# 目次

# 身体運動の創造的学びの一人称研究

堀内隆仁

2023 年 12 月 11 日

# 第Ⅰ部

# 身体運動の創造的学びの物語

# 第 1 章

# 背景：学びを描くということ

## 1.1 Gendlin の experiencing, focusing

## 1.2 いいい

堀内 [?] は、廣松渉 [?] は尼ヶ崎 [?] は [?] しかしここで、「能動性」の観点から忘れてはならないのが、「気持つ」という心の作用である。注意（志向）や気分や情や欲といったものが、知覚と行為のカップリングに、「作用」してくるのだ。よく知られたものに、「カクテルパーティー効果」がある。ほかにもダックアンドラビットやといったゲシュタルト心理学の知見の蓄積を参照すれば、「気持つ」が知覚に「作用」しうることは明らかだ[*1]。この能動性は、能動性とはいえど、「表象」という存在の前には、**ボトムアップ能動性**とも呼びうるものだ。さしずめ表象は**トップダウンな能動性**と呼んでもよいだろう。つまり、知覚はトップダウンとボトムアップ両方向の能動性によって成り立つものだと主張したい。知覚・行為・思考からなる渦、とはこういうことである。

[*1] 「A が B に作用する」は、「A と B がカップリングにある」と微妙に異なるありようとして書き分けている。知覚と行為はともに、物的世界と心的世界の境界に生起する記号現象だが、気持つは心的世界での面が強い（もちろん厳密には、気持つというのも、興奮して汗が出るとか怖くて鳥肌が立つなど、両世界の境界に生起するものではある。）

# 第 2 章

# 学びの物語

## 2.1　読み方と書き方について

本章では、著者自身をさす一人称を、「私」と称する。

随所で、その知見に関係するであろう、経験科学的知見を引用する。科学に「帰着」する、科学のほうが偉いという考え方では決してないが、多面的・深い理解を促すためには、既存の科学的知見との関連づけることは重要であろう。

# 第 3 章

# 走りはどう変わったのか？連続写真からの考察

## 3.1　連続写真からの考察を行う。

本章では、著者自身をさす一人称を、「私」と称する。

随所で、その知見に関係するであろう、経験科学的知見を引用する。科学に「帰着」する、科学のほうが偉いという考え方では決してないが、多面的・深い理解を促すためには、既存の科学的知見との関連づけることは重要であろう。

# 第 4 章

# 物語の考察

## 4.1　学びの野生化

レヴィ=ストロース

# 第 5 章

# まとめ

## 5.1　物語の意義

レヴィ=ストロース

# 第 II 部

# 身体の「表情」感得を促すアプリの制作と、その実践

# 第 1 章

# 背景：身体の醸しだす表情

## 1.1　動く身体を鑑賞するとき、なにをどうみるのがよいのか？

第一部でも表れているように、実践家は学ぶときに、自身の運動を撮影した動画や、他者の運動をみたり、あるいは YouTube で視聴したりする。

繰り返すが、実践家の学びとは、SOMA 的な、一人称視点から「感じる」身体を研ぎ澄ますのである。そのためには、その「動く身体」になにをどうみるのが良いのか？こういう問題意識を、私（著者）は培ってきたのだった。一人称研究として、これをデザインするようになっていた。付録にまわすが、物語のもとになった期間のあと、そーゆープログラミング的なことをはじめるようになったのだ。

動画視聴というときに、もっともありがちなのは、「三人称的な身体」をみてしまうということである。これは、ーーー。であり、スポーツバイオメカニクスの考え方とも直結する。むろん、こうした見方が果たす意義も大きい。前述したようにーーーだからだ。そして、実践家は（私たちは）、そうした態度をとるのがやりやすいというのもまた事実である。

これは主に、「視覚」という知覚の性格と密接に関連しているのだろう。中村 [?] は、視覚という知覚は、ほかの諸知覚から「独走」して、私たちの「明晰的な意識」と結びつきやすいことを指摘している。近代科学的なものごとの捉え方もこの最たる例である。だが、視覚をはじめとした諸知覚はほんらい、体性感覚を中心にして互いに統合されて「共通感覚」として成り立っていると中村は言う。

身体運動も視覚独走型に、明晰的な部分志向となってしまうというわけだ。

共通感覚的にみるということは、視覚だけではなく、みているんだけれども、見ているわたしの身体にも感覚が生じるということである。ちょうどミラーニューロンシステムはこのことを支持する。ここで日常の経験をつぶさに顧みてみると、たしかに、ーーやーーということがある。

