1. アプリ概要と主な機能

Expo NaviFlow は、大規模イベント（今回は万博をベースとした）会場内で、人の混雑を避けながら快適に目的地を回れるよう設計された「ルート最適化ナビゲーションアプリ」です。

＜主な機能＞

・目的地（スタート、経由、ゴール）の選択と、最適な回遊ルートの自動計算

・混雑状況をリアルタイムに反映し、より空いているルートを提示

・地図上へのルート描画（白→青で進行順が分かるグラデーション線）

・ルートの再選択・再計算の対応

・現在の状況をライブカメラで視聴可能（今回は連携ができなかったため Unity 上のマップで代用）

2. 操作手順（基本的な流れ）

(1) スタート地点を選択

(2) 必要に応じて経由地点を複数選択（再クリックで解除も可能）

(3) ゴール地点を選択

(4) [complete] を押すことで、最適化ルートを自動描画

(5) 地図上にルート線と説明文が表示される

(6) [リセット] を押すとすべての選択・表示が初期化され、再選択が可能

3. 今後の改善案（拡張できる機能例）

・混雑予測表示（今後混み始める施設の予告機能）

・ルートの回る順番を文字、AR で表示

・滞在時間の目安表示（各施設の平均所要時間の可視化）

・ユーザー設定に応じたルート最適化（例：年齢、性別、グループに応じたルートの表示）

・QR コード共有機能（家族や友人と同じルートを共有可能に）

・色覚バリアフリーモード（混雑度表示を色以外でも伝える工夫）

- 地図に施設写真や CG モデルを重ねた視覚強化

4. 開発技術とシステム構成の概要

■ 使用開発環境と技術一覧

・OS：Windows 11

・Unity：2022.3.38f1

→ 地図 UI、カメラ切替、UI 操作、ルート描画などクライアント側の中核機能を構築

・Python：3.9

→ FastAPI サーバーによるルート最適化と混雑処理を担当

・WSL 環境：Ubuntu 20.04

→ Unity と FastAPI 間の通信処理（Web 連携）を構築

・FastAPI：ルート API・混雑 API として構成

・Blender + BlenderGIS：Expo 地形マップの構成補助・レイアウト設計

■ システムの仕組み

1. ユーザーが「スタート」「経由」「ゴール」の地点を選択

2. Unity → FastAPI にルート情報を送信（POST /route）

3. Python 側で混雑度を反映した最適化ルートを計算（ACO アルゴリズム）

4. Unity がルート・混雑情報を受信 → 地図上にルート線を描画（白→青）

5. パビリオンの色分け（青〜赤）で混雑度をリアルタイムに可視化

■ 混雑度の表示方法

- 各施設に赤〜青の色を割り当て（人の多さによって変化）

- 混雑人数はライブカメラ（YouTube）映像を解析し、YOLOv5 でカウント

- 結果を ExpoRouteVisualizer.cs 経由でパビリオンの色に反映

５．想定される質問と回答例

Q1. このアプリは何を目的に作られていますか？

A. Expo NaviFlow は、万博などの大規模イベントで「人の流れ」「施設の混雑状況」を考慮しながら、回る順番を最適化できるナビゲーションアプリです。現地でも事前計画でも使えるよう、操作性と視認性にこだわって設計しました。

Q2. 混雑状況のデータはどこから取得しているのですか？

A. Expo NaviFlow では、実際の会場に設置されたライブカメラ（YouTube Live など）の映像を元に、AI が施設ごとの混雑状況をリアルタイムに解析しています。　具体的には、Python 側の aco_realtime.py にて OpenCV と YOLOv5（画像認識 AI）を使用し、カメラ映像上の人物を検出・人数をカウントしています。
その人数データは施設ごとの「混雑度」として数値化され、Unity 側の ExpoRouteVisualizer.cs でパビリオンごとの色（青〜赤）に変換されて表示されます。この混雑データはルート最適化の計算にも反映されており、実際の人の流れを考慮したナビゲーションが可能になっています。（ただ、現在は三つのライブカメラしか使用できていないので情報の確実性は少し怪しいです。）

Q3. パビリオンに表示されている色の意味は何ですか？

A. 色は混雑度を表しています。青が空いている、赤が混雑しているなど、混雑レベルに応じて段階的に色分けしています。一目で混み具合が分かるようになっています。

Q4. ルートを描画するときの線の色に意味はありますか？

A. はい、線は白から青にグラデーションで変化し、進む順番（スタート → ゴール）を視覚的に示しています。番号ラベルも併せて表示されるので迷うことがありません。

Q5. 最適なルートはどうやって計算しているのですか？

A.アリコロニー最適化（ACO：Ant Colony Optimization）というアルゴリズムを使って、スタート・経由・ゴールの順路の中で最も効率的なルートをリアルタイムで計算しています。ACO は、実際のアリが餌場までの道のりを探索する行動をモデルにした「群知能アルゴリズム」です。複数の仮想的な「アリ」が経由地の並べ方（順列）を探索しながら、良いルートには「フェロモン」を残し、その情報を次の探索に活かします。

