筑波大学大学院博士課程

システム情報工学研究科特定課題研究報告書

施設内での経路案内を可能にする ナビゲーションシステムの開発

―複数のモバイルプラットフォームで
共通動作を実現する実装方式の開発―

萬成 亮太

修士（工学）

（コンピュータサイエンス専攻）

指導教員　三末 和男

2015 年 3 月

概要

スマートフォンの普及に伴い、様々なナビゲーションアプリが広く利用されている。このような製品の多くは国土地理院などが提供している地図やデータベースを利用してナビゲーションを行っているが、公共施設内の施設情報は提供されていないことが多く、大学や遊園地のような場所では利用することが出来ない。そのため、顧客である和田教授と山際准教授は、広大な敷地を持つ筑波大学への来客者の案内に苦慮しており、自身も用務先の建物の位置が学内のどこにあるのか分からないといった問題を抱えている。この問題を解決するため、本プロジェクトでは大規模な施設を対象とした歩行者ナビゲーションシステムを開発した。筆者は本システムのうち、利用者がサービスを利用するためのスマートフォンアプリケーションの設計と開発を担当した。開発に際し、複数のモバイルプラットフォームでの開発に考慮すべき事項があることが判明した。本報告書は、Android と iOS のプラットフォーム間の差異を吸収し、共有動作を実現するための方式について述べる。

目 次

第 1 章　はじめに　1

第 2 章　開発背景とプロジェクト概要　4
2.1 歩行者ナビゲーションシステムの現状 4
2.2 顧客が抱える課題と要望 5
2.3 課題の解決策 5
2.4 想定する利用者 7
2.5 学内ガイドを可能にするナビゲーションシステムの開発 7
2.5.1 システム構成 7
2.5.2 歩行者ナビゲーションアプリ 9
2.5.3 データベース 12
2.5.4 管理アプリケーション 14
2.5.5 課題と解決策 15
2.6 施設内経路探索と施設内検索を可能にするナビゲーションシステムの開発 . . 16
2.6.1 機能 16
1. 施設情報の管理 16
2. 施設情報の検索 17
3. 経路案内 17
4. 道情報の自動登録 18
2.6.2 システム構成 18
2.6.3 開発体制とスケジュール 18

第 3 章　複数のモバイルプラットフォーム向けの実装方式の開発　21
3.1 実装方式の開発手順 21
3.1.1 アプリケーションの開発方針 22
クラス構造の統一 22
画像リソースの効率的な生成 23
3.1.2 クラス対応表の作成 25
3.1.3 画像変換ツールの開発 25
画像の変換 26
9-Patch 28

3.2 ナビゲーションシステムの開発 29
3.2.1 二つのプラットフォームにおけるアプリケーションの実装手順 29
3.2.2 機能要件 31
3.2.3 画面設計 33
標準画面 33
メニュー 34
検索結果画面 34
案内画面 36
案内画面の切り替え 37
ブックマーク画面 37
3.3 開発した実装方式 37

第 4 章 システム評価 **40**
4.1 実装の考察 40
4.1.1 画像リソースの統一 40
4.1.2 クラス構造の統一 41
4.2 ユーザビリティテスト 42
4.2.1 実験の目的 42
4.2.2 実験内容 42
4.2.3 実験結果 43
4.2.4 アプリケーションの改良 43
4.3 考察 46

第 5 章 おわりに **48**

謝辞 **49**

参考文献 **50**

図 目 次

2.1 Google マップに登録されている建物情報の例 6
2.2 Google マップを利用して筑波大学内で経路検索をした時の例 6
2.3 パークの案内図 6
2.4 東山動物園 6
2.5 システム構成 8
2.6 施設切り替え機能 10
2.7 検索機能 10
2.8 ナビゲーション機能（地図） 11
2.9 ナビゲーション機能（AR） 11
2.10 ブックマーク機能 11
2.11 履歴機能 11
2.12 大学の論理構造 12
2.13 学内情報を保持するためのデータベーススキーマ 13
2.14 本システムの全体構成 19
2.15 全体スケジュール 19

3.1 標準解像度の端末向けの画像リソースと表示結果 24
3.2 4 倍解像度の端末向けの画像リソースと表示結果 24
3.3 mdpi、@1x を基準とした変換（アップコンバート） 27
3.4 十分に大きい画像を基準とした変換（ダウンコンバート） 27
3.5 9-Patch なしで横方向にスケールした例 28
3.6 9-Patch ありで横方向にスケールした例 28
3.7 縮小前の 9-Patch 画像 29
3.8 通常のリサイズ 29
3.9 画像変換ツールのリサイズ 29
3.10 本システムの内部設計 30
3.11 プラットフォーム間で処理の流れが異なる例 31
3.12 プラットフォーム同士の処理の流れを統一した例 32
3.13 状態の遷移 33
3.14 標準画面 34
3.15 メニュー 35

3.16 検索結果（Android 版） 35
3.17 検索結果（iOS 版） 35
3.18 案内画面（地図） 36
3.19 案内画面（AR） 36
3.20 履歴・お気に入り 37
3.21 プロジェクト構成 39

4.1 地図画面を表示するジェスチャ 44
4.2 AR 画面を表示するジェスチャ 44
4.3 チュートリアル 1 ページ目（赤枠内、Android 版） 45
4.4 チュートリアル 2 ページ目（Android 版） 45
4.5 チュートリアル後（Android 版） 45
4.6 チュートリアル 1 ページ目（赤枠内、iOS 版） 45
4.7 チュートリアル 2 ページ目（iOS 版） 45
4.8 チュートリアル後（iOS 版） 45
4.9 追加したメソッド（Android 版） 47
4.10 追加したメソッド（iOS 版） 47

第1章　はじめに

本報告は、筑波大学大学院コンピュータサイエンス専攻の高度 IT 人材育成のための実践的ソフトウェア開発専修プログラムにおける特定課題研究として実施した「施設内での経路案内を可能にするナビゲーションシステムの開発」について述べたものである。

スマートフォンの普及により、歩行者用ナビゲーションアプリが広く利用されている。Google Maps[1] を代表としたこれらのアプリケーションには、目的地の名称さえ分かればその場で検索を行い、目的の場所にたどり着くことが出来るという特徴がある。このような製品が数多く出回っている背景には、Google[2] や Apple[3] などのコンテンツプロバイダが、地図や地名といった情報を Application Programming Interface（以下、API）として扱いやすい形で提供していることが挙げられる。これらの API を利用することで、地図情報を持たないサービス提供者が、手軽にナビゲーションアプリを開発することが可能となっている。

しかし、これらの API が持つ情報は公道や公共交通機関向けの物が多く、大学内の建物や遊園地内のアトラクションと言った大規模施設内の詳細な情報は提供されないことが多い。そのため、例えば筑波大学の場合、学内の施設や履修している科目が開設されている建物まで辿り着くことが、事前に場所を把握していないと困難であるという問題がある。

このような背景の中、本プロジェクトの顧客である筑波大学システム情報系情報工学域の和田耕一教授及び山際伸一准教授（以下、顧客ら）は、筑波大学への来訪者を誘導する方法に苦慮しており、また自身も用務先の建物の位置が学内のどこにあるのか分からない場合があるという問題を抱えている。そこで 2013 年度、顧客らの要望を受け、様々な大学に導入することが出来る歩行者ナビゲーションシステムを開発するプロジェクトを発足した。

このシステムはスマートフォンで使用する歩行者ナビゲーションアプリケーション（以下、歩行者ナビゲーションアプリ）、Web ブラウザから利用する施設情報を管理するための管理アプリケーションおよびデータベースから構成されている。

しかし、このシステムを対象とした評価実験や顧客からの評価、プロジェクト内における議論などの結果、次に上げる問題が浮上した。

1. 歩行者ナビゲーションアプリが Android のみに対応している

前システムで開発した歩行者ナビゲーションアプリが対応するプラットフォームは Android のみであった。しかし、2014 年 3 月末時点での日本国内のプラットフォームの普及率は、Android が 57.1%、iOS が 41.8%と、iOS の普及率が Android に次ぐ大きさとなって

[1] https://www.google.co.jp/maps
[2] https://www.google.co.jp
[3] https://http://www.apple.com/

いる [1]。より多くのユーザを獲得するため、様々なプラットフォームへの対応は必要不可欠である。

2. 施設情報の検索に使用できるワードは目当ての情報の正式名称のみである

前システムでは、施設内を案内する際、キーワード検索によって施設内にある目的地に関する情報を得ることが出来る。しかし、このキーワード検索には問題があり、実際に施設に来場し、本システムを用いて施設内の案内を受ける者（以下、サービス利用者）は、施設内の場所の名称を完全に、または部分的にでも知っている必要がある。本システムは利用者としてその施設に詳しくないユーザを想定しているため、このような制約は実際に利用する際に大きな障害となり得る。

3. 大学以外の施設に導入できない

前システムは、当初の方針として大学におけるナビゲーションを前提として構築された。そのため、施設内の情報を表すモデルも大学向けに作成されており、大学以外の施設、例えば遊園地や動物園といった大規模施設の情報を表現するには向いていなかった。ところが、前システムのプロジェクトの後半で、大学以外の施設でも対応できるようにして欲しいとの要望が顧客らから出された。この要望に対し、前システムでは大学向けに設計されたモデルの中に、大学以外の施設情報を入力することで解決していたが、この方法では根本的な解決にはならない。

4. 目的地までの経路案内が行われない

前システムで開発した歩行者ナビゲーションアプリは、目的地まで案内をする際、現在地と目的地を地図や Augmented Reality（以下、AR）で示すだけのシンプルなものであった。しかし、本プロジェクトで開発するシステムを導入する大規模施設の中には、筑波大学のように道が複雑に入り組んでいる場合が想定される。このような施設で現在地と目的地だけの情報で歩行者に案内を促すのは、歩行者ナビゲーションアプリとして不親切である。

以上の各問題に対応するための新たなプロジェクトを 2014 年 4 月より発足させ、次に挙げる各方策で課題の解決に当たることとした。

1. iOS 向け歩行者ナビゲーションアプリの開発とクラス設計の統一

本システムがサポートするプラットフォームとして、Android に加えて iOS を新規開発する。Android と iOS ではクラス構造の設計ルールや API に差異があるため、Android 向けアプリケーションと iOS 向けアプリケーションの双方でクラス構造を可能な限り統一する。これにより、歩行者ナビゲーションアプリに修正を加える際、統一化によって修正箇所を局所化することが出来、高い保守性を得ることが出来る。

2. 施設情報へのタグ付けによる関連キーワードを用いた検索への対応

施設内情報データベースの施設内の情報に対し、略称や通称、関連するワードなどをタ

グとして登録する。キーワード検索を行う際、これらのタグ情報からもデータを検索することで、より柔軟な検索に対応出来るようにする。

3. 階層数の柔軟な変更を可能とするための施設情報データベーススキーマの再設計

本システムを導入する可能性のある施設は多岐にわたる。これらの施設で、施設内情報を抽象化し、保持出来るようにデータベーススキーマを変更する。また、それと同時に本システムの運用を行う施設の担当者（以下、施設管理者）向けの施設情報管理機能を再設計後のスキーマに対応させる。

4. 施設内の道情報を管理する機能、経路案内機能をおよび道情報自動登録ツールの開発

施設内の道情報を登録・管理する機能、および道情報を用いてサーバ側で経路探索を行う機能を開発する。また、歩行者ナビゲーションアプリ側では、地図と AR による道案内に、新たに経路情報を提示する機能を開発する。さらに、道情報の収集には膨大な時間と労力がかかることが予想されるため、この作業の手間を軽減するための「道情報自動登録ツール」を開発する。

本システムをリリースすることにより、施設管理者は施設の来場者に対するサービスレベルを従来より大きく向上させることが期待できる。また、サービス利用者は、本システムの歩行者ナビゲーションアプリを自身のスマートフォンにインストールするだけで様々な施設において施設内情報の提供や経路案内といったサービスを受けることが出来るというメリットがある。

著者は前システムで Android アプリケーション開発の担当を行い、本プロジェクトにおいても、引き続き Android 開発、および新規開発である iOS の開発を担当する。Android と iOS で実装する 2 つのアプリケーションの保守性を、クラス統一やリソース統一によっていかに保つかが開発の焦点となっている。

本報告は本章を含めて全 5 章と付録から成る。第 2 章では本システムの開発背景、2013 年度に開発されたシステムの概要及び課題点とその解決策について述べる。以降は著者が担当する内容についての章である。第 3 章では Android と iOS 両プラットフォームの開発効率を高めた設計と実装方法について述べる。第 4 章では実装したアプリケーションの妥当性を検証するために行った被験者実験と、実験結果から得られたフィードバックによるアプリケーションの改善について述べる。第 5 章ではまとめと今後の課題について述べる。また、付録として開発したアプリケーションのテスト仕様書とテスト結果、著者が行った被験者実験の手順書およびアンケートの様式を添付する。

第2章　開発背景とプロジェクト概要

2.1　歩行者ナビゲーションシステムの現状

スマートフォンの普及に伴い、歩行者向けナビゲーションアプリが急速に普及している。これらの多種多様なナビゲーションアプリは、スマートフォン端末の高い情報処理能力と、コンテンツプロバイダによって提供されている地図情報に支えられている。ナビゲーションアプリは、その用途に応じて様々な種類が存在している。これらは大別して以下の２つに分けられる。

公道や公共交通機関利用時を想定したシステム

１つ目は、利用シーンとして公道や公共交通機関を想定しているサービスである。これらのサービスとして、Google マップや NAVITIME[1]、いつも NAVI[2] などが挙げられる。これらは地図の情報が広い範囲をカバーしており、大規模施設までの道案内や、公共交通機関を使った乗り継ぎ方などを総合的にナビゲーションすることができる。ただし、これらがサポートする範囲はほとんどの場合街中に限られており、大規模施設内の詳しい情報は提供されていない。なぜなら、多くの地図アプリケーションが地図としての機能を提供するために利用している Google Maps API[3] や Apple Maps API[4] などが、大規模施設内の情報を提供していないからである。そのため例えば筑波大学の場合、Google Maps API が提供できる道情報の限界は筑波大学循環バスが走っているループ道路までであり、学内にある一部の建物には名前が表示されない。そのため、筑波大学に初めて訪れた人が徒歩で目的の建物に行きたいと考えても、その建物が敷地内のどこに、どのような道で通っているのかを、Google Maps が提供している情報から読み取ることは出来ない。

施設内部での利用を想定したシステム

２つ目は、特定の施設でのナビゲーションに特化したサービスである。これらのサービスとして、パークの案内図[5]（図 2.3）や東山動物園アプリ[6]（図 2.2）などが挙げられる。これらは前述の問題点を解決しており、遊園地や動物園といった大規模施設が、そ

[1] http://www.navitime.co.jp/
[2] http://www.its-mo.com/
[3] https://developers.google.com/maps/
[4] https://developer.apple.com/maps/
[5] https://itunes.apple.com/jp/app/pakuno-an-nei-tu/id837170889
[6] https://play.google.com/store/apps/details?id=jp.ne.ctv.HigashiyamaZoo

れぞれ独自にナビゲーションアプリを提供している。各施設が提供しているこれらのアプリケーションを利用することで、Google Maps などでは案内出来ないような場所でもナビゲーションが可能になる。また、これらのアプリケーションは各施設に特化した作りとなっているため、例えば遊園地の場合、園内のアトラクションの位置はもちろん、待ち時間やイベントのタイムスケジュールといった仔細な情報提供が可能となっている。しかし、これらのアプリケーションは各施設に特化しているため、汎用性が効かないというデメリットがある。ディズニーランドならパークの案内図、東山動物園なら東山動物園アプリといったように、各施設に訪れた際に、都度アプリケーションをダウンロードし、そのアプリケーションの使い方を学習する必要が生じる。

2.2 顧客が抱える課題と要望

本プロジェクトの顧客は、筑波大学システム情報系情報工学域の和田耕一教授及び山際伸一准教授である。

両名は筑波大学に勤務し、以前から筑波大学への来訪者を誘導する方法に苦慮していた。また、自身も目的地の建物や授業の教室の場所が分からない場合があるという問題を抱えている。

この問題は広大な敷地面積を有する筑波大学では特に顕著に現れており、複雑に入り組んだ道や林立する建物の位置情報は、Google マップや NAVITIME、いつも NAVI などでは提供されていない。そのため、筑波大学の学内における位置情報は、同学が公開しているキャンパスマップを参照する他ない。キャンパスマップは紙媒体や Web ページによって提供されているものの、検索が出来ない、地図を読むことが苦手な利用者が目的地に辿り着くことが困難、といった問題がある。

このような問題から、顧客らから以下の 3 点の要望を受けた。

- 目的地の場所を素早く知ることが出来ること
- 筑波大学を含む様々な大学にも適用できること
- 地図が苦手な利用者でも案内できるよう、AR による案内機能を設けること

これらの課題を解決するためのプロジェクトチームを発足し、IT ソリューションを検討、提供する。

2.3 課題の解決策

先述の課題および顧客らからの要望を受け、本プロジェクトでは次の方針で課題の解決を図ることとした。



図 2.1: Google マップに登録されている建物情報の例



図 2.2: Google マップを利用して筑波大学内で経路検索をした時の例

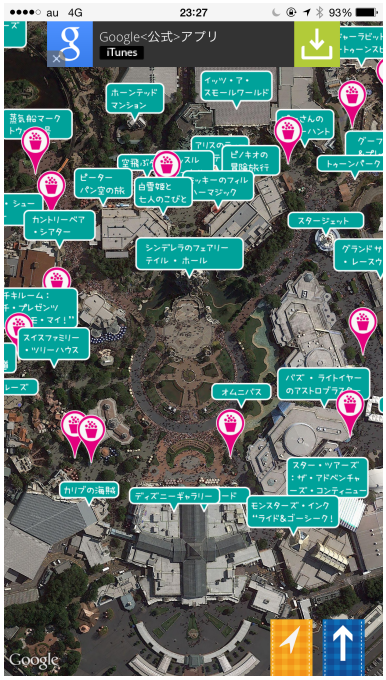


図 2.3: パークの案内図



図 2.4: 東山動物園

学内情報を管理する機能

サービス利用はによる検索の対象となる学内情報を施設管理者が管理するための機能を開発する。

目的地へ案内する機能

目的地と現在地の場所を示すことにより、サービス利用者を目的地へ案内する歩行者ナビゲーションアプリを開発する。さらに、AR を用いることで直感的な案内を可能にする。

学内情報を検索する機能

本システムに登録されている学内情報を歩行者ナビゲーションアプリで検索し情報を得る機能を開発する。

2.4 想定する利用者

施設管理者

施設管理者は、厚生施設や教室など大学の情報を管理する者が該当し、本システムが保持する学内情報を管理する役割を担う。施設管理者には、管理アプリケーションにログインするための認証情報が付与される。この認証情報を持たない限り、本システムの学内情報を更新することは出来ない。認証情報には、本システムに複数登録されている大学のうち、どの大学の管理者であるかを識別する役割もある。施設管理者が登録した学内情報が、サービス利用者が本システムの歩行者向けナビゲーションアプリを介してアクセスする情報となる。

サービス利用者

サービス利用者は、本システムの歩行者向けナビゲーションアプリを利用し、学内の案内を受ける者が該当する。Android がインストールされた端末を持つユーザは、Google Play[7] から本システムの歩行者向けナビゲーションアプリをインストールすることで、サービスを利用することが出来る。特に大学内に慣れていない新入生や来客者など、目的地がどこにあるか把握していない人を対象としている。

