

卒業論文 2021年度（令和3年度）

ユーザインターフェースの設計をサポートする  
ノーデザインツールの研究

慶應義塾大学 環境情報学部

尾崎 正和

全世界インターフェースデザイン（増井研究会）

2022年1月



## 卒業論文 2021 年度（令和 3 年度）

# ユーザインターフェースの設計をサポートするノーデザインツールの研究

## 論文要旨

スマートフォンやパーソナルコンピュータの急激な普及につれ、様々な応用ソフト（アプリケーション）が開発され人々が毎日利用している。近年ではアプリケーション開発において十分な開発時間、人員、技術が確保することが難しく質の担保されたアプリケーションを設計し開発することが難しくなっている。そして個人開発者、スタートアップ企業、アイディアはあるが実装する力のない個人などではさらにこれを行うことが難しくなる。

その流れに伴い「ノーコード」と呼ばれるプログラミングやコンピュータサイエンスに精通していない人でも簡単にアプリケーションを開発できるサービスが登場した。しかしながら現在のノーコードにおいてもアプリケーションのユーザインターフェースデザインはユーザが行うか、テンプレートを利用する他ないためユーザインターフェースに精通していないユーザでは自由度の高く質の担保されたユーザインターフェースのアプリケーションを開発するには困難である。

そこで本研究では、本来大規模な開発でしか行われない専門のユーザインターフェースデザインを自動化し、ユーザインターフェースに関する専門的な知識がなくても容易にユーザインターフェースを開発できるようにするためのシステムの開発をおこなった。ユーザインターフェースの設計の方法については確立された手法はなくデザイナのセンスと経験に委ねられていた部分が多くあったが、既存のユーザインターフェースを分析していくことでユーザインターフェースは画面に表示されている要素の数と種類、各要素のグルーピング、画面内の要素がもつ優先度が決定されればインターフェースを制作できるのではないかという仮説を立てた。その仮説に基づき iOS 向けのネイティブアプリケーションのユーザインターフェースを専門的知識なしに制作できるシステムを開発し有用性を明らかにした。

## キーワード

UI, ノーデザイン, ノーコード, プロダクト開発, UI 自動生成, インターフェースビルダー, 人間中心設計

慶應義塾大学 環境情報学部

尾崎 正和

# **Abstract Of Graduation Thesis Academic Year 2021**

## **Research on no-design tools to support the design of user interfaces**

### **Summary**

High-quality, easy-to-use UI design requires in-depth professional knowledge and experience. We will propose a system that builds UI automatically. Our system enables designing UI without professional knowledge and experience.

There is no established method for UI design. So, it has often been left to the sense and experience of the designer.

However, the UI of smartphone applications is similar and has not changed significantly for decades. Hence, we thought that there must be some kind of rule.

By analyzing existing interfaces, we hypothesized that UI can be built if the number and types of elements, the grouping of each element, and the priority of each element are determined. Also, in general, when we talk about "design" in software, we tend to focus on the appearance of the interface.

By automating the appearance of interfaces, a designer can focus on the essential design. Based on this hypothesis, we developed a system that allows users to generate user interface source codes for native iOS applications (SwiftUI) without specialized knowledge. After that, evaluated the usefulness of this system.

### **Keywords**

User Interface, No-design, No-code, Product Development, UI Generation, Interface Builder, Human Centered Design

Faculty of Environment and Information Studies  
Keio University

Masakaz Ozaki



# 目 次

<b>第1章 序論</b>	<b>1</b>
1.1 背景 . . . . .	1
1.1.1 システム開発におけるデザイナの役割と現状 . . . . .	1
1.2 目的 . . . . .	3
1.3 問題の所在 . . . . .	3
1.4 本論文の構成 . . . . .	4
<b>第2章 ノーデザイン: 要素, 優先度, グルーピングの情報からのUI自動生成手法の提案</b>	<b>5</b>
2.1 既存インターフェースの分析 . . . . .	5
2.1.1 画面遷移を司るUI . . . . .	5
2.1.2 画面の内容を表示するUI . . . . .	7
2.2 アルゴリズムの構築 . . . . .	7
<b>第3章 関連研究と関連プロダクトの整理</b>	<b>11</b>
3.1 ノーコードでのシステム開発 . . . . .	11
3.1.1 MIT App Inventor . . . . .	11
3.1.2 STUDIO . . . . .	11
3.1.3 Glide . . . . .	12
3.2 デザインの自動化 . . . . .	12
3.2.1 機械学習を用いたUIの自動生成 . . . . .	12
<b>第4章 iOSネイティブアプリケーションのインターフェース自動生成システムの開発</b>	<b>13</b>
4.1 システムの設計・開発 . . . . .	13
4.1.1 基本的な配置アルゴリズム . . . . .	16
4.1.2 優先度 . . . . .	16
4.1.3 グルーピング . . . . .	22
4.2 システムの有効性についての実験 . . . . .	22
4.2.1 既存UIの再現 . . . . .	22
4.2.2 ユーザテスト . . . . .	27
4.2.3 対象と手続き . . . . .	27
4.2.4 結果と考察 . . . . .	29
<b>第5章 結論</b>	<b>35</b>

