

# PC同好会

ついに編纂された！  
PC部の歴史

プログラミング初心者でもできる！  
お絵描きアプリの作り方

大好評！ VRジェットコースターの全て  
その他webにも多数掲載！！

2022年

# 音楽と展覧の会

# PC部の歴史

われわれは、三つの問題を立てることにしたい。

- PC部とは何か？—全てである。
- PC部は、これまで、甲陽においてどのようなものであったか。—無であった。
- PC部は何を要求しているのか。—何がしかのものになることを。

PC部の歴史を編纂したいと思っていたが去年はなんだかんだで部誌が出せずできなかつたので、今年こそはと思ってまとめた。

形式

- ・西暦
- ・105回生(今の高2生)の学年
- ・説明

注: 正確には「PC同好会」だがここでは「PC部」と表記する。

## 2019年

### 105回生 中1三学期～中2二学期

1月

現部長のK君がバスケ部をやめ、PC部を設立した。初期メンバーは五人で全員が中1だった。

5月

甲陽には自由の学びの奨学金という制度があり、自主的な活動にお金が欲しくなった時に審査に通れば奨学金がもらえる。PC部はパソコン一台を申請し、ありがたいことに奨学金で買っていただいた。

結構スペックのいい物を無理して買っていただいた

これはパソコンを学校に持てて来られない部員が使ったり音展の展示用のパソコンとして使った。

6月

少し部員が増えて7人になった。(なお全員105回生)

先述のパソコンが届いた。

7月

このころ105回生は中2となり、地理の授業でケッペンの気候区分を習っていたので、気候区分を判定するプログラムを書いてパソコンに判定させるのが流行った。

先に30を行った方が勝ちになるゲームをいろいろな言語で作った。

五目並べ、オセロ、マイクラのmod、ホームページなども作った。

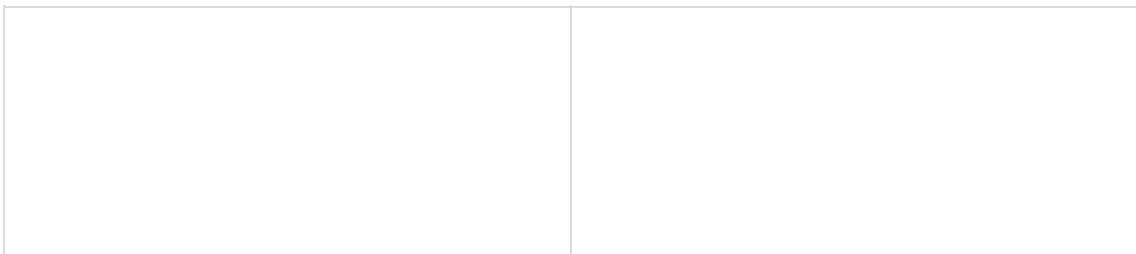
11月

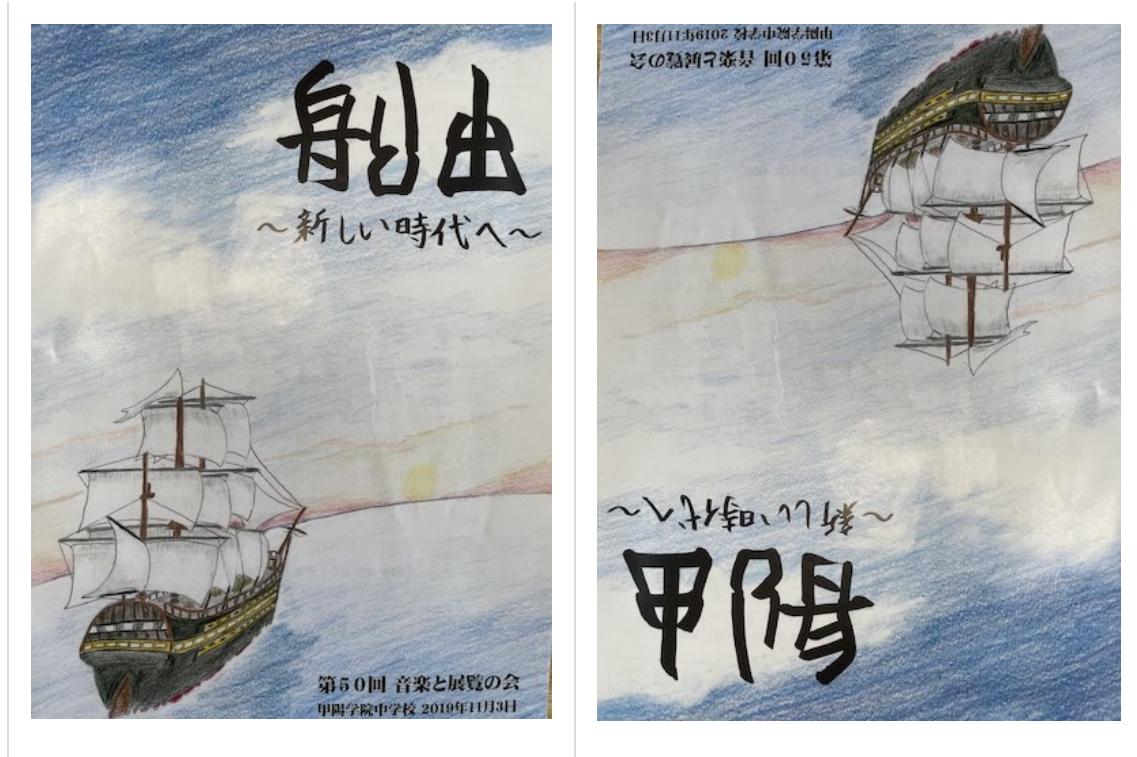
初めての音展となった。

写真: 2019年の音展パンフレット

2019年のテーマは「船出」で、中学では珍しい二文字だった。

このパンフレットは逆から読むと「船出」が「甲陽」になるというすごいデザインとなっている。





教室の半分を借り、プログラミング言語についてのポスター数枚、ホームページ、オセロ(コンピューターと対戦)を展示した。

ホームページでは他の団体の展示の紹介や準備中の写真などを掲載した。

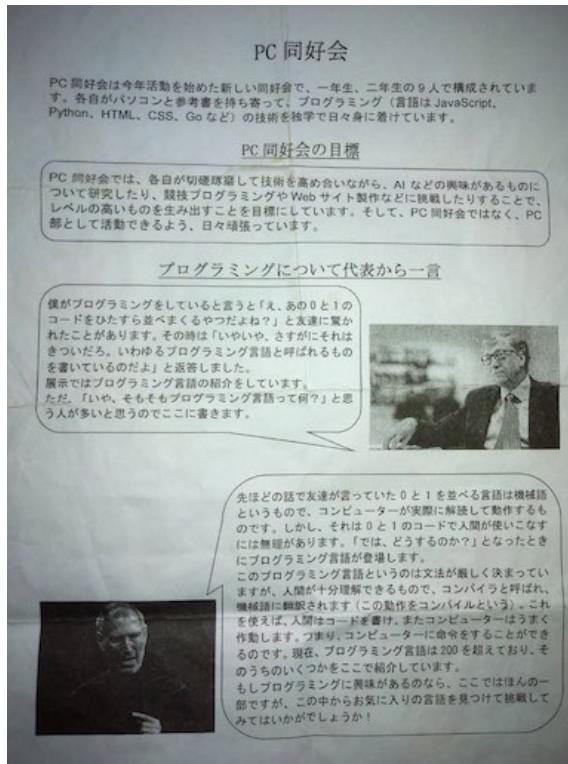
写真: PC部準備中の様子

手前のパソコンでホームページ、オセロの展示をし、右側の仕切り板でポスターを展示した。



オセロに関してはオセロ全国大会一位の子とそのお父様が来られて、ボコボコにされた。

写真: 配布した部誌



また、106回生が数名入部して部員は10人になった。

音展後にある部員はポールを転がして迷路のゴールを目指すというゲームを作った。

12月

パソコンを自作する部員が出てきたり、競技プログラミング(以下競プロ)を始める部員が出てきた。

## 2020年

### 105回生 中2三学期～中3二学期

1月

国語で漢文の返り点を学んだので、ある部員が返り点から読み順を出力するプログラムを書いた。

2月

学年末考査があるとおもいきや...

#### 新型コロナウイルスの拡大で学校が休みになった

休みが続きゲームにも飽き出したので競プロに勤しむ部員が増え、競プロブームが起った。

学校が再開して忙しくなる7月まで続いた。

5月

この頃部員がGitHubという、ファイルの変更履歴を保存できるサービスの勉強を始めた。

オンライン授業で三角関数を習った。

7月

中学校はスマホの持ち込みが禁止されている。もちろんパソコンも駄目なのでこの頃までは登校した時に先生にパソコンを預かってもらっていたが、ある日PC部専用ロッカーを買っていただくことになり、それが届いた。