2.5 学内ガイドを可能にするナビゲーションシステムの開発

2.5.1 システム構成

本システムの構成を図 2.5 に示す。本システムはクライアント・サーバー型アプリケーションとして実装されている。

[7] https://play.google.com/

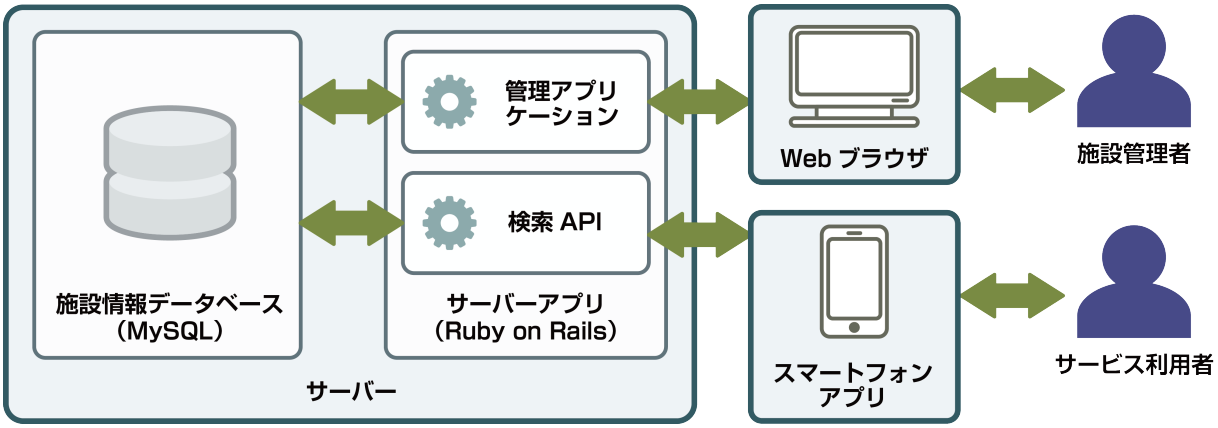


図 2.5: システム構成

クライアント

クライアントはユーザに応じて 2 種類が存在している。

一つ目はサービス利用者向けのスマートフォンアプリケーションである。スマートフォンアプリケーションはサーバーに設けられている検索 API と通信し、目的地の情報を取得する。取得した情報を元にサービス利用者を地図や AR を用いて分かりやすく案内する。

二つ目は施設管理者向けの管理アプリケーションである。管理アプリケーションは本システムが保持している学内情報を簡便に管理するために用意されている。Web アプリケーションとして実装されており、Web ブラウザからアクセスすることで利用が可能である。施設管理者が修正を行うと、サーバーに施設情報の修正処理がリクエストされる。

サーバー

サーバーは大きく分けて施設情報データベースとサーバーアプリケーションの 2 つに分けられる。

施設情報データベースは、本システムの肝となる学内情報や管理者アカウントなどを保持するデータベースで、MySQL が利用されている。データベースはクライアントアプリケーションから直接参照されることはなく、後述のサーバーアプリケーションから参照される。

サーバーアプリケーションは、本システムにおいてサーバーとクライアントとを結び、クライアントからの要求を適切にデータベースに反映、参照する役割を担う。Ruby on Rails アプリケーションとして実装されている。サーバーアプリケーションには大きく別けて検索 API と管理システムの 2 つの機能が用意されている。検索 API は、サービス利用者が利用するクライアントアプリケーションからの検索を受ける機能である。こ

の機能は、検索クエリを元にデータベースから施設情報の検索を行い、クライアントアプリケーションが扱いやすい形でデータを返却する API である。管理システムは、施設管理者が利用する管理アプリケーションからのリクエストを受ける機能である。この機能は、検索 API のようにデータベースの情報を読み取るだけでなく、施設管理者の修正リクエストに応じてデータベースに情報を書き込む役割を持つ。

2.5.2 歩行者ナビゲーションアプリ

本プロジェクトが開発する歩行者ナビゲーションアプリは、利用者に現在地を通知する必要がある。また、インターネットに接続して本システムのサーバーから様々な情報を取得する必要があることから、歩行者向けナビゲーションアプリをスマートフォン向けに実装することに決定した。スマートフォン向けのプラットフォームとして Android や iOS、フレームワークとして PhoneGap などが挙げられる。それぞれのメリット、デメリットを表 2.1 に示す。

各プラットフォームのメリット、デメリットを考慮した上で、開発するプラットフォームを検討した結果、実装の自由度の高さやランニングコストの低さ、処理速度の安定性などから Android で実装することを決定した。

表 2.1: 各プラットフォーム、フレームワークのメリットとデメリット

プラットフォーム	メリット	デメリット
Android	・iOS に比べて実装の制限が少ない	・端末毎に動作が異なる場合がある
iOS	・端末毎で動作の差異がほとんどない	・開発、リリースするための費用が高い ・開発環境が限られる
フレームワーク	メリット	デメリット
PhoneGap	・Android と iOS の両方にコンパイルできる	・カメラなどのハードウェアの機能が遅い ・アプリ自体の処理速度が低い

歩行者ナビゲーションアプリは、顧客の要望を叶えるために、以下の機能を実装した。

1. 施設切り替え機能 本システムでは、施設内を検索するために、利用者がいる施設を切り替える必要がある。施設切り替えボタンをタップすると、データベースに登録されている施設がリストに一覧表示される（図 2.6）。一覧から利用者がいる施設名を選択すると、選択した施設を対象とした施設内情報検索が利用できる。

2. 検索機能 検索ボックスにキーワードを入力し、キーボードの検索ボタンをタップすると、本システムのサーバーからキーワードにマッチする学内情報が検索結果として返却され

る。検索結果は、図 2.7 のように、建物、教室、授業のカテゴリ毎にページ分けされている。各タブをタップすることで、カテゴリを切り替えることが出来る。検索結果一覧には、現在地から目的地候補までの距離と方角が表示される。

3. ナビゲーション機能 検索結果一覧から目的地を選択すると、ナビゲーション画面に切り替わる。ナビゲーション画面は、図 2.8 のように地図上に現在地と目的地のマーカーと、２つのマーカーを結ぶ線が表示される。また、端末を地面と垂直になるように起こすと、図 2.9 のように AR によるナビゲーション画面に切り替わる。この画面では端末が向いている方向を基準とした目的地までの方向と、目的地のマーカーが表示される。端末を再び地面と水平になるように下ろすと、図 2.8 で示す地図画面に切り替わる。

4. ブックマーク、履歴機能 事前に場所を調べたり、同じ場所に再度案内して欲しい場合がある。このような時は図 2.10 に示すブックマーク機能や、図 2.11 に示す履歴機能を利用する。一度ブックマーク登録した場所や、案内を行った場所は、ブックマークや履歴として登録され、以後一覧からすぐに呼び出すことが出来る。



図 2.6: 施設切り替え機能



図 2.7: 検索機能

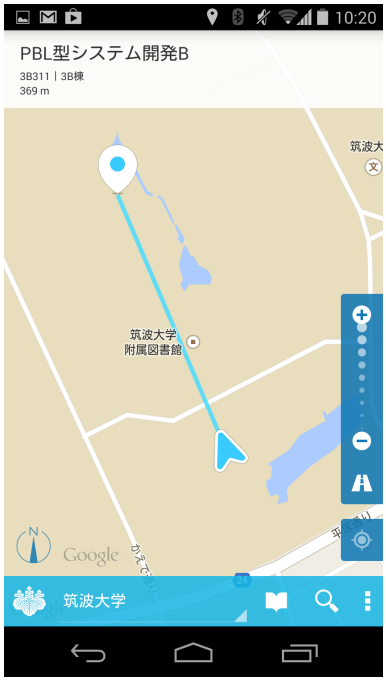


図 2.8: ナビゲーション機能（地図）

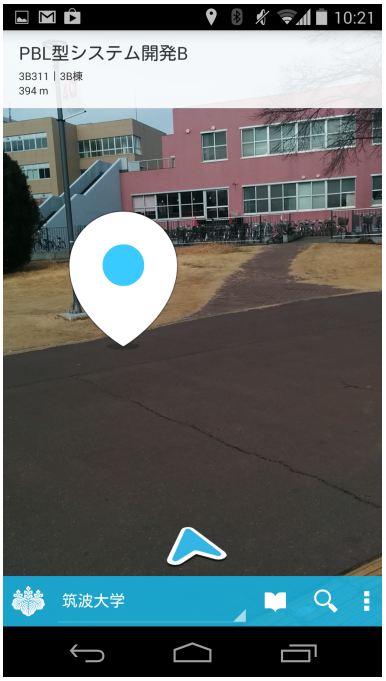


図 2.9: ナビゲーション機能（AR）



図 2.10: ブックマーク機能



図 2.11: 履歴機能

2.5.3　データベース

複数の大学のデータ構造を調査した所、学内の建物や教室、授業などは各校で異なる方法で管理されていた。学内情報は種類ごとに電子データで独自に管理されており、教室データと授業データの 2 種類が存在することが判明した。教室データには、教室の名称やその教室がある建物、教室の広さなどが記載されていた。授業データには、授業の名称やその授業が開講される教室、開講時期などが記載されていた。しかし、経緯度に関する情報はどちらにも記載されていなかった。そのため、学内向け歩行者ナビゲーション用のデータベースを構築する際、経緯度情報は独自に用意する必要がある。

学内情報を検索する機能や目的地へ案内する機能を実現するためには、検索対象とする学内情報と、その情報の位置情報を登録する必要がある。構築するデータベースの構造を決定するために、各校の電子データをさらに調査した結果、大学は図 2.12 に示すように、建物の中に教室があり、教室の中に授業がある、という階層構造となっていることが判明した。本システムで構築するデータベースもこの階層構造を表現するのだが、実際にデータベースで表現する論理構造からエリア、キャンパスに該当する情報は除外することにした。なぜなら、建物や教室を目的地として検索する場合は想定できても、エリアやキャンパス自体を目的地として検索するというユースケースは考えられないためである。

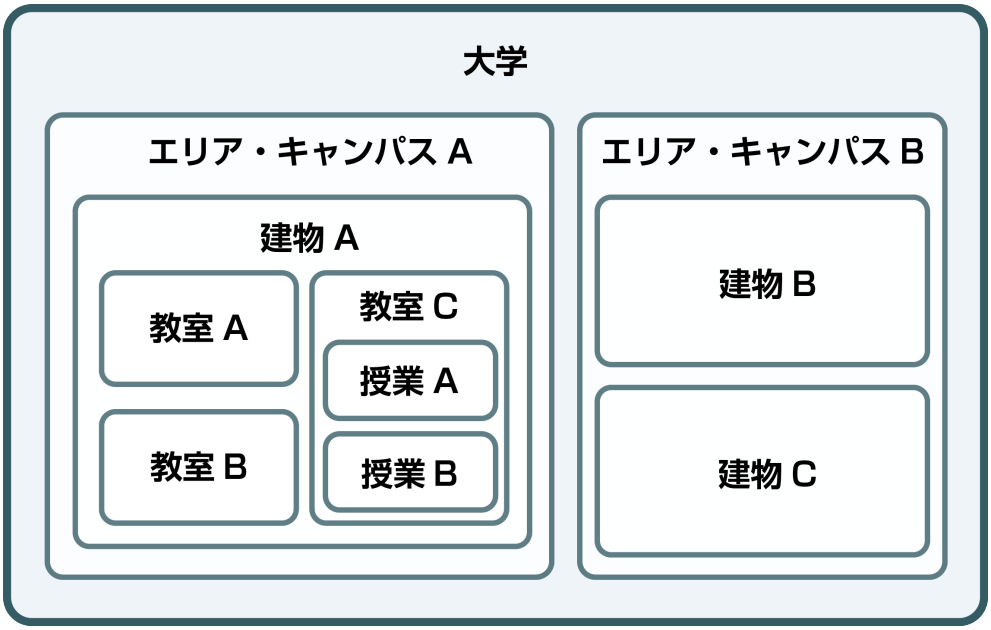


図 2.12: 大学の論理構造

筑波大学向けに登録すべきデータを数えた所、建物、教室、授業の情報は合計で 8000 件近いデータがあることが判明した。これらの情報に利用者がたどり着けるようにするには、各情報の物理的な経緯度を入力しなければならない。しかし、8000 件近いデータの一つ一つに

経緯度を入力していくのは、膨大な時間と労力がかかる。今後、他の大学にも導入を行う場合に障害になることが予想された。そこで、データベースの論理構造を図 2.13 のようにすることで、経緯度の入力されていない情報の経緯度を、論理的に親となる情報の経緯度から補完出来るようにした。

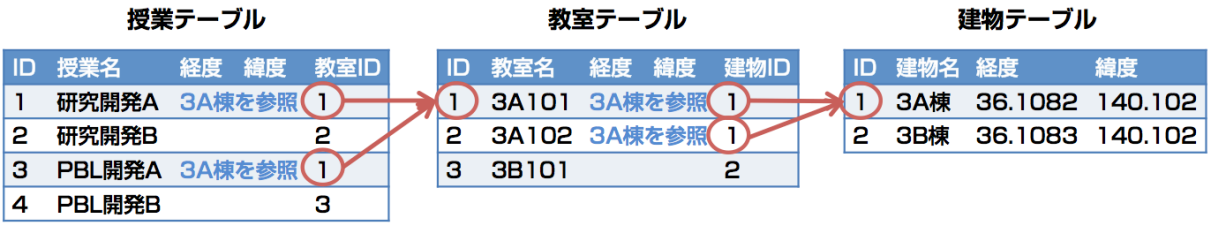


図 2.13: 学内情報を保持するためのデータベーススキーマ

筑波大学だけでなく、他大学にも対応できるようにするためには、以下の機能の開発も必要となる。

大学情報

建物や教室、授業のテーブルに複数の大学の情報を登録すると、その情報がどの大学のものであるのかがわからなくなり、利便性が低下する。筑波大学にいる場合は筑波大学の情報、といったように大学を絞り込んで検索が出来るようにする必要がある。そこで、大学情報を登録し、建物や教室といった情報にひもづけることで、大学毎に検索を行うことができる。

管理者アカウント

複数の大学で導入する場合、施設管理者が所属する大学の情報のみを編集対象とする必要がある。また、本システムの管理アプリケーションにおいて、管理者アカウントの認証情報を持つもののみが、自身の大学の情報を編集できるようにする必要が有る。

アクセス制限

大学によっては、建物や教室、授業を管理する担当者が別々になっている場合がある。このような場合、建物管理者は建物情報のみを管理出来るようにする必要がある。管理者アカウントに対し、そのアカウントが編集できる範囲を制限するために管理者権限機能が必要である。

前システムが完成した後、運用試験を実施した。その際、サービス利用者がどのようにサービスを利用したかを記録することで、サービス改善の糸口をつかめると考えた。そこで、データベースにログを保存するテーブルと、ユーザーのアクションをログに保存する機能を追加した。記録する利用ログは表 2.2 の項目である。

表 2.2: 利用ログの項目と概要

項目	概要
検索ワード	サービス利用者が検索しているワードの傾向を把握することが出来る
IP アドレス	利用ログの内、利用者別に行動を絞り込むことが出来る
検索日時	利用されている時間帯を把握することが出来る
大学	大学毎の本システムの利用頻度を把握することが出来る
その他	OS、デバイス名、カメラの有無、解像度を取得する

2.5.4 管理アプリケーション

学内情報データベースは MySQL で管理されている。MySQL で管理されている情報を修正する場合、一般的な方法ではコマンドラインから MySQL サーバーに接続し、SQL クエリを発行することで中の情報を書き換える。しかし、コマンドラインでの施設情報の修正は容易ではない。そこで、施設管理者の負担を減らすため、Web 上のインターフェースから簡便に学内情報を修正することが出来る管理アプリケーションを開発する必要がある。開発した機能は以下の４つである。

施設管理者の管理機能

管理アプリケーションを使って施設情報を修正することが出来る施設管理者の管理を行う。ログイン名やパスワード、所属する大学などの情報を編集することが出来る。

大学の管理機能

歩行者ナビゲーションアプリが利用できる大学を管理する。本システムを新たに大学に導入する際に利用する。

学内情報の管理機能

建物や教室、授業といった学内情報の登録、修正を行う。名称や経緯度、論理構造上で親となる情報などを、入力フォームから簡単に編集することが出来る。また、建物や授業などの電子データで存在している情報をファイルからインポートすることも可能である。大量に存在するデータを１つずつ手入力する作業を、インポート機能によって一括で行う。

検索ログの閲覧機能

歩行者ナビゲーションアプリから送信されたキーワード検索のログを閲覧することが出来る。ログを分析することで、アプリケーションの改善などに役立てる。

2.5.5 課題と解決策

前システムの完成後、運用試験及びシステムの評価実験を実施した所、４つの課題が挙がった。以下に課題と解決策について述べる。

1. 歩行者ナビゲーションアプリのプラットフォームの制約

前システムで提供しているスマートフォンアプリケーションは Android 向けのみであった。しかし、日本国内におけるスマートフォン向けプラットフォームの普及率は、iOS が Android に次ぐ大きさとなっており、昨今 iOS の普及率が更に増加している。このような背景から、Android のみの対応では不十分であり、Android と iOS の両プラットフォームの対応が必要であると言える。両プラットフォームに対応することで、多くのサービス利用者に本サービスを提供することが出来る。

2. 施設内情報を保持するデータベーススキーマに柔軟性がない

前システムでは、大学を対象としたモデル設計を行った結果、3 階層固定のデータベーススキーマとなった。今年度は、大学以外の大規模施設への対応も視野に入れている。しかし、前システムのデータベーススキーマはデータの階層構造が 3 階層に固定されており、施設の構造が 3 階層以外の施設に導入する際には不都合が生じる。そのため、データベーススキーマを見直し、導入する施設の論理構造に柔軟に対応可能なモデルを新たに構築する必要がある。

3. 正式名称でしか検索が行えない

前システムでは、施設内の情報を検索する際、検索キーワードがデータベースに登録されている名前に部分一致、または完全一致している場所情報を検索していた。しかし、昨年度の運用試験の後に行った利用ログの分析により、学内情報の別称や略称、関連するワードによる検索ログが多いことが判明した。前システムの検索方法では、これらの検索キーワードではヒットしない。本システムが想定している利用者は学内に詳しくないユーザを対象としているため、このような制約は実際の利用シーンでは障害となる。そのため、学内情報に関連する略称や関連するワードなど、施設内の情報の名前以外のキーワードから検索できるようにする必要がある。

4. 経路案内が行えない

前システムでは、利用者を目的地まで案内する画面において、地図上の現在地と目的地にマーカーを立て、その間を線で結ぶという方法で案内を行った。しかし、システムの評価実験の結果、筑波大学のように大規模な施設では通路構成が複雑なため、前システムの案内機能のみでは歩行者ナビゲーションアプリとして不十分であることが判明した。そのため、目的地までの歩行経路を表示することにより、詳細な案内機能を提供する必要がある。

2.6 施設内経路探索と施設内検索を可能にするナビゲーションシステムの開発

本プロジェクトで、顧客は新たに大学以外の大規模施設への対応を要望している。また、前システムの問題点を解決することで、より利便性の高い歩行者ナビゲーションアプリを提供する。

2.6.1 機能

1. 施設情報の管理

施設情報の管理機能は、施設内情報や施設内道情報などの情報を登録、編集する機能である。大学以外の施設に対応するために、前年度設計したデータベーススキーマを再設計し、管理アプリケーションも修正する必要がある。以下に、開発する機能と概要について述べる。

施設内情報

施設内情報とは、前システムの建物、教室、授業情報に該当する。前システムは大学を対象としたシステムであったため、保持する施設内情報も大学に特化したものとなっていた。本プロジェクトでは、大学以外にも対応するために様々な施設に対応出来る汎用的なデータベーススキーマと、新しいスキーマに合わせた管理アプリケーションを開発する。

カテゴリ

前システムでは、教室、授業といった階層毎にデータベースのテーブルが分かれていたため、各階層の名前を表現することが出来た。しかし、本プロジェクトでは様々な施設に対応するため、階層の名前を施設に応じて変更しなければならない。そこで、施設ごとに各階層の名前を設定するテーブルを設け、施設に応じた階層名を表現できるようにする。

