SWEST24 9/2(金) 12:30〜13:40 セッションs4c
https://swest.toppers.jp/SWEST24/program/s4c.html#s4

WEBブラウザ上で動作する ETロボコンシミュレータの仕組みについて

ETロボコン東京地区　実行委員
新　吉高（あたらし　よしたか）

1.1 講演者紹介

- 2015年よりETロボコン実行委員会に東京地区実行委員として参加
- 2017年東京地区実行副委員長を経て、2018年より東京地区実行委員長
- 2020年大会で用いられたETロボコンシミュレータ(Unity)の設計・実装を担当

- 本業は某企業で組込みソフトからクラウド上でのデータ分析まで薄く広く対応するエンジニア兼マネージャ。
- 技術士(情報工学部門)

東京地区大会懇親会後の集合写真

1.2　ETロボコンとは

Embedded Technology Software Design Robot Contestの通称で、LEGO社のLEGO Mindstormsを使用した自律型ライントレースロボットの設計とタイムを競うコンテストです。

総合部門

モデル部門
システム分析、
ソフトウェア設計モデル

＋

競技部門
走行競技による性能

HackEV

1.3 コロナ禍でのETロボコン

- ETロボコン実行委員会はコロナ禍のため、2020年度の大会をシミュレータ大会とすると決定
- TOPPERS/箱庭WGが組込みCPUシミュレータAthrillを開発
- 発表者は3Dゲーム開発環境として有名なUnityを用いて、Athrillと連携可能なETロボコンシミュレータを開発
- 2020~2021年の大会はシミュレータのみで実施
- 2022年の大会は実機（アドバンストクラス、プライマリークラス）とシミュレータ（エントリークラス）の併用となった

1.4 大会版ETロボコンシミュレータの構成

参加チーム開発

走行体制御ソフトウェア
プログラミング言語：C/C++

新が開発した部分

ETロボコンシミュレータ
(参加者限定)

棚橋先生の開発

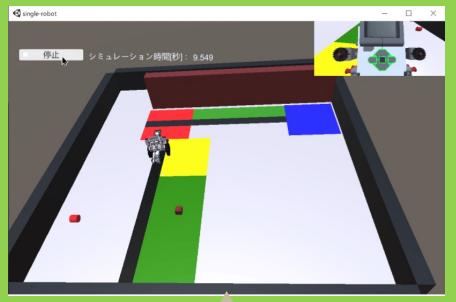
etrobo環境

TOPPERSの成果物
EV3RT/ASP3
エミュレータ向け移植版

箱庭WGの成果物＋土樋さんの改善
Athrill
CPU エミュレータ

箱庭WGの成果物

単体ロボット向け
シミュレータ(一般公開)



同期・通信(UDPのみ)
ETロボコンの独自拡張あり

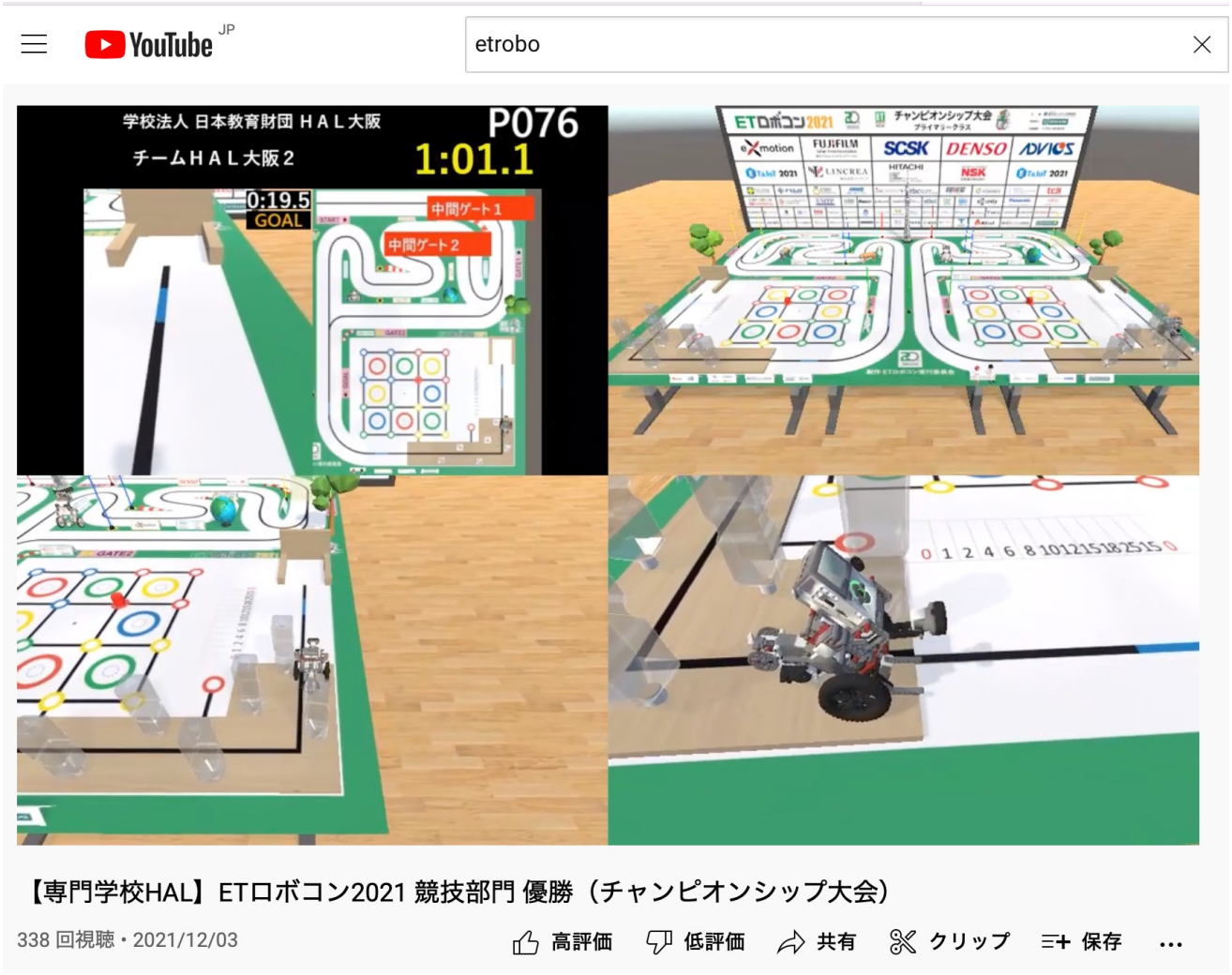
同期・通信(UDP, ROS …)
箱庭WGの独自拡張あり

https://github.com/ETrobocon/etrobo

参照： https://toppers.github.io/hakoniwa/prototypes/single-robot/

1.5 ETロボコン2021 CS大会競技優勝走行

https://youtu.be/lh1g9Nbq-aU

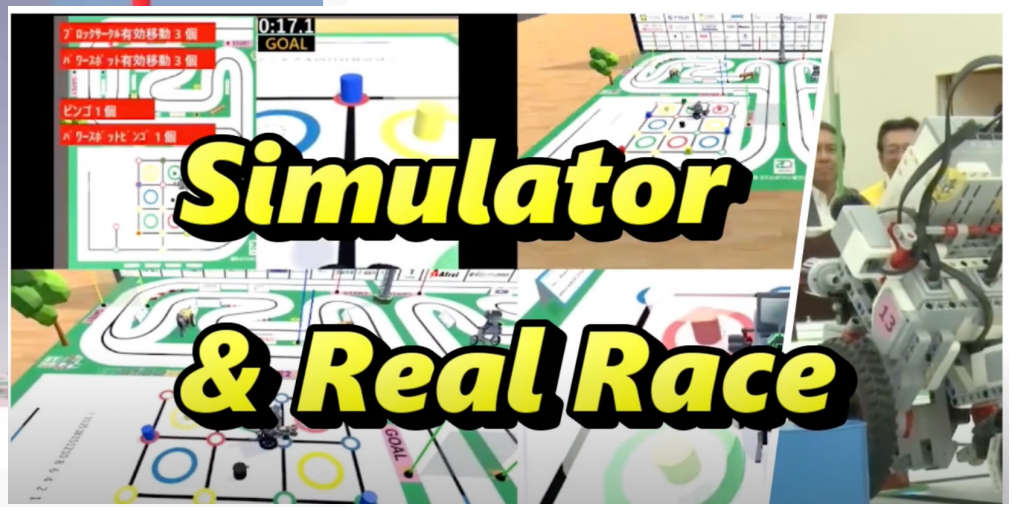


1.6 関係者の要求とその評価

- ETロボコン実行委員会の要求：　学びの歩みを止めない！
  - シミュレータ大会を実施すること。→ **OK**
  - 箱庭WGの成果物を活用すること。→ **OK**
  - ラインとレースのみの簡易なものでも十分。 → **難所攻略も可能なレベルに**
  - 競技判定は目視でもよい。 → **自動判定可能**
- 箱庭WGの要求：　成果物を多くの参加者にアピールしたい！
  - Athrillと単体ロボットシミュレータを大会で利用すること。→ **Athrillのみ利用**
  - 箱庭WGにHackEV走行体の3Dモデルを提供すること。→ **OK**
  - 十分な開発時間を用意することはできない。→ **土樋さんが開発を支援**
- 発表者の狙い：　リアル大会の雰囲気を再現したい！
  - エントリー、プライマリー、アドバンストの3クラスをすべてサポートする。→ **OK**
  - 走行体の動作をすべてAPIで制御できるようにする。→ **音と画面以外OK**
  - Unityの物理エンジンで作り切る。→ **OK**

