



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
- 本業は某企業で組み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 ETRobot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が組み込み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

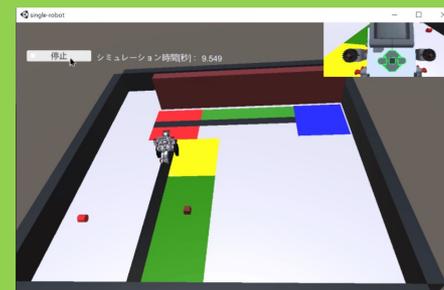
etrobo環境

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
Attrill
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>

☰ YouTube JP etrobo ×

学校法人 日本教育財団 HAL大阪 P076
チームHAL大阪2 1:01.1
0:19.5 GOAL
中間ゲート1
中間ゲート2

ETロボコン2021 チャンピオンシップ大会
eXmotion FUJIFILM SCSK DENSO ADVICS
Tabot 2021 LINCREA HITACHI NSK Tabot 2021
ETロボコン実行委員会

【専門学校HAL】ETロボコン2021 競技部門 優勝 (チャンピオンシップ大会)

338 回視聴 · 2021/12/03

👍 高評価 🗑️ 低評価 ➦ 共有 📄 クリップ ⌵ 保存 ...

1.6 関係者の要求とその評価

- ETRobot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**
 - ライント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再現したい！
 - エントリー、プライマリー、アドバンスの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成する。→ **OK**

1.7 ETロボコン 2022 プロモーションビデオ

<https://youtu.be/v8O5J65B4uA>

YouTube JP etrobo

Go To The Start!

Simulator & Real Race

ETロボコン2022 プロモーションビデオ

532 回視聴 · 2022/02/15

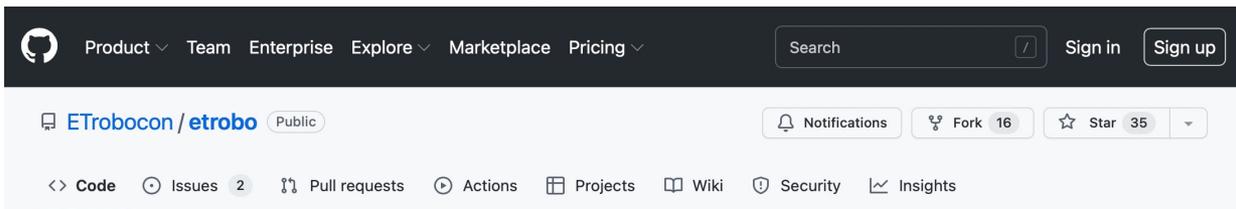
4 低評価 共有 クリップ 保存 ...