道情報

経路案内を実現するためには、施設内をどのように道が通っているかを保持する必要がある。道情報のデータ構造は、カーナビゲーションシステムで利用される道情報ネットワークのようなグラフ構造で表現されており、交差点などの分岐点を表すノードと、ノード同士を道がどのように繋がっているかを表すエッジによって構成される。本システムの道情報は、経路探索を行うためにノードには表 2.3 に示す役割が存在し、探索精度を向上する目的でエッジには表 2.4 に示す属性が付加出来る。これらのグラフ構造を編集するためのインターフェースが管理アプリケーションに用意されており、地図上に表示されるノードやエッジをマウス操作によって簡単に追加、編集することが出来る。

表 2.3: ノードの種類

ノードの種類	役割
施設出入口	大規模施設に入る際に必ず通過するノード。校門などが該当する
交差点	エッジが 3 本以上接続されているノード
カーブ	通路の変曲点となるノード
建物出入口	経路案内の際に目的地となるノード

表 2.4: エッジの属性

エッジの属性	概要
舗装路 / 未舗装路	未舗装路のような歩きにくい道か否か
主要路 / 非主要路	人の往来の多い道か否か
階段あり / なし	階段を含むか否か

2. 施設情報の検索

前システムでは、検索キーワードがデータベースに登録している名称（以下、正式名称と呼ぶ）に完全一致、または部分一致している情報しか検索結果が得られなかった。この問題に対し、本システムでは施設内情報にタグ情報を付加し、タグからもキーワード検索を行うことで、正式名称以外のワードでも目的の情報を得ることを可能とする。タグ情報とは、サービス利用者が検索する可能性の高いキーワードを、関連する施設内情報に紐付けることで利用する。

3. 経路案内

経路案内機能は歩行者ナビゲーションアプリでユーザを目的地まで案内する際に、従来のような目的地の位置だけでなく、案内を開始した位置から目的地までの道順を地図上に表示する機能である。AR 表示においても目的地までの経路を AR を用いて三次元で指し示すことで、地図を読むことが苦手なユーザでも直感的に分かりやすく辿り着くことが出来る。

経路の種類は利用者のニーズに応じて 4 種類の組み合わせが用意されている。1 つ目は最短経路で、この経路は経路探索の計算で現在地から目的地までのエッジの総距離が最も短くなる経路である。2 つ目は推奨経度で、この経路はエッジに割り当てられた属性（表 2.4）を加味した、歩きやすい経路が選択される。前述の 2 つの経路は、経路の中から階段を除外して計算することも出来る。この「階段を含まない」オプションは、階段の歩行が難しいユーザのために用意されている。ユーザは、以上の 4 つの組み合わせの中から好みの経路を選択し、経路案内のサービスを受けることが出来る。

4. 道情報の自動登録

経路案内を実現するには、施設内の道をグラフ構造で表現したデータとして一から構築しなければならない。道情報の登録は、通路構成と建物の出入口の位置を現地調査した後、管理アプリケーションを用いて行う必要がある。しかし、筑波大学のように広大な敷地を持つ施設では、通路構成が複雑かつ建物の出入口も多いため、道情報の登録に膨大な手間が掛かる。そこで、道を現地調査する際に歩いた軌跡を、GPS を用いて記録するアプリケーションを用いることで、道情報を登録を補助する機能を提供する。登録の際、GPS ログの誤差から出るノイズを、独自のアルゴリズムに基づいて除去しグラフ構造に変換することで、自動的にデータベースに登録することが可能となる。

2.6.2 システム構成

本システムは前システムと同様に、クライアント・サーバ型のアプリケーションである。本システムの構成を図 2.14 に示す。

クライアントは、歩行者ナビゲーションアプリと管理アプリケーション、道情報を自動的に登録するための GPS 情報を収集する GPS ロガーから構成される。歩行者ナビゲーションアプリのプラットフォームは、Android と iOS に対応する。管理アプリケーションは、新たに施設内情報管理画面と施設内道情報管理画面が追加される。

サーバは、データベースと管理アプリケーション向けの制御プログラム、歩行者ナビゲーション用の検索 API、道情報を自動登録するプログラムから構成される。DBMS には MySQL と PostgreSQL を併用している。施設情報やタグ、類語辞書は MySQL に、道情報は PostgreSQL 向けの経路探索ライブラリである pgRouting[8] を用いるために同データベースに格納する。

2.6.3 開発体制とスケジュール

本プロジェクトでは、前システムの問題点と顧客からの新たな要望を実現するために、新たな歩行者ナビゲーションシステムを開発する。開発チームとして、本プログラム所属の学生 4 名でチームを発足した。本プロジェクトの顧客兼課題担当教員は、和田耕一教授と山際伸一准教授であり、システムに対する意見やプロジェクトに対する指導をいただいた。各開発項目と担当者を表 2.5 に示す。

次に全体スケジュールについて述べる。本プロジェクトでは、表 2.5 に示すように各者の担当範囲を分担し、各担当者がスケジュールに沿って開発を進めていく（図 2.15）。各機能は要件定義、開発、テストの順に進めていく。すべての機能の開発及びテストは 11 月の中間発表までに終了し、11 月の中間発表後からシステムの評価実験を実施した。各機能はチケットという単位に分割し、チケット管理システムである Redmine を利用して進捗の管理を行った。

[8] http://pgrouting.org/

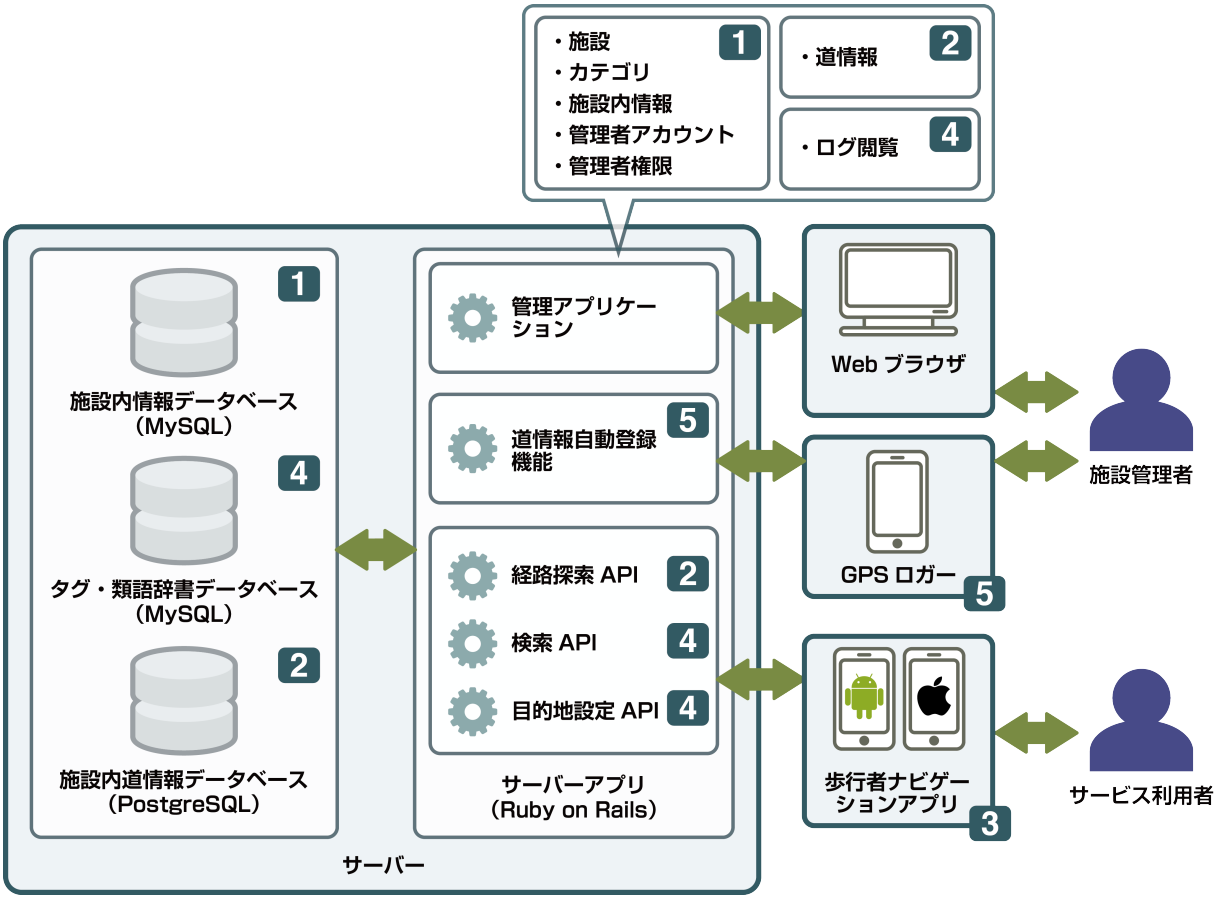


図 2.14: 本システムの全体構成

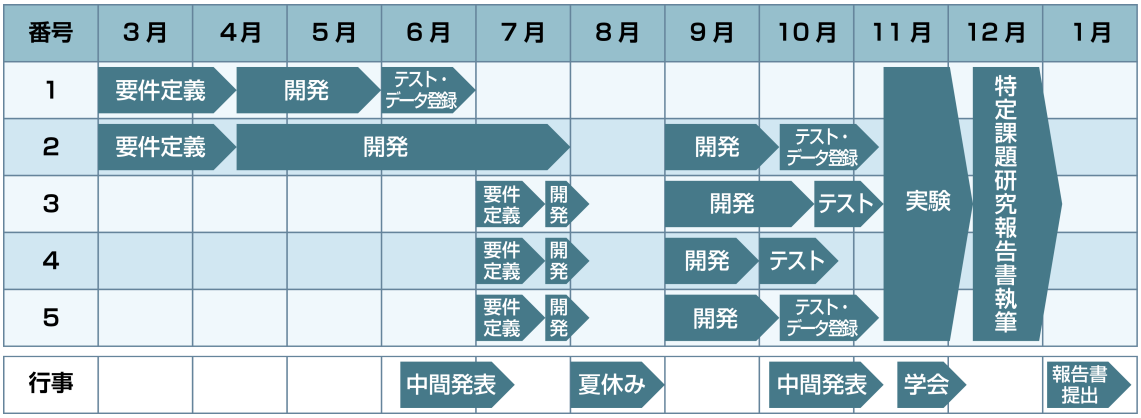


図 2.15: 全体スケジュール

表 2.5: 各開発項目の機能一覧

番号	機能	担当	開発項目
1	施設内情報管理機能	畑中裕太 芳賀隼人	データベース（施設内情報データベース）
			管理アプリケーション（管理者アカウント、管理者権限、施設、カテゴリ、施設内情報）
2	施設内道情報管理機能 経路探索機能	千田隼人	データベース（施設内情報データベース）
			管理アプリケーション（道情報）
			通信 API（経路探索 API）
3	歩行者ナビゲーションアプリ	萬成亮太	歩行者ナビゲーションアプリ（Android、iOS）
4	施設内情報検索機能	畑中裕太	データベース（タグ・類語辞典データベース）
			管理アプリケーション（ログ閲覧）
			通信 API（検索 API、目的地設定 API）
5	道情報自動登録機能	芳賀隼人	GPS ロガー
			道情報自動登録機能

第3章　複数のモバイルプラットフォーム向けの実装方式の開発

3.1　実装方式の開発手順

モバイル向けアプリケーションの開発方法はいくつか存在する。大きく分けて以下2つに大別される。

1. モバイルフレームワークを利用する方法

この方法は、PhoneGap[1] や Titanium Mobile[2] といったモバイルフレームワークを利用する方法である。モバイルフレームワークを利用すると、各プラットフォーム上で動作するバイナリをコンパイルすることが出来る。１つのソースコードを書くだけで複数のプラットフォームに対応できる利点がある。一方、プラットフォームを跨いでプログラムを実装するため、各プラットフォームに特有の機能が利用できないことがあるという欠点もある。

2. ネイティブで開発する方法

この方法は、Android 向けに Java、iOS 向けに Objective-C を用いて開発を行う方法である。この方法は各プラットフォームの標準的な開発方法であるため、前述の方法と違い、プラットフォームのすべての機能を利用することが出来る。しかし、プラットフォーム間で API や開発形体は異なっており、ソースコードを再利用することは出来ない。結果、プラットフォームの数だけアプリケーションを実装する必要があるという欠点がある。

これらの内、どちらの開発方法を取るか検討した結果、“2. ネイティブで開発する方法”を採用することに決定した。“1. モバイルフレームワークを利用する方法”は、前システムの初期の実装で利用した経験があり、その際に、

- 地図の描画が遅い
- カメラが正常に動作しない
- センサーの値が取りづらい

[1] http://phonegap.com/
[2] http://www.appcelerator.com/titanium/

といった問題が起こったためである。本システムでも地図やカメラなどを利用予定であるため、ネイティブで実装することに決定した。

一方、“2. ネイティブで開発する方法”で Android と iOS 向けのアプリケーションを開発した場合、単一のプラットフォームと比較して、

- 設計工数が 2 倍になる
- アプリケーションの実装工数が 2 倍になる

という問題が起きる。

このような問題から、本システムの歩行者向けナビゲーションアプリの開発では、ネイティブで実装をしつつ、モバイルフレームワークのように複数プラットフォーム間でリソースを共有する効率化を図った。この効率化を行うために取った工夫を “実装方式”と呼ぶ。二つのプラットフォームにおけるアプリケーションの開発を通して得た知見を実装方式としてまとめた。次節以降では、本開発で実際に使用した実装方式の開発方法を述べる。

3.1.1 アプリケーションの開発方針

クラス構造の統一

本方式ではまずクラス構造の統一を図る。Android と iOS をネイティブで開発すると、単一プラットフォームの場合と比べ以下の様な事が起こる。

- 内部設計を 2 度行う必要がある
- 実装を 2 度行う必要がある
- 修正の際、ソースコードを 2 度修正する必要がある

このようにすべての作業工程が 2 倍になってしまう。

しかし、クラス構造を統一することで、作業工程の肥大化をある程度抑えることが可能である。なぜならクラス構造が同一の 2 つのプログラムは、プログラミング言語は異なるが表現するプログラムの処理は同一であり、2 度の実装作業で近いプログラムを書くことが出来るためである。また、内容の近い 2 つのプログラムは修正の際も、同様に作業を執り行うことが出来るためコストも低く抑えることが出来る。

しかしながら、Android と iOS ではプラットフォームが提供しているクラスが異なっており、そのままではクラス構造を統一することは出来ない。そこで、クラスの役割に応じて Android と iOS のクラスを対応付け、クラス構造を統一して表現できるよう “クラス対応表”を作成する。クラス対応表を元に内部設計を行うことで、クラス構造を統一することが可能である。

画像リソースの効率的な生成

画像リソースとは、アプリケーションのインターフェースを構成する画像のことである。例えば画面内に星を描画する場合、プログラムによって星形を描画するのではなく、星が描かれた画像を画像リソースとしてインターフェース内に読み込むことで、簡単にインターフェースをデザインすることが出来る。Android、iOS 共に画像リソースという概念が存在し、インターフェースを作る時はどちらのプラットフォームでも同じ中身の画像を再利用することが出来る。

スマートフォン端末の処理能力は年々進歩しており、それに合わせて解像度も高いものが増えてきている。端末の画面サイズは同じでも、ピクセル数は様々である。1 インチあたりのピクセル数のことを dpi と呼び、dpi の大きさによって Android、iOS 共に分類分けされている（表 3.1）。

表 3.1: dpi による解像度の分類

OS	基準となる dpi	解像度名
Android	〜120dpi	ldpi
Android	〜160dpi	mdpi
Android	〜240dpi	hdpi
Android	〜320dpi	xhdpi
Android	〜480dpi	xxhdpi
Android	〜640dpi	xxxhdpi
iOS	163dpi	@1x
iOS	326dpi	@2x
iOS	401dpi	@3x

画像リソースは分類された解像度に適した大きさで作成する必要がある。例えば横 8cm、縦 12cm の画面に縦横 6cm で画像を表示する場合、図 3.1 のように画面のピクセル数が横 16px、縦 24px の場合は一辺が 12px の画像で済むが、図 3.2 のように画面のピクセル数が横 32px、縦 48px の場合は一辺の長さを 24px で用意しなければならない。Android、iOS 共に解像度毎に大きさの異なる画像リソースを用意することが推奨されている [7][8]。解像度の種類は Android では 6 種類、iOS では 3 種類があり、一枚の画像に付き合計 9 種類の解像度の画像を書き出さなければならない。さらに、Android と iOS では書き出した画像の保存先やファイルの命名に規則があり、それぞれの規則に則る必要がある。

この作業はすべての画像に対して実施しなければならず、10 枚の画像リソースでも 90 枚の画像を書き出す必要がある。そこで開発の効率を高めるため、“画像変換ツール”というバッチ処理ツールを開発して運用する。

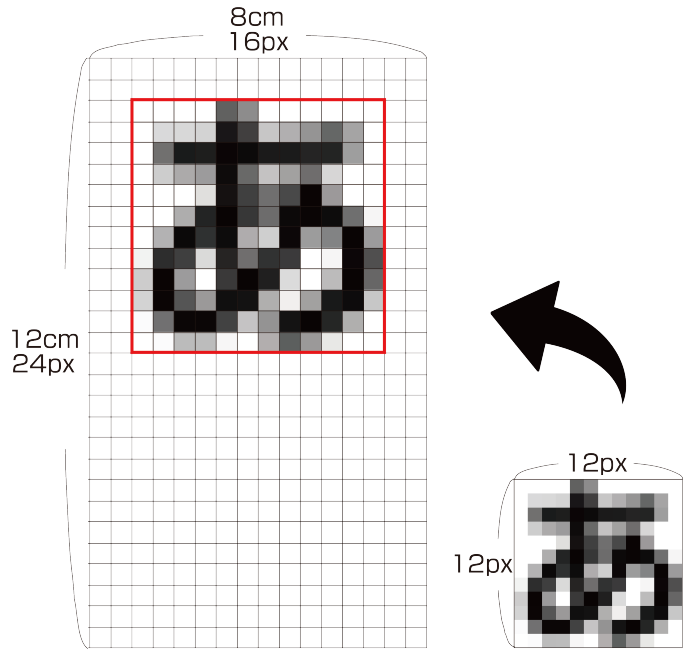


図 3.1: 標準解像度の端末向けの画像リソースと表示結果

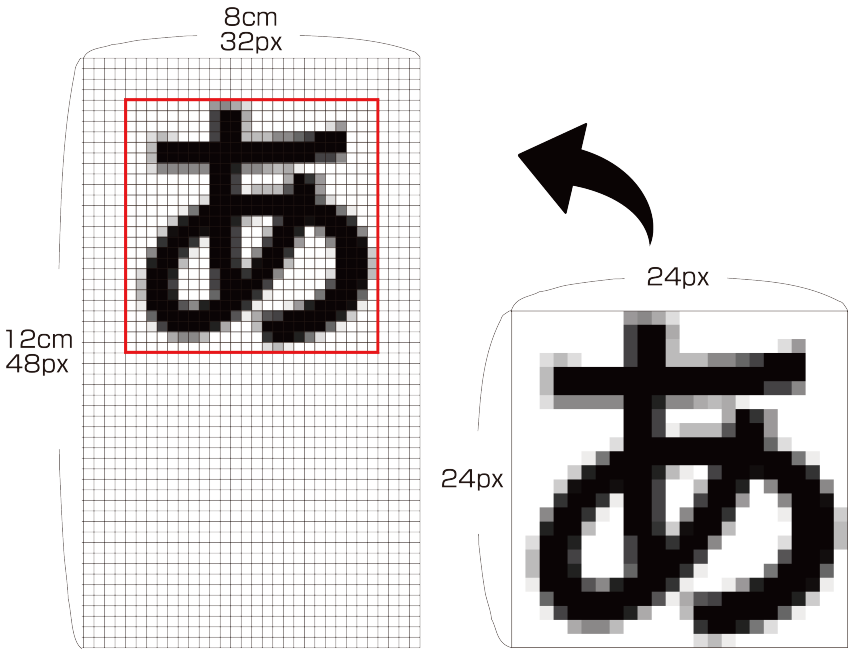


図 3.2: 4 倍解像度の端末向けの画像リソースと表示結果

3.1.2 クラス対応表の作成

本プロジェクトでは、2 つのプラットフォームの実装を近づけることで保守性の向上を目指す。調査の結果、2 つのプラットフォームは、近い概念を持つクラスが提供されていることが判明した [2]。そこで、プラットフォーム間のクラスの対応付けと、どちらか一方のプラットフォームにしか存在しないクラスの補完方法について事前に調査した。