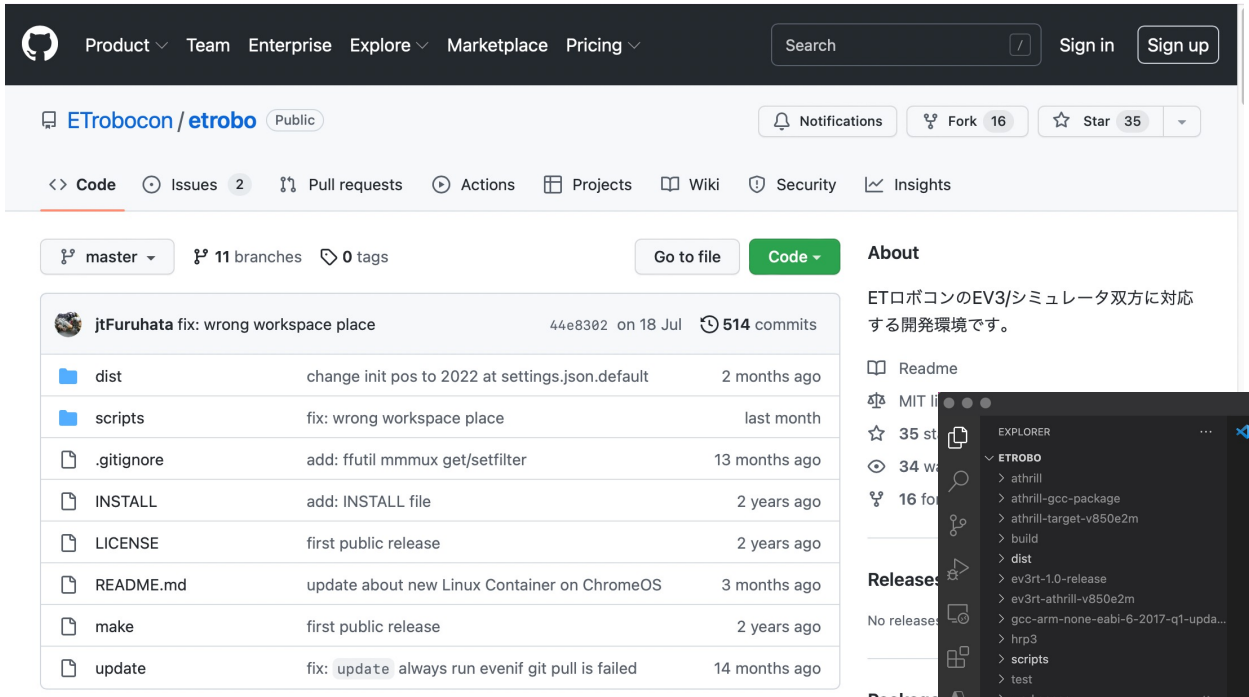
1.7 ETロボコン 2022 プロモーションビデオ

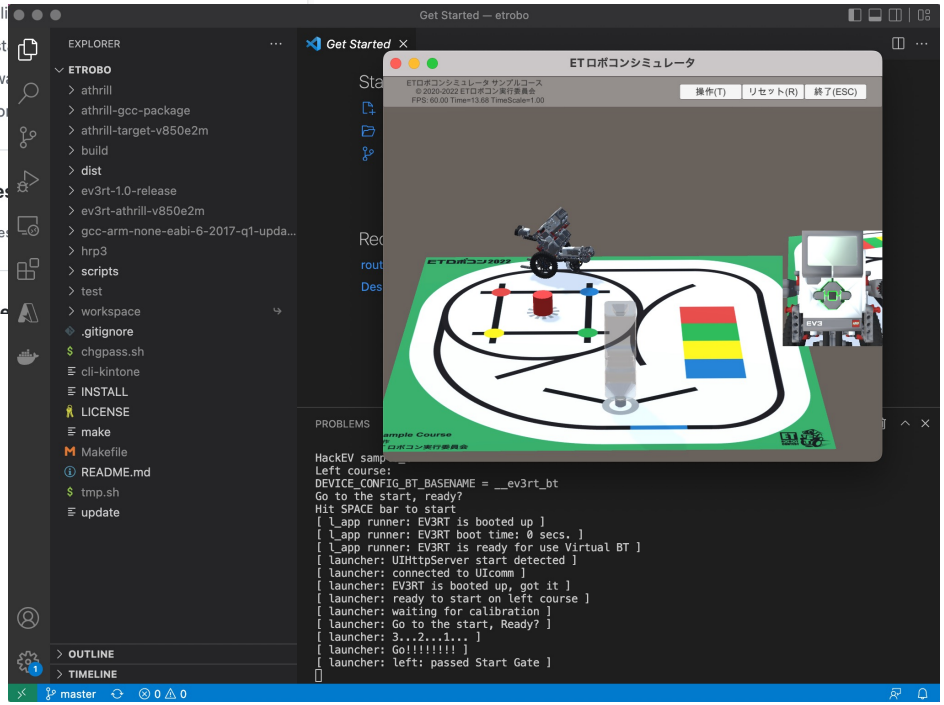
https://youtu.be/v8O5J65B4uA



1.8 ETロボコンシミュレータ評価環境

https://github.com/ETrobocon/etrobo





サンプルコースのみ走行可能なシミュレータを一般公開している。
誰でも自由にインストール可能。
ワンコマンドでインストールできるが、一度失敗すると結構ハマる。

2.1 ETロボコンシミュレータの課題と解決案

多くの制約の中で開発したため、様々な課題が残っている。

ETロボコンシミュレータの課題	解決案
複数OS対応によりインストーラが複雑化し、バージョンアップ毎に不具合報告がある（ruby、M1 Mac、WSL関係が多い)	**WebGLによるOS非依存化、インストールレス環境の実現**
参加者メリットの維持のため、シミュレータのソースコードを公開できない	ソースコードを公開可能なバージョンを別に開発する。
マイコンシミュレータAthrillとのUDP通信周期を短くできず、倒立振子制御の実現が難しい	**Athrillの代わりに軽量なスクリプト言語を用いる（この場合、EV3用コード生成のメリットは無くなる）**
JSONファイルでコースをカスタマイズできるものの、難所の配置の自由度が低い	スクリプト言語で自由に配置できるようする

2.2 Web版ETロボコンシミュレータを作ってみた

- WebGLによるOS非依存化、インストールレス環境の実現
- Athrillの代わりに軽量なスクリプト言語(MiniScript)を用いる

https://webglsim.etrobo.jp

API一覧

MiniScript仕様

操作	機能
カーソルキー	走行体を動かす（スクリプト非実行時）
SHIFTキー	アームを上げる（スクリプト非実行時）
ALTキー	しっぽを伸ばす（スクリプト非実行時）
Vキー	カメラを走行体追従モード（正面⇔側面）にする
Cキー	カメラをマウス操作モード⇒パノラマモード⇒コクピットモードに変更する
カンマ(,)キー	走行体の位置を変更する

