e2m
└---ev3rt-athrill-v850e2m
```

$ git clone https://github.com/toppers/ev3rt-athrill-v850e2m.git

5.箱庭用EV3RT環境のダウンロード

- 今回使用するEV3制御アプリをダウンロードし，箱庭EV3RT環境にコピーします

  サンプルアプリのclone

  $ git clone https://github.com/toppers/hakoniwa-scenario-samples.git

  コピー元

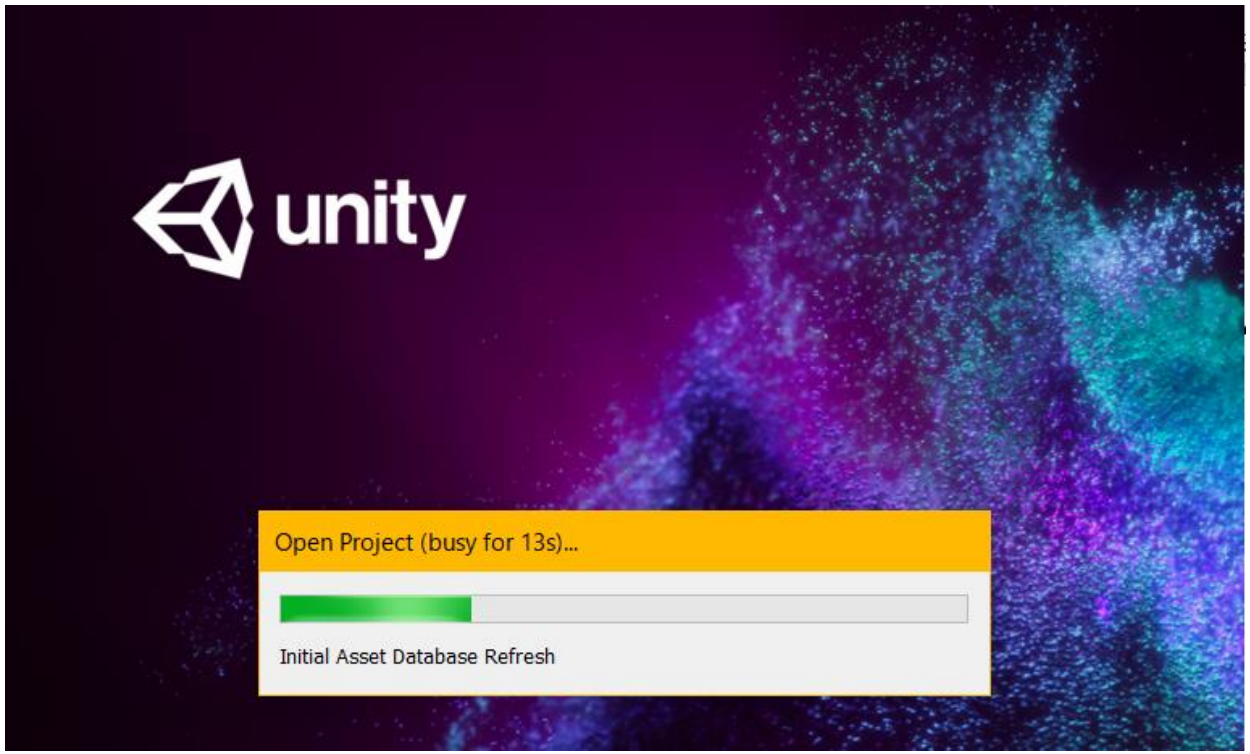
  hakoniwa-scenario-samples/single-robot 配下の全てのフォルダ

  コピー先

  ev3rt-athrill-v850e2m/sdk/workspace 配下

6. Unityのインストール

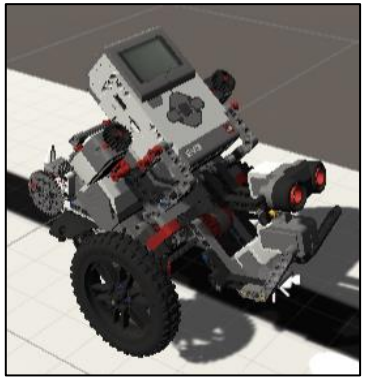
UnityのバージョンはUnity 2020.1.0b9 (64-bit)以降をインストールしてください



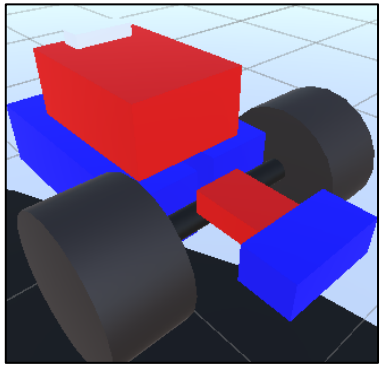