そういう見方のほうが、「一人称視点から感じる身体」とはリンクしているように考えられもする。少なくとも、みているわたしの「身体」がじかに関わっている見方ということになる。では、そういう見方とはいったいなんなのか？

どう問題か？では、そこにためには、一体なにをどう見れば良いのか？

### 1.1.1　現象学身体学から

金子らは、潜性的にーーーーということを主張する。’

## 1.2　『表情』

### 1.2.1　表情論導入

この問いによく答えるための重大なヒントが、哲学者・廣松渉 [?] の提唱した「表情」という概念にある。表情とは、「顔の表情」をメタフォリカルに拡張した概念である。一言で問いに答えるなら、「実践家は身体の表情をみるべき」となるのだが、その理解のために、まずは表情について本節でじっくりと説明する。表情とはなんなのか。廣松は説いている。

> 風景に眼を向けて見よう。われわれの日常如実の体験相においては、いま例えば、「いま裏山の松の樹はガッシリとしているが大枝はノタウッテいる。崖にかけて淡竹がスクスクと伸びており、刃先はピンと張っている。・・・小川はサラサラと流れ、魚はスイスイと泳いでいる。雪がヒラヒラと舞い始め、やがてシズシズと降りしきる。松はコンモリと雪帽子を被り、いよいよドッシリと落ち付いて見える。一陣の風がサッと捲き起こり、雪がパッと散る。が、松はカタクナに立っている。竹はタワワに軋み、雀がピョンピョンと枝渡りすると、ドタドタと雪が零れる。夕陽がノンビリと傾き、月影がソッと忍び寄って来る・・・。」
> 環界的情景は、表情性に満ち充ちている。
> — 出典: 廣松渉, 『表情』, 弘文堂, p.9

味わい深い語りであろう。カタカナで表記された部分が表情に相当する。末尾の「環界的情景」とは、「私たちに生きられ・体験された、情景としての環境・世界」のことだと解釈してよいだろう[*1]。廣松は続ける。

> 直接的な体験意識に即するとき、事物（というものが在って、それ）が表情性を帯びている、という表現方式は実態には合わない。右の文章では、松がグネグネしているとか、淡竹がスクスク伸びているとか、事物的分節体が表情性を呈するかのような表現方式になっているが、現実的にはむしろ、グネグネしているあれ、スクスク伸びているこれの覚知が先であって、その覚知与件が松・竹として事物的に認知・命名されるというのが実情であろう。
> — 出典: 廣松渉, 『表情』, 弘文堂, pp.9-10

私たちの生の体験では、（1）「グネグネ」のような表情こそをまず先に感得して、そのあと事後的

[*1] 廣松渉はこのように、「廣松語」とも呼びうるような、さまざまな独自の用語をもちいて論述をしており、その多くは、彼による解説が与えられていない。廣松語が廣松らしさを味わうスパイスになっていることは言うまでもないが、一見してその意味するところがつかみにくい語もある。本論文では、こうした廣松語が登場するたびに解説をほどこすことにする。

に、（２）その表情を「竹」のような事物的なものとして捉える（捉えなおす）のだ、と廣松は説いているのだ。そう、この順番は、私たちの常識と反するのだ。廣松のこの主張は、常識と反するかなりラディカルな主張である。私たちの常識では、（１）「竹」という物がまずあって、（２）私がそれを認識したときに「グネグネ」という解釈を与えているのだと考えがちだからだ。しかし、体験をよくよく振り返ってみれば、そういう体験があるということにも気付かされる。

### 1.2.2　表情の定義のようなもの

[**?**] においては、表情とはなにかという定義然とした記述もいくつか見られる。それらを掲載する。

> 表情感得とは、情緒価と即応価とを内自化せる知覚的現認（p.79）
>
> 「表情感得」こそが、（中略）知・情・意の分化以前的な本源的体験相の原基（p.79）
>
> 知覚相・情緒価・即応価という三契機の各々が、質態値・度量値・趨勢値を内自化せしめた相で現前する（p.79）