また、PC部のロゴのデザインはこのロッカーの鍵や物理講義室の鍵が付けてあったオブジェのようなものが元になっている。

8月

この頃からある部員が音展のために、中学校を360度カメラで撮影し、中学のストリートビューを作り始めた。  
three.jsというものを使った。

10月

ある部員がSVGという画像形式からPNGという画像形式に変換するプログラムを書いた。

ストリートビュー作成のため長い廊下を通行止めにし(顔が写るのが許されていなかったため)、校内のあちこちを撮った。

11月

2回目の音展を迎えた。写真: 2020年の音展パンフレット

2020年のテーマは「道」だった。

写真の絵はPC部員のH君が描いた。



第51回 音楽と展覧の会

2020年11月3日

甲陽学院中学校

3年生: 9:00~11:00 1年生: 11:00~13:00 2年生: 13:00~15:00

この音展では展示スペースを設けず、学校のサーバーを借りてオンラインで発表した。

ホームページやストリートビューを校内のQRコードからアクセスして見ていただいた。

音展後にはマリオの1-1を自作したりUnityという開発環境でリズムゲームを作る部員が出た。

## 2021年

105回生 中3三学期～高1二学期

2月

先述のマリオが完成した。

この頃PC部の部員数が最も多く、12名いた。

3月

105回生は全員高校へ進学となり、中学のPC部は実質消滅した。

PC部のロッカーはおそらくどこかの団体に渡されたと思う。

高校ではスマホやパソコンの持ち込みは許可される。

4月

高校でPC部が再び設立された。

部員は9人となった。

このタイミングで複数の部員はパソコンを買い替えた。

6月

ある部員はVue.jsというものの学習を始めた。

ある部員は高校版のストリートビューを作り始めた。

7月

ある部員はカメラを[Raspberry Pi](#)というスマホサイズのコンピューターで制御することで定点撮影をし、タイムラプスを作った。

写真: その部員が使ったRaspberry Pi



[全国中学高校Webコンテスト](#)(以下webコン)に参加登録をした。都市鉱山について調べることにした。

8月

ある部員は高校を3Dモデルで再現した。

9月

高校の音展は通常9月だが、この年は延期となって11月開催となった。

10月

数学でベクトルを習った

ある部員はベクトルを使ってお絵描きができるプログラムを書いた。

また[ウェブデザイン技能検定](#)(3級)に合格した部員もいた。

webコンのためにある部員はMarkdownという言語をホームページなどで使われる

HTMLという言語に変換するプログラムを書いた。

11月

音展のホームページやwebコンのための作品作りをした。

webコンに関しては最初の審査まで時間がなかったため、11月19日に行われた校外学習の摩耶山ハイキングでパソコンを持参し、山頂で調整をした。

本来このハイキングは105回生が中一の時に行われるはずだったが、三年連続で雨で中止となり高一ですることになった。

写真 右: 山頂にある掬星台 左: 山頂からの眺め





3度目の音展を迎えた。

写真: 2021年の音展パンフレット

2021年のテーマは「男祭り音 the moun 展」だった。



音展のホームページや高校のストリートビューを公開した。

音展の頃は学校で風邪が流行っており、PC部の半分はしんどかった。

12月

PC部員3名はポートアイランドの理化学研究所にある富嶽を使ってプログラミングができる会に参加した。

とは言つても使うのは富嶽のごく一部だが

冊子の小問の誘導に沿ってプログラムを組んでいくことで素数ゼミ ([wiki](#)) の謎を解こうという形式だった。そこそこ大変だった。

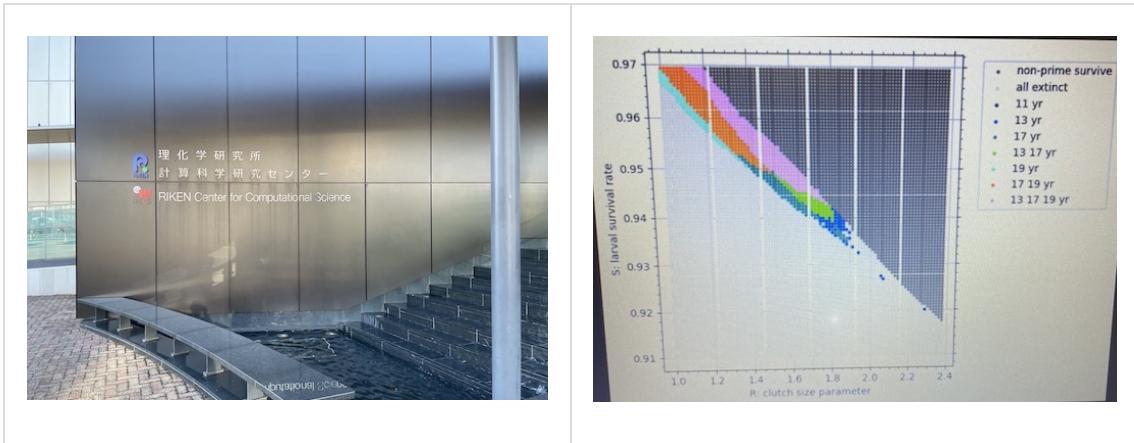
富嶽を使えば並列して多くの処理を行えるので普通のパソコンでは到底終わらない計算量でも数十秒でできた。

当日は正月前で、多くの研究者が休暇中に計算してもらえるよう富嶽の予約を入れていたのでなかなか順番が回って

こなかった。

また、NHKの取材が入り、部員の一人のインタビューが放送された。

写真 右:理化学研究所入口 左: 計算結果を表にしたもの



また、ある部員はウェブデザイン技能検定(2級)に合格した

## 2022年

### 105回生 高1三学期～高2二学期

1月

ある部員はPHPの勉強を始めた

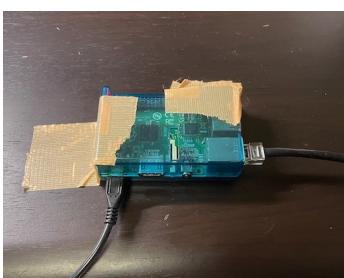
PC部はwebコンの審査を次々と通過していった。

最後の審査に向けて、ページの英訳やその他最終調整をした。

2月

ある部員はRaspberry Pi(先述)が家にあるのを思い出し、発掘して使い出した。

写真: そのRaspberry Pi(小学生の時に買ったもの)



音楽のために買い替えようと思ったが半導体不足で新しいものが販売されておらず買えなかった。

3月

webコンで金賞をとった。

作品 [右ポケットの中の資源を求めて](#)

校長先生に直接表彰していただいた。

ある部員はボディトラッカーを自作しようとした。また、Gaia EDR3という星のデータを使って星空の画像を作った。

4月

ある部員はPHPを使って掲示板を作った。また通信方式やWebSocketという技術の勉強をした。

ある部員はMIDI形式の音楽ファイルを再生するプログラムを書いた。(キーボードが表示される凝ったもの)

5月

ある部員はMikuMikuDanceを使って3Dモデルを動かした。

ある部員はnode.jsとWebSocketの技術を使って掲示板を作ろうと思った。

ある部員はNeovimというものの学習を始めた。

6月

105回生は修学旅行で北海道を行った。広かった。

写真 右:北海道の写真 左: さっぽろテレビ塔



7月

夏休みに入り、ある部員は掲示板を完成させた。

写真: 掲示板のイメージ画像

ホーム スレッド一覧

#0 祝 掲示板完成

投稿内容を入力(256文字まで)

0/256 送信

コメントを読み込みました

<<前50コメ

次50コメ>>



1 2022年7月30日1時6分13秒: ID: HCxyEtCj

完成しました。自由に使ってください。

2 2022年7月30日2時27分23秒: ID: ujFGeBop

うおおおおお

3 2022年7月30日2時27分33秒: ID: ujFGeBop

あ、下向きになったんや

4 2022年7月30日2時28分33秒: ID: HCxyEtCj

やっぱり1コメが読めたほうがいい

5 2022年7月30日2時30分32秒: ID: HCxyEtCj

PC部には先行公開して音展のときに一般公開します

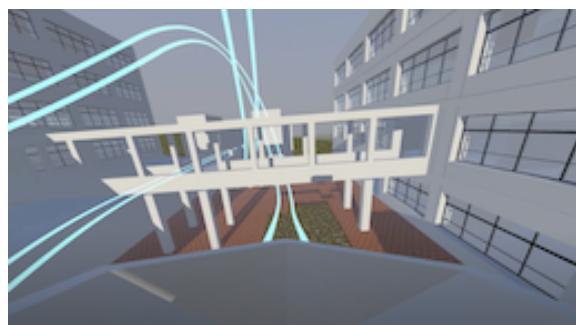
8月

部員が一名増えた。

[プログラミング甲子園](#)のモバイル部門に出場することになった。ある程度の基礎は完成させた。

ある部員はVRジェットコースターの制作を始めた。VRゴーグルで流す映像を作った。

写真: 映像からの切り抜き

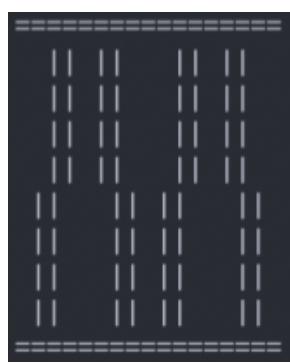


PC部公式Tシャツを作った。写真 右:シャツの前面 左:シャツの後面



全面の「PC」は実行できるコードでできている。

写真: 出力の文字列(PC部のロゴ)

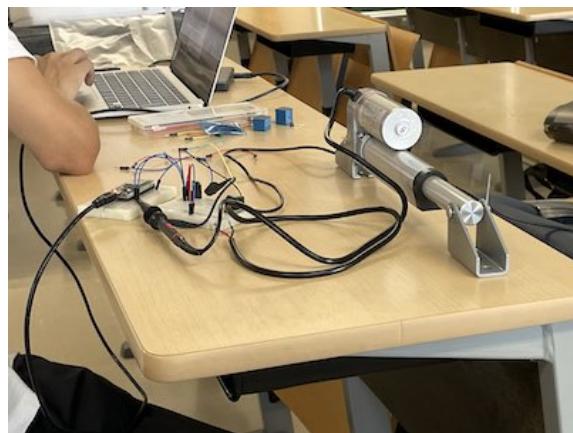


多くの部員が旅行に行った。

9月

音展の準備を進めた。ある部員はVRジェットコースターの椅子を動かす回路を作った。

写真: 制作の様子



ある部員は仕事がなくなったので二人で対戦できる早押しゲームを作った

写真: ゲームの画面

「開始」ボタンを押して再試合



また、画像をボトルキャップで表現した時の完成イメージを出力するプログラムを書いた

写真: 出力例



ある部員は音展の公式ホームページを作った

写真: ホーム画面



# パソコン甲子園途中経過報告

## 参加

PC同好会はいま、パソコン甲子園という大会に参加しています。大会には、2人チームでプログラミング技術を競うプログラミング部門、1~3人チームでAndroidアプリモバイル部門、個人でCG作品を描き上げるいちまいの絵CG部門の3部門があるのですが、1番可能性がありそうなモバイル部門にPC同好会から3人出場しています。