クラスの対応付けを行う際、ソースコードを見直した時にクラスの役割が分かりやすいよう、クラスを継承した場合のクラスの命名規則を統一しておく。また、利用予定のクラスがどちらかのプラットフォームでしか提供されていない場合は、もう一方のプラットフォームで代用する方法も予め決めておく。2 つのプラットフォームについて表 3.2 のようなデザインコンセプトを事前に定義しておくことで、後の開発を円滑に進めることが出来る。

表 3.2: クラスの対応表

役割	Android	iOS
画面	Activity	UIViewController
画面内の画面	Fragment	UIViewController
タブ	ViewPager	UIPageViewController
カスタムビュー	View	UIView
テキストビュー	TextView	UILabel
テキスト入力	EditText	UITextField
ボタン	Button	UIButton
画像ボタン	ImageButton	UIButton
リストビュー	ListView	UITableView
Web ビュー	WebView	UIWebView
リストダイアログ	Dialog + ListView	UIActionSheet
配列	ArrayList	NSArray
辞書	TreeMap	NSDictionary
恒久保持情報	PreferenceManager	NSUserDefaults

3.1.3 画像変換ツールの開発

スマートフォンアプリケーションの実装に入る前に、開発を円滑に進めるための画像変換ツールを実装した。本ツールは今後、他のプロジェクトでも役立てたいと考えていたため、様々なプラットフォームで実行することが望ましい。そこで、本プロジェクトではこのツールを Node.js[3] で実装することにした。Node.js は Mac OS X や Linux、Windows 上で動作する

[3] http://nodejs.org/

ため、プラットフォームに依存すること無く利用することが出来る。新しい環境で本ツールを利用する場合は、本ツールをプロジェクトフォルダに配置し、Node.js 向けに提供されている各種モジュールをインストールするだけで導入することが出来る。各種モジュールのインストールは、本ツールに付属するモジュールの一覧を元に、コマンドラインから一括で行うことが可能である。

画像の変換

各プラットフォームにおける画像リソースの拡大率は表 3.3 のとおりである。

画像を書き出す際は、解像度が近い mdpi と@1x 向けに作成された画像リソースを基準として、表 3.3 に記載された拡大率の画像を作成し、プラットフォームの規則通りの場所、ファイル名に書き出す。画像変換ツールは当初、基準である mdpi や@1x 向けに作成された画像リソースを各解像度向けに変換する予定であったが、例えば xhdpi 用の画像を書き出す場合、mdpi 向けに作成された画像リソースを縦横 2 倍に拡大することになる。画像を拡大すると、図 3.3 のようにぼやけた画像になってしまい、最適な結果が得られないという問題があった。この問題を解決するため、設定ファイルを用いた基準サイズの直接的な指定に対応した。mdpi や@1x の画像サイズを基準とするのではなく、設定ファイルに記述された縦横のピクセルサイズを基準として各解像度の画像を書き出す方法である。画像変換ツールは画像を変換する際、この設定ファイルを参照して各画像リソースが mdpi や@1x の解像度においてどのくらいのサイズで表示されるかを認識し、書き出しを行う。変換元となる画像サイズを十分に大きなサイズで用意することにより、すべての画像サイズで最良な品質を得ることが出来る（図 3.4）。

表 3.3: 画像リソースの拡大率と書き出し方法

OS	解像度名	サイズ	書き出し方法
Android	ldpi	基準 x0.75	res/drawable-ldpi/ に書き出し
Android	mdpi	基準	res/drawable-mdpi/ に書き出し
Android	hdpi	基準 x1.5	res/drawable-hdpi/ に書き出し
Android	xhdpi	基準 x2	res/drawable-xhdpi/ に書き出し
Android	xxhdpi	基準 x3	res/drawable-xxhdpi/ に書き出し
Android	xxxhdpi	基準 x4	res/drawable-xxxhdpi/ に書き出し
iOS	@1x	基準	任意の場所に書き出し
iOS	@2x	基準 x2	任意の場所に「(ファイル名)@2x.(拡張子)」で書き出し
iOS	@3x	基準 x3	任意の場所に「(ファイル名)@3x.(拡張子)」で書き出し

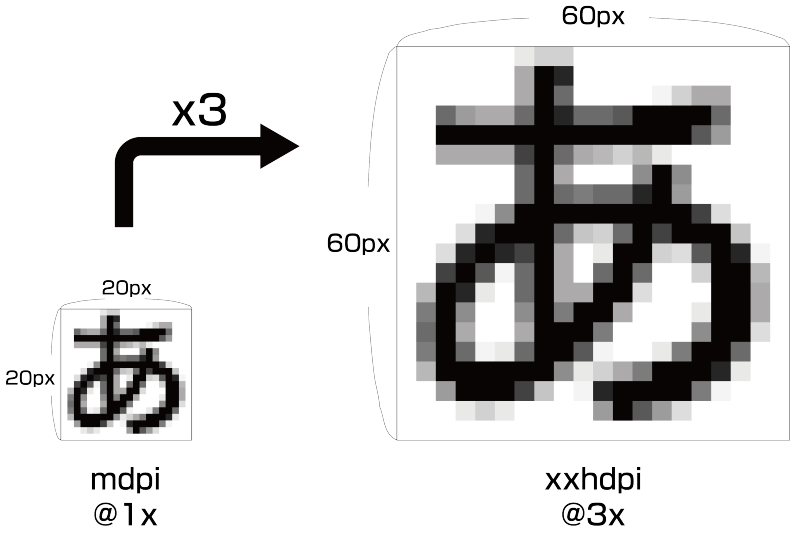


図 3.3: mdpi、@1x を基準とした変換（アップコンバート）

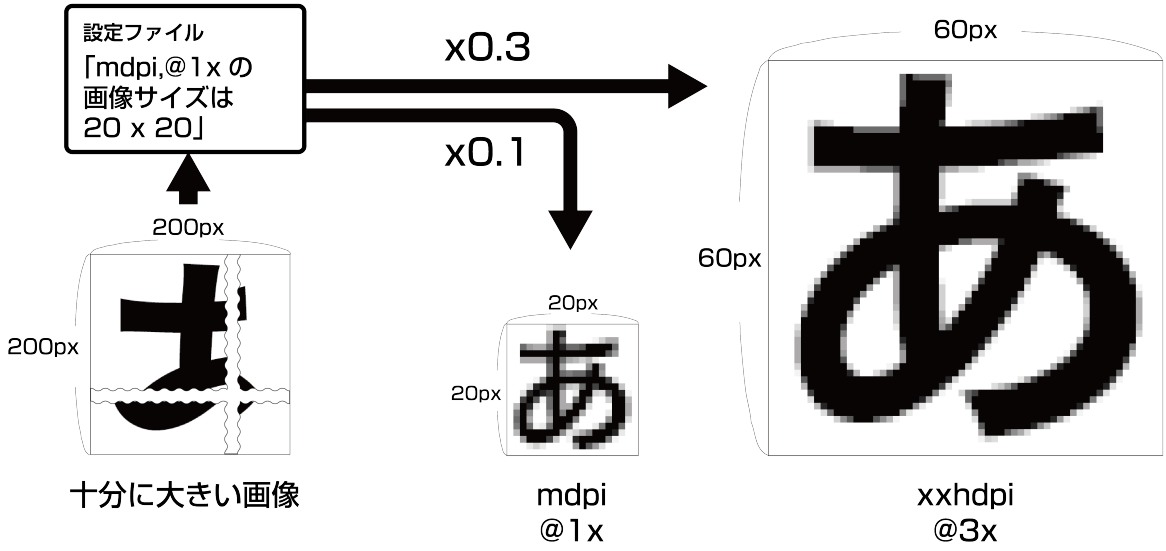


図 3.4: 十分に大きい画像を基準とした変換（ダウンコンバート）

9-Patch

9-Patch とは Android がサポートしている、画像リソースをリサイズした時に画像の特定領域の形状を保持する機能である。角丸（図 3.5）や影付きの画像のサイズを変えると、画像内の一部のデザインが崩れてしまうことがある。9-Patch を用いると、これらの画像をリサイズしても最適な結果が得られる。

元の画像を 9-Patch に対応させるには、画像の周囲を 1 ピクセルずつ拡張する。拡張したピクセルのうち、スケールを破壊してもよい行また列を図 3.6 のように黒色で塗る（以後、マーカーと呼ぶ）。マーカーがある列と行ピクセル（図 3.6 の青色部分）はリサイズした際に引き伸ばされ、それ以外のピクセルは形状が守られる。9-Patch の画像をリサイズする際、画像内にあるマーカーから引き伸ばすピクセルの列と行を読み取り、画像をリサイズするため、マーカーが重要な意味を持つ。

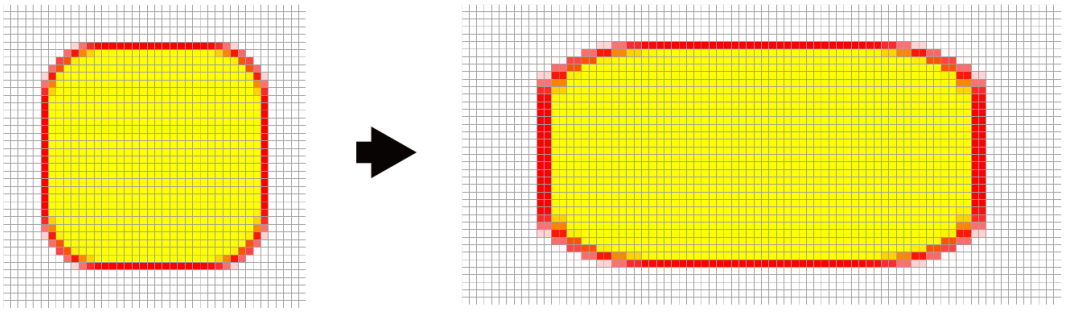


図 3.5: 9-Patch なしで横方向にスケールした例

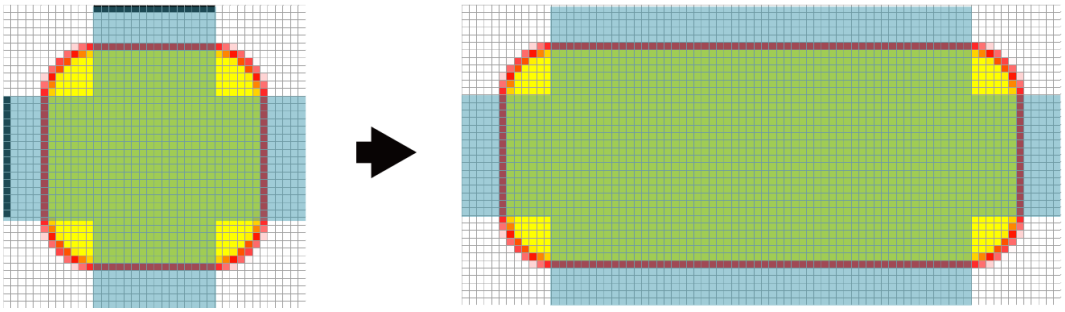


図 3.6: 9-Patch ありで横方向にスケールした例

ここで画像変換ツールに話を戻す。画像変換ツールでは元画像をプラットフォーム毎に様々な画像サイズで書き出す。この時、9-Patch 化された画像（図 3.7）を通常の方法でリサイズすると、リサイズ時の補完の影響によって図 3.8 のようにマーカーが滲んでしまう。マーカーは不透明な黒色で塗られている必要があるため、滲んだマーカーは 9-Patch の誤動作を引き起こす。

従来、画像変換ツールを使わない方法での 9-Patch 画像のリサイズは、リサイズ後に手作業でマーカーの滲みを修復しなければならなかった。一方、本プロジェクトで開発した画像変換ツールでリサイズを行うと、画像の周囲 1 ピクセルを収縮してマーカーを一旦削除し、リサイズ後に画像の周囲 1 ピクセルを拡張してマーカーを復元する。この手順により、9-Patch 画像のマーカーを滲ませること無くリサイズすることが出来る（図 3.9）。

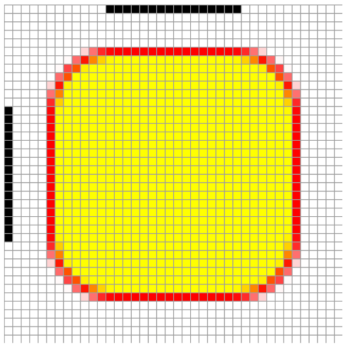


図 3.7: 縮小前の 9-Patch 画像

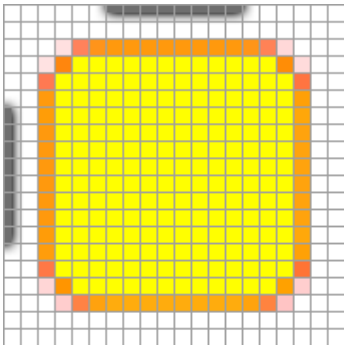


図 3.8: 通常のリサイズ

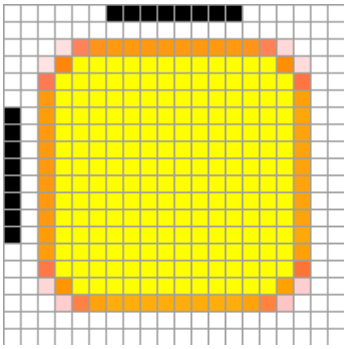


図 3.9: 画像変換ツールのリサイズ

3.2 ナビゲーションシステムの開発

3.2.1 二つのプラットフォームにおけるアプリケーションの実装手順

スマートフォンアプリケーションの実装は、Android と iOS において似た構造のプログラムに成るように実装する必要がある。そのために、以下の手順で実装を行った。

実装と移植

２つのプラットフォームのうち、最初にどちらか一方のプラットフォームに的を絞り、内部設計（図 3.10）に従ってアプリケーションをすべて実装する。この時、もう一方のプラットフォームでどのように実装するかを考慮し、出来るだけ処理の流れが一致するように注意する。今回はプラットフォームによる制限が多い iOS を先に実装した。

iOS 向けのアプリケーションを実装した後、Android 向けのアプリケーションを、iOS から移植する形で実装する。iOS 向けにすでに完成されたアプリケーションが存在するため、iOS から Android 向けにプログラムを翻訳する作業となる。プラットフォームの差異から来るプログラム構造の違いは、次の修正作業で埋めていく。

プラットフォーム間の差異を修正

プラットフォームが提供している API を通常の方法で扱うと、API の思想の違いから、構造に差が生じることがある。これらの差は、後にプログラムの規模が大きくなった時、

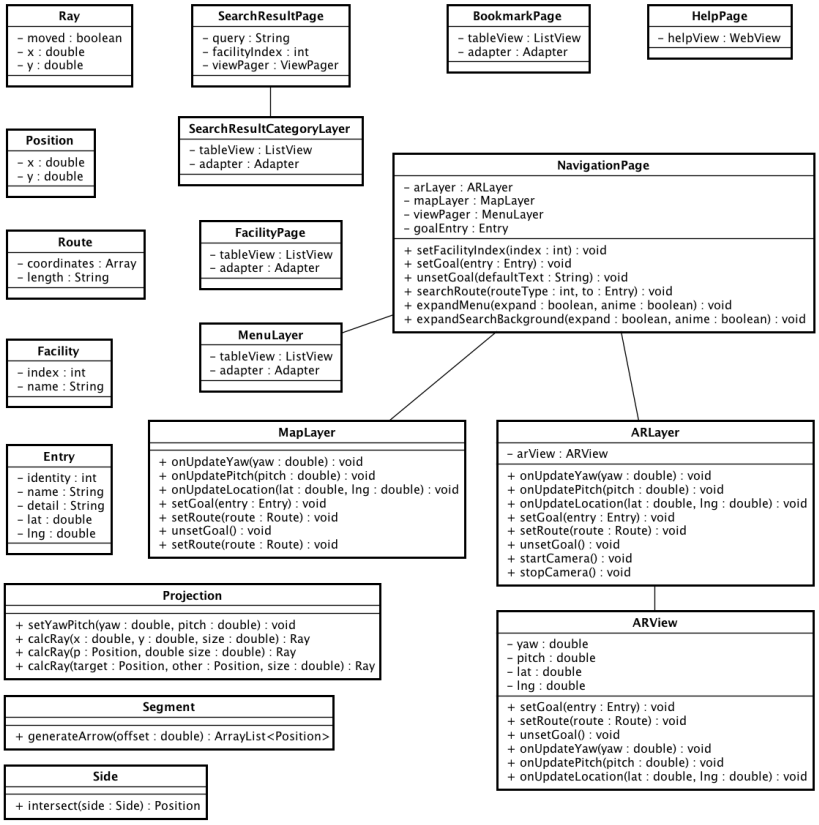


図 3.10: 本システムの内部設計

大きな差に膨れ上がる可能性がある。この問題を、どちらかのプラットフォームでもう一方のプラットフォームが提供している API を再現することで解決する。

例えば、Android と iOS にはどちらも表を実装するためのクラスが提供されている。Android と iOS の一般的な方法で表を実装すると、図 3.11 のように Android では表が Adapter のサブクラスを参照し、このサブクラスで行の表示方法やデータを定義するのに対し、iOS では表が任意のクラス（図 3.11 では UIViewController）を参照し、そのクラスに実装されている行の表示方法やデータを間接的に参照する、という差が生じる。

そこで、iOS のプログラムで Android の Adapter を再現する。Android では表を扱う際、必ず Adapter というクラスを用いる必要がある。このようなクラスは iOS には存在しないが、Adapter クラスを再現したクラスを用意することで、Android と同様の方法で表を実現することが出来る（図 3.12）。このような修正によって、２つのアプリケーションのプログラム構造を一致させることが出来る。

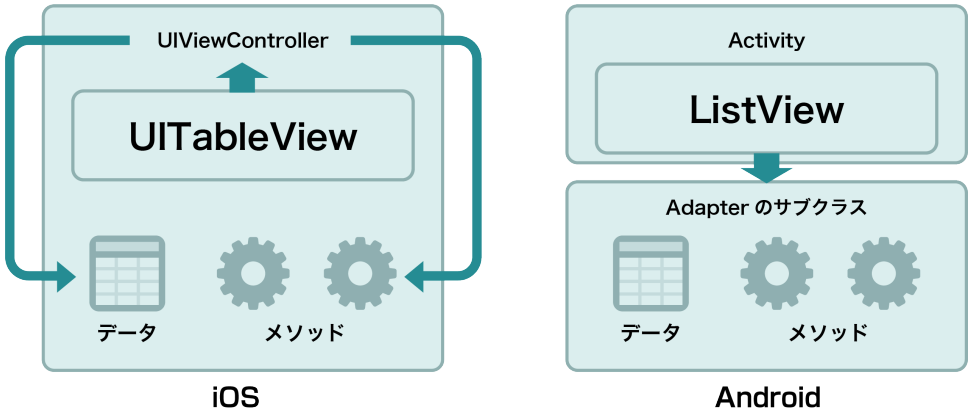


図 3.11: プラットフォーム間で処理の流れが異なる例

3.2.2 機能要件

以下に本プロジェクトで開発する歩行者ナビゲーションアプリの機能を挙げる。

地図

地図は現在地や目的地をユーザに提示するために提供する（図 3.13-）。前システムでは地図を操作するためのツールが用意されていたが、本プロジェクトでは保守性を重視し、最小限の構成とする。地図機能には Android と iOS のどちらにも対応している Google Maps API を利用する。Google Maps API も Android 版と iOS 版で細かな違いはあるが、クラス構造の統一に影響はない。