ETロボコン
チャンネル登録者数 262人

登録済み

1.8 ETロボコンシミュレータ評価環境

<https://github.com/ETrobocon/etrobo>

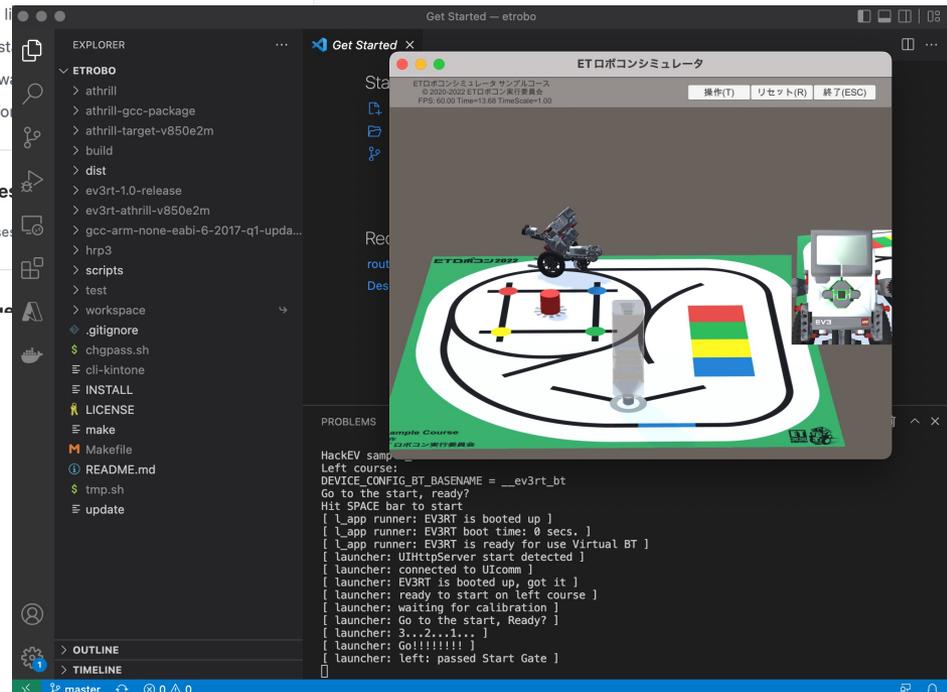


File	Commit Message	Author	Date	Commits
dist	change init pos to 2022 at settings.json.default	jtFuruhata	2 months ago	514
scripts	fix: wrong workspace place	jtFuruhata	last month	514
.gitignore	add: ffutil mmmx get/setfilter	jtFuruhata	13 months ago	514
INSTALL	add: INSTALL file	jtFuruhata	2 years ago	514
LICENSE	first public release	jtFuruhata	2 years ago	514
README.md	update about new Linux Container on ChromeOS	jtFuruhata	3 months ago	514
make	first public release	jtFuruhata	2 years ago	514
update	fix: update always run evenif git pull is failed	jtFuruhata	14 months ago	514

About
ETロボコンのEV3/シミュレータ双方に対応する開発環境です。

Readme
MIT License
35 stars
34 watchers
16 forks
Releases
No releases published

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にする

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

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

MiniScript仕様

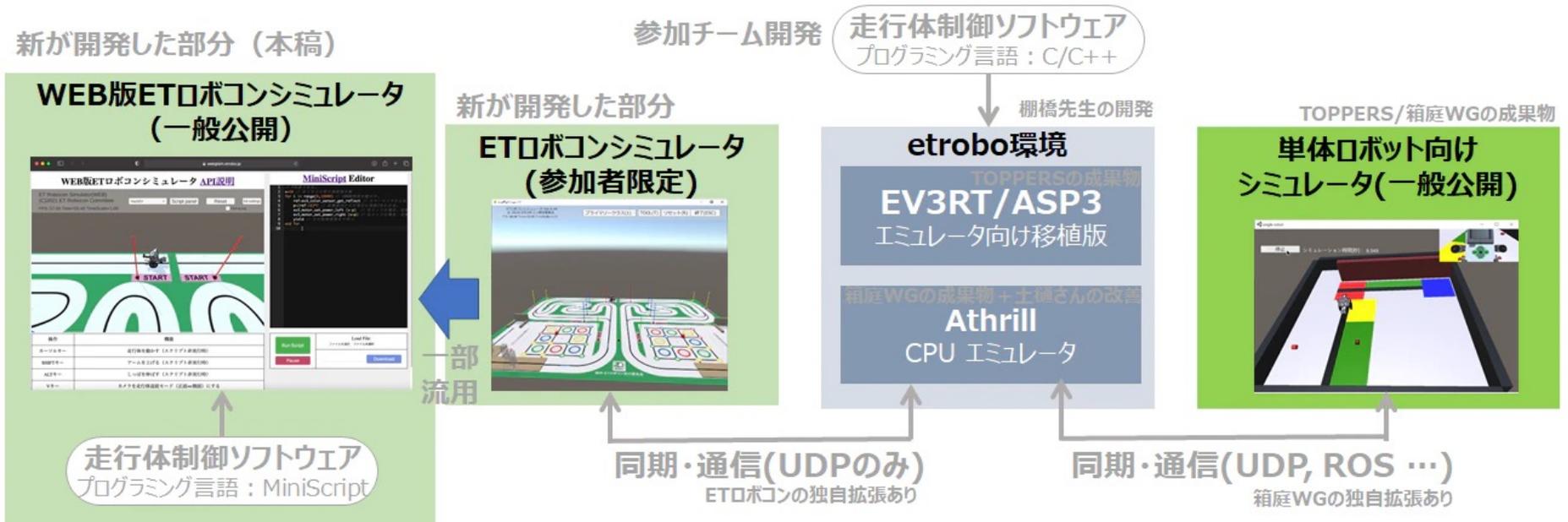
Aceエディタ
(MiniScript用)