今のところいい感じにことが運んでいるのでそのことについて書きたいと思います。

## 予選

モバイル部門では毎年アプリを作る上でのテーマが出されます。今年は「温故知新」でした。温故知新と言われてもすぐにいいアイデアが思い浮かぶ感じでもないので、とりあえず100個くらいブレインストーミングで出してみました。その中からまともなものを1割ほど抽出し、その中から〇〇するアプリに決め、「〇〇」という名前にしました(なんとなく本選で戦う他チームに知られたら不利なのではないかと思ってなんとなく伏せています)。それからこのアプリについての企画書を一生懸命書いて提出してみました。これが意外とハードでPC同好会あるあるの期限ギリギリ提出になりました。その後、パソコン甲子園公式ページから参加チームがみれたのでみてみると、30チームくらい参加していました。しかもお隣さんのあの灘高校も参加してて怖かったです。プログラミング部門でどうせ今年も無双するんだからモバイル部門で本戦出場枠ひとつ減らすのやめてほしいです。(※後日談:無双しました)

## 本戦出場決定~今

本戦通ってました。よかったです。ちなみに灘さんも通ってました。しかし、本戦出場チームは甲陽や灘のような進学校は少なめで工業高校系がやはり多い感じでした。そのせいで運営さんから「甲陽工業高等学校」って書類で間違えられちゃうレベルです。甲陽の知名度の低さも如実に示す悲しい出来事でした。

本戦出場が決まったチームは運営さんが作ってくれたslackのグループに入ります(この時灘のチームの人からDMも来ました。なんやかんやでお互い気にしてるんですかね?)そのグループでいろいろと必要書類とかが送られて来ます。貸与されるスマホの取り合いに乗り遅れてほぼ残り物みたいなやつになったり、シンプルに提出書類の提出をすっかり忘れてたりちょこちょこヘマしつつもがんばってます。ちなみに本戦は福島県の会津大学でやるので会津に無料で連れてってもらいます。やったー。当日は会津大学でプレゼンもあるし、アプリ紹介ボードを作ってきて自分たちのアプリをアピール時間もあります。今からでも緊張します。何はともあれアプリを完成させないことには始まらないので製作中なんですが、なにせ音展が被っちゃってるで滞っています(これまたPC同好会あるある)。

しかし、せっかく運営さんに「夢あるアプリ」だとして本戦出場の権利をもらったからには、絶対間に合わせて会津行って来ます。そして優勝します。楽しみにしていてください。数ヶ月後パソコン甲子園のホームページには高らかに「甲陽工業高等学校」の名が掲げられているはずです。

# ガチ初心者のためのプログラミング入門

by ZOI

## 自己紹介など

こんにちは。ZOIといいます。パソコン同好会でいろいろつくって遊んでる人です。突然ですが、「プログラミングやってみたいけど難しそう」と思ったことはありませんか?「数学や英語の能力がいる」とか、「めっちゃタイミング早くないといけない」とか思われている(かもしれない)プログラミングですが、実際にやってみると(めっちゃ高度なことをしない限り)そんなに難しくはないんです!まずは、騙されたと思ってやってみてください。

## お絵描きアプリを作つてみる

### 関数と変数

さっそくプログラミングをしていきます。特別に必要なものはほとんど無いです。ひとまずパソコンとインターネット回線、そしてYouTubeで動画を探せる程度のタイピング能力さえあればどうにかなります。

ネットで「プログラミング 始めたか」とかで検索すると、黒い画面に「Hello World」とか表示させるようなページがいっぱい出てくると思います。それはそれで後で役に立つのですが、驚くほど楽しくないです。なので、そんな地味なことはせず、絵を描くアプリを作つてみましょう。

まずは、Googleとかで「Open Processing」と検索して、上のほうにでてきたページを開いてください。これはプログラムを書いたり実行したりする場所です。うわーいきなり英語かーと思うかもしれません、そんなことは気にせず右上の「Create a Sketch」というボタンを押してみましょう。はい。さっきのデザイナーみたいなページとは一転、いきなりプログラミング感あふれるページになりましたね。なにやら最初からコードが書いてあります。これを実行してみましょう。上部中央の三角ボタンを押すと実行されます。

画面が灰色になるだけではなく、マウスを振ると丸がいっぱい描画されます。なんでしょうこれ。ではコードを見てみましょう。上部中央の `</>` というボタンでコードの画面に戻れます。こんなコードがかいてあるはずです。

```
function setup() {
  createCanvas(windowWidth, windowHeight);
  background(100);
}

function draw() {
  circle(mouseX, mouseY, 20);
}
```

なんかすごいプログラミングってかんじの見た目ですが、落ち着いて見てみるとそこまで読めないものではないんじゃないでしょうか。まず、`{}` で大きく2つのブロックに分かれていますね。`function` は関数という意味です。プログラミングでは、とりあえず現時点では「関数」 = 「コードをまとめたカタマリ」と思ってもらって大丈夫です。関数は、呼び出されたとき、その内容を上から順番に実行してくれます。

この2つの関数は、それぞれ `setup()`、`draw()` と名前がついています。このコードが実行される順番ですが、OpenProcessingが勝手に`setup()→draw()→draw()→draw()→draw()...`というふうに呼び出してくれます。(なので、この`draw`を勝手に他の名前にすると動きません。)

まず、1行しかない`draw()`の中を見てみましょう。さっきマウスの場所に丸が描かれていましたが、`draw()` という名前から想像できるように、この関数の中で描かれています。`circle(~, ~, ~)` というのがありますが、この `circle()` ってのも関数です。そんな関数どこにも定義されてないぞ?って思うかもしれません、この関数は

OpenProcessingが勝手に定義しているものです。 `circle(x, y, r)` と実行すれば、中心(x, y)、半径rの円が描かれることになっています。今回、xやyには `mouseX`、`mouseY` という謎の文字が書いてますね。この場所には数字が入るはずなのですが。このような、数を表している文字を「変数」といいます。(厳密には違いますが...)これらの変数は、名前から分かる通り、マウスのXY座標を表しています。(「関数」「変数」と立て続けに出てきましたが、関数は処理をするのに対して、変数は数値などある値を表します。`mouseX()` のように、カッコの有無を間違えないように気をつけましょう。)これによって、「マウス位置に丸を描く」という処理ができるのです。ちなみに、`circle`を2つ、こんな感じで書いてみると...

```
// setup() は省略 (実際は書いてね)

function draw() {
    circle(mouseX, mouseY, 20);
    circle(mouseX + 100, mouseY, 20); // x座標を少しずらしてみた
}
```

同時に2つの円が描かれるコードができました。

ちなみに、僕がコードに書いた `//` ですが、これは「この行のここから先は無視してね」という記号です。こういう無視される部分はコメントと呼ばれ、今回のようにちょっとした注釈を書いたり、その行を一時的に無効にするという使い方もできます。この行は消しても普通に動作します。

次に`setup()`を見てみましょう。これは最初に1回だけ実行されるコードなので、描く前の初期設定などが書かれています。

```
function setup() {
    // キャンバス(描く部分)を作成
    // 高さは画面の高さ、幅は画面の幅
    createCanvas(windowWidth, windowHeight);

    // 背景色を「100(グレー)」に設定
    // 0で真っ黒、255で真っ白
    background(100);
}

// draw() は省略
```

まあここについてはあんまり考えなくていいです。

## 公式ドキュメント

最初から書かれていたコードは大体読めたと思いますが、自分で書くとなればまだ問題があるはずです。さつき、`circle()` や `createCanvas` などはOpenProcessingが定義している関数であると言いましたが、もちろん他にもいろいろあります。一体どんな関数や変数を定義しているのでしょうか?もちろん勘で見つけるものではなく、公式がまとめてくれています。

コードを書く画面の右上に赤い「Save」というボタンがありますが、そのちょうど下にある本のマークを押してみましょう。ここに表示されている関数・変数はすでに定義されているので使えます。

(<https://p5js.org/reference>にも同じ内容が書かれています。)

なにか一つ使ってみましょう。たとえば線をひいてみましょう。検索に「line」と打ち込んでみると、Shapeというカテゴリに`line`というのがありますね。この状態では使いかたが全くわからないので、リンクをクリックして詳

しい情報を見てみましょう。 英語のページですが、最悪Exampleさえ見ておけばなんとかなります。 どうやら  $(x_1, y_1)$  から  $(x_2, y_2)$  まで線を引いてくれるようです。 つまり、

```
// setup() は省略

function draw() {
    // circleは一旦動かないようにしておく
    // circle(mouseX, mouseY, 20);
    line(0, 0, mouseX, mouseY);
}
```

として実行してみれば、原点(左上)からマウス位置まで線が引かれます。 公式ドキュメントが読めるようになれば、できることが大きく広がります。

## 変数の利用

さて、ここでの目的は絵を描くことでした。そのためには、「一瞬前のマウスの位置」から「今のマウスの位置」に線を引きたいですよね。さて、「一瞬前のマウスの位置」ってどう取得するんでしょう。さっき、`draw()` は繰り返し呼ばれるといいました。一回前に `draw()` が呼ばれたときの `mouseX`, `mouseY` が使えたらしいのですが。なので、次のときに使うために、毎回 `mouseX`, `mouseY` を「メモ」しておくことにしましょう。

こんなときに役にたつのが、さっきも紹介した「変数」です。いま使っている変数「`mouseX`」は OpenProcessing が定義してくれたものですが、自分で勝手に定義することもできます。やってみましょう。

```
let x = 20;
let y = 30;

// setup() 省略

function draw() {
    circle(x, y, 20);
}
```

これを実行してみると、 $(20, 30)$  に円が描かれます。`let` という行が増えていますね。`let` というのは変数を定義するやつです。変数はなにか値を表すのですが、1行目では `x` という名前の変数をつくり、20を表すように設定しています。