表情論で廣松は、さまざまな経験科学的なことにもふれちえる。

以上を踏まえると、さきほどの（１）（２）は単に「逆転」しているのではない。表情は全体性なのである。

表情はある種のパースペクティブである。「一切の現相が悉く表情生を帯びて感得される（p.79）」と主張する。廣松は以下のように書いている。

> 駄目押しするまでもなく、「感得される表情現相」というのは、決して人物や動物の顔面表情や身体的挙措表情には限られない。原基的な相においては、前章で概説した通り、一切の現相が悉（ことごと）く表情性を帯びて[*2] 感得される。—なるほど、現相のうちには、これというほどの感情価やこれというほどの即応価を帯びていないものもある。だが、その場合でも、表情価が端的に零（ゼロ）なのではない。よしんば零としか言いようのない“欠如態”の相にあるとしても、欠如態は（いわゆる“無色透明”が一種の色であるのと類比的に）それ自身、れっきとした価値態であることを忘れてはなるまい。
> (『表情』, p.79)

[*2] 鋭い読者は「X が表情性を帯びる」という表現方式は表情にふさわしくないのでは？とツッコんだかもしれない。補足しておくが、廣松は、「表現の便宜上、以下では事物が表情性を帯びた相で現前するかのように記す方式をも辞せないようにしよう（『表情』, p.10)」と断りをいれている。

### 1.2.3 「惣」が＜立ち現れる＞

「物」が＜ある＞、「心」が＜思う＞。常識的に私たちは、この二段構えの構図において、自身の体験を理解しようとしてしまう。この構図はもはや、体験を理解しようとするときの前提・出発点となっており、ふだん、私たちはそれじたいを疑うことはあまりない。言い換えるなら、私たちはこの構図に「囚われている」。私たちが表情のなんたるかを深く納得するには、この構図から脱出しなければならないと著者は考える。[*3]。

現象学者・大森荘蔵の一連の哲学「立ち現れ一元論」は、上記の構図をまるごとひっくり返そうとする。この新しい構図における「惣」とは、「表情」に相当すると考えて差し支えない（本研究の範疇においては少なくとも）。

ただここで言っているのは、私たちが自身の体験を

### 1.2.4 簡単な事例：色の近く

視覚の原始的な例として、色の知覚を取り上げる。色とは私たちの意識にのぼる「質」である。色そのものは、世界には存在しない。光刺激は、波とみなせ、波長をもつ。人間は、およそ 400〜800nm の光を知覚することができる。「光波長」と「色」の対応があり、およそ、400nm が紫、500nm が緑、800nm が赤として、色のグラデーションとなる。なぜこうなるか。受容器には錐体細胞がある。錐体細胞は、次のグラフのようになる。

つまり、マゼンダ色に対応する単色光は、世界には存在しない。いわば私たちが脳で作り出している色なのだ。このように、色の知覚という原始的な例において、しかも処理の「上流」ですでに、私たちは「世界をそのままコピー」してはいない。世界を構成しているのだ。

知覚相は、認識以前の世界ではない。世界の「姿形」という「質」を、作り出したものである。

### 1.2.5 表情のまとめと図式化

以上をもって、。表情論を端的にまとめる。

## 1.3 身体の「表情」

表情論を踏まえれば、「身体」の表情があることになる。これが、1.1 節で述べたような「生々しさ」に相当するものだと本研究では考える。一人称から感じる身体を研ぎ澄ます実践家は、鑑賞対象となる動く身体の「表情」を感得することが重要である。

動きのなかにある身体は、その観測に先立つ所与の「姿形」などもっていない。そこには、無数の姿形が潜んでいる、と言い換えてもよい。であれば、身体をそなえた鑑賞者である私が、姿形を

[*3] 心身二元論の構図を「捨て去る」とまでは言っていない。それだけに囚われていては、「表情」に迫れないという問題意識である。

与えるのだ。私との共振的関係のさなかにおいて、その関係性そのものとして、立ち現れるゲシュタルト、それが身体の表情である。

私たちが所与として受けいれがちな「解剖学的身体」も、実はそうした無数に潜在する姿形のたったひとつが顕在化したものにすぎない！しかも解剖学的身体は、広松が表情でいうところの、色でいえば「無色透明」に相当するものである。それ自体がのっぺらぼうというだけの、ひとつの表情なのである。

たとえば私の物語でいえば、こういう連続写真（動画）に対して、ーーということを感じたのである。これは、一種の表情の感得がかかわっている。

ほかにも、たとえば、バキシリーズから、宮本武蔵の動きが。

## 1.4　ツールづくりへの洞察

ここまで実践と、哲学書読み漁りと、表情との出会いによって、実装が固まったのだった。

たしかにそう考えてみれば、Euphrates と ISSEY MIYAKE によ映像作品『ISSEY MIYAKE A-POC INSIDE』[?] かなり似ている。「ファッションデザイナーのイキイキとした動きを如実に表現する」という手法としても作品。