6. Unityのインストール

アセットのダウンロードをします

- single-robot-HackEV.unitypackage
  - https://github.com/toppers/hakoniwa-Unity-HackEV/releases

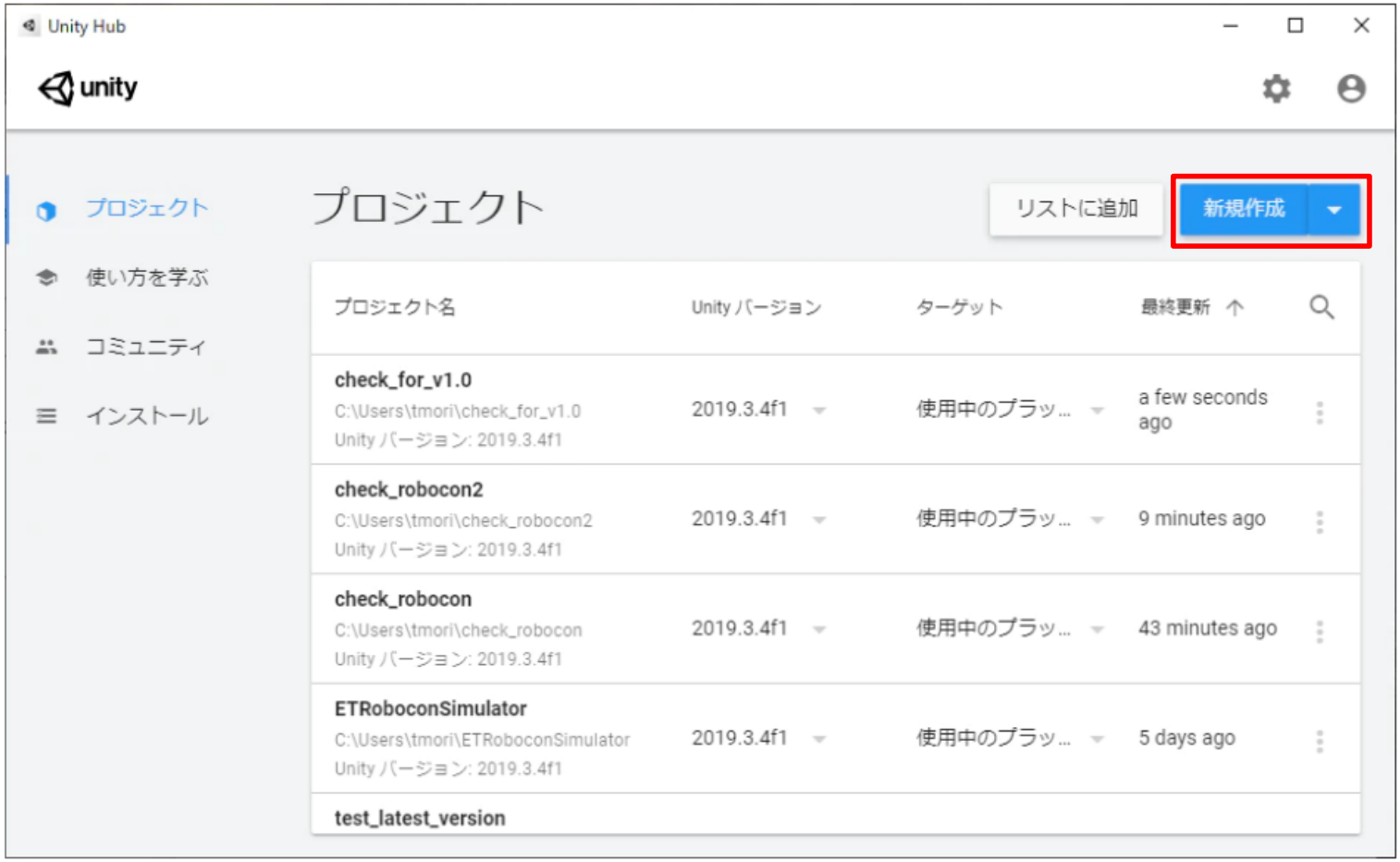


- ev3rt-simple-robot.unitypackage
  - https://github.com/toppers/hakoniwa-Unity-SimpleCar/releases



6. Unityのインストール

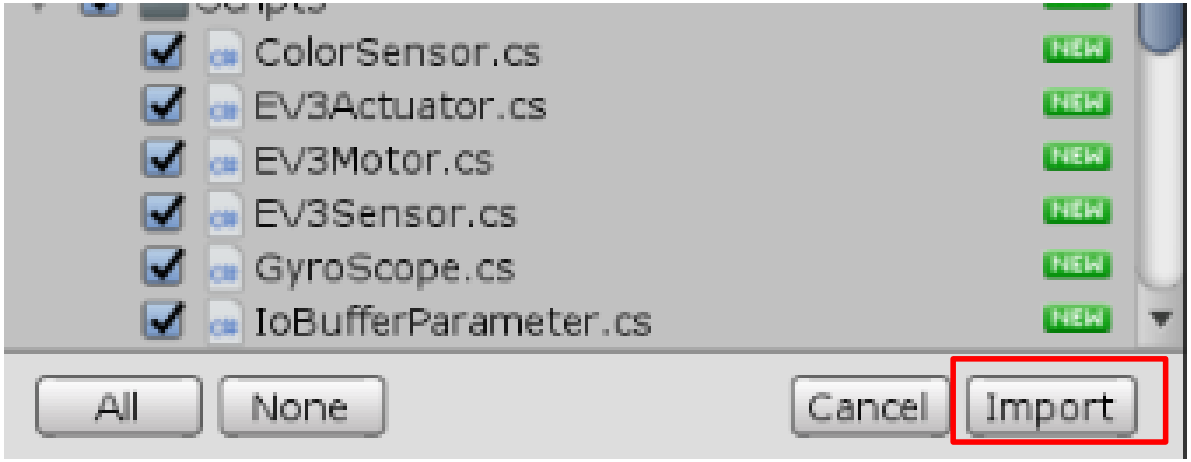
Unityを起動し，新規プロジェクトを作成します



6. Unityのインストール

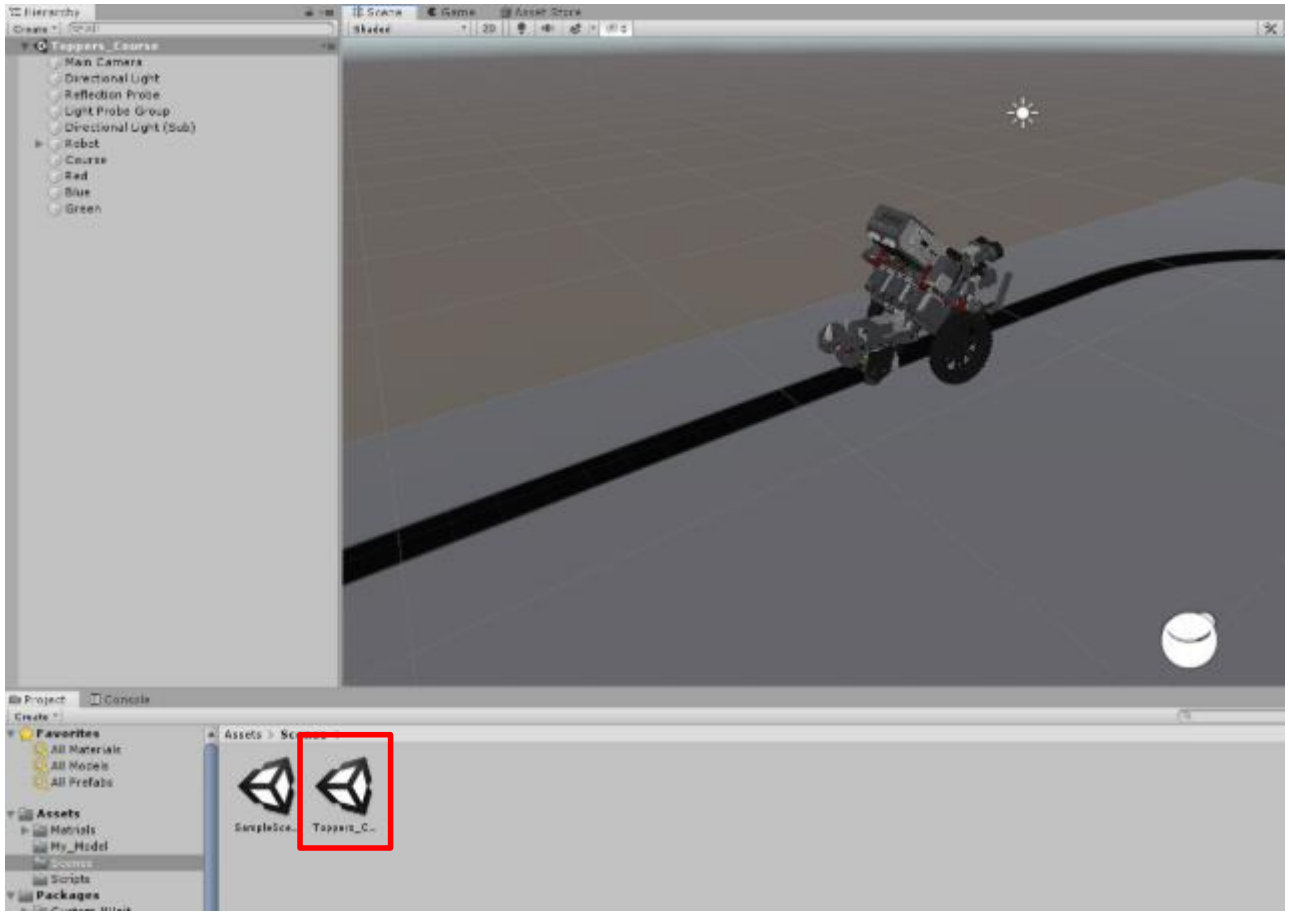
パッケージのインポート

- Unity のメニューから，「Assets」⇒「Import Package」⇒「Custom Package…」と選択し，任意の unitypackageファイルをインポートします



6. Unityのインストール