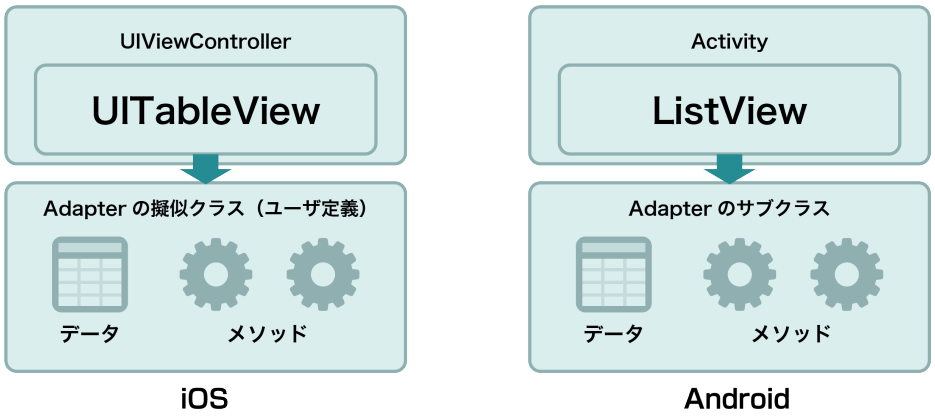


図 3.12: プラットフォーム同士の処理の流れを統一した例

施設内情報検索

キーワード検索により、施設内の情報をユーザに提供する（図 3.13-）。本プロジェクトでは、大学以外の様々な大規模施設での案内も可能にする。前システムで開発した歩行者ナビゲーションアプリは大学に特化していたが、本プロジェクトでは様々な施設に対応できるように汎用性を持たせる必要がある。

ブックマーク

検索結果から案内を開始すると、履歴に案内した場所が登録される。一度案内をしたことのある場所へは、再度検索せず、履歴からすぐに案内を開始することが出来る（図 3.13-）。

よく利用する場所は、履歴とは別にお気に入りとして登録することが出来る。お気に入りからも履歴と同様にすぐに案内を呼び出すことが出来る（図 3.13-）。

経路案内

前システムで挙げられた問題点として、目的地までの経路が表示されないという点があった。本プロジェクトでは独自に道情報を構築し、経路探索を行うことでこの問題を解決する（図 3.13-）。今回開発する歩行者ナビゲーションアプリでは、経路探索で計算した経路を地図上に線で指し示すことで、サービス利用者を案内する。また、AR による案内でも同等の機能を提供する。

従来の技術では、AR の実現のために大掛かりな各種センサーによる実装が必要だったが、このような装置はナビゲーションを始めとする屋外での利用には向いていなかった [3]。しかし、近年のスマートフォン端末の普及により、AR 技術を使った分かりやすいナビゲーションが手軽に利用できるようになった。これらのナビゲーションは、前システムでも実装した「AR によって目的地までの方向を示す」ものが大半である [4, 5, 6] が、本プロジェクトではさらに発展させ、AR による経路案内を実現する（図 3.13-）。

AR の描画は Android では View の Canvas、iOS では UIView の CoreGraphics がそれぞれ対応する。センサーから受け取った情報を元に全く同じ計算式を使い、Canvas や CoreGraphics を用いて AR として描画する。

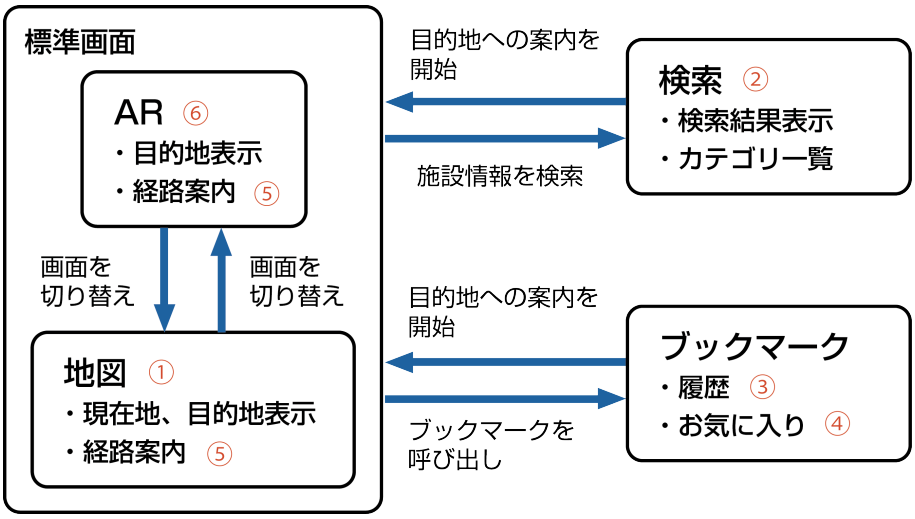


図 3.13: 状態の遷移

3.2.3 画面設計

本プロジェクトで開発する歩行者ナビゲーションアプリの画面設計を以下に示す。

標準画面

標準画面を図 3.14 に示す。標準画面はスマートフォン向け歩行者ナビゲーションアプリの代表である Google Maps のデザインを参考にした。標準画面では地図上に現在地を示すマーカーを表示する。

画面上部にある検索ボタンをタップすることで、キーワード入力による施設内情報検索が出来る。

画面右下のボタンは、現在地が画面の中心に来るように地図を自動的に移動するボタンである。現在地が画面の中心にある状態で再度このボタンをタップすると、端末が向いている方角に合わせて地図を自動的に回転するモードへ移行する。

画面左下にあるボタンをタップすると、後述のメニューを呼び出すことが出来る。



図 3.14: 標準画面

メニュー

メニューは、標準画面の左下のボタンをタップすることで図 3.15 のように呼び出すことが出来る。メニューでは、施設切り替え、お気に入り、履歴画面への移動が出来る。また、歩行者ナビゲーションアプリのヘルプページを閲覧することも出来る。画面の右半分をタップするか、Android の場合はバックボタンをタップすることで、メニューを閉じることが出来る。

検索結果画面

検索結果画面を図 3.16、3.17 に示す。検索結果画面はプラットフォームが提供している UI 部品に依存するため、Android と iOS とでインターフェースが異なる。

検索結果はカテゴリ毎に一覧表示される。カテゴリは一覧を左右にスワイプすることで切り替えることが出来る。カテゴリ分けは対象となる施設ごとに異なり、例えば筑波大学の場合は建物、授業、教室、ディズニーランドの場合はエリア、アトラクション、ショーという分け方になる。



図 3.15: メニュー

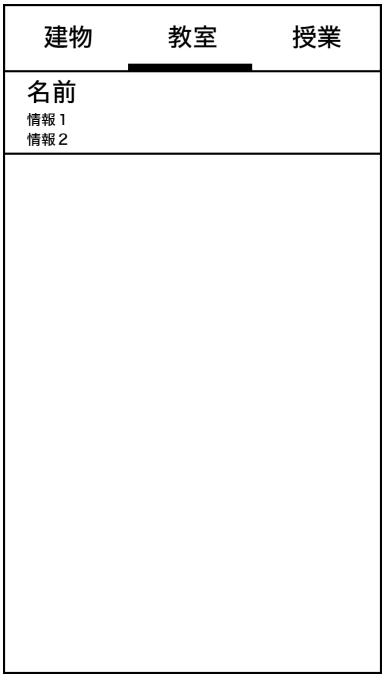


図 3.16: 検索結果（Android 版）

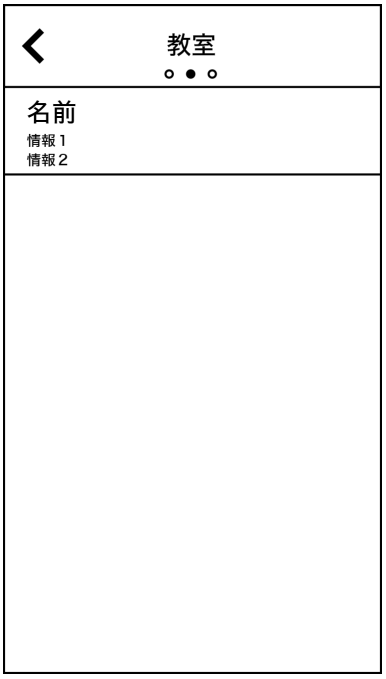


図 3.17: 検索結果（iOS 版）

案内画面

地図による案内画面を図 3.18 に示す。案内を開始すると、地図上に目的地の場所を示すマーカーと、現在地から目的地までの経路が線で表示される。表示される経路はデフォルトでは「推奨経路」の探索結果である。

画面上部には目的地に関する情報と、各種ツールボタンが配置されている。目的地の名前の右にある足あとのボタンをタップすると、経路の種類を選択出来るリストが表示される。本システムでは推奨経路、最短経路と、それぞれ階段を含まないオプションの計 4 つが用意されている。ユーザが経路の種類を選択すると、サーバーに経路探索を要求し、新しい経路を再描画する。星のボタンをタップすると、この案内中の目的地の場所をお気に入りに登録することが出来る。お気に入りに登録した場所は、以後メニューから呼び出すことが出来るお気に入り一覧からすぐに案内を開始することが出来る。このボタンはトグルとなっており、再度タップすることでお気に入りから消去される。バツ印のボタンをタップすると、案内を終了する。案内を終了すると画面上部の目的地情報やツールボタンが消えて標準画面のキーワード入力ボックスに戻り、目的地マーカーと経路を示す線は消去される。

AR による案内画面を図 3.19 に示す。AR 画面では地図画面同様、画面上部に目的地の情報やツールボタンがある。この画面では現在地と端末が向いている方向から、経路を三次元で表示する。表示される経路はカメラ越しに見える現実の道に重畳表示され、どの道を歩けば良いのかを直感的に把握することが出来る。



図 3.18: 案内画面（地図）



図 3.19: 案内画面（AR）

案内画面の切り替え

歩行者ナビゲーションアプリの案内画面は地図画面と AR 画面の 2 種類が存在する。これらの 2 つの画面は、端末を下に向けている時は地図画面、端末を上に向けている時は AR 画面に自動的に切り替わる。

ブックマーク画面

ブックマーク機能は、履歴やお気に入りを一覧表示する機能である（図 3.20）。一覧から目的地を選択することで、再検索すること無く目的地までの案内を開始できる。一覧には本システムが対応しているすべての施設、すべてのカテゴリを別け隔てなく表示する。

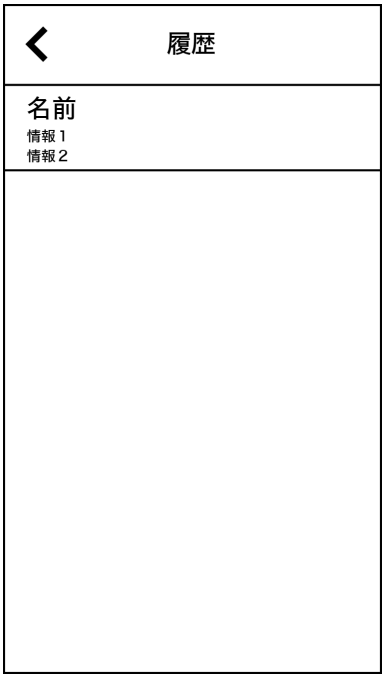


図 3.20: 履歴・お気に入り

3.3 開発した実装方式

本方式を利用したマルチプラットフォーム開発のワークフローを以下に示す。

1. 内部設計

実装するアプリケーションの内部設計を作成する。内部設計は Android と iOS で共有するため、両プラットフォームで役割が対応付けられたクラスを用いて設計する。クラス

対応表（表 3.2）を参照し、どちらのプラットフォームでもアプリケーションを同様に実装可能か検証しつつ内部設計を行う。検証には、それぞれのプラットフォームで対応しているクラスが、アプリケーションの実装に必要な機能を持っているか確認する。

2. プロジェクトの準備

実装のための準備を行う。本方式では、統合開発環境（以下、IDE）として Android の実装に Android Studio[4]、iOS の実装に Xcode[5] を利用する。各 IDE でプロジェクトを初期化すると、アプリケーションをコンパイルするために必要なソースコードやインターフェースなどを含むプロジェクトディレクトリが作成される。各プラットフォームのプロジェクトディレクトリは図 3.21 のように同一の場所（以後、プロジェクトルート）に置き、管理する。

画像変換ツールを利用するために、プロジェクトルートに画像変換ツールと画像変換ツールのための設定ファイルを置く。“画像変換ツール”には、変換対象の元画像ファイルが格納される “入力元フォルダ”のパスや、変換後の画像リソースを書き出す “出力先フォルダ”のパスを設定する。“画像変換ツールの設定ファイル”には、“入力元フォルダ”内の画像の、基準（mdpi、@1x）となる縦横のピクセルサイズを画像毎に定義する。画像変換ツールを実行すると、入力元フォルダの中から画像ファイルを読み取り、画像変換ツールの設定ファイルで定義されたピクセルサイズを基準に各画像の大きさを変換し、Android と iOS の出力先フォルダへ画像リソースを書き出す。

プロジェクトルートには、画像変換ツールが参照する “画像リソースの元画像”を保存するフォルダも配置する。アプリケーションのインターフェース内で利用する画像リソースは、すべてこのフォルダに保存する。このフォルダに保存された画像は画像変換ツールの変換対象となり、画像リソースの書き出しを利用することが出来るようになる。

3. プラットフォーム A の実装

まずどちらかのプラットフォームでアプリケーションをすべて実装する。実装の際、もう一方のプラットフォームでどのように実装するかを考えながら実装すると、後の実装を円滑に進めることが出来る。

実装中、画像リソースに変更があった場合は画像変換ツールを実行し、必要な画像リソースを出力する。画像リソースを出力すると、次にアプリケーションをコンパイルした際に、出力された画像がアプリケーションの画面に反映される。画像リソースを新たに追加する場合は、図 3.21 の “画像リソースの元画像”フォルダに画像を保存し、“画像変換ツールの設定ファイル”に新しく追加した画像リソースの基準となる縦横のピクセルサイズを追記する。

4. プラットフォーム B の移植

もう一方のプラットフォームでアプリケーションを実装する。ここでの実装は、実装済

[4]http://developer.android.com/sdk/index.html
[5]https://developer.apple.com/jp/xcode/

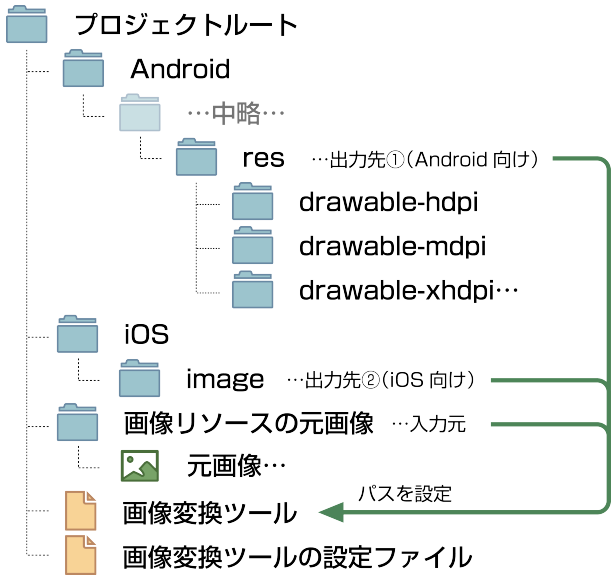


図 3.21: プロジェクト構成

みのアプリケーションを参考に“翻訳”する形で実装することで、ミスの少ない実装が可能になる。

5. ふるまいの統一

3.2.2 節「プラットフォーム間の差異を修正」の通り、手順 3 と 4 で実装した 2 つのアプリケーションを比較し、ふるまいを統一出来る場合は修正する。どちらかのプラットフォーム API と同等の動作をするクラスを用意することで、アプリケーションの構造を更に近づけることが出来る。

第4章　システム評価

4.1　実装の考察

本方式による実装が保守性の維持にどの程度貢献したのかを、画像リソースとクラス構造のそれぞれについて考察する。

4.1.1　画像リソースの統一

画像変換ツールの有効性を検証するため、1 枚の画像を各解像度に書き出す所要時間を、画像変換ツールを使う場合と使わない場合のそれぞれについて計測した。実行するタスクはそれぞれ以下のとおりである。

画像編集ツールを使う場合

1. 画像変換ツールを利用するための設定を行う 2. 画像変換ツールを用いて書き出しを実行する

画像変換ツールを使わない場合

1. 元となる画像を画像編集ツールで開く 2. 対象となるプラットフォームの解像度にあった大きさに元画像をリサイズする 3. リサイズした画像を適切な場所に適切な名称で書き出す 4. 2〜3 の手順をすべての解像度（全 9 種類）について実行する

前述のタスクを筆者が行った所、画像変換ツールを使った場合と使わなかった場合のそれぞれの所要時間は表 4.1 のようになった。画像変換ツールを用いることで、画像の書き出しに掛かる手間を 3 割程度に削減することが出来た。また、9-Patch 画像の場合はリサイズ後にマーカーを書き直す必要があるため、画像変換ツールを使わない場合はさらに時間を要する。

表 4.1: 画像書き出しの所要時間

画像変換ツールを使った場合	50 秒
画像変換ツールを使わなかった場合	146 秒

4.1.2 クラス構造の統一

本方式によるクラス構造の統一によって、Android と iOS を別々に実装する方法に比べて高い保守性が得られると考えられる。本方式を用いない従来の方法で別々に実装を行うと、アプリケーションに同一の機能を新しく追加するといった場合に、構造が異なるために実装方法が大きく異なるという問題が起きる。本方式を用いると、修正方法の検討はおおよそ 1 つのプラットフォームについて考えるだけで済み、実装も片方のプラットフォームで行った修正をもう一方のプラットフォームで同様に行うだけで済む。

実装の結果、Android と iOS のプログラムで定義したメソッドのうち、約 7 割を統一することが出来た（表 4.2）。ここで言う統一とは、メソッドの役割と入力、出力の対応が取れていることを表す。残りの約 3 割のメソッドについて考察する。

表 4.2: Android と iOS のメソッドの対応

プラットフォーム	一致したメソッド	メソッドの総数
Android	60	86
iOS	60	84

汎用通信クラス

このクラスは HTTP によってサーバーと通信を行い、データを送受信するための汎用クラスである。このクラスは各プラットフォームのネイティブ API を参照しているため、プラットフォームが提供している HTTP 通信に関するクラスの実装に依存してしまう。そのため、実装に大きな違いが出てしまっている。しかし、このような汎用クラスは頻繁に修正が加えられることが少ないため、保守性を損ねることにはならない。

ライフサイクルの違い

Android と iOS とでは、アプリケーションがポーズ状態に移行した時[1]や、ポーズ状態から復帰した時の挙動に違いがある。iOS では GPS や地磁気といった各種センサーやカメラを動かしたままポーズ状態に移行した場合、それらは自動的にストップする。これは端末の電池を無駄遣いしないための対策で、アプリケーションがアクティブではない時は、これらのセンサーを取得する必要がないため、このような動きをする。一方、Android ではアプリケーションがポーズ状態に移行した時、動いていたセンサーやカメラは自動的にはストップしない。Android の場合はアプリケーションがポーズ状態になった際、「アプリケーションがポーズに移行した」というイベントが OS から発火される。このイベントを経由して、ポーズ中に不必要なセンサーの読み取りを手動でストップしなければならない。この処理を怠ると、OS はアプリケーションが使われていないにも関わらず、センサーやカメラを動かし続けてしまう。

[1]タスク切り替えで別のアプリケーションに移動した時や、アプリケーションを起動したままスリープボタンを押した時

この仕様は Google Maps API にも存在し、iOS 版の API ではポーズ時の処理は特に定義する必要はないが、Android 版の API ではポーズ時やポーズ復帰時、画面初期化時など OS から呼び出される特定のイベントをすべて API にリダイレクトしなければならない。