ダンス教育家のラバンは、ーー正二十面体によってバレエのフォームを創造しようとした。

また、一連の作品『Strand Beast』も、この線で非常に興味深い。風だけを動力として、木製の機構がただ作動しているだけである。にもかかわらず、そこに不思議な「情」を感じてしまうのだえる。

こうした「最小限の見た目」でこそかえって如実に表現できるのはなぜなのか？キネティックアーティスト・George Rickey の弁を借りよう。

# 第 2 章

# アプリのしくみ

## 2.1　どういうアプリか？

図 2.1　アプリのようす

図被験者は自身の運動を撮影し、本アプリに流し込み、撮影データをもとに、抽象的な展の映像が出てくるそれを自由ん線でむすんだり、点を非表示にしたりを直感的な UI によってやることで、ユーザが「形態」を編集する。この形態のことを、「表情フォルム」と呼ぶ。ただし、表情が一元的な現象（つまり物でも心でもない）のことである。言うなれば、コインのような関係性である。コインという全体（表情）における、「片面」という意味で「表情形態」とよぶのである。

編集してビジュアライズされた形態そのものは物世界に属するのだから、それ自体を表情と呼ぶべきではないだろう。本アプリにおけるある種の物性をもっている。ゲシュタルトが、ノート機能に書く。これらの仕様によって、自分との関係性に帰着させ、これが表情感得を促し、また、表情感得能力を培うことを目論んでいる。

・身体運動の撮影

モーションキャプチャもしくは通常のカメラで、身体運動を動画撮影する。

・アプリ用データへの整形し、DB に読み込ませる。（撮影データをアプリに読み込ませる）

モーションキャプチャで出力したデータを、アプリ用 3 次元データに整形する。この整形には、

ノイズの削除やマーカデータの統合なども含めている。撮影を通常カメラで行った場合は、動画データをもとに 2D データを生成する「データ整形プログラム」によって行う（web アプリとは異なる)。データ整形プログラムは、プログラミング言語 python で記述し、マーカレスの身体運動動画から骨格検出する処理には、Google 社製の「MediaPipe」[?] を Python ライブラリとしてもちいた。このプログラムは、アプリ本体というよりは、データ整形プログラムである。

・実践家が、アプリをプレイする（撮影した自身のデータや他者のデータで戯れる）

アプリをプレイする。データ選択画面から DB からデータを読み込むとプレイ画面に遷移するので、そこでプレイする。

・再び撮影する

モーキャプは python 処理プログラムによって、行っている。詳しくは付録を参照されたい。手動で行い、

## 2.2 システム構成

まずは全体の構成を説明する。2.2 のとおりである。本アプリは、web ブラウザ上で作動する web アプリである。[*1]

図 2.2 システム構成

[*1] web アプリは一般的に、フロントエンドとバックエンドの 2 つから構築される。フロントエンドとは、ユーザの PC やスマホなどで動いているプログラム（すなわち、ユーザが直接目にふれ操作できる、インターネットにつながっていなくても作動する部分）であり、バックエンドとは、インターネットのむこう側にある（巨大な）コンピュータ（サーバ）で動いているプログラムである。たとえば Youtube であれば、動画を選択・視聴・再生制御したり、コメントを書き込んだりなど、私たちユーザが目に見えて直接操作している部分が、フロントエンドに相当する。いっぽうで、ユーザが動画を視聴できるのは、「Youtube のサーバが、選択された動画を、DB からユーザのスマホへ動画データを送り返している」からである。ユーザがコメント欄にコメントを投稿できるのは、「Youtube のサーバが、送られてきたコメントを、DB に書き込み、DB に書き込まれたコメントをユーザのスマホへ送り返している」からである。これら「」内に相当するのが、バックエンドである。

### 2.2.1 バックエンドについて

バックエンドは、その大部分を Google 社の Firebase におまかせしている。Firebase とは、Google 社が提供する web アプリ開発プラットフォームである[*2]。データベース（以下 DB と書く）とホスティング（web サイトとして稼働させる）は、それぞれ Firebase のサービスである「Firestore」と「Firebase Hosting」をもちいている。