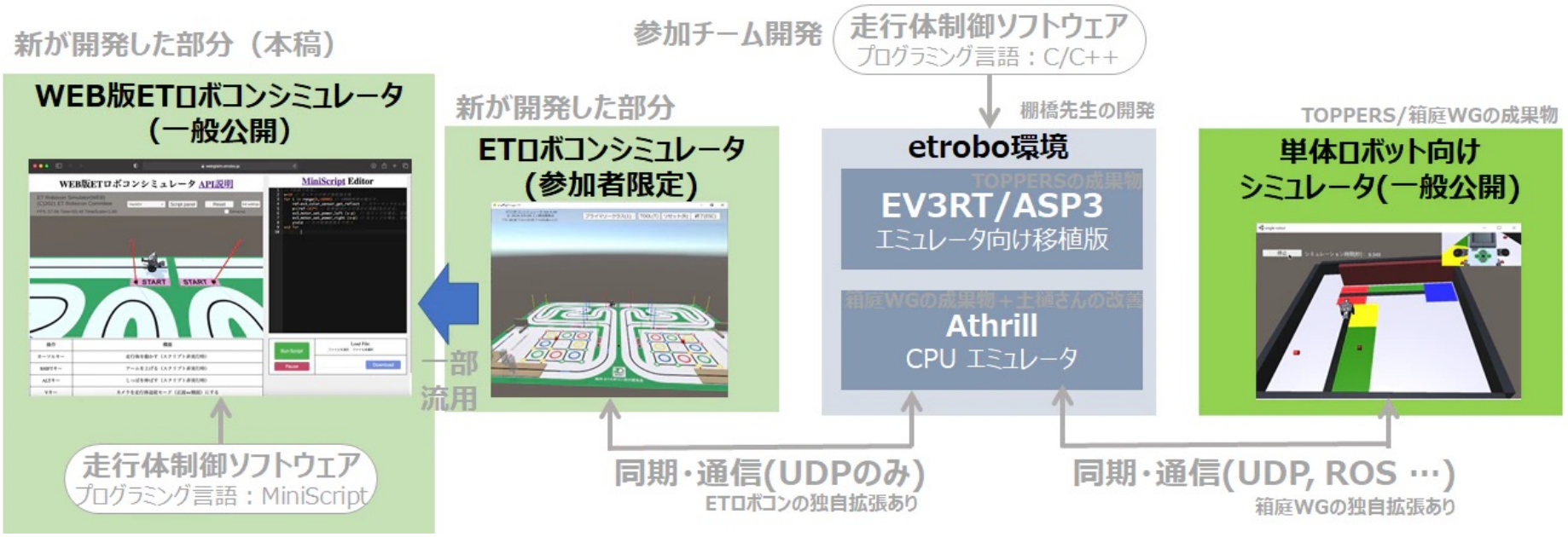
Aceエディタ
(MiniScript用)

シミュレータ操作

スクリプト実行

2.3 WEB版ETロボコンシミュレータの位置付け

大会用ETロボコンシミュレータのコードを一部流用しているが、
WebGLでビルド・実行できるようにかなり書き換えている。

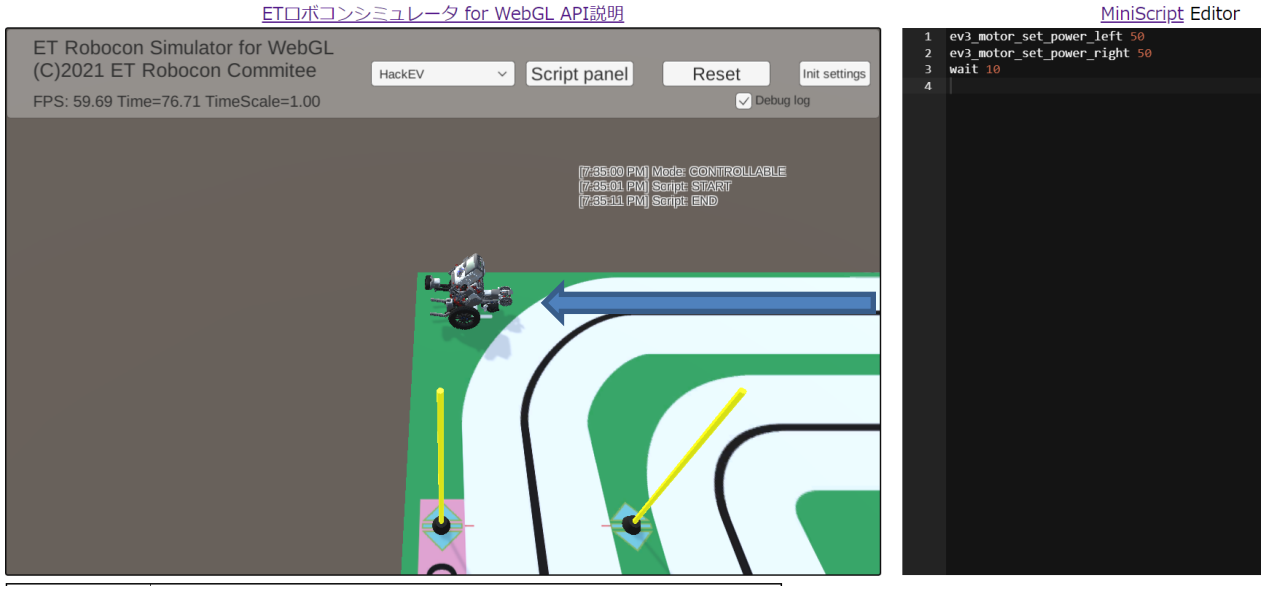


練習1: HackEV(ロボット）を動かそう

① Web版ETロボコンシミュレータをブラウザで起動する。
② エディタに下記プログラムをコピペする。

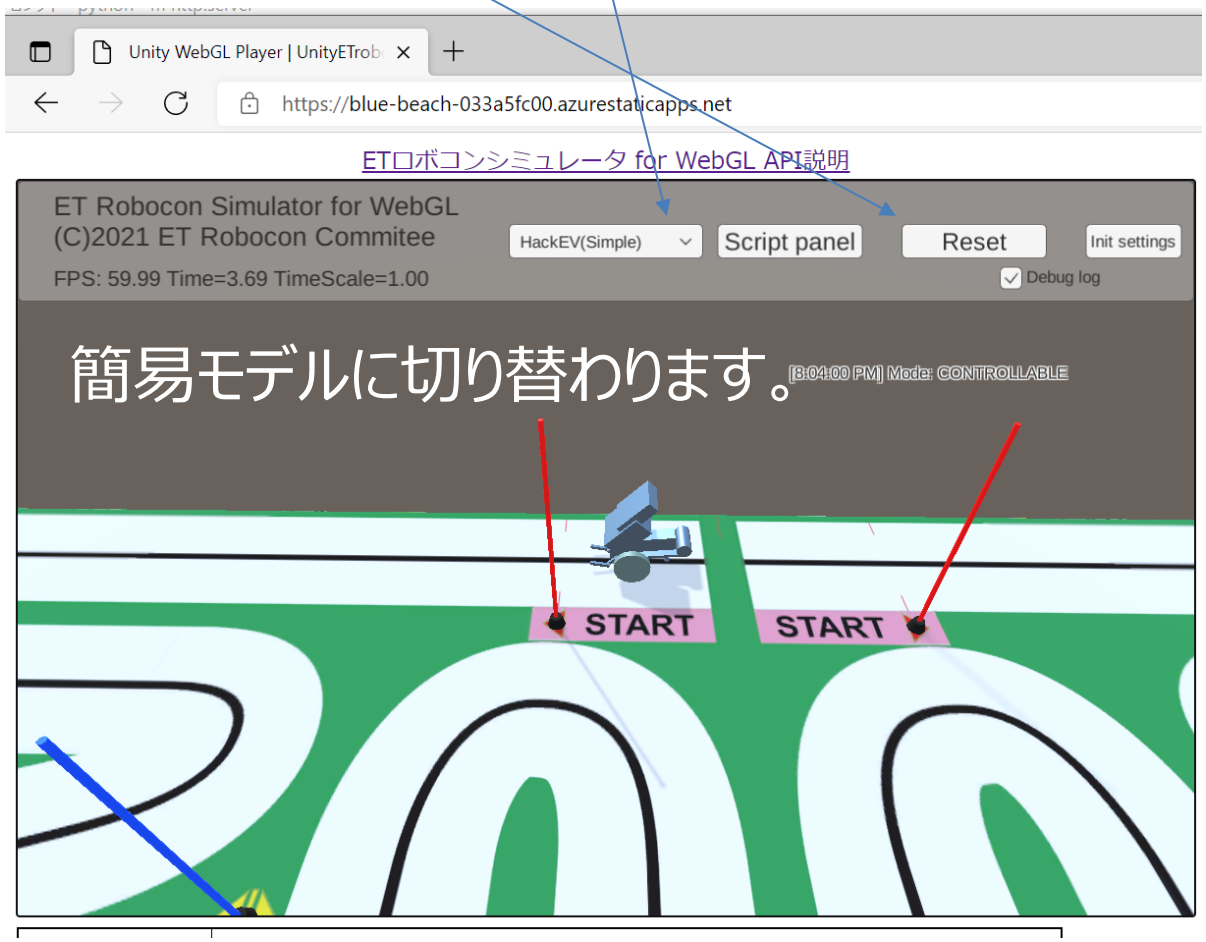
```
ev3_motor_set_power_left 50
ev3_motor_set_power_right 50
wait 10
```