具体的な処理の流れは以下の通りです：

1. **複数の経由地パターンを生成**

   → たとえば 3 つの経由地がある場合、その並び順は 6 通り（3!）になります

   それぞれの順番で「移動コスト」を評価します

2. **評価に使うコスト行列を構成**

   → コストは「距離」+「混雑度」+「滞在時間」で加算され、

   混雑が多い施設を経由するほどルート全体の所要時間が増える設計です

3. **アリたちが複数回ルート探索を繰り返す**

   → 各ルートパターンに「フェロモン」と「評価値（1/コスト）」を使って

   良いルートに多くの探索が集中するようになります（収束）

4. **最終的に、最も評価の高いルート順が選ばれる**

   → この順番が、アプリに描画される「おすすめ順路」として表示されます

   番号ラベルや白→青の線で視覚的に案内されます

この仕組みによって、距離が短いだけでなく「混んでいない順路」や「スムーズに回れる順番」を自動で判断し、来場者にとって最もストレスの少ないルートを提示できるようになっています。

Q6. 他のナビアプリと比べて、何が強みですか？

A. 混雑度を考慮して「できるだけ空いているルート」を提案できる点と、操作がシンプルで結果が即座に視覚化される点が特長です。また、施設間の距離・滞在時間も反映されるため、リアルな来場体験に近いルート提案が可能です。

【Expo NaviFlow アプリ完成までの全工程記録】

1. マップ設計・地形モデル作成（Blender + BlenderGIS）

・使用ツール：Blender + BlenderGIS

・地形取得元：OpenStreetMap（OSM）、Google Satellite

・手順：

- BlenderGIS の「Basemap」で衛星画像を表示

- 「地名検索（例：夢洲）」→範囲調整→地形の切り出し

- 「Get OSM」で建物・道路・施設情報を取得

- building/highway/leisure などの層を取り込み、Expo レイアウトに合わせて配置調整

- 各施設ごとに空オブジェクト「drawpoint_◯◯」を配置し、ルート描画用の拠点に設定

- 完成モデルを .fbx 形式でエクスポートし、Unity へ導入

2. Unity でのマップ構築と Scene 設計

・Blender からインポートした地形・施設モデルを Unity に配置

・描画ポイントは "DrawPointContainer" にまとめ、Tag = "DrawPoint" を付与

・Terrain やカメラ設定を整え、Expo 会場全体を俯瞰できる Scene を設計

3. UI 選択処理と状態管理（主要スクリプト構成）

・RouteSelectionManager.cs：

- ユーザーの選択（startID, viaIDs[], goalID）を管理

- 選択施設に応じて色変更（緑＝スタート、青＝経由地、赤＝ゴール）

- TextMeshPro に現在の選択状態を反映

4. カメラビュー切替と表示調整（CameraSwitcher.cs）

・メインビュー ⇔ 施設ビュー（ひかりの森／東ゲートなど）切替

・ビュー切替と同時に UI 表示（ルートパネル・ミニマップ）も制御

・miniMapManuallyEnabled により、マニュアル表示調整を可能に

5. FastAPI によるバックエンド通信構築

・バックエンド：Python + FastAPI

・主要エンドポイント：

- POST `/route`：start/via/goal を受け取り → 最適経路と描画順を返却

- GET `/expo-route`：施設ごとの混雑人数を返却 → 色分け表示に反映

- GET `/video`：JPEG ストリーム映像を配信 → Unity 側で受け取り表示

6. ライブ映像の取得と表示方法

・Python 側（FastAPI）：

- OpenCV で RTSP 映像を取得 → JPEG 形式にエンコード

- StreamingResponse で `multipart/x-mixed-replace` 形式として配信

・Unity 側：

- RawImage で Texture2D に映像を描画

- `/video` に定期アクセス → UI 上でライブ状況表示が可能

7. ルート描画と混雑情報の可視化（ExpoRouteVisualizer.cs）

・LineRenderer で施設間の経路を白→青のグラデーションで描画

・TextMeshPro 3D で順番ラベルを番号表示

・混雑人数に応じて施設オブジェクトの色を赤〜青の 5 段階で変更

・ルート確定時にラベル・線・色が即時更新されるよう処理

8. 蟻コロニー最適化（ACO）によるルート探索

・ACO（Ant Colony Optimization）アルゴリズムを実装（aco_realtime.py）

・探索対象：施設順の経路パターン（巡回順列）を評価

・各経路に「移動距離 × 混雑係数 × 滞在時間」のコストを計算

・YOLOv5 でカメラ映像から人数をリアルタイム検出 → 混雑度に反映

・複数の仮想蟻が探索し、高品質ルートにフェロモンを蓄積

・最終的に「最も効率的な順番リスト」を JSON で Unity 側へ返却

9. UI 調整と最終整合チェック

・UI ボタン／テキスト／ミニマップ配置を調整し、視認性と操作性を向上

・「全リセット」操作では ID／色／UI／描画線をすべて初期化

・施設ビュー切替時にパネルや説明欄が正しく非表示になるよう制御