Project/Scenes配下にToppers_Courseというシーンが追加されます

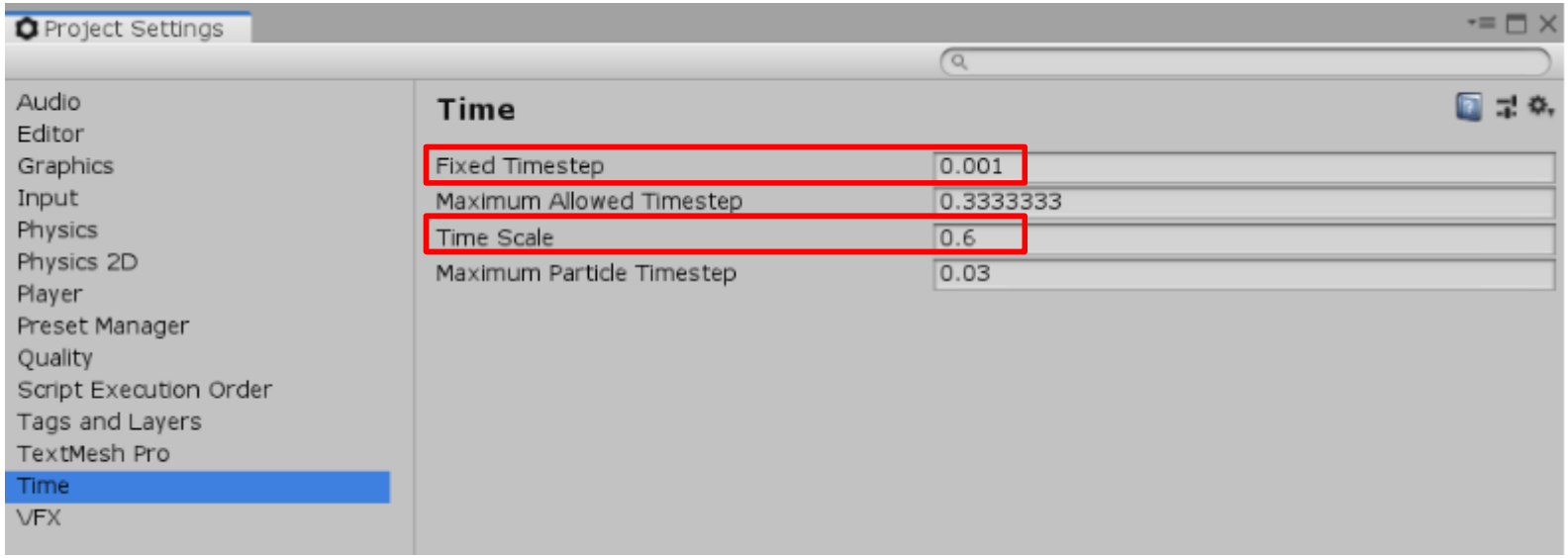


6. Unityの初期設定

シミュレーションに関わる設定をします

- Unity のメニューから，「Edit」⇒「Project Settings」を選択します

Time　Fixed Timestep を 0.001に，　Time Scale を 0.6に設定します

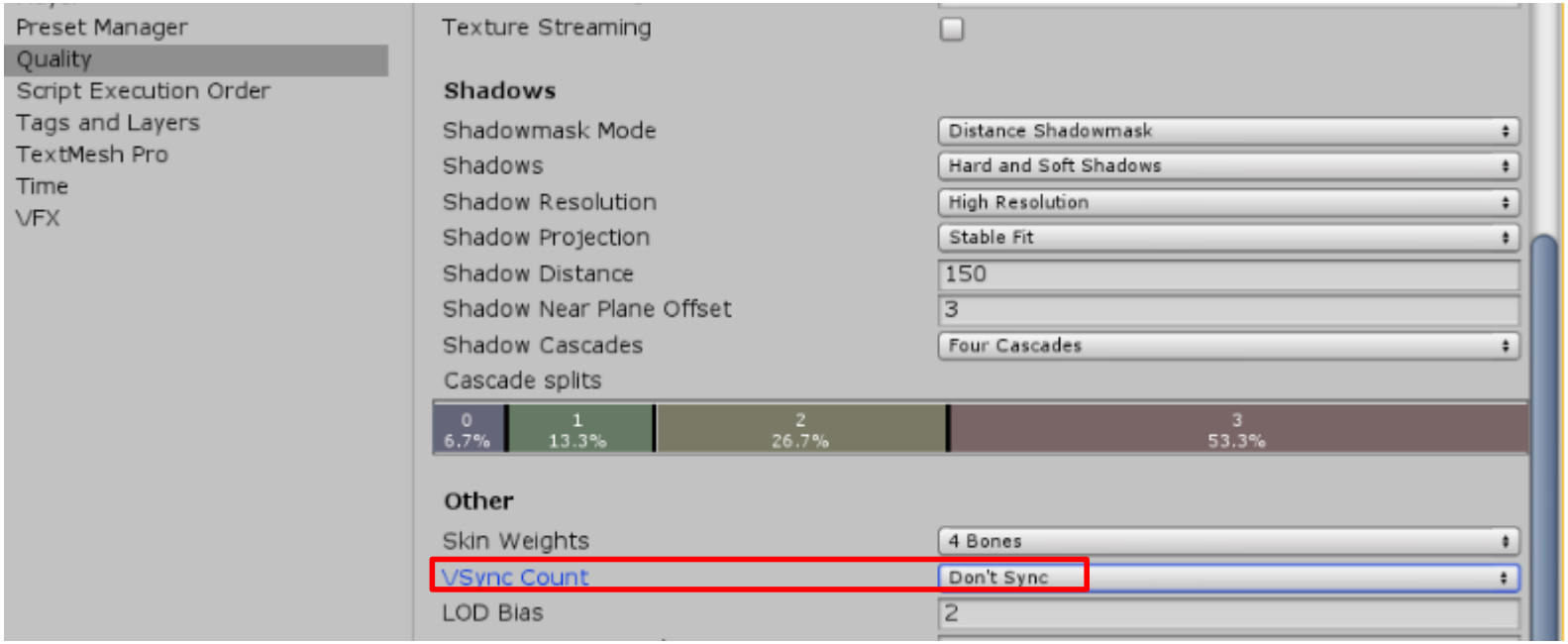


6. Unityの初期設定

シミュレーションに関わる設定をします

- Unity のメニューから，「Edit」⇒「Project Settings」を選択します

Quality　OtherのVSync Count を Don’t Sync に設定します



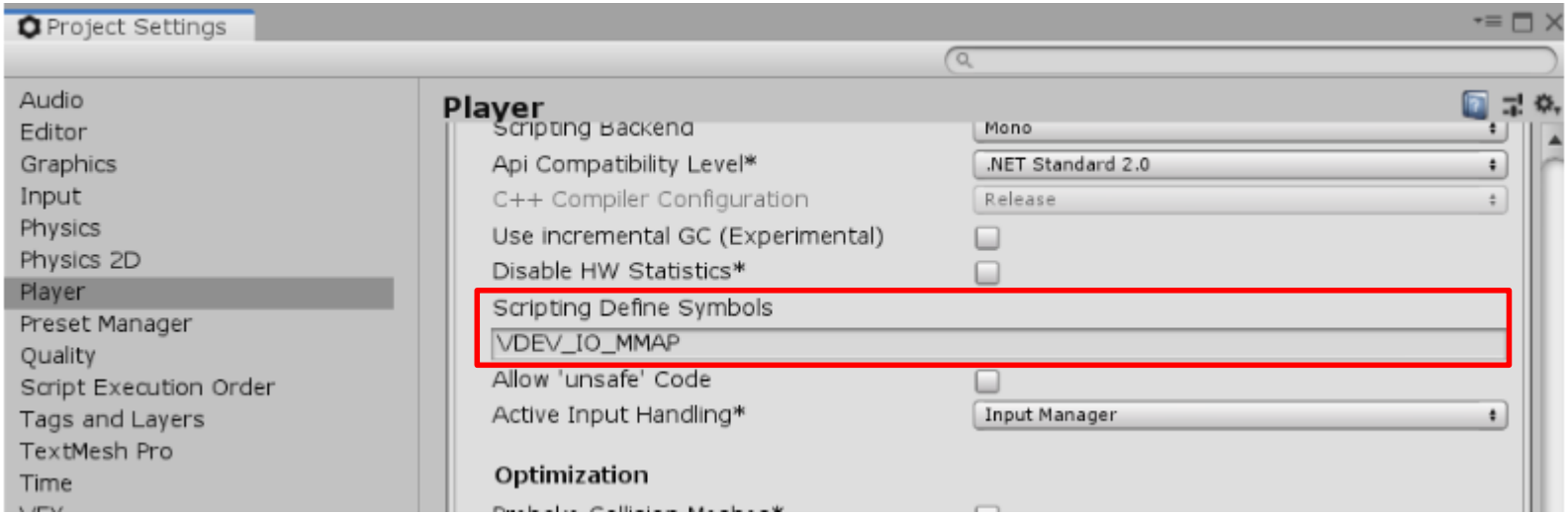
6. Unityの初期設定

シミュレーションの通信方式(MMAP)に関わる設定をします

- Unity のメニューから，「Edit」⇒「Project Settings」を選択します

Player