③ シミュレータ部分をマウスでクリックし、Vキーを押す。
④ Run Scriptボタンを押す。



ヒント1：　FPSが20以下の場合(遅いPCの場合)

① HackEV(Simple)を選択する。
② Resetを押す。



実習1: HackEV(ロボット）を自由に動かそう

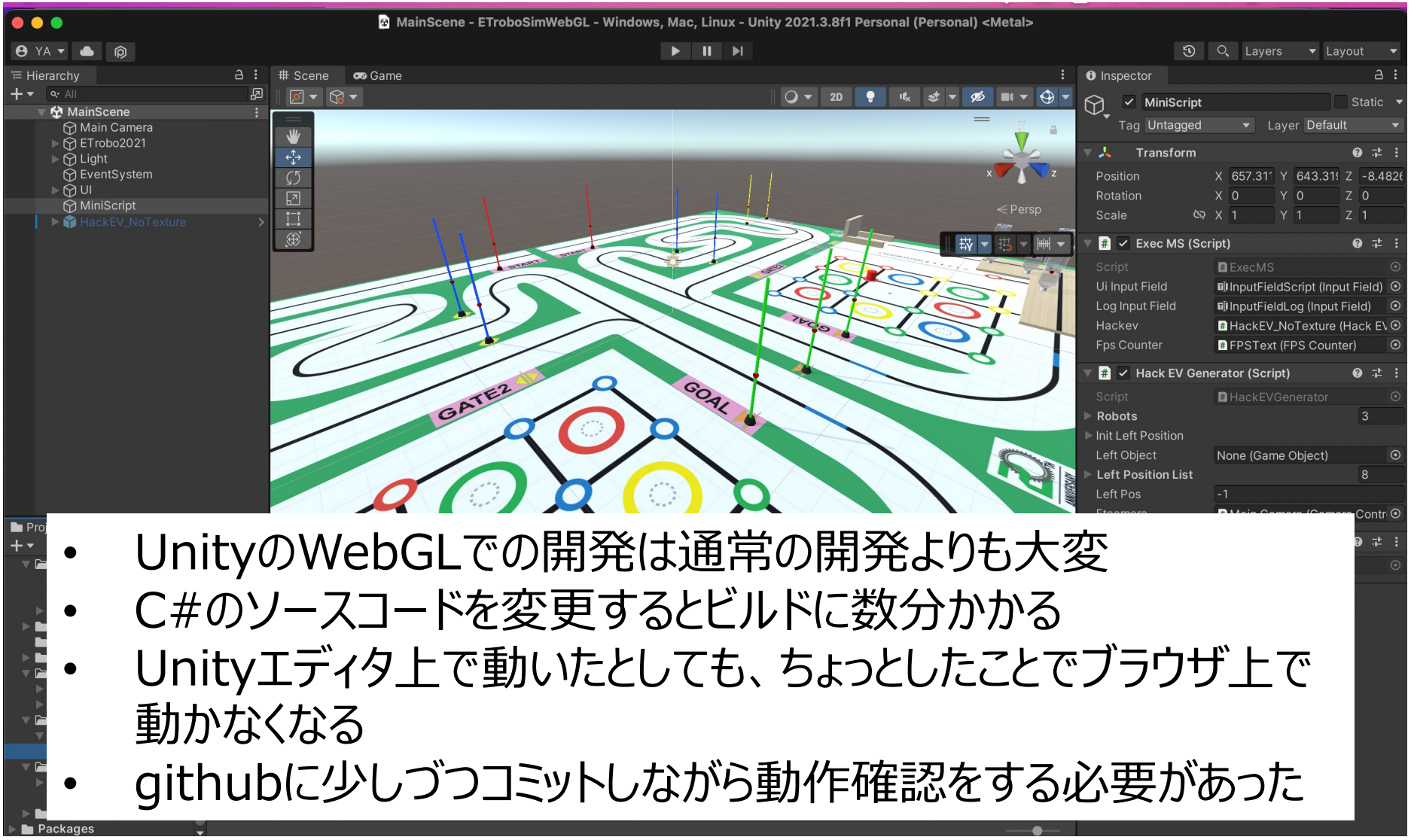
① シミュレータ上のResetを押す。
② エディタ上でev3_motor_set_power_left、ev3_motor_set_power_right、 waitを使って、自由にプログラムを書く。for 文を使って動きを繰り返してみるもよし。

```
for i in range(0,10)
    ev3_motor_set_power_left 50
    ev3_motor_set_power_right 50
    wait 2
    ev3_motor_set_power_left -50
    ev3_motor_set_power_right 50
    wait 0.3
end for
```

③ シミュレータ部分をマウスでクリックし、Vキーを押す。
④ Run Scriptボタンを押す。

この方法でゲート１と２を通過し、ゴールまで走らせてみよう。

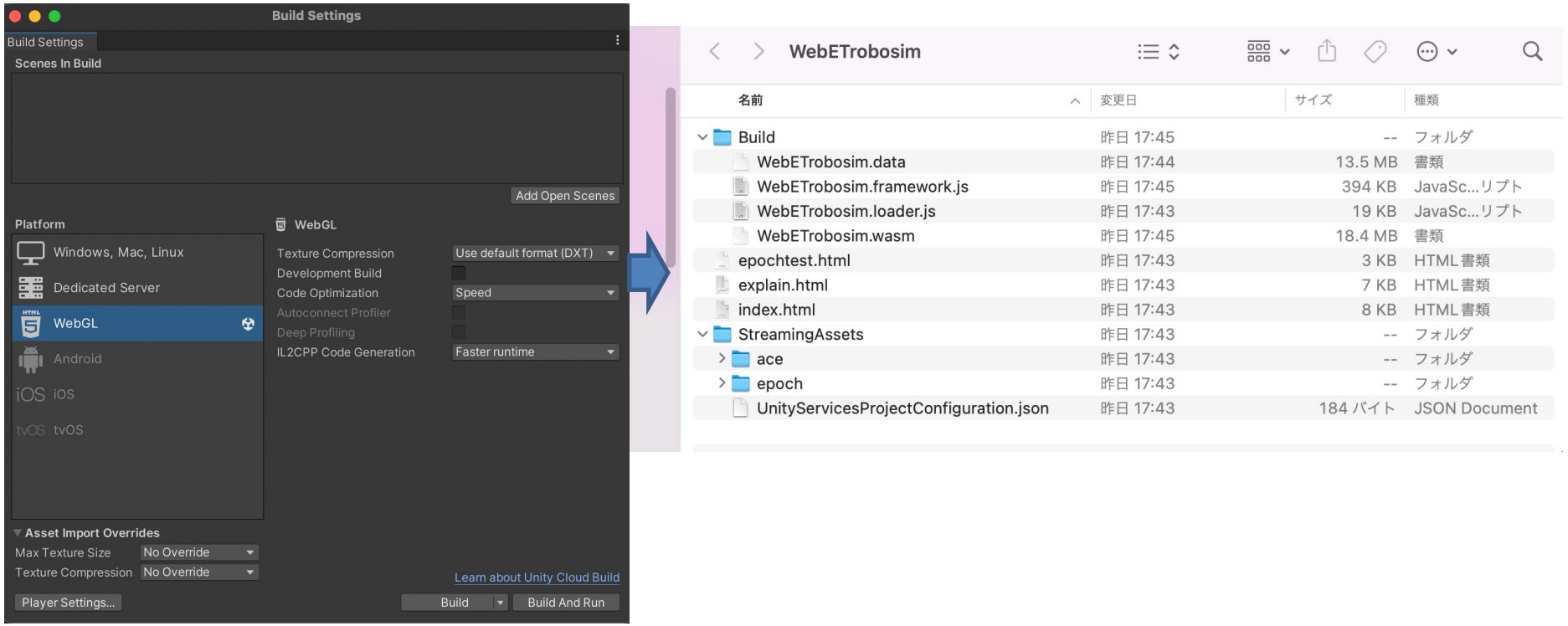
2.4 WebGLによる開発



- UnityのWebGLでの開発は通常の開発よりも大変
- C#のソースコードを変更するとビルドに数分かかる
- Unityエディタ上で動いたとしても、ちょっとしたことでブラウザ上で動かなくなる
- githubに少しづつコミットしながら動作確認をする必要があった