### 2.2.2 フロントエンドについて

アプリの実質的な仕様はこのフロントエンドである。本アプリは、フロントエンド部分は、主に以下のモジュールからなる。それぞれのソースコードは付録を参照されたい。

**メインプログラム**

DB から読み込んだいろんなデータをもっておく。ここから各モジュールで「参照」する。

**映像描画モジュール**

抽象映像（点や線）の描画を担当。グレー背景領域に対しての処理が本モジュールである。グレー背景領域は、HTML の「canvas」要素である。

**再生制御モジュール**

映像の再生/一時停止/停止、コマ送り/戻し、再生速度設定、指定コマへの移動。

**ノートモジュール**

ノート機能を担うプログラム

**ディスプレイ設定モジュール**

その他ディスプレイ設定を担うプログラム

これらのプログラムは、プログラミング言語 JavaScript・HTML・CSS により記述した。以下の JavaScript ライブラリ[*3] をもちいた。() 内にバージョン情報を付記する。

Vue.js(2.6.14)

モダンな web 開発における UI デザインに優れたライブラリである。HTML と JavaScript との連携を高め、より少ないコードで書くことができる。

---

[*2] いわゆる「mBaaS」（mobile Backend as a Service）に分類される。Firebase は、バックエンドのハードウェア環境として、Google 社保有のクラウド上サーバをもちいることができる

[*3] 当該プログラミング言語の、なんらか特定の処理群を寄せ集めてまとめた、いわば機能キットのようなファイルのことである。しばしば、ライブラリは「パッケージ」や「アドオン」とも表記ゆれすることがある。ライブラリを読み込めば、たとえばゼロからコードを書けば 1000 行を要するような処理さえも、わずか数行で書けてしまったりする。開発をスピーディにしてくれ、モダンな開発では（とくに大規模な開発になればなるほど）、ライブラリを積極的に使用するのが主流と言える。

しかしである。「知」とは自分自身の手や身体でつくりあげてゆくべきものだ、という本研究の基本思想はここで強調しておきたい。ライブラリの便利さに甘えすぎた結果「知の受け売り」に慣れてしまう、などということにないよう、自ら気をつけておく態度は重要であろう。

Vuetify(2.6.0)

Vue と連携する形で、モダンな web 開発における UI デザインに優れたライブラリである。テキストフィールド、テキストボックス、ボタンなど、基本的な UI の表現にもちいた。メインプログラムや、再生制御モジュールの UI デザインには、本ライブラリに頼った部分が大きい。なお、UI すべてをこのモジュールに頼ったわけではなく、後述する一部の UI パーツは著者自身で CSS で設計した。

p5.js(1.4.1)

インタラクティブなスケッチを描画できるライブラリである。[*4] 本アプリでは、映像描画部の表現にこれをもちいた。[?] 映像描画モジュールは主に本ライブラリを活用して制作した。

## 2.3 機能とユーザインタフェースの詳細

本アプリは、身体知の学びを支援したりや表情を感得を促すという目的があるからこそ、著者はそれを UI の細部の設計にも反映している。ここでは操作方法に関して説明しよう。ユーザが負担少なく直感的に操作できることは、UI において重要である。マウス

### 2.3.1 再生制御

図 2.3 を参照してほしい。

図 2.3 再生コントローラー

**再生/一時停止（スペースキー）**

スペースキーをトグルスイッチにしている。これは、YouTube や Quick Time Player（macOS のデフォルトの動画プレイヤー）など、よく使われる動画プレイヤーと同様にした。

[*4] Java ベースのプログラミング言語 Processing がもとになっており、それを JavaScript に移植したものである。Processing は、コンピュータアートやメディアアートの制作にもちいられることも少なくない。