前述のような理由で、約 3 割のメソッドの対応付けが取れないという結果になった。しかし、前述の差異はプログラムの処理の流れに深く関わって来ることは無いため、アプリケーションの保守性を損なうような致命的な違いにはならないと考えられる。本方式を用いることで、保守性の高いマルチプラットフォーム開発を行うことが可能である。

4.2 ユーザビリティテスト

本プロジェクトで開発した歩行者ナビゲーションアプリの、アプリケーションとしての問題点を探すため、ユーザビリティテストを実施した。

4.2.1 実験の目的

今回の実験は、開発した歩行者向けナビゲーションアプリのインターフェースのデザインが妥当であるかどうかを検証するために行う。アプリケーション利用者が、アプリ内で目的の機能に辿り着くことが出来るかどうかを検証する。アプリケーションを被験者に利用してもらい、その行動観察から評価を行う。

4.2.2 実験内容

実験観察者は、被験者に本プロジェクトで開発した歩行者向けナビゲーションアプリがインストールされた端末を渡した上で、表 4.3 に示す 12 種類のタスクをアプリケーション上で実施するように指示する。被験者には指示されたタスクをすべて実施してもらう。実験終了後、それぞれのタスクのわかりやすさについての 5 段階評価と、自由記述によるアンケートに回答してもらった。5 段階評価の内訳は、そのタスクが最も分かりにくいと感じた場合は “1”、最も分かりやすいと感じた場合は “5”を付けてもらう。本実験では、顧客の合意のもと、評価の平均が 3 以上のタスクを合格とし、3 未満のタスクは改善の余地ありとして、該当する機能を修正する。

実験にはヘルプドキュメントを用意し、被験者は好きなタイミングで参照することが出来る。尚、実験では紙媒体のヘルプを用意したが、アプリケーション内にも同じ内容のヘルプが組み込まれている。

実験は、本システムの背景や前システムを知っている学生 12 名、全く知らない学生 12 名の合計 24 名を対象に行った。

表 4.3: 被験者実験のタスクと結果の一覧

タスク	タスクの指示	評価の平均
1	ボタン操作で地図上の現在地を画面中央に移動させる	4.67
2	ボタン操作で地図を自動回転モードに切り替える	2.42
3	施設を筑波大学に変更する	3.29
4	「スターバックス」を検索する	4.25
5	「スターバックス」の案内を開始する	4.29
6	「スターバックス」をお気に入りに登録する	4.71
7	案内を終了する	4.38
8	お気に入りに登録されている「スターバックス」への案内を開始する	4.58
9	「スターバックス」をお気に入りから削除する	4.46
10	履歴に登録されている「スターバックス」までの案内を開始する	4.63
11	経路の種類を「最短ルート」に変更する	3.50
12	表示を AR 画面に切り替える	2.96

4.2.3 実験結果

実験結果を表 4.3 に示す。多くのタスクが平均して”4”以上の評価を獲得しており、ユーザーの期待した通りのインターフェースを備えていたと言える。一方で、タスク 2 番やタスク 12 番は”3”を下回っており、どちらかと言うと悪い結果となっている。タスク 2 番は、その前のタスク 1 番でタップするボタンを再度タップしなければならず、混乱の原因となっていた。タスク 12 番については、およそ半数の被験者がヘルプドキュメントを参照した上でタスクをこなしていた。

4.2.4 アプリケーションの改良

実験結果のうち、評価の低かった項目について修正を行う。修正対象として挙げられたのが表 4.3 中のタスク 2 と 12 であるが、2 についてはボタンの動作をスマートフォン向けの Google Maps に習って実装したため、修正の対象外とした。今回はタスク 12 番の「地図と AR 画面の切替」について改良を行う。

この機能で問題になっているのは、図 4.1、4.2 のような端末の向きによる切り替えが直感的に分かりにくく、AR 画面の存在に気が付けない可能性があるという点である。この問題に対し、実験後の被験者へのインタビューで特に意見が多かった、「初回案内時のみチュートリアルを表示する」という方法で解決することにした。

修正は以下の手順で行った。

実装方針の検討

まず、本修正の実装方針を検討した。Android と iOS でチュートリアルを表示する方法



図 4.1: 地図画面を表示するジェスチャ



図 4.2: AR 画面を表示するジェスチャ

は様々に考えられるが、両者のプラットフォームを考慮しないと、プラットフォームの API の差異から実装結果に差が生まれてしまうことがある。Android と iOS の両方で同様に実装可能な修正方針を模索する。

アプリケーションの修正

実装方針が固まったら、実際にアプリケーションの修正を行っていく。今回は iOS、Android の順に実施した。iOS で実装をしつつ、Android 側での実装もイメージすることで、実装結果に差が生じないように心がけた。また、本修正では新たに画像 2 枚を追加したが、画像変換ツールによって数分で所定の 18 枚の画像リソースを出力することが出来た。

上記の手順により、アプリケーションの修正を Android、iOS 合わせて 1 時間かからずに終わらせることが出来た。

修正後のアプリケーションでは、案内開始後に地図画面から AR 画面への切り替えをチュートリアルによって促す（図 4.3、4.6 の赤枠内）。AR 画面へ切り替えると、チュートリアルの内容が変わり、再度地図画面へ切り替える動作を促す（図 4.4、4.7）。地図画面へ切り替えるとチュートリアルは終了となり、チュートリアル表示は消え、通常通りに利用できるようになる（図 4.5、4.8）。チュートリアルが実施されるのは最初の 1 回目だけである。



図 4.3: チュートリアル 1 ページ目（赤枠内、Android 版）

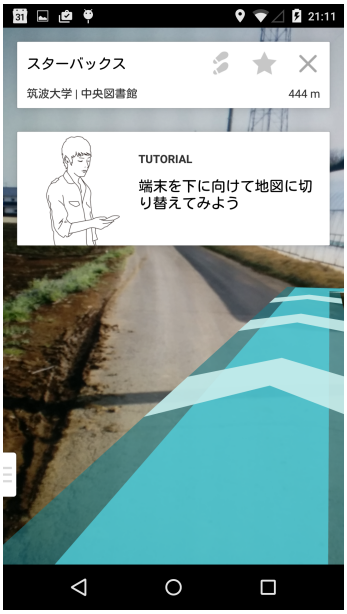


図 4.4: チュートリアル 2 ページ目（Android 版）



図 4.5: チュートリアル後（Android 版）



図 4.6: チュートリアル 1 ページ目（赤枠内、iOS 版）

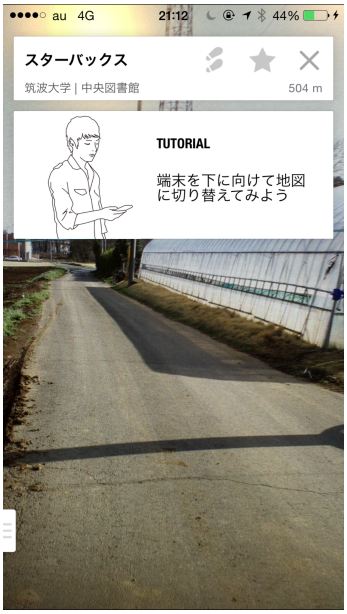


図 4.7: チュートリアル 2 ページ目（iOS 版）

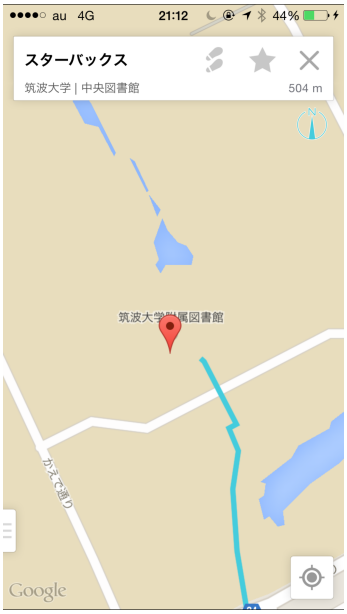


図 4.8: チュートリアル後（iOS 版）

4.3 考察

今回の改良によって追加された 3 つのメソッドはほとんどが一致しており（図 4.10、**??**）、メソッドが呼び出されるタイミングは完全に対応している。このような対応は実装済みのプログラム全体を通して施されているため、別の修正箇所でも今回と似たような効果が得られると考えられる。

開発した歩行者ナビゲーションアプリに対して実施した修正結果から、当初の期待通り効率的な修正が可能であること、修正後も保守性を維持していることが確認出来たため、本方式を用いたクラス構造の統一が有用であると言える。また、本プロジェクトで開発した画像変換ツールも効率的な開発をサポートしており、当初の目標をすべて実現出来たと言える。

```java
// チュートリアル済みかどうか
private boolean isTutorialPlayed(Context context) {
    return PrefUtil.getBoolean(context, "tutorial", false);
}

// チュートリアル開始
private void startTutorial() {
    tutorialProgress = 0;
    nextTutorial();
}

// チュートリアルを進める
private void nextTutorial() {
    switch (tutorialProgress) {
        case 0:
            tutorialView.setVisibility(View.VISIBLE);
            tutorialImage.setImageResource(R.drawable.tutorial_ar);
            tutorialMessage.setText("端末を上に向けてARに切り替えてみよう");
            tutorialProgress++;
            break;
        case 1:
            tutorialImage.setImageResource(R.drawable.tutorial_map);
            tutorialMessage.setText("端末を下に向けて地図に切り替えてみよう");
            tutorialProgress++;
            break;
        case 2:
            tutorialView.setVisibility(View.GONE);
            PrefUtil.putBoolean(this, "tutorial", true);
            tutorialProgress = -1;
            break;
    }
}
```

図 4.9: 追加したメソッド（Android 版）

```objc
// チュートリアル済みかどうか
- (bool)isTutorialPlayed {
    return [[NSUserDefaults standardUserDefaults] objectForKey:@"tutorial"] != nil;
}

// チュートリアル開始
- (void)startTutorial {
    self.tutorialProgress = 0;
    [self nextTutorial];
}

// チュートリアルを進める
- (void)nextTutorial {
    switch (self.tutorialProgress) {
        case 0 :
            self.tutorialView.hidden = false;
            self.tutorialImage.image = [UIImage imageNamed:@"tutorial-ar"];
            self.tutorialMessage.text = @"端末を上に向けてARに切り替えてみよう";
            self.tutorialProgress++;
            break;
        case 1 :
            self.tutorialImage.image = [UIImage imageNamed:@"tutorial-map"];
            self.tutorialMessage.text = @"端末を下に向けて地図に切り替えてみよう";
            self.tutorialProgress++;
            break;
        case 2 :
            self.tutorialView.hidden = true;
            [[NSUserDefaults standardUserDefaults] setObject:@1 forKey:@"tutorial"];
            self.tutorialProgress = -1;
            break;
    }
}
```

図 4.10: 追加したメソッド（iOS 版）

第5章　おわりに

本プロジェクトでは、「施設内での経路案内を可能にするナビゲーションシステムの開発」として、Android と iOS の 2 つのプラットフォームに対応した歩行者ナビゲーションアプリを開発した。

本システムは、大学のみを対象としていた前システムを発展させ、様々な施設に導入出来るようになった。歩行者ナビゲーションアプリとしては、クラス構造を統一することで、高い保守性、拡張性を保つことが出来た。このことは、被験者実験後に行ったアプリケーション改修の容易さからも裏付けられている。機能面でも経路案内の対応や AR 機能の充実により、より歩行者向けナビゲーションアプリとして進歩させることが出来た。また、本プロジェクトで開発した画像変換ツールは、様々なプロジェクトに導入できる汎用的なシステムとして開発した。従来の画像書き出しに掛かる時間を大幅に短縮出来、高効率な開発が可能になる。

今後の展望として、大規模施設における屋内対応が挙げられる。現状、GPS を用いて利用者の現在地を測位しているため、GPS の届かない屋内での利用は難しい。しかし、GPS 以外の情報を用いて絶対位置を取得する研究もあり、今後屋内の案内にも対応することが考えられる。また、筆者の担当領域においては、今回対応しなかったプラットフォームへのサポートが挙げられる。Android や iOS 以外にも、Firefox OS や Windows Phone といった新しいプラットフォームが存在し、今後日本国内でのシェアが拡大する可能性もある。筆者が実施した実装方法を拡張することで、新しいプラットフォームへも柔軟に対応出来ることが期待される。

謝辞

本報告書を執筆するにあたり、指導教員である三末和男准教授には、ゼミなどを通して貴重なご指導やご意見をいただきました。深くお礼申し上げます。本プロジェクトの課題担当教員である和田耕一教授、山際伸一准教授には、熱心な御指導、御助言を頂きました。ここに深く感謝致します。並列分散処理研究室秘書の森田美紀さん、石原美奈子さんには本プロジェクトに関す る事務手続きなどでお世話になりました。また、評価実験に協力していただいた方々に心から感謝致します。プロジェクトメンバーである千田隼人君、芳賀隼人君、畑中裕太君には、ともにプロジェクトに励み、グループワークの大切さを学ばせていただきました。ありがとうございました。最後に、私を暖かく見守って下さいました両親や友人に深く感謝致します。

参考文献

[1] スマートフォン市場規模の推移・予測（2014 年 4 月） - 株式会社 MM総研
http://www.m2ri.jp/newsreleases/main.php?id=010120140423500

[2] 八木 俊広、原 昇平、かわかみ ひろき, 実践 スマートフォンアプリケーション開発 —iOS、Android、Windows Phone の比較. 株式会社ブリリアントサービス, 2012.

[3] F. Liarokapis, I. Greatbatch, D. Mountain, A. Gunesh, V. Brujic-Okretic, and J. Raper. Mobile augmented reality techniques for geovisualisation. *Proceedings of the 9th International Conference on Information Visualisation*, pp. 745-751, 2005.

[4] F. Liarokapis, V. Brujic-Okretic, and S. Papakonstantinou. Exploring urban environments using virtual and augmented reality. *Journal of Virtual Reality and Broadcasting*, vol. 3, no. 5, pp. 1-13, 2007.

[5] T. Jackson, F. Angermann, and P. Meier. Survey of Use Cases for Mobile Augmented Reality Browsers. *Handbook of Augmented Reality*, Springer New York, pp. 409–431, 2011.

[6] P. Belimpasakis, Y. You, and P. Selonen. Enabling Rapid Creation of Content for Consumption in Mobile Augmented Reality *Fourth International Conference on Next Generation Mobile Applications, Services and Technologies*, pp. 1–6, 2010.

[7] Supporting Multiple Screens — Android Developers
http://developer.android.com/guide/practices/screens_support.html

[8] Resource Programming Guide: Image, Sound, and Video Resources
https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/LoadingResources/ImageSoundResources/ImageSoundResources.html

付録A ユーザーマニュアル

CampusARユーザーマニュアル

平成 26 年 11 月 11 日



キャンパス AR の基盤の構築

ART

文責: 畑中裕太、萬成亮太　版: 4.0

目 次

1 アプリ概要 3

1.1 何ができるのか？ 3

1.2 動作環境 3

2 アプリで使うインターフェース 4

2.1 標準画面 4

2.2 メニュー 5

2.3 施設選択画面 6

2.4 お気に入り一覧画面 7

2.5 履歴一覧画面 8

2.6 検索画面 9

2.7 検索結果画面 10

2.8 検索結果のカテゴリ切り替え 11

2.9 ナビゲーション画面（地図） 12

2.10 経路選択ウィンドウ 13

2.11 お気に入りに登録・削除 14

2.12 ナビゲーション画面（AR） 15

2.12.1 地図と AR の切り替え 16

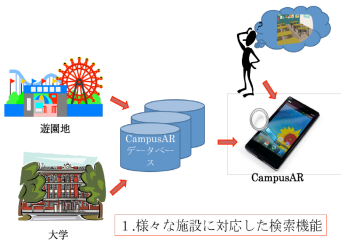
3 注意事項 17

1　アプリ概要

1.1　何ができるのか？

CampusAR が実現する機能は主に以下の 4 つです。

1. どこに目的地があるかわからない時に、目的地の検索をすることができます。
2. 2 次元地図と AR によって行きたい建物にガイドしてくれます。
3. お気に入り機能と履歴機能により再度検索する手間を省くことができます。



1 .様々な施設に対応した検索機能


2.地図とARシステムを用いたナビゲーション機能


3. お気に入りと履歴に登録

1.2　動作環境

Android4.1 以降、または iOS7 以降で、背面カメラ、モーションセンサと GPS を内蔵しており、モバイルデータ通信、または Wi-Fi での常時通信機能が使える携帯端末とタブレットで動作可能です。動作確認している機種を以下に示します。

- iPhone 6 Plus
- Galaxy note
- XPERIA Z1

2　アプリで使うインターフェース

2.1　標準画面



Android 版



iOS 版

① 検索ボックス
検索ボックスをタップすると、検索ボックスに文字を入力することができるようになります。
② メニューボタン
メニューボタンをタップすると、施設選択、お気に入り、履歴にアクセスできるメニューが表示されます。
③ 状態遷移ボタン
状態遷移ボタンは以下の状態遷移をします。

- 現在位置に移動するボタン：現在位置が画面の中心になるように画面移動します。
- 画面回転ボタン：端末の向きに合わせて画面が回転するようになります。地図が 45 °仰角の俯瞰で見ているように表示が変わります。
- 画面回転停止ボタン：画面の回転を止め、地図の向きを北向きに変えることができます。

④ 現在位置の表示
GPS で取得した現在位置を地図上に表示します。端末の向いている方向に従い矢印の向きが変化します。
⑤ コンパス
N と書かれている方角が北を指しています。

2.2 メニュー



Android 版



iOS 版

① 施設選択
施設選択ボタンをタップすると、施設選択画面に移動します。
② お気に入り
お気に入りボタンをタップすると、お気に入り一覧画面に移動します。
③ 履歴
履歴ボタンをタップすると、履歴一覧画面に移動します。
④ ヘルプ
ヘルプボタンをタップすると、アプリマニュアルを表示します。
⑤ メニューを閉じる
半透明のスクリーンをタップすると、メニューを閉じます。

2.3 施設選択画面



Android 版



iOS 版

① 施設一覧
施設名をタップすると、選択した施設で検索できるようになります。
② バックボタン
バックボタンをタップすると、前の画面に戻ります。

2.4　お気に入り一覧画面



Android 版



iOS 版

① お気に入り一覧
場所の名前をタップすると、その場所への案内を開始します。
② バックボタン
バックボタンをタップすると、前の画面に戻ります。

2.5 履歴一覧画面



Android 版



iOS 版

① 履歴一覧
場所の名前をタップすると、その場所への案内を開始します。履歴一覧には検索結果から案内を開始した場所が自動的に登録されます。
② バックボタン
バックボタンをタップすると、前の画面に戻ります。

2.6 検索画面



Android 版



iOS 版

① 検索ボックス
検索ボックスをタップすると、キーボードが表示されます。キーボードから検索したいワードを入力し、検索を行うことができます。
② 検索ボタン
キーボードの検索ボタンをタップすると、入力したキーワードで検索されます。
③ キーボードを閉じる
地図をタップすると、キーボードが閉じます。

2.7　検索結果画面



Android 版



iOS 版

① 検索結果
場所の名前をタップすると、その場所への案内を開始します。
② カテゴリ
一覧に表示されているカテゴリを示します。
③ バックボタン
バックボタンをタップすると、前の画面に戻ります。

2.8　検索結果のカテゴリ切り替え



Android 版



iOS 版

① 検索結果
検索一覧を横方向にスワイプすることで、検索結果のカテゴリを切り替えることが出来ます。

2.9　ナビゲーション画面（地図）



Android 版



iOS 版

① 目的地名
目的地の名前を表示します。目的地名をタップすると、案内を終了し、新たに検索することが出来ます。
② 詳細情報
目的地の詳しい情報を表示します。
③ 総距離
案内を開始した地点から目的地までの距離を表示します。
④ 経路選択ボタン
案内経路の種類を切り替えるウィンドウを表示します。
⑤ お気に入りボタン
案内中の場所をお気に入りに登録、または解除します。
⑥ 案内終了ボタン
案内を終了します。
⑦ 目的地
案内中の目的地の場所を表示します。
⑧ 経路
案内を開始した地点から目的地までの道のりを表示します。