2.5 WebGLでのビルドと実行

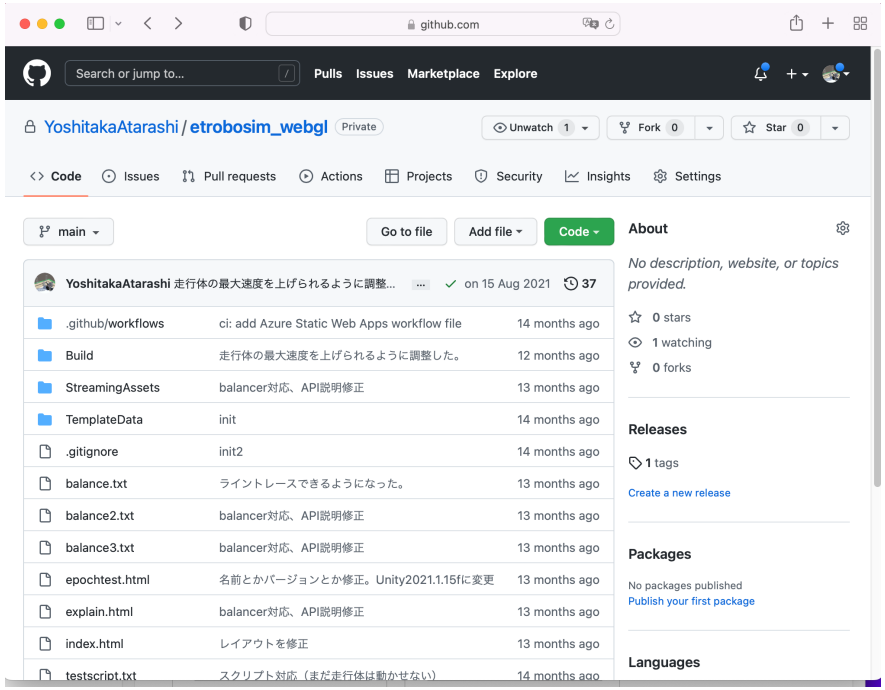
ビルドが成功すると index.html を含むファイル群が生成される。



index.html をHTTP経由でアクセスする必要がある。例えば、python -m http.server を用いることでローカルPC上での動作確認ができる。

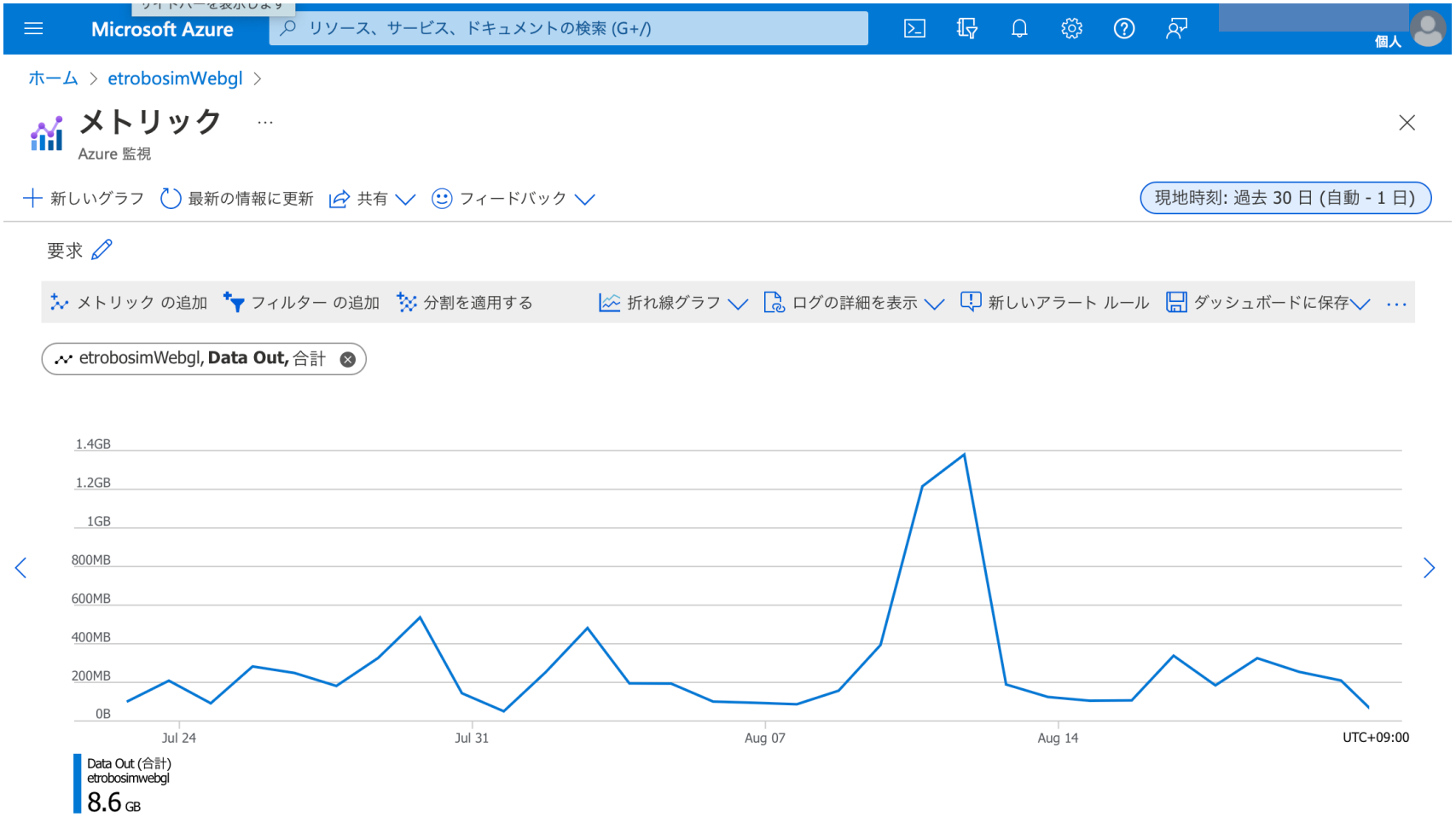
2.6 Azure Static Web Apps へのデプロイ

ビルド結果をgithubにコミットすると、Azure Static Web Appsが自動的にデプロイしてくれる



2.7 メトリックの確認

定常的に、1日当たり200~400MB(4~8回)のアクセスがある

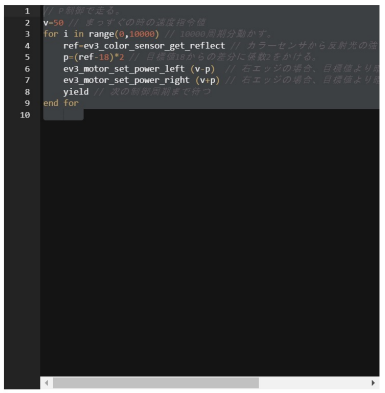


練習2: HackEV(ロボット）でライントレース

① Web版ETロボコンシミュレータをブラウザで起動する。
② エディタに下記プログラムをコピペする。

```
// P制御で走る。
v=50 // まっすぐの時の速度指令値
for i in range(0,10000) // 10000周期分動かす。
    ref=ev3_color_sensor_get_reflect // カラーセンサから反射光の強さを取得する。
    p=(ref-18)*2 // 目標値18からの差分に係数2をかける。
    ev3_motor_set_power_left (v-p)  // 右エッジの場合、目標値より暗い場合は右に曲がるように制御する。
    ev3_motor_set_power_right (v+p) // 右エッジの場合、目標値より暗い場合は右に曲がるように制御する。
    yield // 次の制御周期(5ms)まで待つ
end for
```