また、変数は `=` を使うことでその値を書き変えることができます。なので、こんなふうに書き変えてみると、`x` や `y` が1ずつ増えていき、円が少しづつずれて描かれるはずです。

```
// let 省略
// setup() 省略

function draw() {
    circle(x, y, 20);
    x = x + 1;
    y = y + 1;
}
```

ここまでできたら後は簡単です。

さっきのコードを少し変えてみます。

```
let x = 0;
let y = 0;

// setup() 省略

function draw() {
  // circle(x, y, 20);
  line(x, y, mouseX, mouseY);
  x = mouseX;
  y = mouseY;
}
```

マウスの動きに沿って線を描くことができました。(いまさらですが、「一瞬前のマウス位置」は pmouseX, pmouseY で取得できるらしいです。まあ勉強ということで。)

## if分岐とフラグ

ただ、線が左上から始まってしまうのは嫌ですね。最初の1回は線を描かないようにしましょう。初めの1回かどうかをメモっておく、ということで、変数が使えそうです。とりあえず、今何回目かを表す変数「count」を作りましょう。

```
// let x, y 省略
let count = 0;

// setup() 省略

function draw() {
  // 省略
  count = count + 1;
}
```

ここで、「countが0のときは線を描かない」という処理がしたいですよね。こんなときには if が使えます。if(条件){処理}という形でつかいます。実際にやってみたほうが早いです。

```
// let 省略

// setup() 省略

function draw() {

  // A != B で「AとBが等しくない」という意味です

  if(count != 0) {
    line(x, y, mouseX, mouseY);
  }
  x = mouseX;
  y = mouseY;
  count = count + 1;
}
```

ただ、これではうまくいきません。マウスが枠のなかに無いのにdraw()が呼びだされることがあります。なので、「今までに一度でもマウス位置が更新されたか」を取得したいですね。(直前のマウスの位置が0,0でない、というのでも動くのですが、マウスがたまたま0,0にいったとき動かないで...)もし一度でもマウスに反応したら「`x != 0`」もしくは「`y != 0`」となるので、これを使いましょう。2つの条件A、BをORでつなぎたい場合、「`A || B`」と表せます。ちなみにANDのときは「`A && B`」です。

あともう一つ。今回、「すでに動いたか」を表す変数として「`isMoved`」を作るのですが、これには数字ではなく「`はい`」または「`いいえ`」という値を入れたいです。こんなときは、変数に数字ではなく「`true`」(はい)、「`false`」(いいえ)を入れることができます。もちろん四則演算はできなくなりますが、ifの括弧の中に直接入れることができます。

```
// let x, y 省略
let isMoved = false;

// setup() 省略

function draw() {
  // このifはなくても動くのですが、既にisMovedなら判定する必要がないので、一応書いています。
  if(!isMoved){
    if(x != 0 || y != 0){
      isMoved = true;
    }
  }
  if(isMoved) {
    line(x, y, mouseX, mouseY);
  }
  x = mouseX;
  y = mouseY;
}
```

さて、これでマウスの動きに合わせて線を引くコードが書けました。ですが、これでは一筆書きしか描けないので不便です。マウスが押されている間だけ描くことにしましょう。マウスが押されている間は `mouseIsPressed` が`true`になるので、これを使いましょう。

```
// let x, y 省略
let isMoved = false;

// setup() 省略

function draw() {
  // if(!isMoved) 省略
  if(isMoved) {
    if(mouseIsPressed){
      line(x, y, mouseX, mouseY);
    }
  }
  x = mouseX;
  y = mouseY;
}
```

もちろん、if()が深くならないように、2行のifを `if(isMoved && mouseIsPressed)` とまとめることもできます。

## イベントの利用

描いた画像を保存できるようにしましょう。

画像の保存自体は `saveCanvas()` という関数が用意されているのでできるのですが、ユーザーの操作に応じて保存しなければいけません。このような時に使えるのが、イベントという考え方です。今回はキーボードのSキーを押したときに保存するようにします。

まず、`setup()`や`draw()`と同様に`keyPressed()`をつくりましょう。`setup`は開始時に、`draw`は一定間隔で呼び出されました。が、`keypressed`は何らかのキーが押されたときに呼び出されます。

`keyPressed()`の中では、特別な変数「key」が使えます。keyには押されたキーが入っているので、これを使えば目的のキーが押されたのか判別することができます。

これまでに説明したことを使えば、こんなコードが書けるのではないかでしょうか。

```
// let 省略  
// setup(), draw() 省略  
  
function keyPressed() {  
  if(key == "s") {  
    saveCanvas('test', 'png');  
  }  
}
```

## 完成

さて、なんとなく絵が描けるプログラムができました。全体のコードはこんな感じです。

```
let x = 0;  
let y = 0;  
let isMoved = false;  
  
function setup() {  
  createCanvas(windowWidth, windowHeight);  
  background(255);  
}  
  
function draw() {  
  if(!isMoved){  
    if(x != 0 || y != 0){  
      isMoved = true;  
    }  
  }  
  if(isMoved && mouseIsPressed) {  
    line(x, y, mouseX, mouseY);  
  }  
  x = mouseX;  
  y = mouseY;
```

```
}
```

```
function keyPressed() {
    if(key == "s") {
        saveCanvas();
    }
}
```

## あとがき

分量にしては中身がないですね…すみません。今回作ったお絵かきアプリは全く実用性がないものでしたが、ドキュメントを睨み、関数を書き、変数を定義するといった行為は、皆さんが日々使っているアプリやゲーム、さらにはiOS、Android、Windows、macOSといったものの開発者もやっていることなんです。そう思うとすごいですね。

もしこの記事を読んでプログラミングに興味が出てきたなら、ぜひ勉強してみてください。プログラミングの世界は広く、さまざまな分野があります。例えばUnityなどのゲーム開発ならこの記事とよく似た構造になっていますし、もっと見た目にこだわりたいと思ったならWebに手を出してもいいかもしれません。他にも、論理が好きな方なら競技プログラミング、友達とゲームで遊びたいならModやサーバー構築など、業界が広いが故に多くの入り口があるというのは、プログラミングの長所の一つなんじゃないかと思います。

一度手を出してしまえば癖になる（個人差あり）プログラミング、ぜひやってみませんか？！

またまたこんにちは、ZOIです。 星を一つ一つ描画してめちゃくちゃリアルな星空を作つてみたので、その話をします。

## 星の再現

さて、星空を作るということで、まず星が必要です。 星空をカメラで撮ってきていっぱい貼ればいいんですが、それだと面白くないですよね～。(あと綺麗に撮れない) というわけでせっかくなので、現実の星のデータを使って星を表示してみようと思います! 星のデータを Wikipedia かどこかで一つずつ調べていくと死んでしまうので、ESA(ヨーロッパ宇宙機関)がまとめてくれた便利なデータを使います。 今回使うのは Hipparcos、Gaia という2つの衛星で観測されたものになります。 それぞれHipparcos星表、Gaia星表と呼ぶことになると、Hipparcos星表は主に明るい星(12等星以上)、Gaia星表は暗い星(20等星以上)の情報を含んでいます。 このデータを使ってどのように星を表示するか考えましょう。もちろん星を一つ一つ3Dデータとして表示するのもいいのですが、さっき紹介したHipparcos星表は約12万、Gaia星表は約18億の星のデータがあります。こんなものを3Dデータとして使ってしまふと、たとえ星を1ポリゴンで表せたとしても、18億ポリゴンになってしまいます。これをOBJ形式(3Dモデル)で表した場合、少なめに(1ポリゴン=50バイトで)見積もっても90GiBになります。現実的じゃないです。そこで、画像として星空を用意しておいて、それを球に貼り付けて表示することにしましょう。

## 画像の作成

というわけで、Gaia星表とHipparcos星表から星空を描くプログラムを作ります。しかし、一回自分で作つてみて一応完成したのですが、出力するとどう見てもミスつてました。天文の知識がなさすぎて、一体どこが間違っているのかよくわからん。それだけで5000文字くらい書いてたけど、治すのはめんどくさそう難しそうだったので、とりあえず自分で作るのは諦めました。

そこで「Gaia・Hipparcos星表を使って星空を描くアプリないかな～」とか思いながらWebの海を彷徨っていたところ、なんとありました!(なんであるの???) こちらのプロジェクトです。(GitHubってのは「エンジニア版のPixiv」って言われてたりします。宝の山。)

<https://github.com/misohena/drawstars>

しかし、このコードを動かすだけなら簡単すぎるので、ちょっと手を加えてみます。 実は、このプロジェクトが作られた後に、「Gaia EDR3」というGaia星表の新バージョンが作られてるんです。僕としてはEDR3の方で描きたかったので、EDR3への対応をしていきます。 GitHubってのは一種のSNSなのですが、「他人が作ったプロジェクトに勝手に派生作品を作る」っていう謎機能があります。(Fork。Twitterのリツイートに近い?) この機能をありがたく使わせてもらいます。 派生先リンクはこれです。

[https://github.com/ZOI-dayo/drawstars\\_GaiaDR3](https://github.com/ZOI-dayo/drawstars_GaiaDR3)

ただ、元のコードがしっかりしてるので、リンクの変更以外あんまり変更するところはなかったです。強いて言えば、元々使ってあるGaia DR2というバージョンでは星の表面温度が取得できるのですが、Gaia EDR3にはそれがなさそうです。(なんで消した...?) なので、それっぽい論文(<https://arxiv.org/pdf/1008.0815.pdf>)にある式 $\log(T_{\text{eff}}) = 3.999 - 0.654(C_{\text{XP}}) + 0.709(C_{\text{XP}})^2 - 0.316(C_{\text{XP}})^3$ を使って捻り出しました。誤差はあるようですが、星の色が目に見えて変わるほどではないはず...