Other SettingのScripting Define Symbols に”VDEV_IO_MMAP”と設定します

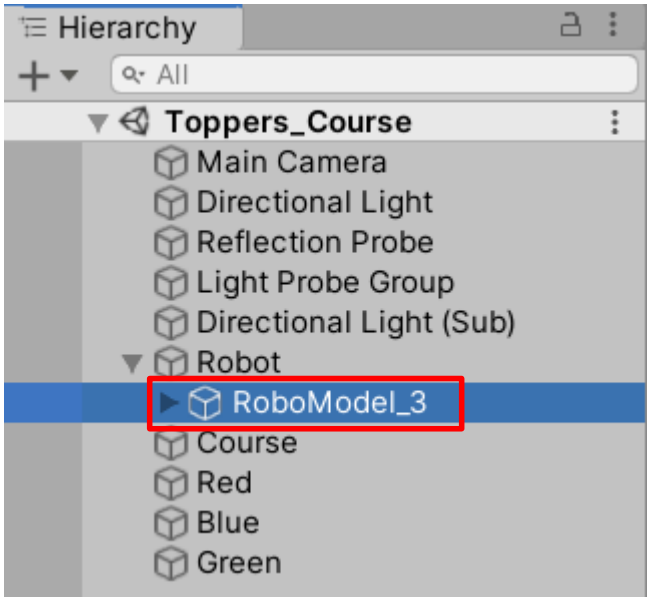


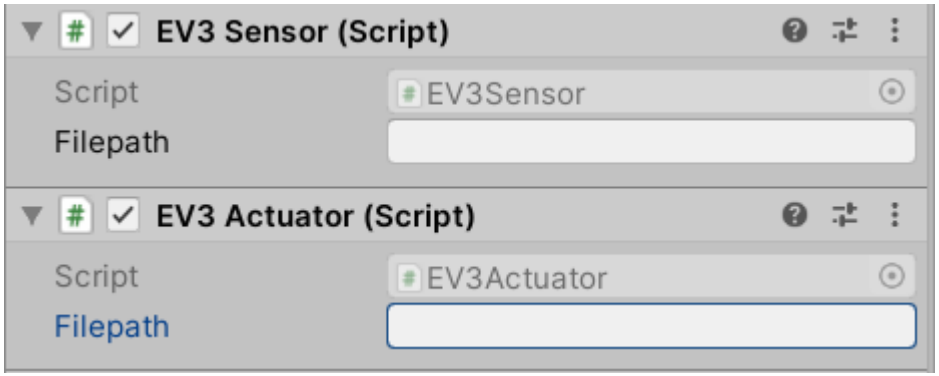
! 入力したあとは，カーソルを外すなどしてください．うまく設定が反映されない場合があります

6. Unityの初期設定

シミュレーションの通信方式(MMAP)に関わる設定をします

- Unity のHierarchyビューにてRoboModel_3を選択します
- 選択すると，画面右のInspectorビューに[EV3 Sensor (Script)]と[EV3 Actuator (Script)]が表示されます

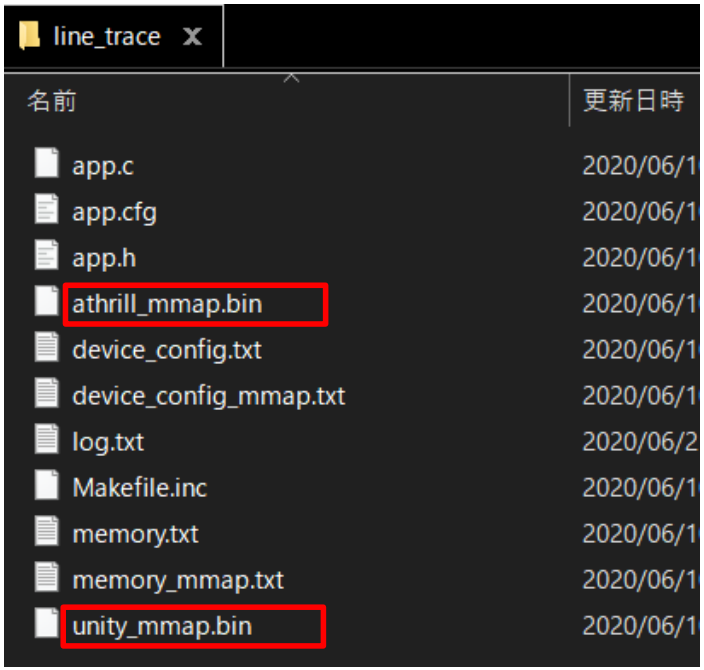


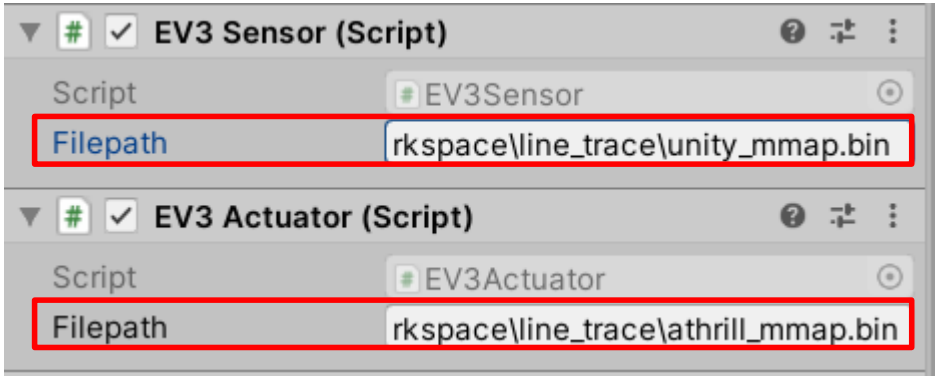


6. Unityの初期設定

シミュレーションの通信方式(MMAP)に関わる設定をします

- 画面右のInspectorビューにて[EV3 Sensor (Script)]と[EV3 Actuator (Script)]のFilePathにmmapファイルの絶対パスを入力します