スクリプト実行

シミュレータ操作

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ーする。

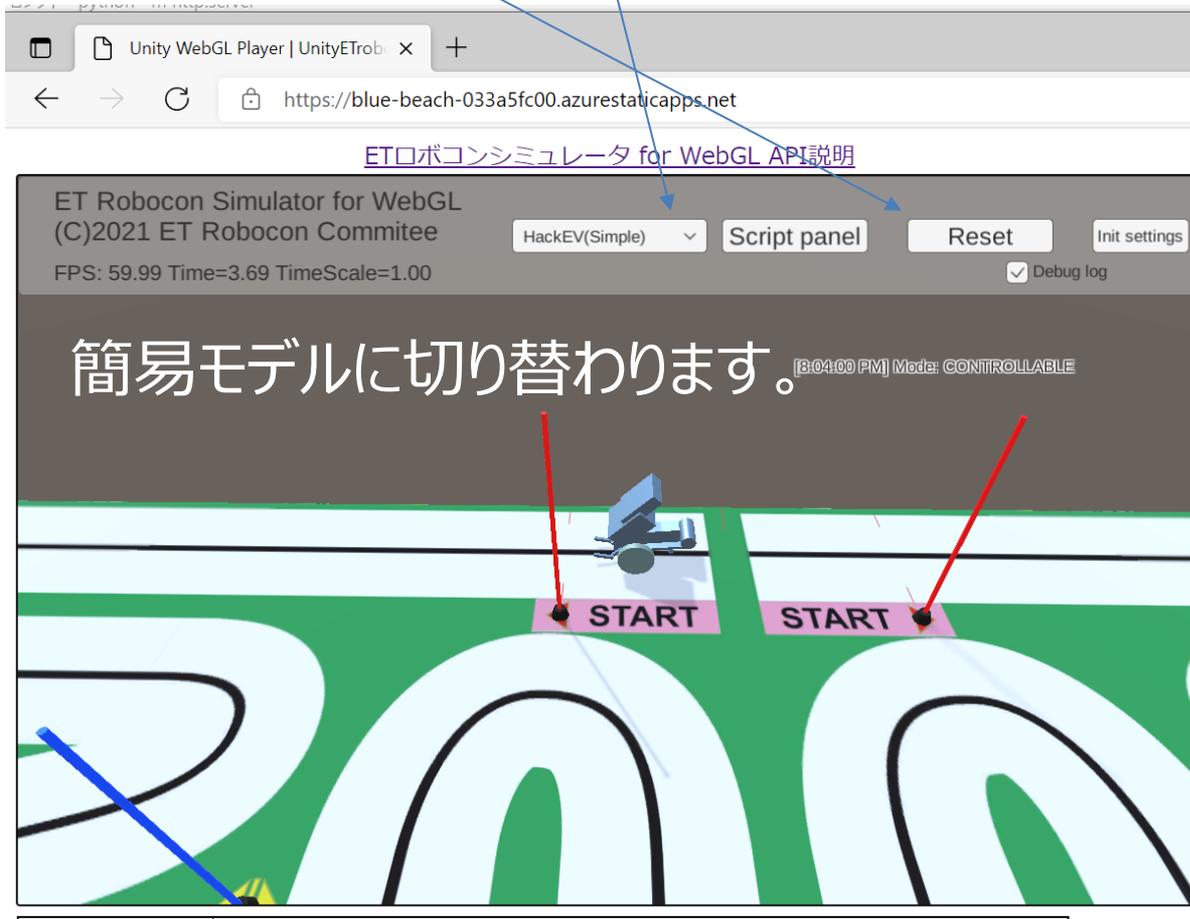
```
ev3_motor_set_power_left 50  
ev3_motor_set_power_right 50  
wait 10
```