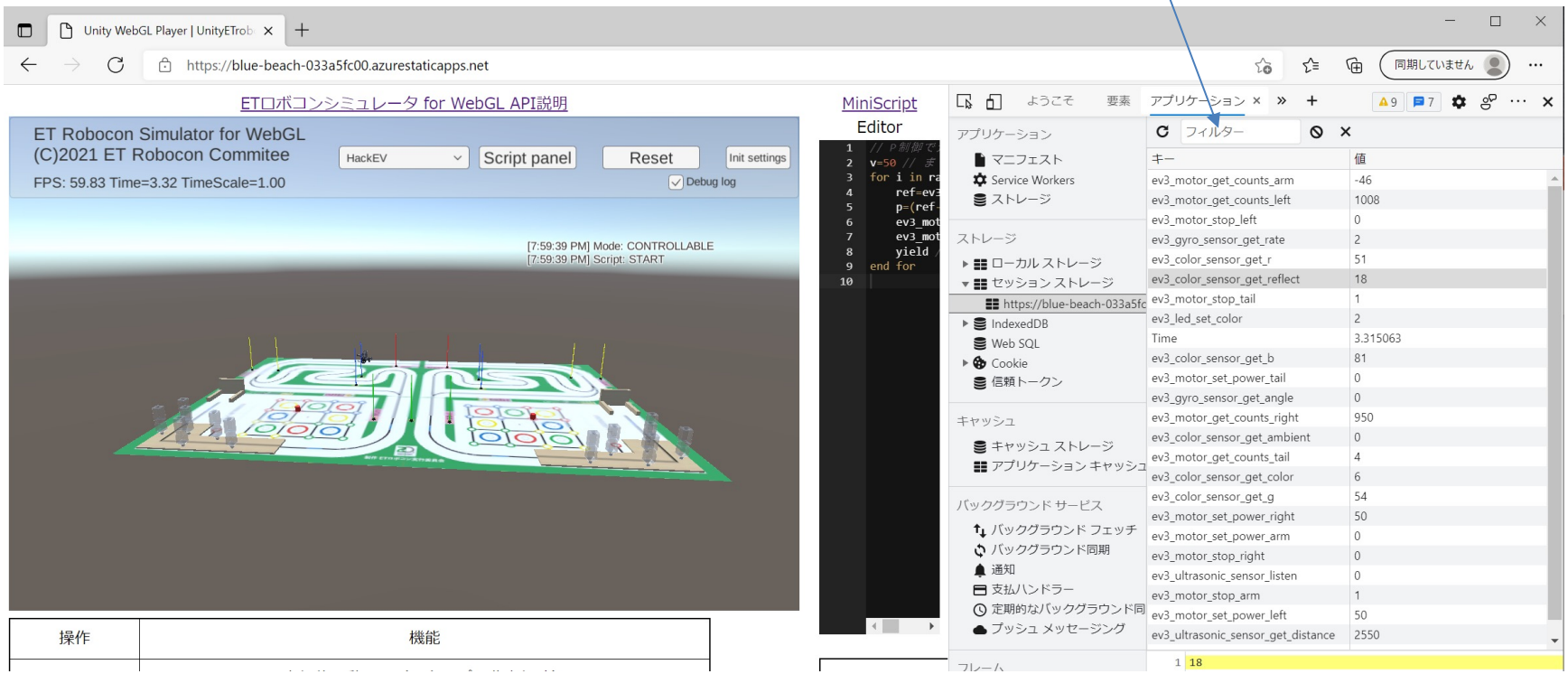
③ シミュレータ部分をマウスでクリックし、Vキーを押す。
④ Run Scriptボタンを押す。



黒いラインの右側に沿って
走っている。

ヒント2：　センサ値の確認方法

① ブラウザでF12キーを押す(IE/Edgeの場合)
② アプリケーション＞ストレージ＞セッションストレージの中のURLをクリックする。
③ スクリプト実行中は、MiniScriptのAPIによって取得可能な値を観測できる。



実習2: HackEV(ロボット）で左エッジライントレース

① Web版ETロボコンシミュレータをブラウザで起動する。
② 練習2のプログラムを改造して、左エッジをライントレースするように変更する。
③ シミュレータ部分をマウスでクリックし、Vキーを押す。
④ Run Scriptボタンを押す。

コース形状が複雑なので、途中でライントレースに失敗するかも。

ゴールまで完走するにはどうすればよいか？

2.8 MiniScript

ロボットを制御するためのプログラミング言語を検討した

・インタプリタ型で簡単に実行できる
・C#で実装されている
・Unity上のオブジェクトを容易に操作できる
・関数拡張が容易である

という利点から、MiniScriptを採用した。

https://miniscript.org

Welcome to MiniScript!

This is the home page for MiniScript, a simple, elegant language for embedding or learning to program.

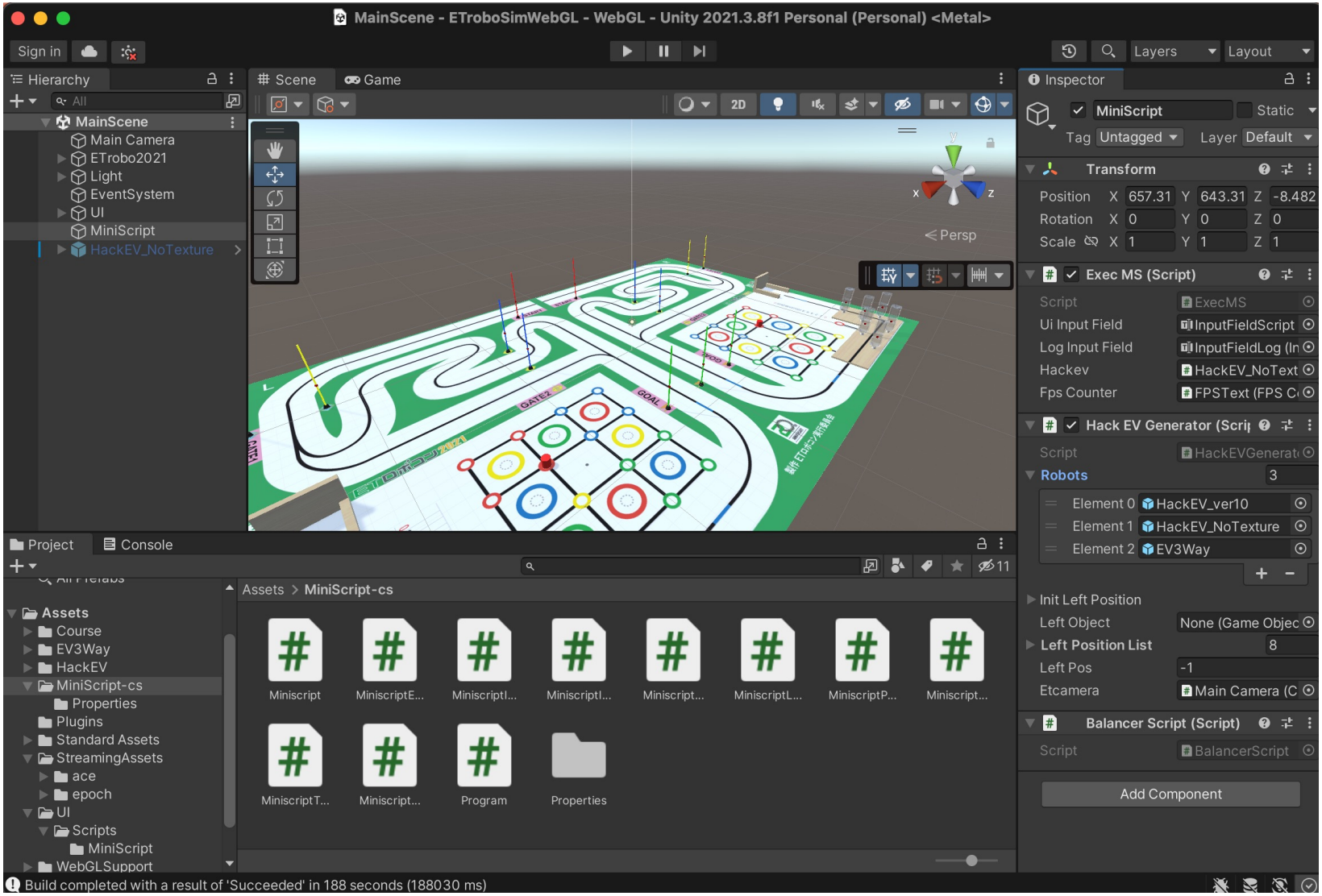
MiniScript is *modern*, *elegant*, *easy to learn*, and *easy to embed* in your own C# or C++ projects. (Or even Kotlin, thanks to a recent third-party project!)

It's also open-source and has been under continuous development since 2016.

Want to learn more? Here's a one-page summary of the language. Also see how it compares side-by-side to Lua, Python, or JavaScript.