EV3 Sensorにはunity_mmap.binを，
EV3 Actuatorにはathrill_mmap.bin
の絶対パスを入力します

単体ロボット向けシミュレータの使用方法

Windows　V850を使用する場合の手順です

1. EV3制御プログラムのビルド
2. Unityのシミュレータの起動
3. athrillの起動

1.EV3制御プログラムのビルド

- WSL上でev3rt-athrill-v850e2m/sdk/workspace に移動します

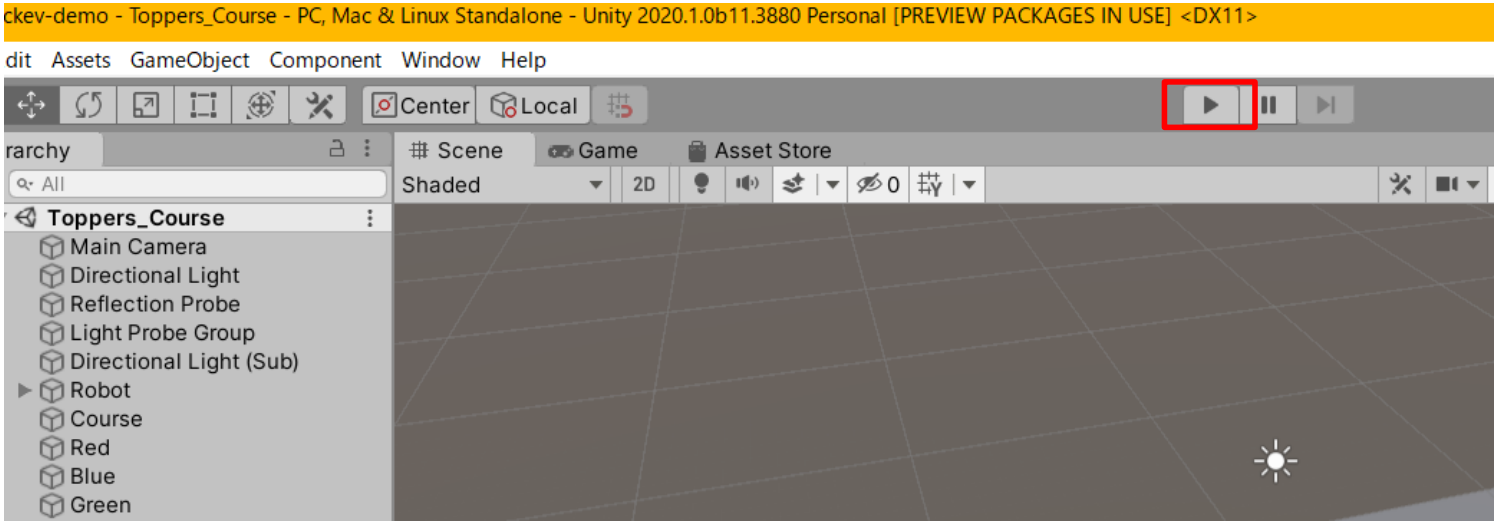
- 下記コマンドでビルドします

  $ make img= <アプリケーションフォルダ名> clean

  $ make img= <アプリケーションフォルダ名>

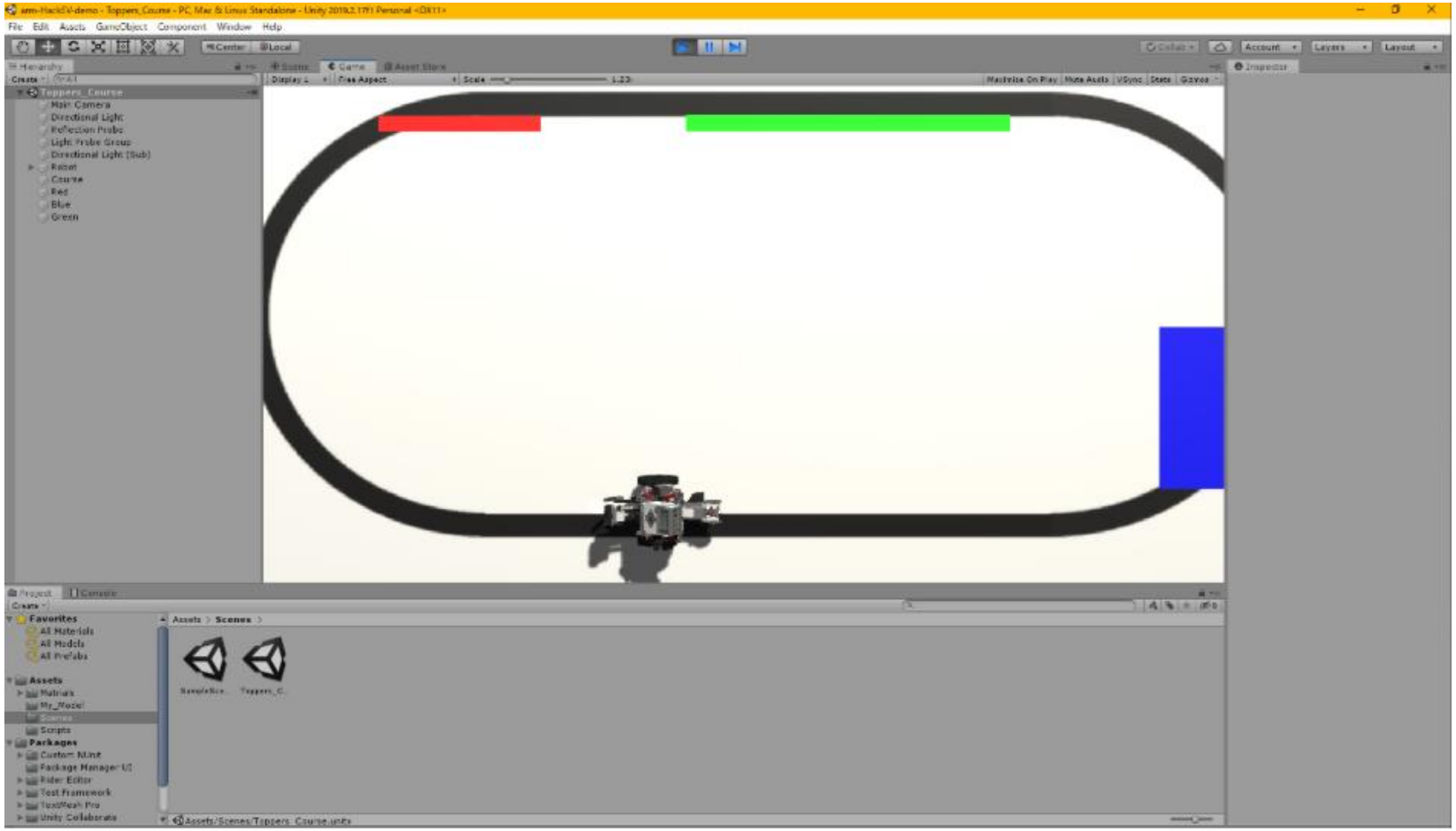
2.Unityのシミュレータの起動

- Unityの画面上で，実行ボタンを押下します



2.Unityのシミュレータの起動

- 下図のように，画面が切り替わります



3. athrillの起動

- WSL上でathrillを起動し，シミュレーションを開始します