**コマ送り/戻し（左右矢印キー）**

右矢印キーで 1 コマ送り、左矢印キーで 1 コマ戻る。

**再生のスライダー（マウスドラッグ）**

再生のスライダーを表示している。スライダーのツマミをマウスドラッグすると、コマを自在に移動できる。

**指定コマへジャンプ（テキストフィールドに入力）**

テキストフィールドに、指定のコマを半角数値で打ち込むと、そのコマにただちに移動できる。

**再生速度変更（セレクタ選択）**

再生速度を 4 段階から選択できる。速度は、× 0.25、× 0.5、× 1、× 2 である。

### 2.3.2 表情形態の編集

図 2.4　表情形態の編集

本アプリ最大の特徴である、表情形態の編集について説明する。macOS なら command キー（windows なら ctrl キー）を押し下げている状態でのみ、ゲシュタルトの編集ができる。この状態を「**編集モード**」と呼ぶ。これらのキーを離すと、自動的に編集モードから抜け、「**再生モード**」に切り替わる。編集モードではないときはすべて再生モードである（モードはこの 2 つのみ）。再生は再生モード時のみできる。再生中に command キーを押すと、その瞬間に一時停止し、編集モードに切り替わる。macOS の command キー・windowsOS の ctrl キーは、様々なアプリにおいて、キーボードショートカット時の修飾キーとしてもっともよく採用されているものである。

編集モード中は、キャンバス背景のグレーが少し暗く・濃くなるようにしてある。ささいな工夫に思えるかもしれないが、ユーザが「いまは編集モードだ」と直感的にわかるためには大事なことである。この仕様そのものが表情に直結するというわけではないが、表情の感得がそもそも簡単ではない。、この種の UI の細かな工夫は、ユーザがストレスなく心身をそこに集中できるよう促す、という観点からして重要だと考える。以降の説明でも、同様の工夫については言及する。編集モー

ド中は、下に書く操作説明も表示されるようになっており、ユーザはいつでも編集のしかたを確認できる。ゲシュタルトの編集は、command キーと他の操作（キーまたはマウス操作）を組み合わせてやる。基本的には編集とは、点どうしを結んだり/外したり、それぞれの点の表示/非表示にしたりすることを指す。したがって、aaaaa

**2 点を連結/解除（点をクリック）**

いずれかのマーカを一回クリック[*5]

すると、そのマーカが選択状態となって色づく。そのまま command キーを離さずにキャンバス上でマウスカーソルを滑らせて、他マーカをマウスオンする。マウスオンすると、両点のあいだに線が x 色で表示される。この状態で 2 点目をクリックすると、両点が線で結ばれる。連結済みの２点どうしに対して以上の操作をすると、２点の連結が解除される。

ユーザがマウスオンしているマーカは赤くなり、かつ、半径が少し大きくなるようにした。

**ある点の表示/非表示を切り替える（D キー）**

ユーザがマウスオンしているマーカの表示/非表示状態を切り替える（D キーがトグルスイッチ的にはたらく）。非表示状態のマーカを実際に見えないようにするのは再生モード時においてのみであり、編集モード時は、表示状態マーカはそのまま見えて、非表示状態のマーカもそれより薄い色で見えるようにしている。

**ある点を中心固定点にする（C キー）**

ユーザがマウスオンしているマーカを中心固定点にする。中心固定点とは、いわゆる 3D カメラの注視点のことであり、中心固定点となったマーカは、再生中でも画面中心で動かない（デフォルトでは、中心固定点はグローバル座標原点である）。このとき他の点・線群は、つねに中心固定点を原点としたローカル座標系に描画されることとなる。[*6] ゆえにどこを中心固定点にするかによって、再生時の全体の点・線群のふるまいは当然異なってくる。この設計では、中心固定点の変化に応じて多彩なゲシュタルトが生まれるのを促したいのである。

**ながめる立場を変える（マウスドラッグ・マウスウィール）**

マウスではなくトラックパッドをつかう場合は、スワイプ動作でみる角度を変え、ピンチ動作でズーム in/out する。

「カメラ」と呼ばないのは、意図的である。本アプリではいわゆる 3D 空間を映し出す「カメラ」は、単なる客観的観測者にとどめていないということだ。いわゆるオブジェクトは、観測者と無関係に存在するが、ゲシュタルト（表情）は、観測者との関係性として成り立つものである。ゆに、

---

[*5] 最近傍マーカは、「独自の計算法」によって「2 次元のマウス位置座標」と「3 次元のマーカ位置座標」との「距離」を算出することで同定している。算出ルールの解説は、コンピュータグラフィックスの描画処理過程にまで踏み入った煩雑な計算を伴うので、付録（何番）に回す。

残念ながら p5.js ライブラリでは、「3 次元空間座標をもつ点群が、最終的にどのように、マウスと同じ画面 2 次元平面へと描画されてゆくか」という処理プロセスがブラックボックス化されている。だから、最近傍マーカ同定法を自前で実装する必要があったのだ。これはライブラリの弱点と言え、脚注*3 で触れた問題と関連する。

