第Ⅰ部

身体の「表情」感得の実践

第 1 章

アプリの仕組みついて

1.1　アプリをつかう流れ

・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。モーションキャプチャは、Trio、OptiTrack をもちいた。

・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、

1.2　システム構成

まずは全体の構成を説明する。図のとおりである。本アプリは、web ブラウザ上で作動する web アプリである。Firebase で開発している。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[*1]。DB は、Firestore を用いている。web アプリのホスティング（web サイトとして開設するためのサーバ）は、Firebase Hosting を用いている。

web アプリなので、プログラミング言語 JavaScript・HTML・css で開発した。メインに用いた JavaScript ライブラリは以下である。() 内にバージョン情報を付記する

Vue.js(2.0.0)
: 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。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デザインした。

p5.js()
: スケッチを描画できるプログラミング言語である Processing を、JavaScript に移植したバージョンである。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。本アプリでは、映像描画部の表現にこれをもちいた。[?]

Firebase
: 抽象映像（点や線）の描画を担当。グレー背景の領域が本モジュールである。

再
: 映像の再生/一時停止/停止、コマ送り/戻し、再生速度設定などを担当。

1.3　アプリのモジュール構成

web アプリは、プログラミング言語 JavaScript で記述した。アプリは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

[*1] いわゆる「mBaaS」（mobile Backend as a Service）に分類される。データベースなど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。

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。

本アプリの開発環境を説明する。Apple 社の iMac2020(Retina 5K, 27-inch) をもちいて開発した。ハードウェア環境の詳細は以下である。

- 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。CPU

1.4 データベース概要

データベースの概要を説明する。Firestore であることは述べたが、Firestore は、データを階層構造にして保管する（いわゆる NoSQL）。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。

1.5 ユーザインタフェース

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

1.5.1　再生制御

図 1.1 を参照してほしい。

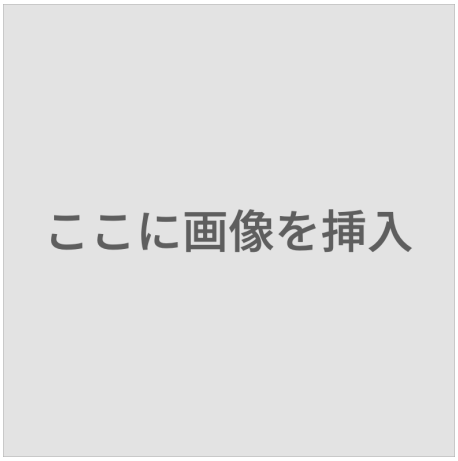

図 1.1　再生コントローラー

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 標準動画プレイヤー）など、よく使われる動画プレイヤーと同様にした。

コマ送り/戻し（左右矢印キー）

左右矢印キーで、コマ送り/コマ戻しができる。右矢印が +1 コマ、左矢印が-1 コマである。

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。

1.5.2　ゲシュタルトの編集

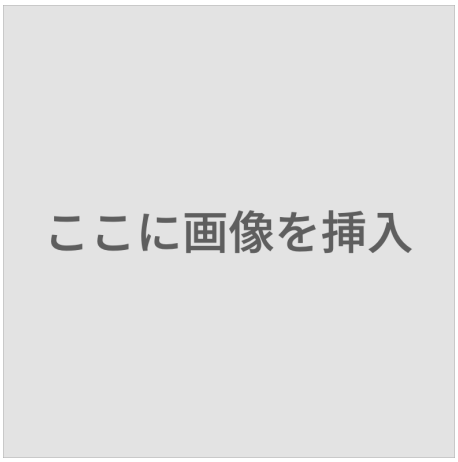

図 1.2　ゲシュタルト編集

macOS なら command キー、windows なら ctrl キーを押し下げた状態のみ、編集モードになる。これらの修飾キーは、それぞれの OS においてキーボードショートカットの修飾キーとしてもっとも頻繁に利用されるものだ。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。小さ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。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、

すべての点をーー

カメラ操作

重要なポイントだが、本アプリではいわゆる 3 D 空間を描写する「カメラ」は、単なる客観的観測者にとどめていない。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。ゆえ

コマ間を自在に移動（スライダーをマウスドラッグ）

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、