GazeDataCollection

概要

このアプリケーションはTobiiアイトラッカーハードウェアとElectronベースのフロントエンドを組み合わせて、視覚的選択実験を行うためのシステムです。参加者は画像セットを閲覧しながらシステムが視線の動きを追跡し、好みの画像や画像内の関心領域を選択することができます。視線パターンやユーザー選択を含むすべてのデータは、後の分析のために保存されます。

システムコンポーネント

バックエンド

- **Pythonアイトラッキングサービス**: Tobiiハードウェアに接続し、WebSocket経由で視線追跡データをストリーミングします
- `high_speed_tobii_websocket.py`

フロントエンド

- **Electronアプリケーション**: クロスプラットフォームのデスクトップアプリケーションフレームワーク
- **Reactベースのユーザーインターフェース**: モジュラーコンポーネントアーキテクチャを採用
- **WebSocketクライアント**: Pythonバックエンドに接続して視線追跡データを受信

システム要件

- **ハードウェア**:
  - Tobiiアイトラッカー
  - 互換性のあるUSBポートを備えたコンピュータ
- **ソフトウェア**:
  - Python 3.10
  - Node.js 14以降
  - Tobii Pro SDK
  - Docker

インストール

Dockerを使用する方法（推奨）

1. **リポジトリをクローン**:

```
git clone https://github.com/mmmsmm16/GazeDataCollection.git
cd GazeDataCollection
```

2. **Dockerコンテナをビルドして実行**:

```
cd frontend
docker-compose build
docker-compose up
```

3. **コンテナ内で依存関係をインストールしアプリケーションをビルド**:

```
npm install
npm run build
```

手動インストール

1. **リポジトリをクローン**:

```
git clone https://github.com/mmmsmm16/GazeDataCollection.git
cd GazeDataCollection
```

2. **Python依存関係をインストール**:

```
cd backend
pip install -r requirements.txt
```

3. **フロントエンド依存関係をインストール**:

```
cd ../frontend
npm install
npm run build
```

設定

画像セット

システムは`frontend/image_sets`ディレクトリから画像を読み込みます。以下のように画像を整理してください：

- `frontend/image_sets`の下にサブディレクトリを作成（例：`set1`、`set2`）
- サブディレクトリに画像（JPG、JPEG、PNG、GIF）を配置
- 各サブディレクトリは個別の画像セットとして扱われます

ディレクトリ構造の例：

```

```

```
frontend/
├── image_sets/
│   ├── set1/
│   │   ├── image1.jpg
│   │   ├── image2.jpg
│   │   └── ...
│   ├── set2/
│   │   ├── image1.png
│   │   └── ...
│   └── ...
```

バックエンド設定

WebSocketサーバーはデフォルトでポート8765で実行されるよう設定されています。変更が必要な場合は：

1. `backend/eye_tracker.py`または`backend/high_speed_tobii_websocket.py`でポートを変更
2. `frontend/src/renderer/hooks/useEyeTracker.js`で対応するWebSocket URLを更新

システムの実行

バックエンド（アイトラッキングサーバー）の起動

1. **Tobiiアイトラッカーをコンピュータに接続**
2. **アイトラッキングサーバーを起動**:

```
python high_speed_tobii_websocket.py
```

3. **アイトラッカーが検出されたことを確認**: コンソールに「Found eyetracker: [モデル名]」および「WebSocket server started on ws://0.0.0.0:8765」と表示されるはずです

フロントエンドアプリケーションの起動

1. **Electronアプリケーションを実行**:

```
cd frontend
npm start
```

2. **ディスプレイを選択**: プロンプトが表示されたら、実験に使用するディスプレイを選択

アプリケーションの使用方法

初期設定画面

1. **画像セットの選択**: `image_sets`ディレクトリから読み込まれた利用可能な画像セットから選択
2. **ユーザータイプ**: 役割に応じて「Host」または「Guest」を選択

3. **総ステップ数**: セッションの画像選択ステップ数を設定
4. **領域選択ステップを含める**: ユーザーが画像内の関心領域を選択すべきかどうかを切り替え
5. **セッション開始**: パラメータを設定した後、実験を開始

実験の実行

1. **カウントダウン**: 各ステップの前にカウントダウンタイマーが表示される
2. **画像選択**: 4つの画像が画面に表示され、ユーザーは「選択」ボタンをクリックして好みの画像を選択
3. **領域選択**（オプション）: 有効にすると、ユーザーは選択した画像上に長方形の領域を描画して、関心領域を示すことができる
    - **ポジティブ/ネガティブ**: ポジティブまたはネガティブな領域としてマークするかを選択
    - **領域の削除**: 以前に描画した領域を削除

セッションの終了

セッションは設定されたすべてのステップを完了すると自動的に終了します。または、アプリケーションを閉じることで早期に終了することもできます。

データストレージとフォーマット

すべてのデータは`frontend/data`ディレクトリに保存され、ユーザータイプとセッションIDによって整理されます。

ディレクトリ構造

```
frontend/
├── data/
│   ├── host/
│   │   ├── 1/
│   │   │   ├── 1_1.json
│   │   │   ├── 1_1.csv
│   │   │   └── ...
│   │   ├── 2/
│   │   │   └── ...
│   │   └── sessions_info.txt
│   ├── guest/
│   │   └── ...
```

データファイル

1. **JSONファイル**（`step_substep.json`）: 各ステップに関するメタデータを含む:
    - 画像情報と位置
    - 選択された画像と領域
    - ユーザーアクションログ
    - タイムスタンプ

    JSON構造の例:

```
{
  "step": 1,
  "subStep": 1,
  "timestamp": "2024-10-09T08:26:40.162Z",
  "isRegionSelectionStep": false,
  "images": [
    {
      "id": "image1",
      "src": "file://path/to/image",
      "alt": "image1",
      "position": 1,
      "isSelected": true,
      "regions": {
        "positive": null,
        "negative": null
      }
    },
    ...
  ],
  "userActionLog": [
    {
      "timestamp": 455927616900,
      "action": "SESSION_START",
      "details": { "sessionId": 1, "userType": "guest", "totalSteps": 10 }
    },
    ...
  ],
  "isEndSession": false
}
```

2. **CSVファイル**（`step_substep.csv`）: 生の視線追跡データを含む:
   - `timestamp`: アイトラッカーのタイムスタンプ（マイクロ秒）
   - `left_x`, `left_y`: 左目の視線座標（0-1の範囲、画面に対する相対値）
   - `right_x`, `right_y`: 右目の視線座標（0-1の範囲、画面に対する相対値）

   CSVフォーマットの例:

```
timestamp,left_x,left_y,right_x,right_y
455927616900,0.3484,0.4429,0.3609,0.4099
455927633566,0.3413,0.3975,0.3622,0.3716
...
```

3. **sessions_info.txt**: ユーザータイプごとのすべてのセッションの概要:

```
SessionID: 1, Data Count: 10
SessionID: 2, Data Count: 5
```