- 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

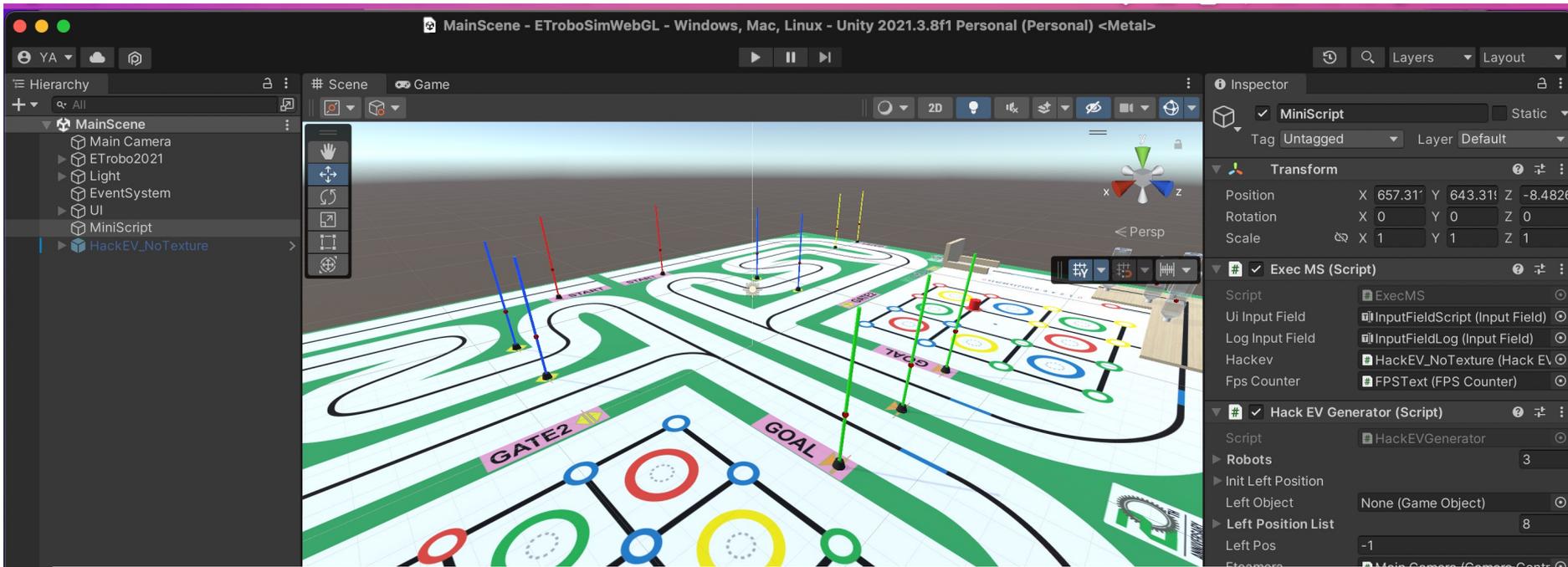
- ① シミュレータ上のResetを押す。
- ② エディタ上でev3_motor_set_power_left、ev3_motor_set_power_right、waitを使って、自由にプログラムを書く。for 文を使って動きを繰り返してみるもよし。

```
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。

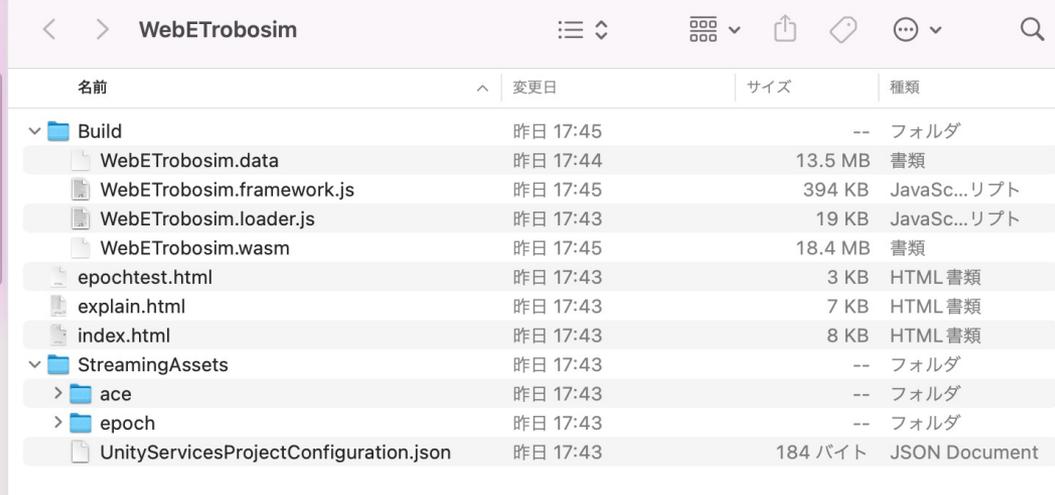
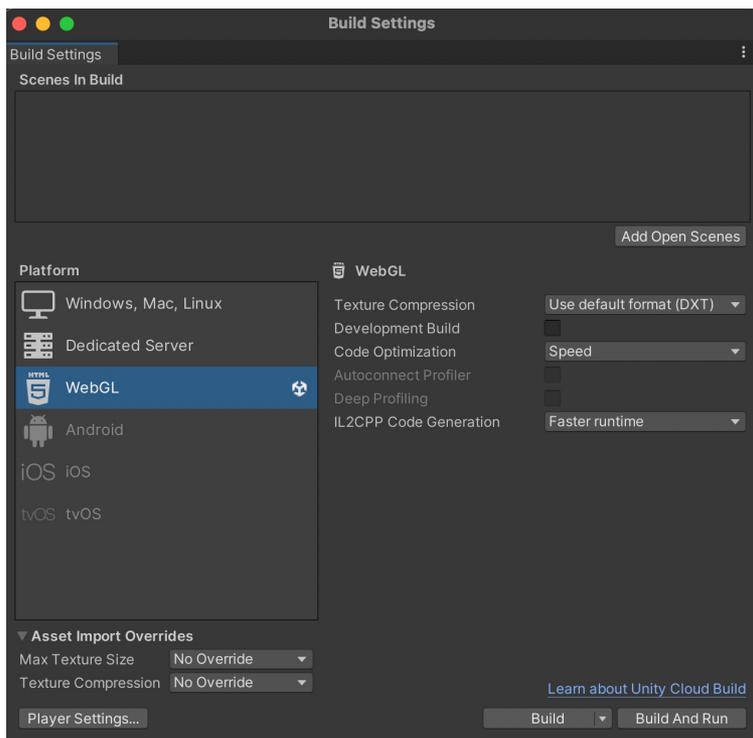
2.4 WebGLによる開発



- 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に少しず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

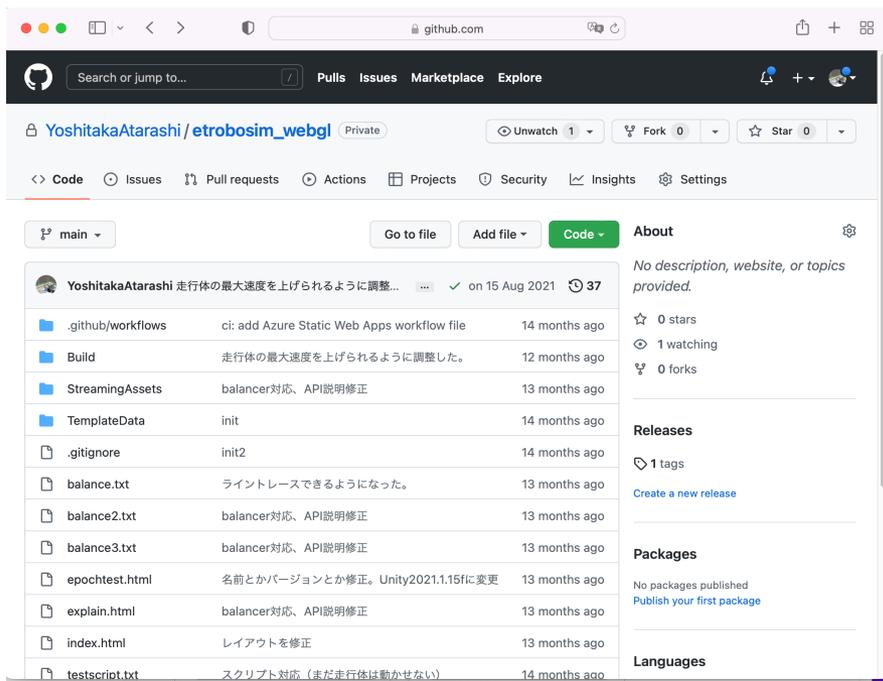
ビルドが成功すると index.html を含むファイル群が生成される。



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。

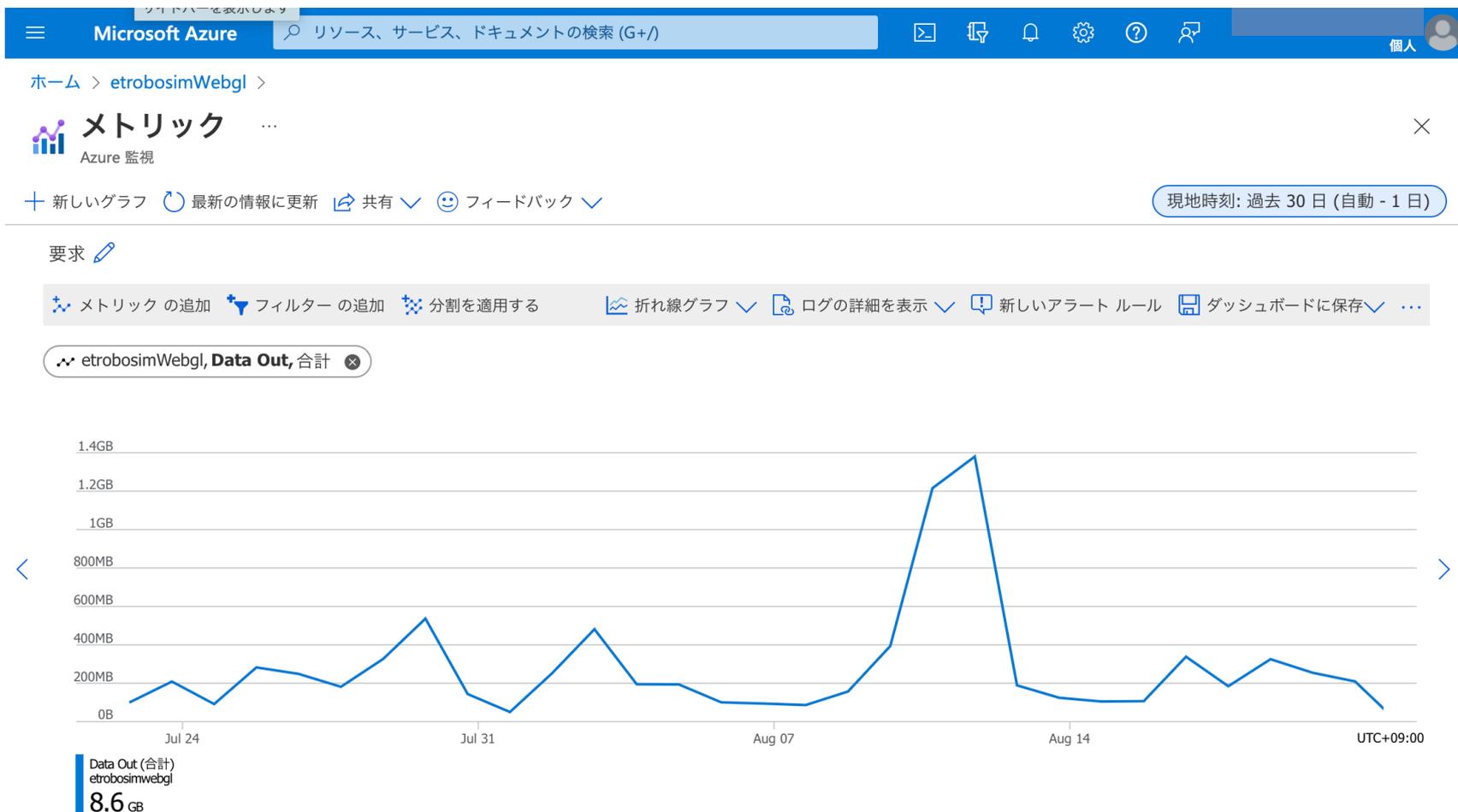
2.6 Azure Static Web Apps へのデプロイ

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