あの使い方はmisohenaさんのプロジェクトと全く同じです。なんやかんやしてできた画像がこちら。

どう見ても星空ですね。天の川がぐにゃぐにゃなのは正距円筒図法という書き方だからです。詳しくはGoogleさんに聞きましょう。

なお星空画像が欲しいという方は、謎コードを書いたり600GBのダウンロードをしたりするような変な真似はせずに、おとなしく下のサイトからダウンロードしましょう。<http://t.nomoto.org/AllSkyHipp2GAIA/>

## まとめ

さていい感じに画像ができたわけですが、一体これ何に使うの? となったと思います。ふと思いつきで作っただけなのでどこにも使う場所がない...かと思いきや、VRジェットコースターの背景にいい感じに設置することに成功しました。詳しくはVRジェットコースターの記事で。

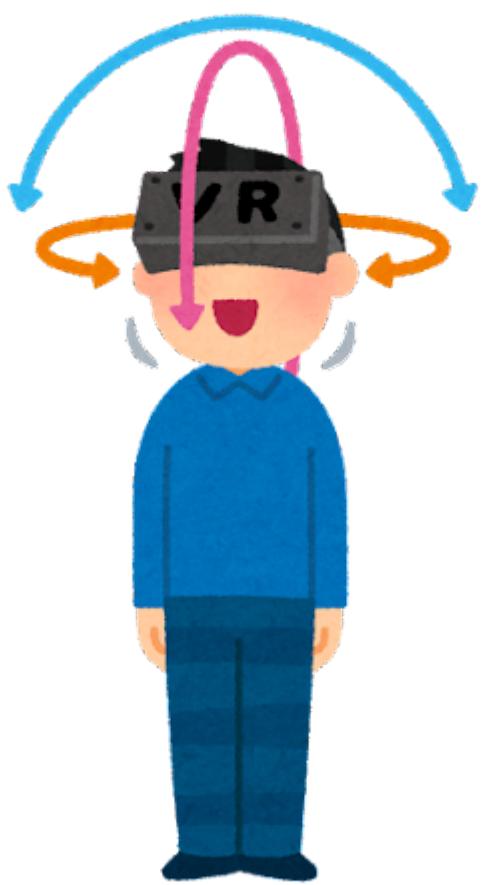
# VRジェットコースター製作の裏話

こんにちは、ZOIです。 みなさんは今回の音展で、VRジェットコースターには乗りましたか？ まだ乗っていない人は、ぜひ乗ってみてください。 たった1分少しの作品ですが、非常に多くの技術が使われているので、読む前と後では全く違って見える…かもしれません！

ただ、開発のほぼほぼ全行程を書いてるので、長い上に技術的な話しかしていません。 これは本当に申し訳ないです… それでも良いという方はぜひご覧ください。

## 開発開始

VRジェットコースターということで、まずはVRの話をします。 VRといっても大きく分けて2種類あります。 専用のホルダーにスマホをはめて使うものと、VRゴーグルを使うものです。 それぞれ基本的には「3DoF」「6DoF」と呼ばれるトラッキングを採用していて、6DoFは3DoFに加えて縦横への移動にも反応してくれます。

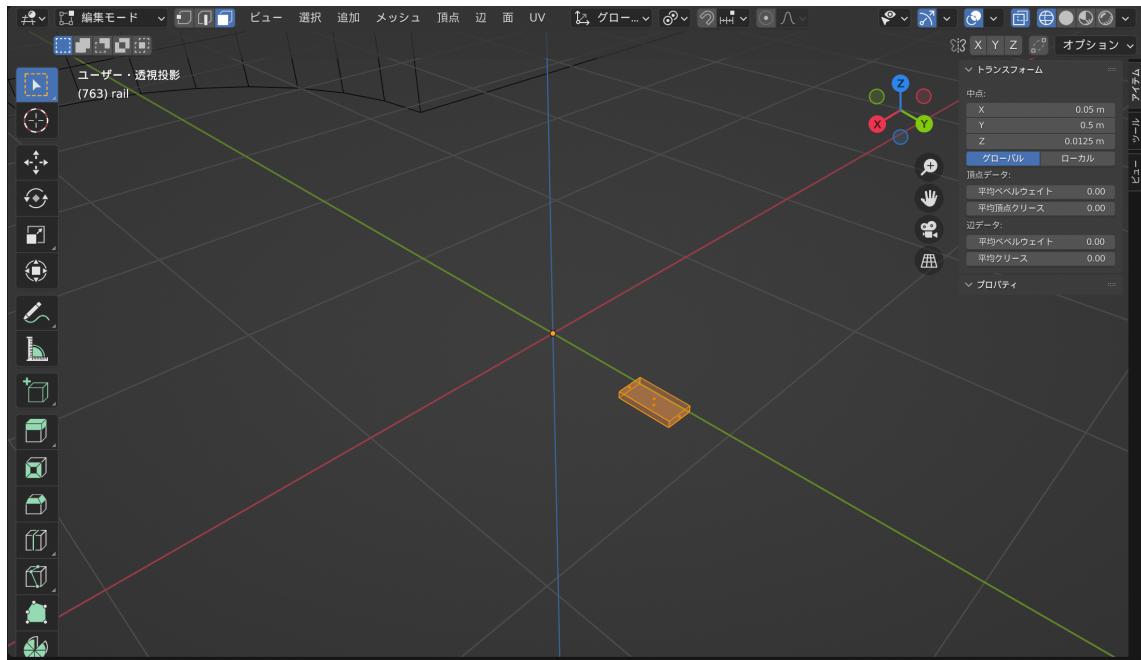


ジェットコースターではレールの下を覗きこむなどの動作もしたいので、今回は6DoFを使うことにしました。実際に触ってみると分かりますが、3DoFと6DoFでは没入感が全く違います。専用のVRゴーグルというのいろいろな種類があるので、「追加で高性能なPCが要らない」「そこまで高くない」という条件から、Meta Quest 2を使うことにしました。これはVRの入門機のような扱いをされているもので、一番売れているゴーグルです。なによりコスパがいいです。あと、中身がAndroidベースなのでアプリを作りやすいです。このあたりは後で詳しく話します。

## 3Dモデル作成

ジェットコースターを作るので、走らせるコースを作る必要があります。去年、PC同好会で「甲陽ストリートビュー」という作品を作ったのですが、そのプロジェクトの一環として校舎のモデルを作ったことがあります。せっかくなのでこれをベースにしましょう。3Dモデルの作成はBlenderで行います。無料なのにプロ向けの機能が使えるすごいやつです。VRChatなどのアバター制作によく使われていますね。

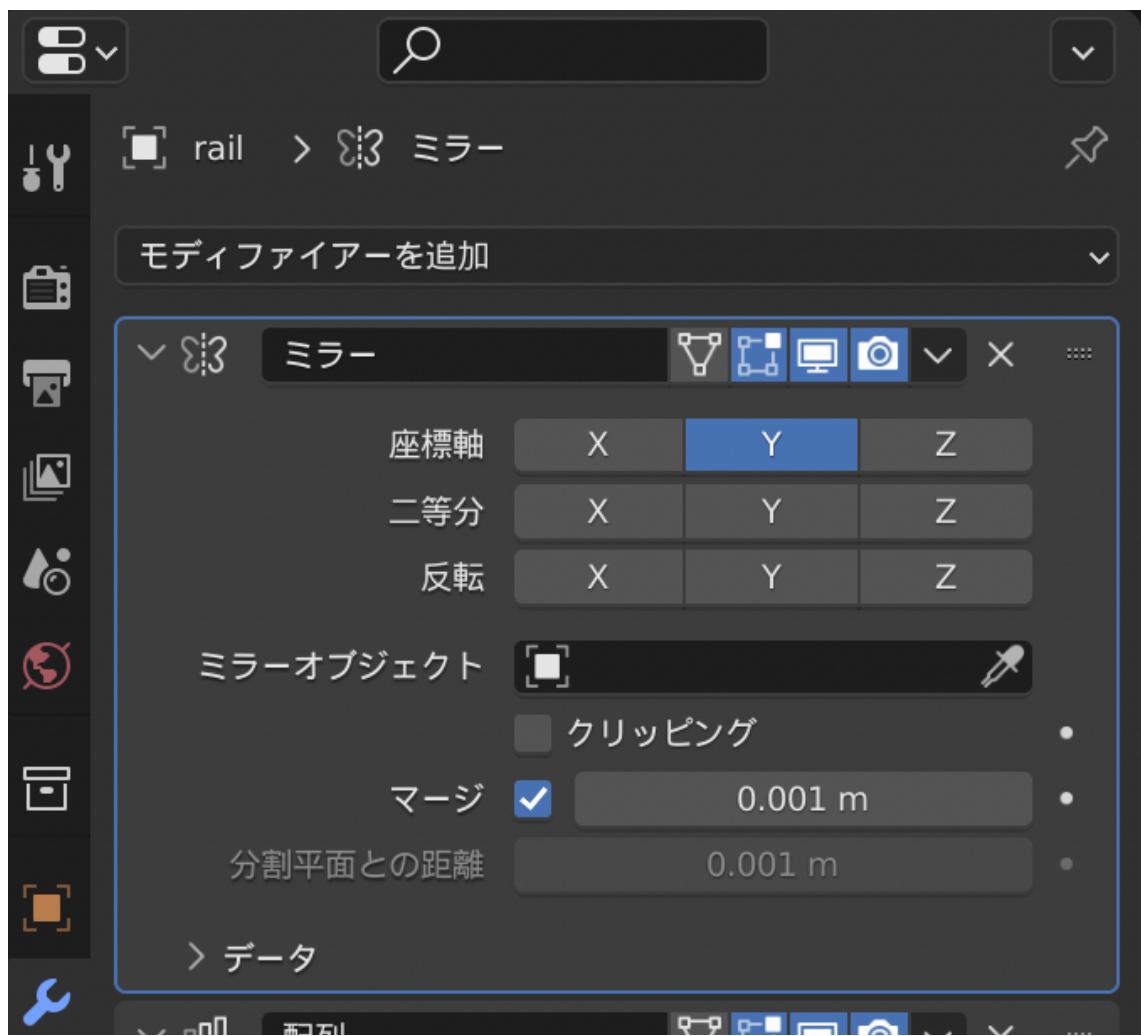
校舎のモデルがあるので、あと作らなければいけないのはレールと車体です。車体は立方体を窪ませてなんとかするとして、レールは平行な棒を2本用意する必要があります。これを手作業でやろうとすると大変なので、Blenderのモディファイアやカーブ、コンストRAINTなどを駆使して楽をします。今回、レールはレーザーのようなイメージにしたいので、薄い長方形のような断面にします。実は、レールのために作らなければいけないモデルはこれだけです。