2.10　経路選択ウィンドウ



Android 版



iOS 版

① 経路の種類
経路の種類をタップすると、現在地から目的地までの経路を再計算し、画面に新しい経路を描画します。
② ダイアログを閉じる
地図やキャンセルボタンをタップすると、経路選択をキャンセルし、ウィンドウを閉じます。

2.11　お気に入りに登録・削除



Android 版　　iOS 版

お気に入りボタンをタップすると、案内中の場所をお気に入り登録、または削除することが出来ます。
アイコンが黄色く点灯している場所は、お気に入りに登録されていることを示します。

2.12　ナビゲーション画面（AR）



Android 版



iOS 版

① 経路
経路を AR で表示します。

2.12.1　地図と AR の切り替え



地図



AR

端末を下に向けると地図、立ち上げると AR に自動的に切り替わります。

3　注意事項

アプリを使用する上での注意事項

- 端末の GPS 機能を ON にして使用してください。
- インターネット通信ができる状態でのみ使用することができます。
- このアプリケーションが案内するのは建物までです。教室までは館内の見取り図などから自力で到達してください。
- 自転車などの乗り物に乗りながら使用しないでください。
- 機種によってはカメラがなかなか起動しないものもあります。

付録B 被験者実験 同意書

システム情報工学研究科 コンピュータサイエンス専攻
ART

CampusAR 被験者実験 同意書

実験の内容

- □ インターフェースのユーザビリティの検証（30 分）
- □ 検索時間の測定（30 分）
- □ 本システムを使った道案内の有効性の検証（１～２時間）

注意事項

- 実験担当者の指示に従って下さい。
- 実験内容を口外しないで下さい。

個人情報とデータの取扱い

取得したデータや個人情報は、研究目的以外には使用しません。特定課題研究報告書などに利用いたしますが、匿名化しますので、個人情報は守られます。

以上の説明を理解し、本実験に参加することに同意してくださる場合、署名をお願いします。

平成　　年　　月　　日

所属：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

付録C 実験手順書

実験手順書（スマートフォンアプリケーション）

作成日：2014/11/12
作成者：萬成亮太

目的

1. 開発したアプリケーションのユーザビリティを検証するため
2. 開発したアプリケーションの改善点を探るため

被験者

高度 IT の学生 12 名

準備物

1. 本書
2. 事後アンケート
3. ボールペン
4. CampusAR ユーザーマニュアル
5. Android 端末（CampusAR が十分に動作するもの）
6. iOS 端末（CampusAR が十分に動作するもの）

被験者の割り当て

被験者には普段利用している携帯端末と同じプラットフォーム（Android/iOS）の端末を割り当てる。

普段利用している端末が CampusAR に対応していない被験者には、Android と iOS を均等に割り当てる。

実験手順

1. 観察者が実験に使用する CampusAR を以下の状態に戻す。
   - 標準画面に移動する
   - 案内を開始している場合は、案内を終了する
   - 検索対象の施設を「東京ディズニーランド」に設定する
   - お気に入りをすべて削除する
   - 地図上の現在地マーカーが見えなくなるように地図を移動する
2. 観察者が被験者に CampusAR のユーザーマニュアルを渡し、実験中に参照しても良いことを伝える。
3. 被験者実験を開始する。被験者に以下のタスクを順番に指示する。各タスク完了時に、終わった旨を伝えてもらうよう促す。
   A) ボタン操作で地図上の現在地を画面中央に移動させる
   B) ボタン操作で地図を自動回転モードに切り替える
   C) 施設を筑波大学に変更する
   D) 「スターバックス」を検索する
   E) 「スターバックス」の案内を開始する
   F) 「スターバックス」をお気に入りに登録する
   G) 案内を終了する
   H) お気に入りに登録されている「スターバックス」への案内を開始する

I)　「スターバックス」をお気に入りから削除する
J)　履歴に登録されている「スターバックス」までの案内を開始する
K)　経路の種類を「最短ルート」に変更する
L)　表示を AR 画面に切り替える

4. 観察者が被験者に「CampusAR のユーザビリティに関するアンケート」を記入してもらう。
5. 記入してもらった結果のうち、満足度のスコアが低いものについて、観察者が被験者に質問する。

注意事項

1. 実験は被験者 1 名ずつ個別に行う。
2. 被験者には実験中の観察項目を教えない。
3. 実験中は被験者に対して実験結果に影響を与える可能性のある助言を行わない。

付録D ユーザビリティに関するアンケート

CampusAR のユーザビリティに関するアンケート

- **普段使用している携帯電話はスマートフォンですか？**

Yes　・　No

- **Yes の場合は、スマートフォンのプラットフォームを教えて下さい**

_____

- **それぞれのシナリオに関してお答えください**

（5:非常にそう思う 4:そう思う 3:どちらとも言えない 2:そう思わない 1:全くそう思わない）

1. 現在地を画面中央に移動するボタンは適切である	5 ・ 4 ・ 3 ・ 2 ・ 1
2. 地図を自動的に回転するボタンには改善の余地がある	5 ・ 4 ・ 3 ・ 2 ・ 1
3. 施設を筑波大学に変更する方法は直感的である	5 ・ 4 ・ 3 ・ 2 ・ 1
4. 「スターバックス」の検索に戸惑った	5 ・ 4 ・ 3 ・ 2 ・ 1
5. 目的地への案内を開始する方法は適切である	5 ・ 4 ・ 3 ・ 2 ・ 1
6. お気に入りに登録する方法が分かりづらかった	5 ・ 4 ・ 3 ・ 2 ・ 1
7. 案内を終了する方法は直感的である	5 ・ 4 ・ 3 ・ 2 ・ 1
8. お気に入りの呼び出し方に戸惑った	5 ・ 4 ・ 3 ・ 2 ・ 1
9. お気に入りを削除する方法はわかりやすかった	5 ・ 4 ・ 3 ・ 2 ・ 1
10. 履歴機能の使用には経験を要する	5 ・ 4 ・ 3 ・ 2 ・ 1
11. 経路の種類を選択する方法は直感的である	5 ・ 4 ・ 3 ・ 2 ・ 1
12. AR 画面の切り替えに戸惑った	5 ・ 4 ・ 3 ・ 2 ・ 1

自由記述：

アンケートは以上です。ありがとうございました。

付録E Androidアプリケーション UIテスト仕様書

Androidアプリケーション UIテスト仕様書

●目的
プログラムの全処理を1回以上実行し、本画面の正常動作を確認すること。
●本テストの合格条件
全テスト項目について判定結果が「合」となること。
●留意事項

テストID	種別	対象機能	前提条件	操作内容
UIT1		アプリケーション起動	A) インターネットに接続可能であること。 B) GPSがONになっていて、位置座標を取得できていること。 ※これ以降、インターネットに接続していない条件がない限りA)B)の条件を満たすこととする。	1) CampusARのアイコンをタップする。
UIT2		メニュー表示	A) 地図画面であること。	1) メニューボタンをタップする。
UIT3		施設選択	A) UIT2の実行直後であること。	1) 施設をタップする。
UIT4		施設選択	A) UIT3の実行直後であること。 B) 施設情報に「筑波大学」が登録されていること。	1) 「筑波大学」をタップする。
UIT5		施設選択	A) UIT4の実行直後であること。 B) 施設情報に「愛媛大学」が登録されていること。	1) 「愛媛大学」をタップする。
UIT6		検索機能(愛媛大学)	A) UIT5の実行直後であること。 B) 施設内情報に「愛大ショップえみか」が登録されていること。	1) 検索ボックスをタップする。 2) 「愛大」と入力する。 3) エンターキーをタップする。
UIT7		検索機能(愛媛大学)	A) UIT6の実行直後であること。 B) 施設内情報の階層が3つあり、カテゴリーはそれぞれ「建物」、「教室」、「授業」で登録されていること。	1) 画面を左にスワイプする。 2) 画面を左にスワイプする。 3) 画面を左にスワイプする。 4) 画面を右にスワイプする。 5) 画面を右にスワイプする。 6) 「愛大ショップえみか」をタップする。
UIT8		検索機能(筑波大学)	A) UIT4の実行直後であること。 B) 施設内情報に「石の広場」が登録されていること。	1) 検索ボックスをタップする。 2) 「石の広場」と入力する。 3) エンターキーをタップする。
UIT9		検索機能(筑波大学)	A) UIT8の実行直後であること。 B) 施設内情報の階層が3つあり、カテゴリーはそれぞれ「建物」、「教室」、「授業」で登録されていること。	1) 画面を左にスワイプする。 2) 画面を左にスワイプする。 3) 画面を左にスワイプする。 4) 画面を右にスワイプする。 5) 画面を右にスワイプする。 6) 「石の広場」をタップする。
UIT10		検索結果確認	A) UIT4の実行直後であること。 B) 施設内情報に「研究開発プロジェクトA」、「3B311」、「3B棟」が登録されていること。 C) 「研究開発プロジェクトA」が「3B311」の子に、「3B311」が「3B」棟の子として登録されていること。	1) 検索ボックスをタップする。 2) 「研究開発」と入力する。 3) エンターキーをタップする。
UIT11		検索結果確認	A) UIT10の実行直後であること。	1) 画面を左にスワイプする。 2) 画面を左にスワイプする。
UIT12		経路案内機能(建物)	A) UIT9の実行直後であること。	
UIT13		経路案内機能(建物)	A) UIT12の実行直後であること。	1) 経路変更ボタンをタップする。
UIT14		経路案内機能(建物)	A) UIT13の実行直後であること。	1) 最短経路をタップする。
UIT15		経路案内機能(建物)	A) UIT14の実行直後であること。	1) 推奨経路をタップする。
UIT16		経路案内機能(建物)	A) UIT15の実行直後であること。	1) 端末を地面から垂直に傾ける。
UIT17		経路案内機能(授業)	A) UIT11の実行直後であること。	1) 「研究開発プロジェクトA」をタップする。
UIT18		経路案内機能(授業)	A) UIT17の実行直後であること。	1) 経路変更ボタンをタップする。
UIT19		経路案内機能(授業)	A) UIT18の実行直後であること。	1) 最短経路をタップする。
UIT20		経路案内機能(授業)	A) UIT19の実行直後であること。	1) 推奨経路をタップする。
UIT21		経路案内機能(授業)	A) UIT20の実行直後であること。	1) 端末を地面から垂直に傾ける。
UIT22		ブックマーク機能	A) UIT20の実行直後であること。	1) ブックマークボタンをタップする。
UIT23		ブックマーク機能	A) UIT22の実行直後であること。	1) メニューボタンをタップする。 2) お気に入りをタップする
UIT24		ブックマーク機能	A) UIT23の実行直後であること。	1) 「研究開発プロジェクトA」をタップする。
UIT25		ブックマーク機能	A) UIT24の実行直後であること。	1) ブックマークボタンをタップする。

Rev.1 2014/11/4 畑中 新規作成

合格条件	実施者	結果	実施日	備考
a) スプラッシュ画面が表示されること。 b) 地図画面が表示されること。	萬成亮太	OK	11/12/14	
a) メニューバーが表示されること。	萬成亮太	OK	11/12/14	
a) 施設一覧表示画面が表示されること。	萬成亮太	OK	11/12/14	
a) 地図画面が表示されること。 b) 検索ボックスの文言が「筑波大学から検索する」になっていること。	萬成亮太	OK	11/12/14	
a) 地図画面が表示されること。 b) 検索ボックスの文言が「愛媛大学から検索する」になっていること。	萬成亮太	OK	11/12/14	
a) 1)の操作で検索ボックスが入力可能状態になること。 b) 2)で入力した文字が検索ボックスに表示されること。 c) 3)の操作で検索結果一覧画面が表示されること。	萬成亮太	OK	11/12/14	
a) 1)の操作でカテゴリーが教室に変更すること。 b) 2)の操作でカテゴリーが授業に変更すること。 c) 3)の操作で画面に変化がないこと。 d) 4)の操作でカテゴリーが教室に変更すること。 e) 5)の操作でカテゴリーが建物に変更すること。 f) 6)の操作で地図画面が表示されること。	萬成亮太	OK	11/12/14	c) 3)操作後、カテゴリ表記が切り替わるため、画面変化あり
a) 1)の操作で検索ボックスが入力可能状態になること。 b) 2)で入力した文字が検索ボックスに表示されること。 c) 3)の操作で検索結果一覧画面が表示されること。	萬成亮太	OK	11/12/14	
a) 1)の操作でカテゴリーが教室に変更すること。 b) 2)の操作でカテゴリーが授業に変更すること。 c) 3)の操作で画面に変化がないこと。 d) 4)の操作でカテゴリーが教室に変更すること。 e) 5)の操作でカテゴリーが建物に変更すること。 f) 6)の操作で地図画面が表示されること。	萬成亮太	OK	11/12/14	c) 3)操作後、カテゴリ表記が切り替わるため、画面変化あり
a) 1)の操作で検索ボックスが入力可能状態になること。 b) 2)で入力した文字が検索ボックスに表示されること。 c) 3)の操作で検索結果一覧画面が表示されること。	萬成亮太	OK	11/12/14	
a) 1)の操作でカテゴリーが教室に変更すること。 b) 2)の操作でカテゴリーが授業に変更すること。 c) 研究開発プロジェクトAが検索結果一覧にあること。 d) 研究開発プロジェクトAのリストボックスの中に、「3B311」、「3B棟」が表示されていること。	萬成亮太	OK	11/12/14	
a) 上部のテキストボックスに文言「石の広場」と表示されていること。 b) 上部のテキストボックスに経路変更ボタンが表示されていること。 c) 上部のテキストボックスにブックマークボタンが表示されていること。 d) 上部のテキストボックスに経路案内終了ボタンが表示されていること。 e) 地図画面の目的地の位置座標にマーカーが表示されていること。 f) 現在位置から目的地のマーカーまでの経路が表示されていること。 g) 現在位置から目的地のマーカーまでの距離が表示されていること。	萬成亮太	OK	11/12/14	
a) モーダルが表示されること。 b) モーダルの表示内容が設計仕様を満足していること。	萬成亮太	OK	11/12/14	
a) モーダルが削除されること。 b) 地図画面に表示されている経路が最短経路に変更されていること。	萬成亮太	OK	11/12/14	
a) モーダルが削除されること。 b) 地図画面に表示されている経路が推奨経路に変更されていること。	萬成亮太	OK	11/12/14	
a) カメラ画面に切り替わること。 b) 地図画面に表示されている経路に沿った形で、ARの経路が重畳表示されていること。	萬成亮太	NG(#1) 2回	11/12/14	
a) 地図画面に切り替わること。 b) テキストボックス内は文言以外UIT12と同じ情報が表示されていること。 c) テキストボックスの文言に「研究開発プロジェクトA」と表示されていること。 d) c)の文言の下に「3B311 | 3B棟」と表示されること。 e) 現在位置から「3B棟」までの経路が表示されること。	萬成亮太	OK	11/12/14	d)「筑波大学 | 3B311 | 3B棟」(仕様変更により)
a) モーダルが表示されること。 b) モーダルの表示内容が設計仕様を満足していること。	萬成亮太	OK	11/12/14	
a) モーダルが削除されること。 b) 地図画面に表示されている経路が最短経路に変更されていること。	萬成亮太	OK	11/12/14	
a) モーダルが削除されること。 b) 地図画面に表示されている経路が推奨経路に変更されていること。	萬成亮太	OK	11/12/14	
a) カメラ画面に切り替わること。 b) 地図画面に表示されている経路に沿った形で、ARの経路が重畳表示されていること。	萬成亮太	NG(#1) 2回	11/12/14	
a) ブックマークボタンが黄色に変わること。	萬成亮太	OK	11/12/14	
a) メニューバーが表示されること。 b) 2)の操作後にブックマーク一覧表示画面が表示されること。 c) 「研究開発プロジェクトA」が表示されること。 d) 研究開発プロジェクトのリストボックスに表示される情報が設計仕様を満足していること。	萬成亮太	OK	11/12/14	
a) UIT17と同じ合格条件	萬成亮太	OK	11/12/14	
a) ブックマークボタンが灰色に変わること。	萬成亮太	OK	11/12/14	

UIT26		ブックマーク機能	A) UIT25の実行直後であること。	1) メニューボタンをタップする。 2) お気に入りをタップする
UIT27		履歴機能	A) UIT20の実行直後であること。	1) メニューボタンをタップする。 2) 履歴をタップする
UIT28		経路案内機能	A) UIT20の実行直後であること。	1) 経路案内終了ボタンをタップする。
UIT29		バックボタン(施設)	A) UIT22の実行直後であること。	1) メニューボタンをタップする。 2) 施設をタップする。
UIT30		バックボタン(施設)	A) UIT29の実行直後であること。	1) 画面上部のバックボタンをタップする。
UIT31		バックボタン(施設)	A) UIT29の実行直後であること。	1) Androidのバックボタンをタップする。
UIT32		バックボタン(ブックマーク)	A) UIT22の実行直後であること。	1) メニューボタンをタップする。 2) お気に入りをタップする。
UIT33		バックボタン(ブックマーク)	A) UIT32の実行直後であること。	1) 画面上部のバックボタンをタップする。
UIT34		バックボタン(ブックマーク)	A) UIT32の実行直後であること。	1) Androidのバックボタンをタップする。
UIT35		バックボタン(履歴)	A) UIT22の実行直後であること。	1) メニューボタンをタップする。 2) 履歴をタップする。
UIT36		バックボタン(履歴)	A) UIT35の実行直後であること。	1) 画面上部のバックボタンをタップする。
UIT37		バックボタン(履歴)	A) UIT35の実行直後であること。	1) Androidのバックボタンをタップする。
UIT38		バックボタン(検索)	A) UIT8の実行直後であること。	1) 画面上部のバックボタンをタップする。
UIT39		バックボタン(検索)	A) UIT8の実行直後であること。	1) Androidのバックボタンをタップする。
UIT40		メニューバー削除	A) UIT1の実行直後であること。	1) メニューボタンをタップする。 2) メニューバー以外の画面をタップする
UIT41		Androidのバックボタンの挙動	A) UIT1の実行直後であること。	1) Androidのバックボタンをタップする。
UIT42		Androidのバックボタンの挙動	A) UIT1の実行直後であること。	1) メニューボタンをタップする。 2) Androidのバックボタンをタップする。
UIT43		Androidのバックボタンの挙動	A) UIT12の実行直後であること。	1) Androidのバックボタンをタップする。
UIT44		地図画面の機能	A) UIT1の実行直後であること。	1) 現在位置に戻るボタンをタップする。
UIT45		地図画面の機能	A) UIT44の実行直後であること。	1) 画面回転ボタンをタップする。
UIT46		地図画面の機能	A) UIT44の実行直後であること。	1) 地図上をタップすること。
UIT47		地図画面の機能	A) UIT45の実行直後であること。	1) 地図上をタップすること。
UIT48		地図画面の機能	A) UIT45の実行直後であること。	1) 端末が向いている方向を変化させる。
UIT49		地図画面の機能	A) UIT45の実行直後であること。	1) 現在位置に戻るボタンをタップする。
UIT50		地図画面の機能	A) UIT1の実行直後であること。	1) 端末を地面と垂直に傾ける。
UIT51				
UIT52				
UIT53				
UIT54				
UIT55				
UIT56				
UIT57				

a) メニューバーが表示されること。 b)「研究開発プロジェクトA」が削除されていること。	萬成亮太	OK	11/12/14	
a) メニューバーが表示されること。 b) 2)の操作後に履歴一覧表示画面が表示されること。 c)「研究開発プロジェクトA」が一番上に表示されること。 d) 研究開発プロジェクトのリストボックスに表示される情報が設計仕様を満足していること。	萬成亮太	OK	11/12/14	
a) UIT4の終了後と同じ画面になっていること。	萬成亮太	OK	11/12/14	
a) メニューバーが表示されること。 b) 2)の操作後に施設一覧表示画面が表示されること。	萬成亮太	OK	11/12/14	
a) 地図画面が表示される。	萬成亮太	OK	11/12/14	
a) 地図画面が表示される。	萬成亮太	OK	11/12/14	
a) メニューバーが表示されること。 b) 2)の操作後にブックマーク一覧表示画面が表示されること。	萬成亮太	OK	11/12/14	
a) 地図画面が表示される。	萬成亮太	OK	11/12/14	
a) 地図画面が表示される。	萬成亮太	OK	11/12/14	
a) メニューバーが表示されること。 b) 2)の操作後に履歴一覧表示画面が表示されること。	萬成亮太	OK	11/12/14	
a) 地図画面が表示される。	萬成亮太	OK	11/12/14	
a) 地図画面が表示される。	萬成亮太	OK	11/12/14	
a) 地図画面が表示される。	萬成亮太	OK	11/12/14	
a) 地図画面が表示される。	萬成亮太	OK	11/12/14	
a) 2)の操作後と1)の操作前の画面から変化していないこと。	萬成亮太	OK	11/12/14	
a) アプリが終了すること。	萬成亮太	OK	11/12/14	
a) アプリが終了すること。	萬成亮太	OK	11/12/14	
a) UIT4の終了後と同じ画面になっていること。	萬成亮太	OK	11/12/14	
a) 現在位置に現在位置マーカーが表示されていること。 b) 現在位置が中心になるように画面が移動すること。 c) 画面の上部が北になっていること。 d) 縮尺が変更しないこと。 e) 現在位置に戻るボタンが画面回転ボタンに変更すること。	萬成亮太	OK	11/12/14	
a) 端末が向いている方向に画面の上部が移動すること。 b) 仰角表示になること。 c) 画面回転ボタンが画面回転終了ボタンに変わること。	萬成亮太	OK	11/12/14	
a) 画面回転ボタンが現在位置に戻るボタンに変更すること。 b) その他の画面の表示が変わらないこと。	萬成亮太	OK	11/12/14	
a) 画面回転終了ボタンが現在位置に戻るボタンに変更すること。 b) その他の画面の表示が変わらないこと。	萬成亮太	OK	11/12/14	
a) 地図画面の北方向にコンパスの指す方向が変更すること。	萬成亮太	OK	11/12/14	
a) 現在位置が中心になるように画面が移動すること。 b) 画面の上部が北になっていること。 c) 縮尺が変更しないこと。 d) 現在位置に戻るボタンが画面回転ボタンに変更すること。	萬成亮太	OK	11/12/14	
a) 画面が変化しないこと。	萬成亮太	OK	11/12/14	