- ev3rt-athrill-v850e2m/sdk/workspace/＜アプリケーションフォルダ＞に移動して，以下のコマンドを実行します

【通信方式がMMAPの場合】

```
$ athrill2 -c1 -t -1 -m memory_mmap.txt -d
device_config_mmap.txt ../asp
```

たった1コマンド叩くだけでシミュレーションが実行できる！

3. athrillの起動

- 成功するとWSL上で以下のメッセージが出力され，Unity上の画面で，EV3のロボットが動き始めます

```
core id num=1
ROM : START=0x0 SIZE=512
RAM : START=0x5ff7000 SIZE=10240
Elf loading was succeeded:0x0 - 0xfd68 : 63.360 KB
ELF SYMBOL SECTION LOADED:index=22
ELF SYMBOL SECTION LOADED:sym_num=964
ELF STRING TABLE SECTION LOADED:index=23
athrill_device_func_call=0x60f7444

TOPPERS/ASP3 Kernel Release 3.2.0 for V850-ESFK3 (Nov  6 2019, 10:56:14)
Copyright (C) 2000-2003 by Embedded and Real-Time Systems Laboratory
                            Toyohashi Univ. of Technology, JAPAN
Copyright (C) 2004-2017 by Embedded and Real-Time Systems Laboratory
            Graduate School of Information Science, Nagoya Univ., JAPAN

brick_dri initialized.
   _____   ______ ___  ______
  / __/ | / /_  // _ \/_  __/
 / _/ | |/ //_ </ , _/ / /
/___/ |___/____/_/|_| /_/



Powered by TOPPERS/HRP2 RTOS
Initialization is completed..
System logging task is started.