原点と少しずれた場所にあるのがポイントです。次に、レールを配置したい形にカーブを引きます。



最後に、レールのオブジェクトに対してミラー、配列、カーブのモディファイアを設定します。



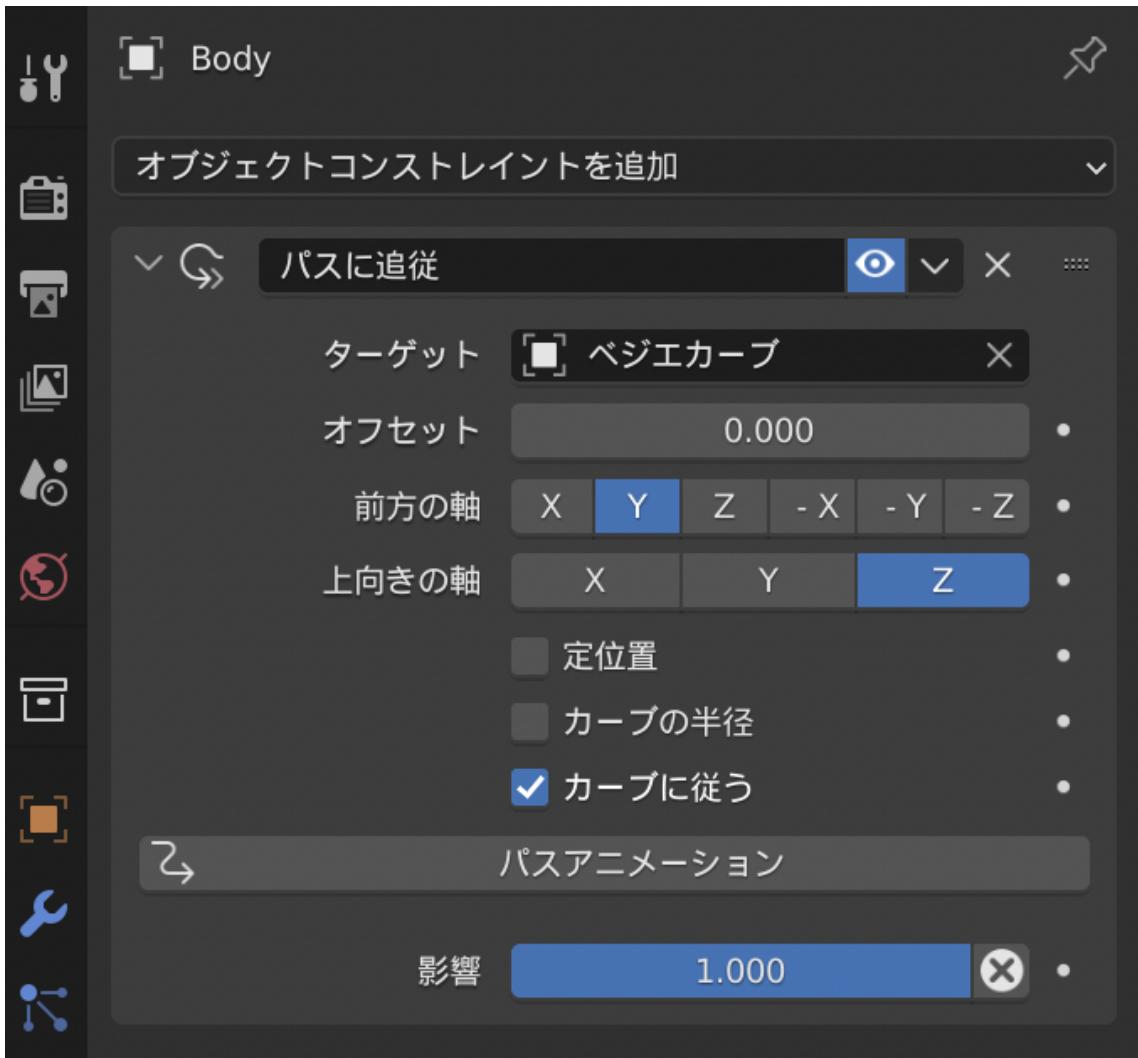


ミラーでレールを左右に配置、配列で直線状に伸ばして、カーブでカーブにそって曲げるといったかんじです。レールが変に捻じれると思うので、カーブの各頂点の「傾き」プロパティをいじって調整してください。

## レール上を滑らせる

最終的にVRゴーグルに入れる際にはUnityというアプリ(後述)を使うので、Unity内蔵の物理エンジンを使えば良いと思っていたのですが、実際にやってみるとガタガタしてうまく使えませんでした。そこで、Blender上でアニメーションを作り、Unityにインポートするという方法をとることにしました。Blenderにはさっく作ったレールのカーブがあるので、これに沿って車体を移動させることで、滑らかに滑らせることができます。車体のモデルに対

してパスに追従のコンストRAINTを設定することで、カーブに沿って動くようになります。



ただし、物理演算に基づいた動きをするわけではないので、手動でそれっぽく見せる必要があります。

## アプリ化

さっき説明を省略したUnityについて話します。Unityとは、「ゲーム開発環境」と呼ばれるアプリの一つです。その名の通りゲームが作れるアプリで、有名どころでは「ポケモンGO」「原神」「ウマ娘」などはこれを使って作られています。原神などをやったことある人ならわかると思うんですが、あれってPCとスマホで同じゲームが動いてますよね。あれって実はすごいことで、本当ならWindows用、mac用、iPhone用、Android用...とほぼ1から書き直さないといけないところを、1つ書くだけで全部作れてしまうという化け物です。まあ今回はQuest2、つまりはAndroid用のアプリしか作らないんですけど。

では、作ったモデルをUnityにインポートです。といっても最近のUnityは.blendファイルの読み込みに対応しているようなので、Unity上でファイルを選択し、プロパティのRigのアニメーションタイプをジェネリックに、Animationのインポートをいい感じにチェックつけたらアニメーションがインポートされます。

シーンにPrefabとして配置したら、Animatorコンポーネントを追加、そして適当にアニメーションコントローラを作つて当ててください。中に入れるアニメーション自体は.blendファイルの中に含まれている感じになっていると思うので、頑張って探しましょう。僕の場合一番最後にありました。

最後に、Unityのカメラを追従させましょう。これは簡単で、シーン中のカートのモデルをなんとかして見つけ出し、その子としてカメラを作るだけです。

できたら再生してみましょう。 ジェットコースターっぽくなりました。

## VR化

さあこのままではタイトル詐欺です。「VR」ジェットコースター正在運営しているからにはVRで動くようにしましょう。 今回はMeta Quest2のスタンドアローン(PCに繋がない)での動作を想定しているので、Android向けにビルドします。 Android向けのビルト環境ができていない人は、Unity Hubを開いて、使っているUnityにAndroid系のモジュールを加えてください。 そしてQuest2を準備しましょう。

(ちなみにQuest2を買おうとしてた時に突如2万円値上げしてめっちゃ困りました。他の予算を削ってなんとかなりましたが。)

Quest2を開発者モードにします。 Quest2のセットアップに使ったスマホを使い、Oculusアプリの設定→その他の設定から開発者モードをONにできます。 開発者としてグループを作れみたいなことを言われるかもしれません。 指示に従っておけば大丈夫です。 その後パソコンとケーブルで繋ぐと、Quest2の画面に「許可しますか?」的な質問が表示されるので「はい」を押して許可してください。 これで、Quest2をパソコンから触れるようになりました。

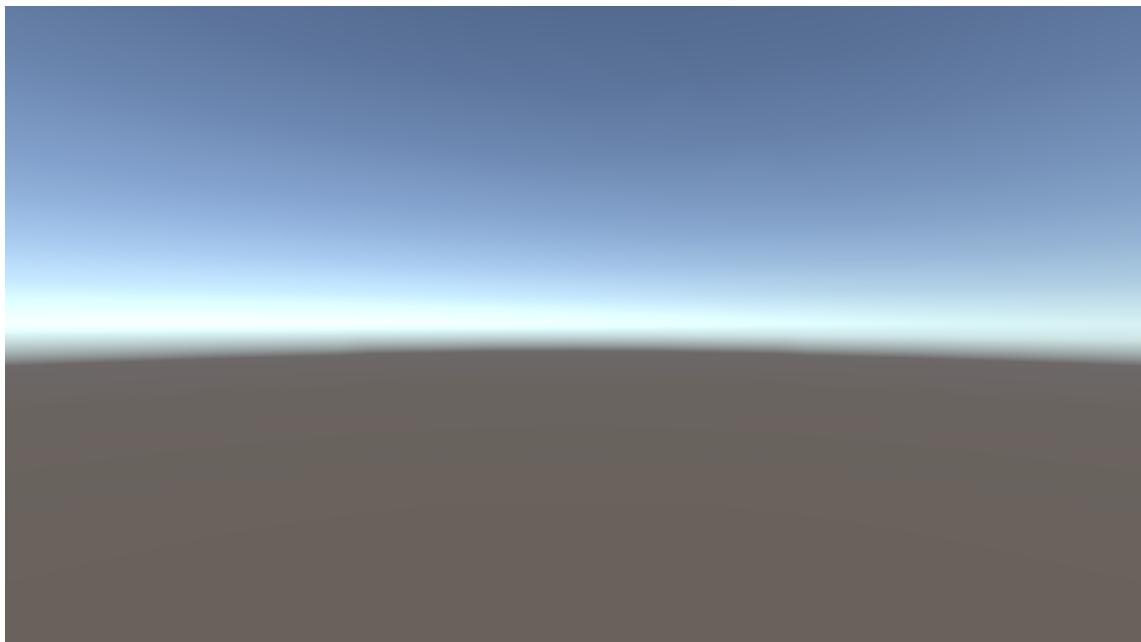
では、Unityのビルト設定をします。 プラットフォームをAndroidに切り替え、プレイヤー設定を開き「企業名」「プロダクト名」「バージョン」を設定です。 ここまでは普通のAndroidアプリと同じですね。 ただ追加で、色空間をリニアに、テクスチャ圧縮をASTCにする必要があります。