2.9 MiniScriptの組み込み

UnityにMiniScript-csを取り込み、MiniScript制御用オブジェクトを追加



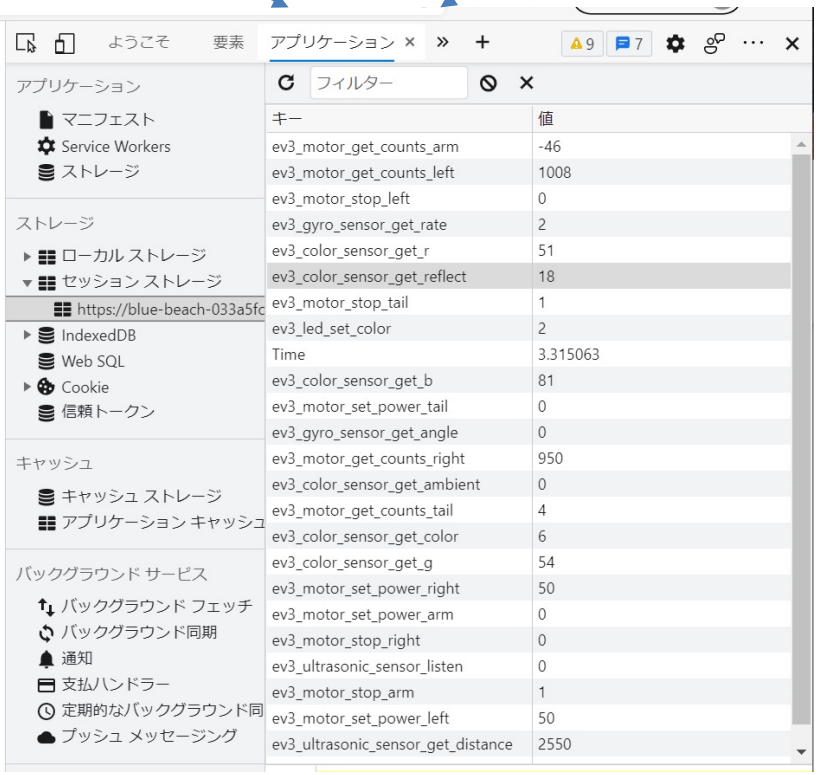
2.10 MiniScriptの関数拡張

miniscriptからセッションストレージを経由し、Unity上のロボットを制御するための関数をC#側に追加

```
static void AddIntrinsics(Dictionary<string, int> sensorMS, Dictionary<string, int> actuatorMS)
{
  if (intrinsicsAdded) return;

  intrinsicsAdded = true;
  Intrinsic f;

  foreach (var key in sensorMS.Keys) {
    f = Intrinsic.Create(key);
    f.code = (context, partialResult) => {
      var rs = context.interpreter.hostData as ExecMS;
      if (rs.isRunning())
      {
        var hv = rs.hackev;
        return new Intrinsic.Result(hv.sensorMS[key]);
      }
      else
      {
        return Intrinsic.Result.Null;
      }
    };
  }

// 以下略
```

Get系関数

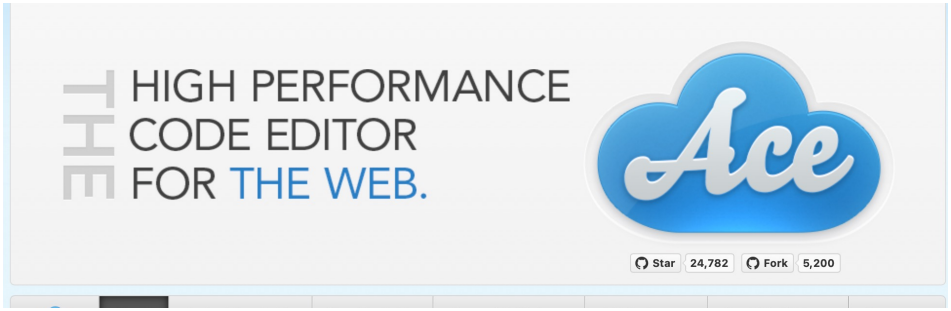
Set系関数



2.11 WEBエディタ Ace

JavaScript+CSSで提供されるWEBブラウザ上で動作するテキストエディタ

https://ace.c9.io



Unityとの連携は、RunCodeボタンを押したときにAceエディタ上の MiniScript コード を送信するJavaScriptを記述することで実現している

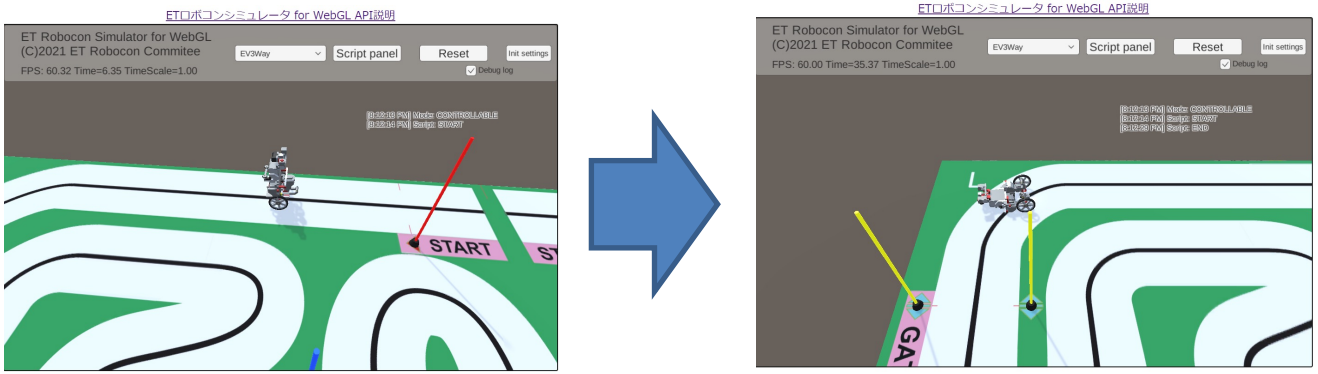
```
var editor = ace.edit("editor");
editor.setTheme("ace/theme/twilight");
editor.session.setMode("ace/mode/miniscript");

function RunCode() {
  code = editor.getValue();
  unityInstance.SendMessage("MiniScript", "RunScript", code);
}
```

練習3: EV3way(倒立振子ロボット）を動かす

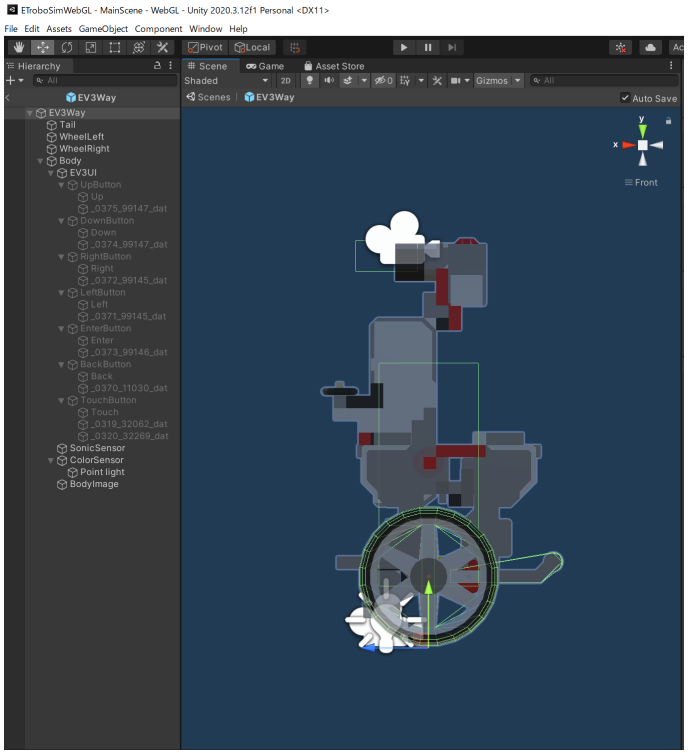
① 上のメニューでEV3Wayを選択し、リセットする。
② エディタに下記プログラムをコピペする。

```
speed=40 //前進速度
str=0 //旋回速度
balance_init // 倒立振子初期化
for i in range(0,3000)
    // 倒立振子制御
    balance_control speed,str, ev3_gyro_sensor_get_rate,0,ev3_motor_get_counts_left,ev3_motor_get_counts_right
    ev3_motor_set_power_left  balance_pwm_l
    ev3_motor_set_power_right balance_pwm_r
    yield
end for
```