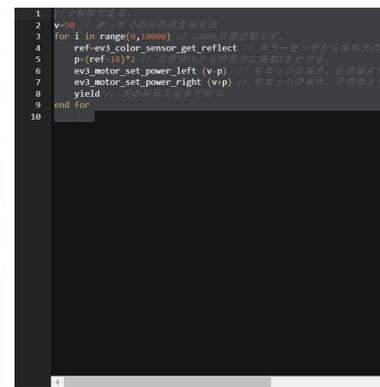
練習2: HackEV(ロボット) でライトレース

- 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。

The screenshot displays the ET Robocon Simulator for WebGL API running in a browser. The browser's developer tools are open, showing the MiniScript Editor and the Application tab. The Application tab is expanded to show Session Storage, and the URL 'https://blue-beach-033a5fc0.azurestaticapps.net' is selected. The right pane shows a list of application data with columns for key and value. A blue arrow points from the text in the hints to the Application tab in the developer tools.

キー	値
ev3_motor_get_counts_arm	-46
ev3_motor_get_counts_left	1008
ev3_motor_stop_left	0
ev3_gyro_sensor_get_rate	2
ev3_color_sensor_get_r	51
ev3_color_sensor_get_reflect	18
ev3_motor_stop_tail	1
ev3_led_set_color	2
Time	3.315063
ev3_color_sensor_get_b	81
ev3_motor_set_power_tail	0
ev3_gyro_sensor_get_angle	0
ev3_motor_get_counts_right	950
ev3_color_sensor_get_ambient	0
ev3_motor_get_counts_tail	4
ev3_color_sensor_get_color	6
ev3_color_sensor_get_g	54
ev3_motor_set_power_right	50
ev3_motor_set_power_arm	0
ev3_motor_stop_right	0
ev3_ultrasonic_sensor_listen	0
ev3_motor_stop_arm	1
ev3_motor_set_power_left	50
ev3_ultrasonic_sensor_get_distance	2550

実習2: HackEV(ロボット) で左エッジライトレース

- 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!](#))

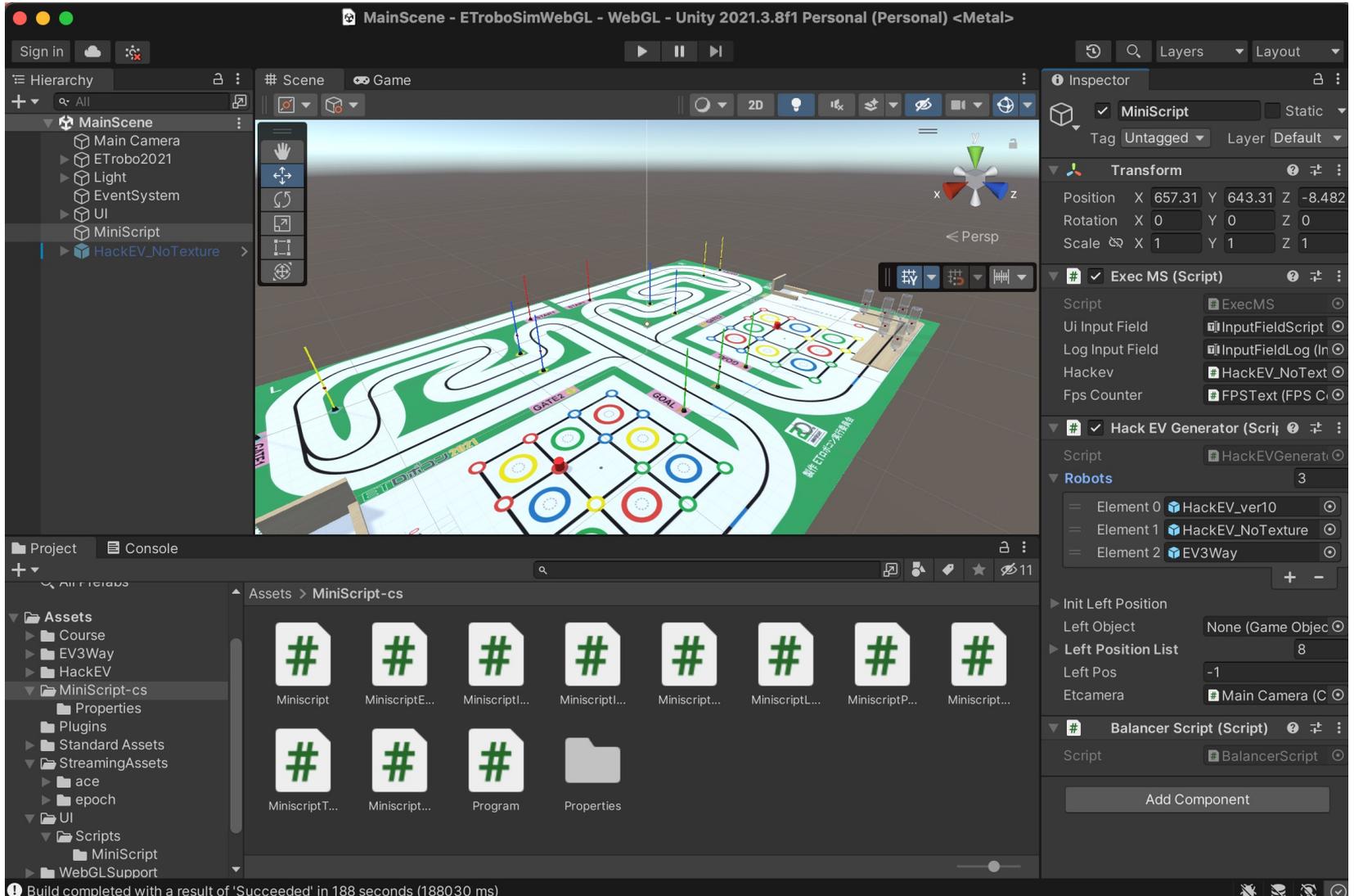
It's also [open-source](#) and has been under continuous development since 2016.

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;
            }
        };
    }
}
```