[*6] たとえば、デフォルト状態で動きまわるマーカ A と静止したマーカ B があったとしようこれを動点 A を中心固定点にすれば、マーカ A が静止点に、マーカ B が動きまわるようになるわけである。

カメラのふるまいの内部的なルールとしては、以下の図のように配置されており、動作ルールにより、直感的操作を可能としている。マウスドラッグとピンチを、「極座標上を動くのとチョクに対応し動作」にしてある。

**表示中のすべての点どうしを結ぶ/外す（A キー）**

表示中の点群の連結状態を一斉に変更する。表示中の点群の連結状態は以下の 3 状態のどれかに相当するので、A キーをトリガーにして、これら 3 状態間でサイクリックに状態遷移が起こる。

どの点どうしも連結されていない場合、すべての点どうしを連結する。

一部の点どうしが連結されている場合は、すべての点どうしの連結を解除する。

すべての点どうしが連結されている場合は、すべての点どうしの連結を解除する。

**すべての点を表示→孤立点のみ非表示→すべての点を非表示→...（繰り返し）**...

孤立点とは、ほかのどの点とも連結されていない点のことである。孤立点の非表示とはすなわち、連結されている点のみ表示するのは、実際の探るときにありがたい機能である。

### 2.3.3 その他ディスプレイ設定メニュー

tab キーがトグルスイッチとなり、本メニューが画面左側から出し入れされる。「その他」とは言っても、表情感得において無関係というわけではもちろんない。図を参照されたい。

図 2.3 を参照してほしい。

こうやってtodoを書くよ。

図 2.5　その他のディスプレイ設定

### 2.3.4 ノート機能

ノート機能について図で説明。一般的な状態を図にする。それぞれのパーツを指し示して、その解説をする。ちゃんとそれぞれの UI が表情とどう関連しているのかを、明確に説明すること！

### 2.3.5　操作ガイド UI

2 種類の操作ガイド UI がある。アップバーのキーボードボタンと、ちなみに哲学が表示される。スナックバーと

## 2.4　データベース概要

DB の構成を説明する。Firestore は、データを階層構造にして保管する[*7]。それにならって、以下に構造を説明する。

・身体運動コレクション

・ノートデータ

・会員コレクション

これらを連携させながら、web アプリとして望ましい諸機能を実現している。具体的にどういう局面で連携しているのかは、ユーザインタフェースの章で説明する。

## 2.5　開発環境

本アプリの開発にもちいた PC は、Apple 社の iMac2020(Retina 5K, 27-inch) である。このハードウェア詳細は以下である。

- プロセッサ: 3.8 GHz 8 コア Intel Core i7
- グラフィックス: AMD Radeon Pro 5500 XT 8 GB +
- メモリ: 40 GB 2667 MHz DDR4

プログラミングのエディタには、Microsoft 社の Visual Studio Code[**?**] をもちいた。

[*7] いわゆる NoSQL である。

# 参考文献

# 付録

# 第 1 章

# アプリについての付録

## 1.1　カメラ操作 UI のしくみ

カメラのふるまい（位置変化）を制御するのは、マウス（トラックパッド）アクションである。直感的なマウスアクションでカメラ位置を制御するために、カメラの位置を、動径 $r$、極角 $\phi$、方位角 $\theta$ をもちいた「極座標形式の世界座標系」で定義している。

$$CameraPosition = (r, \phi, \theta)$$

噛み砕いていうなら、カメラを「半径可変の地球上」で移動させるようなものであり、カメラの位置は、半径 $r$、緯度 $\phi$、経度 $\theta$ で決まる。カメラの上方向は、球面のうちカメラ位置における経線に沿った接線の北極方向（φ で微分して得られる接線）

command キーを押しっぱなしにした状態で（=編集モードで）キャンバス上で、上下にマウスウィール

キャンバス上をマウスウィールアップ（トラックパッドならピンチアウト）：ズームキャンバス上をマウスウィールダウン（トラックパッドならピンチイン）$r$ を操作する。

表 1.1　カメラズーム

| カメラふるまい | マウスの場合 | トラックパッドの場合 |
|---|---|---|
| データ 1 | データ 2 | データ 3 |
| データ 4 | データ 5 | データ 6 |