③ シミュレータ部分をマウスでクリックし、Vキーを押す。
④ Run Scriptボタンを押す。



ヒント３：　倒立振子とは



ETロボコンで長く愛されてきた倒立振子。
タイヤの回転数と回転速度、車体の角度と角速度を入力として、倒立を維持するようにモータの制御値を計算する。

今回は2017年のEV3Wayのモデル(塚本さん作)を利用し、下記APIを提供している。頑張ればMiniScriptを使って自分で実装することもできる。

- balance_init
- balance_control
- balance_pwm_l
- balance_pwm_r

大本のロジック：　NXTway-GS(2輪型倒立振子ロボット) C API
http://lejos-osek.sourceforge.net/jp/nxtway_gs.htm

参考にしたパラメータ：　EV3way ETロボコンで使用している倒立振子APIパラメーターの再計算手順
https://qiita.com/pulmaster2/items/08edce1ab097539b2deb

実習3: EV3way(倒立振子ロボット）でライントレース

① 上のメニューでEV3Wayを選択し、リセットする。
② 練習3と、練習2or実習2のプログラム参考にして右or左エッジをライントレースするように変更する。
③ シミュレータ部分をマウスでクリックし、Vキーを押す。
④ Run Scriptボタンを押す。

ライントレースだけで、右エッジを30秒台、左エッジを40秒台でゴールまで走れればまぁまぁ速い。

2.12 MiniScript+EV3 APIだけで倒立振子を実現

愚直に倒立振子ライブラリのC実装をMiniScriptに移植すればよい。

https://qiita.com/YoshitakaAtarashi/items/2764c840dd3efce4f060

```
balancer={}
// ローパスフィルタ係数(左右車輪の平均回転角度用)
balancer.A_D = 0.8
// ローパスフィルタ係数(左右車輪の目標平均回転角度用)
balancer.A_R = 0.996

// 状態フィードバック係数
// K_F[0]: 車輪回転角度係数
// K_F[1]: 車体傾斜角度係数
// K_F[2]: 車輪回転角速度係数
// K_F[3]: 車体傾斜角速度係数
balancer.K_F = [ -0.870303, -31.9978, -1.1566, -2.78873 ]
// サーボ制御用積分フィードバック係数
balancer.K_I = -0.44721

// 車体目標旋回角速度係数
balancer.K_PHIDOT = 25.0 * 2.5
// モータ目標回転角速度係数
balancer.K_THETADOT = 7.5

// 制御周期[s]
balancer.EXEC_PERIOD = 0.005

// コマンド最大値
balancer.CMD_MAX = 100.0

// 初期化
balancer.ud_err_theta = 0.0
balancer.ud_theta_ref = 0.0
balancer.ud_thetadot_cmd_lpf = 0.0
balancer.ud_psi = 0.0
balancer.ud_theta_lpf = 0.0

balancer.DEG2RAD=(pi * 2) / 360

balancer.ret_pwm_r=0
balancer.ret_pwm_l=0
```

```
balancer.control=function(args_cmd_forward, args_cmd_turn, args_gyro, args_gyro_offset,
args_theta_m_l, args_theta_m_r)
  tmp_thetadot_cmd_lpf = (((args_cmd_forward / balancer.CMD_MAX) * balancer.K_THETADOT) *
(1.0 - balancer.A_R)) + (balancer.A_R * balancer.ud_thetadot_cmd_lpf)
  tmp_theta = (((balancer.DEG2RAD * args_theta_m_l) + balancer.ud_psi) + ((balancer.DEG2RAD
* args_theta_m_r) + balancer.ud_psi)) * 0.5
  tmp_theta_lpf = ((1.0 - balancer.A_D) * tmp_theta) + (balancer.A_D *
balancer.ud_theta_lpf)
  tmp_psidot = (args_gyro - args_gyro_offset) * balancer.DEG2RAD

  tmp = [ balancer.ud_theta_ref, 0.0, tmp_thetadot_cmd_lpf, 0.0 ]
  tmp_theta_0= [ tmp_theta, balancer.ud_psi, (tmp_theta_lpf - balancer.ud_theta_lpf) /
balancer.EXEC_PERIOD, tmp_psidot ]
  tmp_pwm_r_limiter = 0.0
  for tmp_0 in range(0,3)
    tmp_pwm_r_limiter = tmp_pwm_r_limiter + (tmp[tmp_0] - tmp_theta_0[tmp_0]) *
balancer.K_F[tmp_0]
  end for
  tmp_pwm_r_limiter =(tmp_pwm_r_limiter*20 + (balancer.K_I * balancer.ud_err_theta)*2)
  tmp_pwm_turn = (args_cmd_turn / balancer.CMD_MAX) * balancer.K_PHIDOT

  tmp_pwm_l_limiter = tmp_pwm_r_limiter + tmp_pwm_turn
  balancer.ret_pwm_l = tmp_pwm_l_limiter
  tmp_pwm_r_limiter = tmp_pwm_r_limiter-tmp_pwm_turn
  balancer.ret_pwm_r = tmp_pwm_r_limiter

  balancer.ud_err_theta = ((balancer.ud_theta_ref - tmp_theta) * balancer.EXEC_PERIOD)
+balancer.ud_err_theta
  balancer.ud_theta_ref = (balancer.EXEC_PERIOD * tmp_thetadot_cmd_lpf) +
balancer.ud_theta_ref
  balancer.ud_thetadot_cmd_lpf = tmp_thetadot_cmd_lpf
  balancer.ud_psi = (balancer.EXEC_PERIOD * tmp_psidot) + balancer.ud_psi
  balancer.ud_theta_lpf = tmp_theta_lpf
end function

speed=40 //前進速度
str=0 //旋回速度
for i in range(0,10000)
  // 倒立振子制御
  balancer.control(speed,str,
ev3_gyro_sensor_get_rate,0,ev3_motor_get_counts_left,ev3_motor_get_counts_right)
ev3_motor_set_power_left balancer.ret_pwm_l
ev3_motor_set_power_right balancer.ret_pwm_r
  yield
end for
```

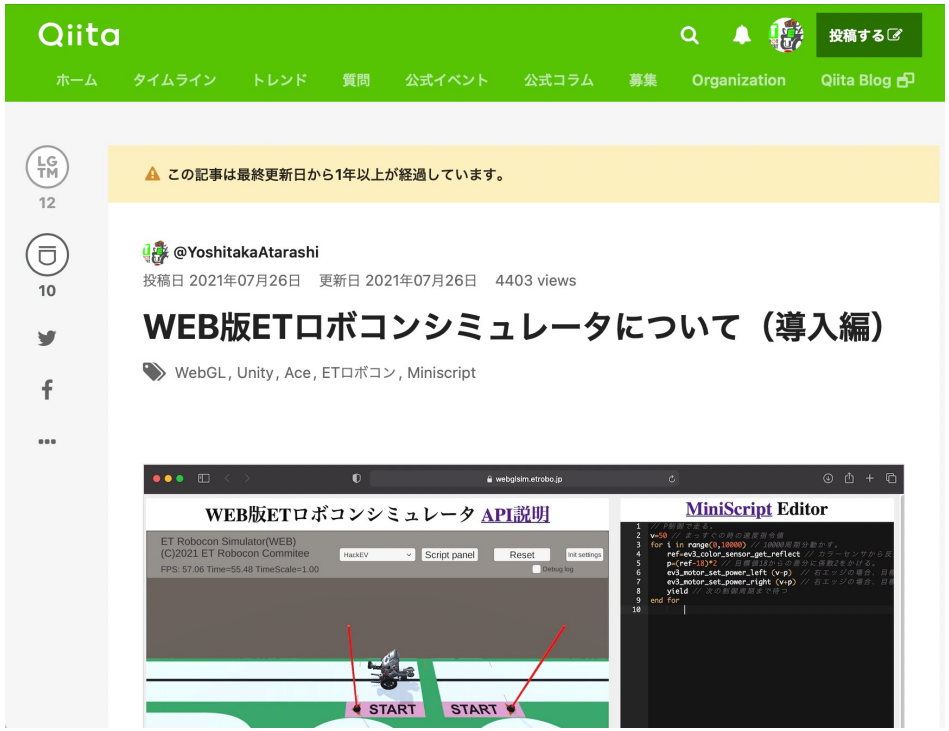
3. さいごに

- 多くのETロボコン実行委員およびTOPPERS/箱庭WGの努力によって、2020年~21年のリモートによるETロボコン大会は無事終了した。リアル大会さながらの興奮を味わうことができたと評判は上々であった。
- WEB版ETロボコンシミュレータは大会向けと比べると機能が限定されているが、インストールレスで簡単に動作させることができるため、様々な応用が考えられる。

A. 参考(Qiita)

https://qiita.com/YoshitakaAtarashi/items/877d1a0cc5ba94a39755

https://qiita.com/YoshitakaAtarashi/items/4e9985f9751260cbe20e



終わり