Get系関数

Set系関数

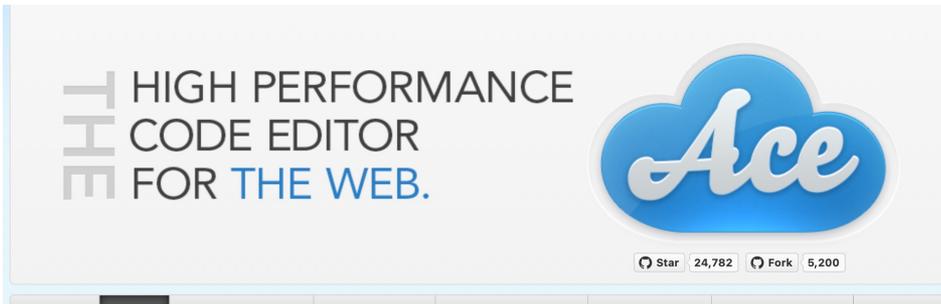
キー	値
ev3_motor_get_counts_arm	-46
ev3_motor_get_counts_left	1008
ev3_motor_stop_left	0
ev3_gyro_sensor_get_rate	2
ev3_color_sensor_get_r	51
ev3_color_sensor_get_reflect	18
ev3_motor_stop_tail	1
ev3_led_set_color	2
Time	3.315063
ev3_color_sensor_get_b	81
ev3_motor_set_power_tail	0
ev3_gyro_sensor_get_angle	0
ev3_motor_get_counts_right	950
ev3_color_sensor_get_ambient	0
ev3_motor_get_counts_tail	4
ev3_color_sensor_get_color	6
ev3_color_sensor_get_g	54
ev3_motor_set_power_right	50
ev3_motor_set_power_arm	0
ev3_motor_stop_right	0
ev3_ultrasonic_sensor_listen	0
ev3_motor_stop_arm	1
ev3_motor_set_power_left	50
ev3_ultrasonic_sensor_get_distance	2550