付録F iOSアプリケーション UIテスト仕様書

●目的
プログラムの全処理を1回以上実行し、本画面の正常動作を確認すること。
●本テストの合格条件
全テスト項目について判定結果が「合」となること。
●留意事項
Androidアプリケーションテスト仕様書を元に作られているため、Android版固有のテスト項目についてはテスト不要。

テストID	種別	対象機能	前提条件	操作内容
UIT1		アプリケーション起動	A) インターネットに接続可能であること。 B) GPSがONになっていて、位置座標を取得できていること。。 ※これ以降、インターネットに接続していない条件がない限りA)B)の条件を満たすこととする。	1) CampusARのアイコンをタップする。
UIT2		メニュー表示	A) 地図画面であること。	1) メニューボタンをタップする。
UIT3		施設選択	A) UIT2の実行直後であること。	1) 施設をタップする。
UIT4		施設選択	A) UIT3の実行直後であること。 B) 施設情報に「筑波大学」が登録されていること。	1) 「筑波大学」をタップする。
UIT5		施設選択	A) UIT4の実行直後であること。 B) 施設情報に「愛媛大学」が登録されていること。	1) 「愛媛大学」をタップする。
UIT6		検索機能(愛媛大学)	A) UIT5の実行直後であること。 B) 施設内情報に「愛大ショップえみか」が登録されていること。	1) 検索ボックスをタップする。 2) 「愛大」と入力する。 3) エンターキーをタップする。
UIT7		検索機能(愛媛大学)	A) UIT6の実行直後であること。 B) 施設内情報の階層が3つあり、カテゴリーはそれぞれ「建物」、「教室」、「授業」で登録されていること。	1) 画面を左にスワイプする。 2) 画面を左にスワイプする。 3) 画面を左にスワイプする。 4) 画面を右にスワイプする。 5) 画面を右にスワイプする。 6) 「愛大ショップえみか」をタップする。
UIT8		検索機能(筑波大学)	A) UIT4の実行直後であること。 B) 施設内情報に「石の広場」が登録されていること。	1) 検索ボックスをタップする。 2) 「石の広場」と入力する。 3) エンターキーをタップする。
UIT9		検索機能(筑波大学)	A) UIT8の実行直後であること。 B) 施設内情報の階層が3つあり、カテゴリーはそれぞれ「建物」、「教室」、「授業」で登録されていること。	1) 画面を左にスワイプする。 2) 画面を左にスワイプする。 3) 画面を左にスワイプする。 4) 画面を右にスワイプする。 5) 画面を右にスワイプする。 6) 「石の広場」をタップする。
UIT10		検索結果確認	A) UIT4の実行直後であること。 B) 施設内情報に「研究開発プロジェクトA」、「3B311」、「3B棟」が登録されていること。 C) 「研究開発プロジェクトA」が「3B311」の子に、「3B311」が「3B」棟の子として登録されていること。	1) 検索ボックスをタップする。 2) 「研究開発」と入力する。 3) エンターキーをタップする。
UIT11		検索結果確認	A) UIT10の実行直後であること。	1) 画面を左にスワイプする。 2) 画面を左にスワイプする。
UIT12		経路案内機能(建物)	A) UIT9の実行直後であること。	
UIT13		経路案内機能(建物)	A) UIT12の実行直後であること。	1) 経路変更ボタンをタップする。
UIT14		経路案内機能(建物)	A) UIT13の実行直後であること。	1) 最短経路をタップする。
UIT15		経路案内機能(建物)	A) UIT14の実行直後であること。	1) 推奨経路をタップする。
UIT16		経路案内機能(建物)	A) UIT15の実行直後であること。	1) 端末を地面から垂直に傾ける。
UIT17		経路案内機能(授業)	A) UIT11の実行直後であること。	1) 「研究開発プロジェクトA」をタップする。
UIT18		経路案内機能(授業)	A) UIT17の実行直後であること。	1) 経路変更ボタンをタップする。
UIT19		経路案内機能(授業)	A) UIT18の実行直後であること。	1) 最短経路をタップする。
UIT20		経路案内機能(授業)	A) UIT19の実行直後であること。	1) 推奨経路をタップする。
UIT21		経路案内機能(授業)	A) UIT20の実行直後であること。	1) 端末を地面から垂直に傾ける。
UIT22		ブックマーク機能	A) UIT20の実行直後であること。	1) ブックマークボタンをタップする。
UIT23		ブックマーク機能	A) UIT22の実行直後であること。	1) メニューボタンをタップする。 2) お気に入りをタップする
UIT24		ブックマーク機能	A) UIT23の実行直後であること。	1) 「研究開発プロジェクトA」をタップする。
UIT25		ブックマーク機能	A) UIT24の実行直後であること。	1) ブックマークボタンをタップする。

書

Rev.1 2014/11/13 萬成 新規作成（Androidアプリケーションテスト仕様書 から継承）

合格条件	実施者	結果	実施日	備考
a) スプラッシュ画面が表示されること。 b) 地図画面が表示されること。	萬成亮太	OK	11/13/14	
a) メニューバーが表示されること。	萬成亮太	OK	11/13/14	
a) 施設一覧表示画面が表示されること。	萬成亮太	OK	11/13/14	
a) 地図画面が表示されること。 b) 検索ボックスの文言が「筑波大学から検索する」になっていること。	萬成亮太	1回目：NG（#1） 2回目：OK	11/13/14	
a) 地図画面が表示されること。 b) 検索ボックスの文言が「愛媛大学から検索する」になっていること。	萬成亮太	OK	11/13/14	b)「愛媛大学 城北地区から検索する」
a) 1)の操作で検索ボックスが入力可能状態になること。 b) 2)で入力した文字が検索ボックスに表示されること。 c) 3)の操作で検索結果一覧画面が表示されること。	萬成亮太	OK	11/13/14	
a) 1)の操作でカテゴリーが教室に変更すること。 b) 2)の操作でカテゴリーが授業に変更すること。 c) 3)の操作で画面に変化がないこと。 d) 4)の操作でカテゴリーが教室に変更すること。 e) 5)の操作でカテゴリーが建物に変更すること。 f) 6)の操作で地図画面が表示されること。	萬成亮太	OK	11/13/14	c) 3)操作後、カテゴリ表記が切り替わるため、画面変化あり
a) 1)の操作で検索ボックスが入力可能状態になること。 b) 2)で入力した文字が検索ボックスに表示されること。 c) 3)の操作で検索結果一覧画面が表示されること。	萬成亮太	OK	11/13/14	
a) 1)の操作でカテゴリーが教室に変更すること。 b) 2)の操作でカテゴリーが授業に変更すること。 c) 3)の操作で画面に変化がないこと。 d) 4)の操作でカテゴリーが教室に変更すること。 e) 5)の操作でカテゴリーが建物に変更すること。 f) 6)の操作で地図画面が表示されること。	萬成亮太	OK	11/13/14	c) 3)操作後、カテゴリ表記が切り替わるため、画面変化あり
a) 1)の操作で検索ボックスが入力可能状態になること。 b) 2)で入力した文字が検索ボックスに表示されること。 c) 3)の操作で検索結果一覧画面が表示されること。	萬成亮太	OK	11/13/14	
a) 1)の操作でカテゴリーが教室に変更すること。 b) 2)の操作でカテゴリーが授業に変更すること。 c) 研究開発プロジェクトAが検索結果一覧にあること。 d) 研究開発プロジェクトAのリストボックスの中に、「3B311」、「3B棟」が表示されていること。	萬成亮太	OK	11/13/14	
a) 上部のテキストボックスに文言「石の広場」と表示されていること。 b) 上部のテキストボックスに経路変更ボタンが表示されていること。 c) 上部のテキストボックスにブックマークボタンが表示されていること。 d) 上部のテキストボックスに経路案内終了ボタンが表示されていること。 e) 地図画面の目的地の位置座標にマーカーが表示されていること。 f) 現在位置から目的地のマーカーまでの経路が表示されていること。 g) 現在位置から目的地のマーカーまでの距離が表示されていること。	萬成亮太	OK	11/13/14	
a) モーダルが表示されること。 b) モーダルの表示内容が設計仕様を満足していること。	萬成亮太	OK	11/13/14	
a) モーダルが削除されること。 b) 地図画面に表示されている経路が最短経路に変更されていること。	萬成亮太	OK	11/13/14	
a) モーダルが削除されること。 b) 地図画面に表示されている経路が推奨経路に変更されていること。	萬成亮太	OK	11/13/14	
a) カメラ画面に切り替わること。 b) 地図画面に表示されている経路に沿った形で、ARの経路が重畳表示されていること。	萬成亮太	OK	11/13/14	
a) 地図画面に切り替わること。 b) テキストボックス内は文言以外UIT12と同じ情報が表示されていること。 c) テキストボックスの文言に「研究開発プロジェクトA」と表示されていること。 d) cの文言の下に「3B311	3B棟」と表示されること。 e) 現在位置から「3B棟」までの経路が表示されること。	萬成亮太	OK	11/13/14	d)「筑波大学	3B311	3B棟」（仕様変更により）
a) モーダルが表示されること。 b) モーダルの表示内容が設計仕様を満足していること。	萬成亮太	OK	11/13/14	
a) モーダルが削除されること。 b) 地図画面に表示されている経路が最短経路に変更されていること。	萬成亮太	OK	11/13/14	
a) モーダルが削除されること。 b) 地図画面に表示されている経路が推奨経路に変更されていること。	萬成亮太	OK	11/13/14	
a) カメラ画面に切り替わること。 b) 地図画面に表示されている経路に沿った形で、ARの経路が重畳表示されていること。	萬成亮太	OK	11/13/14	
a) ブックマークボタンが黄色に変わること。	萬成亮太	OK	11/13/14	
a) メニューバーが表示されること。 b) 2)の操作後にブックマーク一覧表示画面が表示されること。 c) 「研究開発プロジェクトA」が表示されること。 d) 研究開発プロジェクトのリストボックスに表示される情報が設計仕様を満足していること。	萬成亮太	OK	11/13/14	
a) UIT17と同じ合格条件	萬成亮太	OK	11/13/14	
a) ブックマークボタンが灰色に変わること。	萬成亮太	OK	11/13/14	

UIT26		ブックマーク機能	A) UIT25の実行直後であること。	1) メニューボタンをタップする。 2) お気に入りをタップする
UIT27		履歴機能	A) UIT20の実行直後であること。	1) メニューボタンをタップする。 2) 履歴をタップする
UIT28		経路案内機能	A) UIT20の実行直後であること。	1) 経路案内終了ボタンをタップする。
UIT29		バックボタン(施設)	A) UIT22の実行直後であること。	1) メニューボタンをタップする。 2) 施設をタップする。
UIT30		バックボタン(施設)	A) UIT29の実行直後であること。	1) 画面上部のバックボタンをタップする。
UIT31		バックボタン(施設)	A) UIT29の実行直後であること。	1) Androidのバックボタンをタップする。
UIT32		バックボタン(ブックマーク)	A) UIT22の実行直後であること。	1) メニューボタンをタップする。 2) お気に入りをタップする。
UIT33		バックボタン(ブックマーク)	A) UIT32の実行直後であること。	1) 画面上部のバックボタンをタップする。
UIT34		バックボタン(ブックマーク)	A) UIT32の実行直後であること。	1) Androidのバックボタンをタップする。
UIT35		バックボタン(履歴)	A) UIT22の実行直後であること。	1) メニューボタンをタップする。 2) 履歴をタップする。
UIT36		バックボタン(履歴)	A) UIT35の実行直後であること。	1) 画面上部のバックボタンをタップする。
UIT37		バックボタン(履歴)	A) UIT35の実行直後であること。	1) Androidのバックボタンをタップする。
UIT38		バックボタン(検索)	A) UIT8の実行直後であること。	1) 画面上部のバックボタンをタップする。
UIT39		バックボタン(検索)	A) UIT8の実行直後であること。	1) Androidのバックボタンをタップする。
UIT40		メニューバー削除	A) UIT1の実行直後であること。	1) メニューボタンをタップする。 2) メニューバー以外の画面をタップする
UIT41		Androidのバックボタンの挙動	A) UIT1の実行直後であること。	1) Androidのバックボタンをタップする。
UIT42		Androidのバックボタンの挙動	A) UIT1の実行直後であること。	1) メニューボタンをタップする。 2) Androidのバックボタンをタップする。
UIT43		Androidのバックボタンの挙動	A) UIT12の実行直後であること。	1) Androidのバックボタンをタップする。
UIT44		地図画面の機能	A) UIT1の実行直後であること。	1) 現在位置に戻るボタンをタップする。
UIT45		地図画面の機能	A) UIT44の実行直後であること。	1) 画面回転ボタンをタップする。
UIT46		地図画面の機能	A) UIT44の実行直後であること。	1) 地図上をタップすること。
UIT47		地図画面の機能	A) UIT45の実行直後であること。	1) 地図上をタップすること。
UIT48		地図画面の機能	A) UIT45の実行直後であること。	1) 端末が向いている方向を変化させる。
UIT49		地図画面の機能	A) UIT45の実行直後であること。	1) 現在位置に戻るボタンをタップする。
UIT50		地図画面の機能	A) UIT1の実行直後であること。	1) 端末を地面と垂直に傾ける。
UIT51				
UIT52				
UIT53				
UIT54				
UIT55				
UIT56				
UIT57				

a) メニューバーが表示されること。 b)「研究開発プロジェクトA」が削除されていること。	萬成亮太	OK	11/13/14	
a) メニューバーが表示されること。 b) 2)の操作後に履歴一覧表示画面が表示されること。 c)「研究開発プロジェクトA」が一番上に表示されること。 d) 研究開発プロジェクトのリストボックスに表示される情報が設計仕様を満足していること。	萬成亮太	OK	11/13/14	
a) UIT4の終了後と同じ画面になっていること。	萬成亮太	OK	11/13/14	
a) メニューバーが表示されること。 b) 2)の操作後に施設一覧表示画面が表示されること。	萬成亮太	OK	11/13/14	
a) 地図画面が表示される。	萬成亮太	OK	11/13/14	
a) 地図画面が表示される。		N/A		テスト不要
a) メニューバーが表示されること。 b) 2)の操作後にブックマーク一覧表示画面が表示されること。	萬成亮太	OK	11/13/14	
a) 地図画面が表示される。	萬成亮太	OK	11/13/14	
a) 地図画面が表示される。		N/A		テスト不要
a) メニューバーが表示されること。 b) 2)の操作後に履歴一覧表示画面が表示されること。	萬成亮太	OK	11/13/14	
a) 地図画面が表示される。	萬成亮太	OK	11/13/14	
a) 地図画面が表示される。		N/A		テスト不要
a) 地図画面が表示される。	萬成亮太	OK	11/13/14	
a) 地図画面が表示される。		N/A		テスト不要
a) 2)の操作後と1)の操作前の画面から変化していないこと。	萬成亮太	OK	11/13/14	
a) アプリが終了すること。		N/A		テスト不要
a) アプリが終了すること。		N/A		テスト不要
a) UIT4の終了後と同じ画面になっていること。		N/A		テスト不要
a) 現在位置に現在位置マーカーが表示されていること。 b) 現在位置が中心になるように画面が移動すること。 c) 画面の上部が北になっていること。 d) 縮尺が変更しないこと。 e) 現在位置に戻るボタンが画面回転ボタンに変更すること。	萬成亮太	OK	11/13/14	
a) 端末が向いている方向に画面の上部が移動すること。 b) 仰角表示になること。 c) 画面回転ボタンが画面回転終了ボタンに変わること。	萬成亮太	OK	11/13/14	
a) 画面回転ボタンが現在位置に戻るボタンに変更すること。 b) その他の画面の表示が変わらないこと。	萬成亮太	OK	11/13/14	
a) 画面回転終了ボタンが現在位置に戻るボタンに変更すること。 b) その他の画面の表示が変わらないこと。	萬成亮太	OK	11/13/14	
a) 地図画面の北方向にコンパスの指す方向が変更すること。	萬成亮太	OK	11/13/14	
a) 現在位置が中心になるように画面が移動すること。 b) 画面の上部が北になっていること。 c) 縮尺が変更しないこと。 d) 現在位置に戻るボタンが画面回転ボタンに変更すること。	萬成亮太	OK	11/13/14	
a) 画面が変化しないこと。	萬成亮太	1回目：NG(#2) 2回目：OK	11/13/14	

付録G Androidアプリケーション テストバグ管理票

Androidアプリケーション

#	機能	内容	優先度	発見者
1	AR重畳表示	ARで描画される経路が左右反転している。	高	萬成亮太
2				

テストバグ管理票

発見日	修正内容	修正日	修正者
11/12/14	ARViewのlatitudeとlongitudeが入れ替わっていた	11/12/14	萬成亮太

付録H iOSアプリケーション テストバグ管理票

iOSアプリケーション テ

No.	機能	内容	優先度	発見者
1	テキストボックスの施設名表示	プレースホルダの表記が「″(施設名)″ から検索する	低	萬成亮太
2	AR画面への切り替え	案内を開始していない時にもARに切り替えが出来る	中	萬成亮太

ストバグ管理票

発見日	修正内容	修正日	修正者
11/13/14	「(施設名)から検索する」に変更	11/13/14	萬成亮太
11/13/14	案内を開始していない時は端末を起こしてもARに切	11/13/14	萬成亮太