ここからVR対応にしていきます。 まずはプロジェクト設定の中のXR Plugin Managementをインストールします。 インストールが終わったらプラグインプロバイダーをOculusにしてください。

この状態で実行してみたところ、Quest2に映りはするけれども顔の角度が考慮されていないような挙動をしたので、VR用にカメラの設定をします。 Asset Storeから「Oculus Integration」をインストールします。 結構時間がかかります。 終わったら、Assets/Oculus/VR/PrefabsにあるOVRCameraRigを、今カメラを入れている場所に配置します。 カートの子になりますね。 これで実行するとちゃんとしたVRになります。

## 天球を作りたい

Unityに最初から搭載されている空はちょっと安っぽいのでここから変えていきましょう。

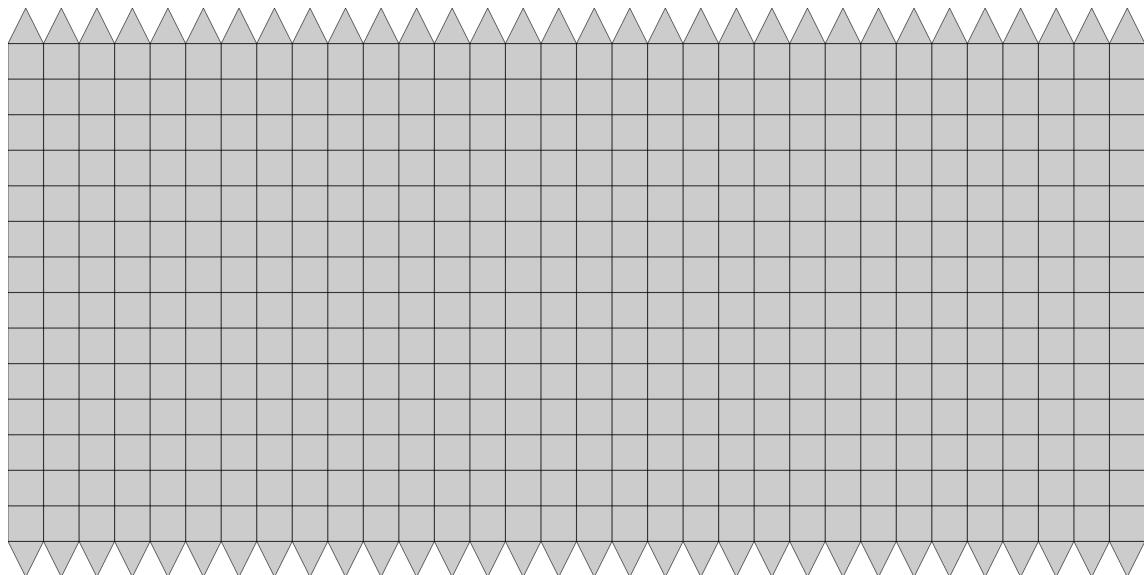


絵を描く才能などはないので、できるだけ現実をパクリます。ただ空を作るとなると面倒だな...と思っていたのですが、そういえばいいのがありました。数ヶ月前に作って放置されていた16Kの星空全天球画像です。詳しくは僕の書いた「めちゃくちゃリアルな星空を作った話」がこの部誌のどこかにあると思うのでそれを見てください。

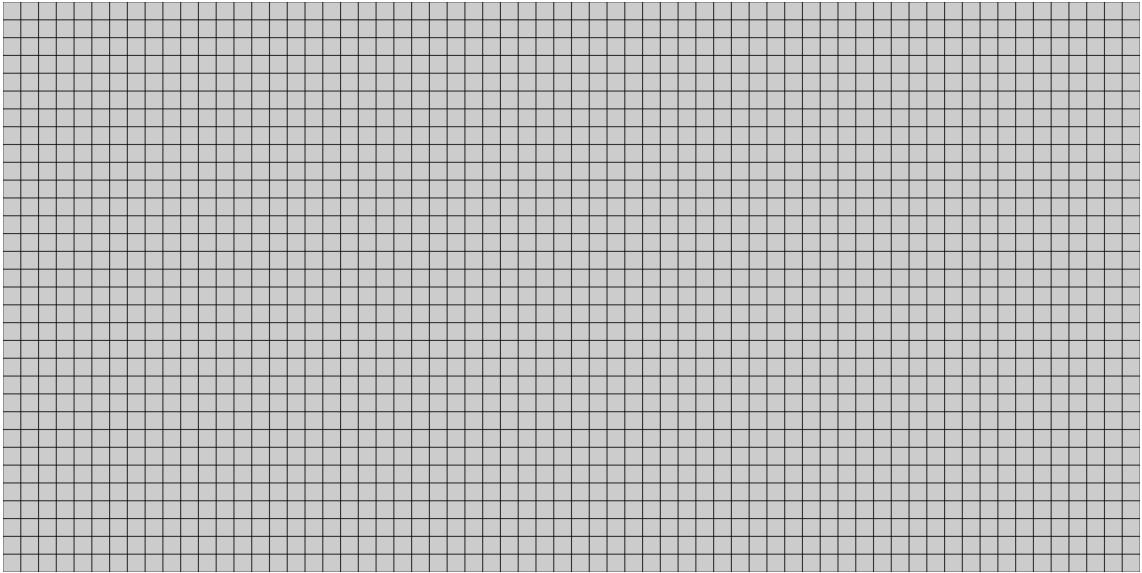
あの画像は天球の緯度、経度を縦横にとったような構造をしていて3Dとは相性がいいです。まずはこれを貼る球を作りましょう。

## 球の作成

球なんて元からあるじゃないかと思われるかもしれません、この場合普通の球は使えません。Blenderで「UV球」として使えるような普通の球は、北極や南極にあたる位置の点がつながっているため、極の隣の面が三角形になっています。このためUV展開(テクスチャのどこを3Dモデルのどこに割り当てるかの設定)したときに使わない領域ができてしまいます。



これを修正した球が必要です。とはいっても作るのは簡単です。Blenderを起動し球を生成、両端の頂点を削除、穴の周囲をEキーで伸ばして、Sキーで0倍に拡大、あとはサイドバーで頂点(たち)の座標を(0,0,±1)にすれば構造は終わり。追加した頂点の分のUVを下のように設定すれば完成です。ついでにメッシュ数も増やしておきました。



## シェーダー

Unity上にこれを置いてテクスチャを貼ると、「外から見た時に星空が見える球」ができるようになります。これは困るので、`Sky.shader`という(名前は任意)ファイルを作り、中に次のようなシェーダーを書きます。

```
Shader "Unlit/Sky"
{
    Properties
    {
        _MainTex ("Texture", 2D) = "white" {}
    }
    SubShader
    {
        Tags { "RenderType"="Opaque" }
        Cull Front // add
        LOD 100
        Pass
        {
            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            #pragma multi_compile_fog
            #include "UnityCG.cginc"
            struct appdata
            {
                float4 vertex : POSITION;
                float2 uv : TEXCOORD0;
            };

```

```

    struct v2f
    {
        float2 uv : TEXCOORD0;
        UNITY_FOG_COORDS(1)
        float4 vertex : SV_POSITION;
    };
    sampler2D _MainTex;
    float4 _MainTex_ST;
    v2f vert (appdata v)
    {
        v.uv.x = 1-v.uv.x;
        v2f o;
        o.vertex = UnityObjectToClipPos(v.vertex);
        o.uv = TRANSFORM_TEX(v.uv, _MainTex);
        UNITY_TRANSFER_FOG(o,o.vertex);
        return o;
    }
    fixed4 frag (v2f i) : SV_Target
    {
        fixed4 col = tex2D(_MainTex, i.uv);
        UNITY_APPLY_FOG(i.fogCoord, col);
        return col;
    }
    ENDCG
}
}

```

これで内側から見ることができるようになりました。

しかし、実際の星空とよく見比べてみると、向きがおかしいです。 実際は、日本から見ると天球は斜めになって回転しています。

まず初めに考えつく方法は、球を斜めに配置するというものです。 これが結構優秀で、後で回転させる時にも楽なのですが、今回は途中で「昼の空のテクスチャは回転しないから夜の部分だけ回転させたい」とよくわからないことを考えたので全部シェーダーで書きます。 これに回転機能などをつけた完成品が以下です。

```

Shader "Unlit/Sky"
{
    Properties
    {
        _MainTex ("DayShader", 2D) = "blue" {}
        _NightTex ("StarShader", 2D) = "white" {}
        _Blend("Blend", Range (0, 1)) = 1
        _Round("Round", Range (0, 1)) = 0
    }
    SubShader
    {
        Tags
        {
            "RenderType"="Opaque"
        }
    }
}

```

```

Cull Front
LOD 100
Pass
{
    CGPROGRAM
#pragma vertex vert
#pragma fragment frag
#pragma multi_compile_fog
#include "UnityCG.cginc"
struct appdata
{
    float4 vertex : POSITION;
    float2 uv : TEXCOORD0;
};
struct v2f
{
    float2 uv : TEXCOORD0;
UNITY_FOG_COORDS(1)
    float4 vertex : SV_POSITION;
    float3 viewDir : TEXCOORD1;
};
sampler2D _MainTex;
sampler2D _NightTex;
float _Blend;
float4 _MainTex_ST;
float _Round;
v2f vert(appdata v)
{
    v.uv.x = 1 - v.uv.x; // add
    v2f o;
    o.vertex = UnityObjectToClipPos(v.vertex);
    o.uv = TRANSFORM_TEX(v.uv, _MainTex);
    o.viewDir = -normalize(_WorldSpaceCameraPos -
mul(unity_ObjectToWorld, v.vertex));
    UNITY_TRANSFER_FOG(o, o.vertex);
    return o;
}
float4 quaternion(float rad, float3 axis)
{
    return float4(normalize(axis) * sin(rad * 0.5), cos(rad * 0.5));
}
float3 rotateQuaternion(float rad, float3 axis, float3 pos)
{
    float4 q = quaternion(rad, axis);
    return (q.w * q.w - dot(q.xyz, q.xyz)) * pos + 2.0 * q.xyz *
dot(q.xyz, pos) + 2 * q.w * cross(
    q.xyz, pos);
}
fixed4 frag(v2f i) : SV_Target
{
    const float PI = 3.14159;
    const float LATITUDE = 34.76; // Koyo
}