```

デモ

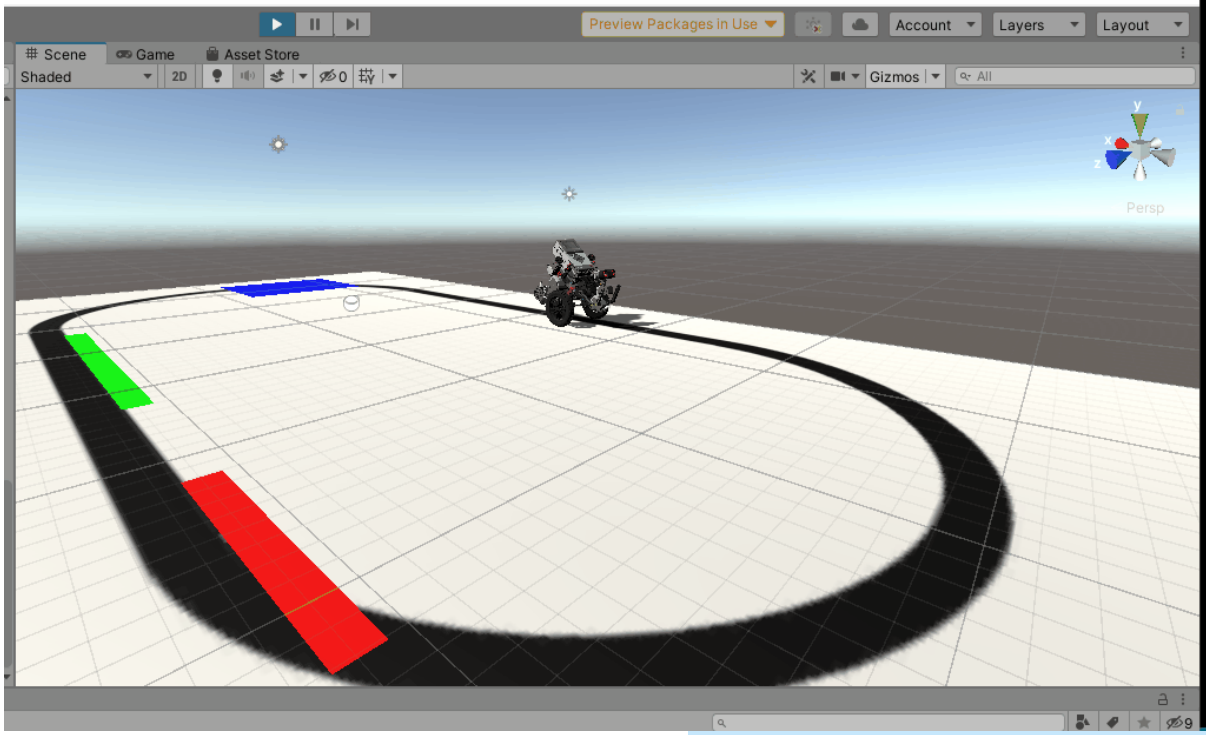
[デモ内容]

制御アプリケーションにより，HackEVがライントレースします．

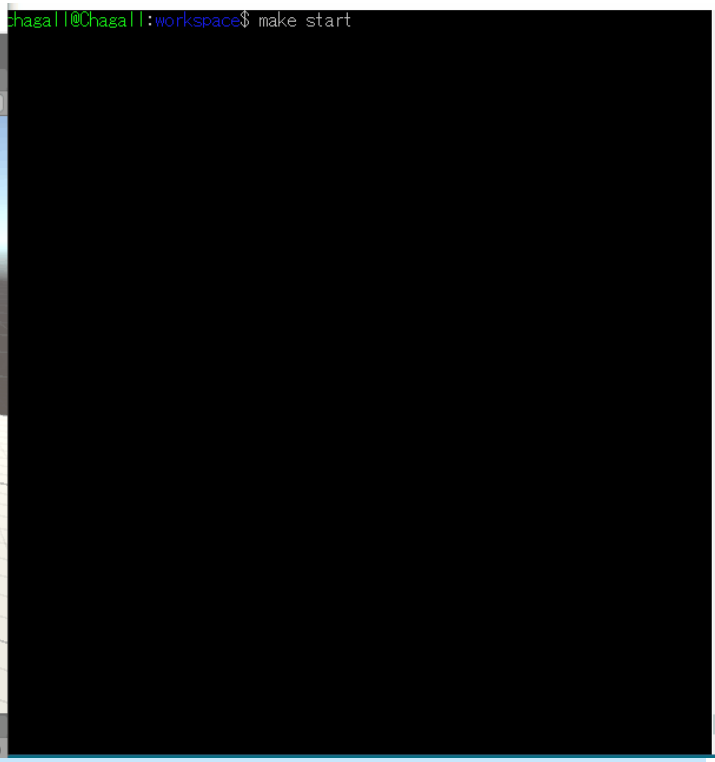
走行路にはカラーコードを配置しており，制御アプリケーションの色認識により，加速/減速/停止します．

赤：加速，緑：減速，青：停止

[Unity]



[Athrill]



[条件]

Unityのシミュレーション精度：　5msec

制御アプリケーション周期　　：10msec

HackEVのUnityアセットは，ETロボコン実行委員会より提供いただいたデータを基に作成しています．
実行委員会の皆さまに深く感謝いたします．
ただし本アセットはETロボコンの本番環境とは異なりますので，大会に参加予定の方はご注意ください．
また，本アセットは，個人利用または教育利用に限定してご利用ください．

謝辞・特記事項

- Unityパッケージの設計と作成にあたっては，宝塚大学 東京メディア芸術学部 吉岡章夫准教授および学部生の杉﨑涼志さん，木村明美さん，千葉純平さんにご協力いただきました.

- HackEVのUnityアセットは，ETロボコン実行委員会より提供いただいたデータを基に作成しています． 実行委員会の皆さまに深く感謝いたします．
ただし本アセットはETロボコンの本番環境とは異なりますので，大会に参加予定の方はご注意ください． また，本アセットは，個人利用または教育利用に限定してご利用ください.

- 本資料は，ユニティ テクノロジーズまたはその関連会社がスポンサーとなったり，ユニティ テクノロジーズまたはその関連会社と提携しているものではありません.
本資料に掲載された Unity の登録商標一覧 に含まれる Unity の登録商標はすべて，ユニティ テクノロジーズまたはその米国や他の国々に所在する関連会社の登録商標または商標です.