// 以下略

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

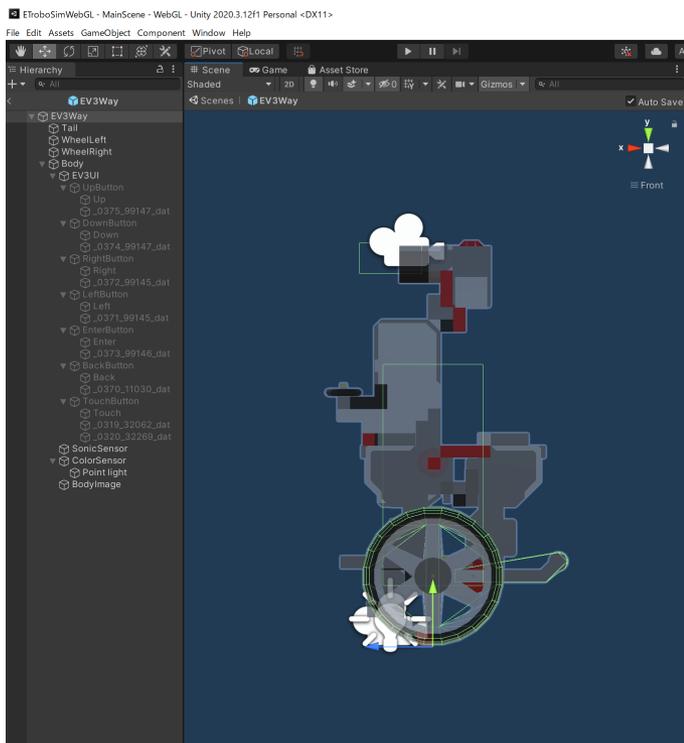
- ① 上のメニューでEV3Wayを選択し、リセットする。
- ② エディタに下記プログラムをコピーする。

```
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 : 倒立振り子とは



ETロボコンで長く愛されてきた倒立振り子。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輪型倒立振り子ロボット) 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り子APIパラメータの再計算手順

<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ばまあまあ速い。

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>

The screenshot shows the Qiita website interface. At the top, there's a green navigation bar with 'Qiita' logo and various menu items like 'ホーム', 'タイムライン', 'トレンド', etc. Below the navigation bar, there's a yellow warning banner stating 'この記事は最終更新日から1年以上が経過しています。' (This article has passed more than 1 year since the last update). The main content area features the article title 'WEB版ETロボコンシミュレータについて (導入編)' by user '@YoshitakaAtarashi', published on July 26, 2021, with 4403 views. The article tags include 'WebGL, Unity, Ace, ETロボコン, Miniscript'. Below the text, there are two screenshots: one of the 'WEB版ETロボコンシミュレータ API説明' interface showing a robot on a track with 'START' buttons, and another of the 'MiniScript Editor' showing code for controlling the robot.

The screenshot shows the Qiita article page for 'WEB版ETロボコンシミュレータについて (解説編)'. The article is by 'WebGL, Unity, ETロボコン, Miniscript'. It includes social media sharing icons for LGTM, Twitter, and Facebook. Below the article content, there's a section titled 'WEB版ETロボコンシミュレータの位置づけ' (Positioning of the WEB ET Robot Competition Simulator). The text explains that searching for ET robot simulators leads to various information, making it confusing, and that this page aims to simplify the positioning. Below the text is a flowchart diagram showing the relationship between different simulators and environments. The diagram includes boxes for 'WEB版ETロボコンシミュレータ (一般公開)', 'ETロボコンシミュレータ (参加者限定)', 'etrobo環境 EV3RT/ASP3 エミュレータ向け移植版', 'Athrill CPU エミュレータ', and '単体ロボット向けシミュレータ(一般公開)'. Arrows indicate dependencies and relationships, such as '同期・通信(UDPのみ) ETロボコンの独自拡張あり' and '同期・通信(UDP, ROS ...) 他WGの独自拡張あり'.



終わり