```

```

        fixed4 day = tex2D(_MainTex, i.uv);
        float3 rotated = rotateQuaternion((90 - LATITUDE) / 180 * PI,
float3(1, 0, 0), i.viewDir);
        float2 starUV = float2((atan2(rotated.x, rotated.z) + PI) / (2 *
PI),
-atan2(sqrt(pow(rotated.x, 2) +
pow(rotated.z, 2)), rotated.y) / PI + 1);
        starUV.x -= _Round;
        fixed4 night = tex2D(_NightTex, starUV);
        fixed4 col = day * (1 - _Blend) + night * _Blend;
        UNITY_APPLY_FOG(i.fogCoord, col);
        return col;
    }
ENDCG
}
}
}

```

うーん、長いし読みにくい。 視線ベクトルをクォータニオンで回転させているのがポイントです。

// ログにとっておいた動画を参考にしながら書く

## ハードウェア

さて、ここで工作に移ります。 今回VRの没入感を高めるために椅子を動かそうと考えていて、その回路を作らないといけないのです。 まずは大雑把な仕組みを決めます。

人を乗せた椅子を動かすのはパワーがいるので、まずここを決めてしまいます。 今回は、アクチュエータという部品を使うことにしました。 これはモーターにギアなどをはめて、回転運動から伸び縮みの運動に変換したものです。 amazonでちょっと探したところ、900Nくらいの力で押すものがあったのでそれにしました。

次に、VRアプリとの通信です。 VRと同期させて動きをつける関係上、どうしても通信は必須です。 ただ、VRからUSBなどを伸ばすのも邪魔なので、無線でモーターの制御を行いたいです。 これを実現してくれるのが「ESP32」というマイコンボードです。 これは、消費電力が低く小さいのにBluetoothとWifiを扱えるというすごいやつです。 性能は低いのでパソコンとしては使えません。 モーター制御くらいなら十分でしょう。

ここで問題が発生します。 今回買ったアクチュエータは12V・3A(Max)というすごい電気を流します。 この経路にマイコンという貧弱な部品が挟まると、多分焼けてしまうでしょう。 そこで、マイコンからの3.3V・12mA程度の弱い電流でどうやって大電流をコントロールするかが課題となります。 このような用途ではリレーかトランジスタが使われることがほとんどです。 リレーは、「弱い電気で電磁石を動かし、物理的にスイッチを作動させる」パート、トランジスタは「2端子間に電圧をかけることで他の端子間の抵抗が低くなる」という特殊な半導体を使ったものです。

リレーでモーターを回路を自作しようとしたのですが、回路設計がガバガバでケーブルから煙が出たところで心が折れました… 急いで大阪の日本橋に走り、モータードライバIC、TB6643KQを買ってきました。 これは、モーターを正負両向きで回すことに対応したICチップというメチャクチャ便利なものです。 (始めからこれ使っていたらよかった…)

ほかに困ったのは、USBと12Vの2電源の管理です。 ESP32にはUSBから5Vを、モーターにはコンセントから12Vを供給しようとしていたのですが、どうやらそれぞれの基準電圧が別のところにあるらしく、思った電圧がかかりませんでした。 問題は電源が2系統あることなので、これを1つにまとめられないか考えてみます。 5VのUSBから3Aを取り出すのはちょっと心配なので、12Vを5Vに降圧します。 これには3端子レギュレータという部品が必要なのですが、もちろんそんなもの持ってません。 ただ、よく考えてみると、ESP32は内部では3.3Vで動いていま

す。ここにUSBの5Vをかけているわけで、この降圧を使えないか?と思って基板をみたところ、あたりです。3端子レギュレータLD1117が乗っていて、これを調べてみると15Vまで入力可能らしい。というわけで流してみると普通に動きました。これで、USB電源は無かったことになり、いい感じに動くようになりました。

## マイコン制御

ESP32に、モーターをコントロールするコードを書き込んでいきます。有名なマイコンボードとしてArduinoの名前を聞いたことがあることのある人もいるかもしれません、ESPはArduinoの仲間ではないので、Arduino用に作られたマイコン書き込みツールを使うことができません。しかし、誰かが「ESPに不足している機能を足してArduinoとして書き込むことができるアプリ」を作ってくれているので、これを使います。

まずは、Arduino公式サイト(<https://www.arduino.cc/>)にアクセスし、Arduino IDEをインストールします。その後、環境設定を開き、追加のボードマネージャURLに

`https://dl.espressif.com/dl/package_esp32_index.json`を追加します。最後に、ボードマネージャを開くとESP32が追加されているのでこれをダウンロードします。これで準備が出来ました。

次のようなコードを書き込みます。

```
#include <WiFi.h>

const char ssid[] = "ESP32AP-WiFi";
const char pass[] = "esp32apwifi";
const IPAddress ip(192, 168, 30, 3);
const IPAddress subnet(255, 255, 255, 0);

const char html[] =
"<!DOCTYPE html><html lang='ja'><head><meta charset='utf-8'></head><body>\n<a href='/?true'>正転</a><br /><a href='/?false'>逆転</a><br /><a href='/?zero'>停止</a></body></html>";

WiFiServer server(80);

void setup()
{
    Serial.begin(115200);

    WiFi.softAP(ssid, pass);
    delay(100);
    WiFi.softAPConfig(ip, ip, subnet);

    IPAddress myIP = WiFi.softAPIP();

    server.begin();

    Serial.print("SSID: ");
    Serial.println(ssid);
    Serial.print("AP IP address: ");
    Serial.println(myIP);
    Serial.println("Server start!");

    pinMode(2, OUTPUT);
```

```

pinMode(0, OUTPUT);
pinMode(12, OUTPUT);
pinMode(14, OUTPUT);

}

void loop() {
    WiFiClient client = server.available();

    if (client) {
        String currentLine = "";
        Serial.println("New Client.");

        while (client.connected()) {
            if (client.available()) {
                char c = client.read();
                Serial.write(c);
                if (c == '\n') {
                    if (currentLine.length() == 0) {
                        client.println("HTTP/1.1 200 OK");
                        client.println("Content-type:text/html");
                        client.println();

                        client.print(html);
                        client.println();
                        break;
                    } else {
                        currentLine = "";
                    }
                } else if (c != '\r') {
                    currentLine += c;
                }
            }

            if (currentLine.endsWith("GET /?true")) {
                digitalWrite(0, LOW);
                digitalWrite(2, HIGH);
                digitalWrite(12, HIGH);
                digitalWrite(14, LOW);
            }
            if (currentLine.endsWith("GET /?false")) {
                digitalWrite(2, LOW);
                digitalWrite(0, HIGH);
                digitalWrite(12, LOW);
                digitalWrite(14, HIGH);
            }
            if (currentLine.endsWith("GET /?zero")) {
                digitalWrite(0, LOW);
                digitalWrite(2, LOW);
                digitalWrite(12, LOW);
                digitalWrite(14, LOW);
            }
        }
    }
}

```

```
        }
        client.stop();
        Serial.println("Client Disconnected.");
    }
}
```

はい。いつも通りクソコード臭がしますね…やってることは簡単で、ESP32をルーターとして起動し、アクセスしたURLに応じてIO出力を変えているだけです。これをモータードライバにつければうまいこと動きます。

…動くはずでした。

## ★死亡★

ここからUnityに対応させようと、Unity側のコードにHTTPリクエストのコードを書いて、急な回転方向の変化は反応してくれないなーとか思っていたんですが、終わりは思いもよらぬ方向からやってきました。

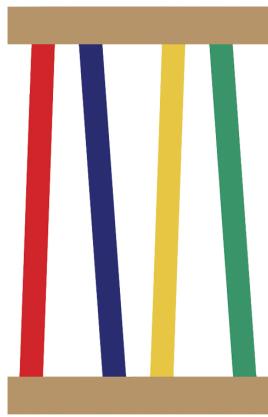
時は音展前日、指導部による最終チェックが入る1時間前… UnityのHTTP送信タイミングを開始からの秒数で指定していて、そこを調整していた時だったんですが、電源を繋げた瞬間 **ESP32が火を吹きました**。な…何を言っているのか わからねーと思うが おれも 何をされたのか わからなかった…

まあ原因はある程度目処がついています。しばらく(1Hくらい)電源を外していくて付けた瞬間に出火したというタイミングと、出火場所が先ほど話した3端子レギュレータだったことから、きっと電圧が一瞬15Vをオーバーしてしまったのでしょう。というのも、使っていたACアダプターがAmazonで最も安い、中華の極みのような製品だったのです。元々コンセントに入れるたびに火花を散らすすごいやつだったんですが、まさか出力が定格オーバーするとは…

どこかがショートしたのか、個体差なのかわからなかつたためもう一方にも刺して見たところ全く同じように炎上したので、この時点で椅子を動かす野望は潰えました。あと1時間って時に部品燃やすのはもはや芸術の域ですね。炎上部品は教室前に展示することとします。

## あとがき

うーーん、あまりにもひどい終わり方です。本当に申し訳ありません…! 一応「VR」「ジェットコースター」の2本柱が折れなかつたので企画としては成立できるのが不幸中の幸いですね。映像だけでもかなり酔うそうなので、まだ遊んでいない人はぜひ!



KOYO PC CLUB