このうえでカメラは、カメラはつねに地球中心あたり（原点 or どれかのマーカ）を注視・追尾する。デフォルトでは、$r = 5[m]$ になっている。本アプリのディスプレイ設定でメッシュ状の球面を描くことができるが、この球面は半径 $3[m]$ であり、多くの運動は、この球面内におさまる。したがって、

この操作で決まった極形式の位置座標を、直交座標形式に変換してから、p5.js のカメラオブジェクトに渡す。

縦か横つまり、カメラの動きと比例関係にある。

## 1.2 最近傍マーカの計算法

コンピュータグラフィックスの処理課程（グラフィックスパイプラインもしくはレンダリングパイプライン）は、以下の流れである。たとえば、ある 3 点があるとして説明する。

1. モデリング：世界座標系において、対象の 3 次元位置座標を決める
2. 視点変換：対象をカメラからの眺めに変換（世界座標系からカメラ座標系への並行移動）
3. （カリング）
4. クリッピング
5. （ラスタライズ）
6. 画面に出力

なお、ベクトルはボールド体で表記する。空間にある 2 つのマーカの 3 次元位置座標 $\boldsymbol{p}$,$\boldsymbol{q}$ と、画面上のマウスカーソルの 2 次元座標 $\boldsymbol{m}$ の「距離」$d$ を算出する。マーカの位置をそれぞれ、

$$\boldsymbol{p} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}, \boldsymbol{q} = \begin{pmatrix} q_x \\ q_y \\ q_z \end{pmatrix}, \boldsymbol{m} = \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix}$$

モデル行列は今回は関係ない。カメラの外部パラメータと内部パラメータによって、変換ができる。

カメラの外部パラメータは、以下 3 つからなる。

- 世界座標系におけるカメラの位置（3 次元ベクトル）
- 世界座標系におけるカメラの注視点（3 次元ベクトル）
- 世界座標系におけるカメラの上方向（3 次元ベクトル）

これら 3 つの情報は、1 つの「ビュー行列」にまとめて表現できる。

$$\mathbf{V} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

カメラの内部パラメータは、以下 4 つ[*1] からなる。

- 焦点距離
- アスペクト比

[*1] これら 4 つの情報をもちいて、図のように、カメラからみた世界全体から、ある「四角錐台」の領域を一意に定めている。この四角錐台の領域のみ、「カメラに映り込む世界」として切り取るわけだ。なぜ直方体ではなく四角錐台なのか？それは「カメラに近ければ近いほど大きく写り、遠ければ遠いほど小さく映る」ということを反映するためである。このあと四角錐台を、２つの底面を押しつぶすように、二次元平面へと圧縮するが、それで反映される。

- ニアクリップとファークリップ平面の距離
- 視野（Field of View, FoV）

これら 4 つの情報は、1 つの「プロジェクション行列」にまとめて表現できる。

$$\mathbf{P} = \begin{pmatrix} \frac{1}{\tan(\mathrm{FOV}/2)\times\mathrm{aspect}} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(\mathrm{FOV}/2)} & 0 & 0 \\ 0 & 0 & \frac{f+n}{f-n} & \frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

したがって、対象点の 3 次元ベクトルに、ビュー行列をかけてからプロジェクション行列をかけることで、変換後の画面座標系で表された 2 次元ベクトル $\boldsymbol{p'} = \begin{pmatrix} p'_x \\ p'_y \end{pmatrix}$ が得られる。

$$\boldsymbol{p'} = \mathbf{PV}\boldsymbol{p} = \begin{pmatrix} \frac{1}{\tan(\mathrm{FoV}/2)\times\mathrm{aspect}} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(\mathrm{FoV}/2)} & 0 & 0 \\ 0 & 0 & \frac{f+n}{f-n} & \frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 0 \end{pmatrix}$$

すべてのマーカ（n 個）に対して、これらの画面上 2 次元座標を求める。すべてのマーカの画面上 2 次元座標とマウスカーソルとの距離をそれぞれ求める（$d_1, d_2, ..., d_n$）。あらかじめ画面 2 次元平面上でマウスカーソルを中心とした一定の半径 $r$ 内にある点群を、「カーソル近傍マーカ」とするようにしておき、$r > d$ を満たす点群を求める。

「カーソル近傍マーカ」群のうちもっとも「カメラからみて手前」にあるものを、「最近傍マーカ」とする。

## 1.3 開発にもちいた JavaScript パッケージ

あかーーいテキスト feiji 以下が npm パッケージ一覧である。

```
function hello(){
  console.log('hello')
}
```