5.1 今後の課題	35
5.1.1 要素, 優先度, グルーピングの入力 UI の検討と開発	35
5.1.2 画面遷移を司る UI の自動生成	35
5.1.3 配色の自動生成	36
5.1.4 スライダー UI による表層デザインの調整	36
5.1.5 統合的なシステムとしての展望	36
謝辞	38
参考文献	40
付録 A UI 自動生成システムユーザテスト アンケート質問紙	43
付録 B UI 自動生成システムユーザテスト アンケート回答	45
B.1 選択問題回答(問 1-3)	45
B.2 普段アプリ開発において、デザインとはどのようなことをおこなっていますか。	45
B.3 今回の自動システムを使った率直な感想をお願いします.	45
B.4 優先度を調整して思い通りの UI に近づきましたか.	46
B.5 普段 UI を考える、実装する上で大変だと思うことは何ですか.	46
B.6 全体を通しての感想や意見などなんでも記入してください.	46

## 図 目 次

1.1	Garret の UX 五段階モデル [3] . . . . .	2
2.1	Hierarchical Navigation[5] . . . . .	6
2.2	Flat Navigation[5] . . . . .	6
2.3	Content-Driven or Experience-Driven Navigation[5] . . . . .	6
2.4	InstagramiOS アプリの UI の分析 . . . . .	8
2.5	facebookiOS アプリの UI の分析 . . . . .	8
2.6	日経電子版 iOS アプリの UI の分析 . . . . .	9
4.1	Label の優先度による視覚的目立たせ具合 . . . . .	17
4.2	Text View の優先度による視覚的目立たせ具合 . . . . .	18
4.3	Button の優先度による視覚的目立たせ具合 . . . . .	18
4.4	Toggle の優先度による視覚的目立たせ具合 . . . . .	19
4.5	Image の優先度による視覚的目立たせ具合 . . . . .	19
4.6	Map の優先度による視覚的目立たせ具合 . . . . .	20
4.7	DatePicker の優先度による視覚的目立たせ具合 . . . . .	21
4.8	Slider の優先度による視覚的目立たせ具合 . . . . .	21
4.9	既存の SNS フィード画面 (Instagram) . . . . .	23
4.10	既存の SNS フィード画面 (Instagram) の要素, 優先度, グルーピング . . . . .	23
4.11	本システムで再構築された SNS フィード画面 . . . . .	24
4.12	既存のログイン画面 (NIKKEI Wave) . . . . .	24
4.13	既存のログイン画面 (NIKKEI Wave) の要素, 優先度, グルーピング . . . . .	25
4.14	本システムで再構築されたログイン画面 . . . . .	25
4.15	リスト画面 (Settings) . . . . .	26
4.16	リスト画面 (Settings) の要素, 優先度, グルーピング . . . . .	26
4.17	リスト画面 . . . . .	27
4.18	地図画面 (Maps) . . . . .	28
4.19	地図画面 (Maps) の要素, 優先度, グルーピング . . . . .	28
4.20	地図画面 . . . . .	29
4.21	被験者 A 記入用紙 . . . . .	30
4.22	被験者 A 自動生成 UI . . . . .	30
4.23	被験者 B 記入用紙 . . . . .	31
4.24	被験者 B 自動生成 UI . . . . .	31

4.25 被験者 C 記入用紙	31
4.26 被験者 D 記入用紙	32
4.27 被験者 D 自動生成 UI	32
4.28 被験者 E 記入用紙	32
4.29 被験者 E 自動生成 UI	33
4.30 被験者 E 自動生成 UI-2	33
5.1 統合デザインシステムのプロトタイプ	36
5.2 統合デザインシステムのプロトタイプ 2	37

## 表 目 次

4.1 利用可能な要素一覧 . . . . .	15
4.2 各要素の配置係数 . . . . .	16
B.1 各要素の配置係数 . . . . .	45



# 第1章 序論

## 1.1 背景

近年、スマートフォンの急速な普及により数多くのアプリケーションが開発され、人々が毎日利用している、そのアプリケーションの開発には人々が感じている課題の洗い出し、適切な仕様の確定、使いやすいインターフェースの設計、実装など専門的で複雑な工程が多数存在する。中でもインターフェース設計とその実装は開発全体の中で設計段階で 45 %、実装段階で 50 %、保守段階で 37 % の時間を占めると言われている [1]。

またデザインカンパニーである Goodpatch の社長である土屋は創業の原点について次のように述べており、デザインの重要性について述べている。

Goodpatch は 2011 年に UI デザインに特化したデザイン会社としてスタートしました。2011 年に起業前に渡ったサンフランシスコでは創業期の Instagram や Uber、Airbnb など多くのスタートアップの共同創業者にデザイナーがおり、ベータ版から UI デザインに力を入れ、ユーザー体験を最初から考え、デザインを重要な差別化要素としてプロダクト開発を行っておりました。日本の環境と大きなギャップを感じた土屋は日本に帰り、創業したことが Goodpatch の原点です。[2]

大手企業の大規模開発では社内に専門知識を持ったデザイナが在籍していたり、Goodpatch のようなデザインカンパニーと共に設計を行うことが可能であるが専門のデザイナが在籍しないスタートアップ企業やインディーズ開発、個人開発ではこれらを行うのは非常に困難である。

また、日本ではデザインが装飾などの表層的なもののみを指すと誤解されていると指摘されている [2]。本来のデザインの役割とは、プロダクトの本質的な価値を見出し、価値を最大化させることである。James Garret の UX 五段階モデル（図 1.1）は、具象と抽象を行き来しながら、戦略、要件、構造、骨格、表層全てを設計することがデザインであると示している。

### 1.1.1 システム開発におけるデザイナの役割と現状

資金や人的リソースが十分な大規模開発では役割としてのデザイナが参画している場合も多いが、前述のようなデザインの能力を有しているとは限らない。職業としてのデザイナを

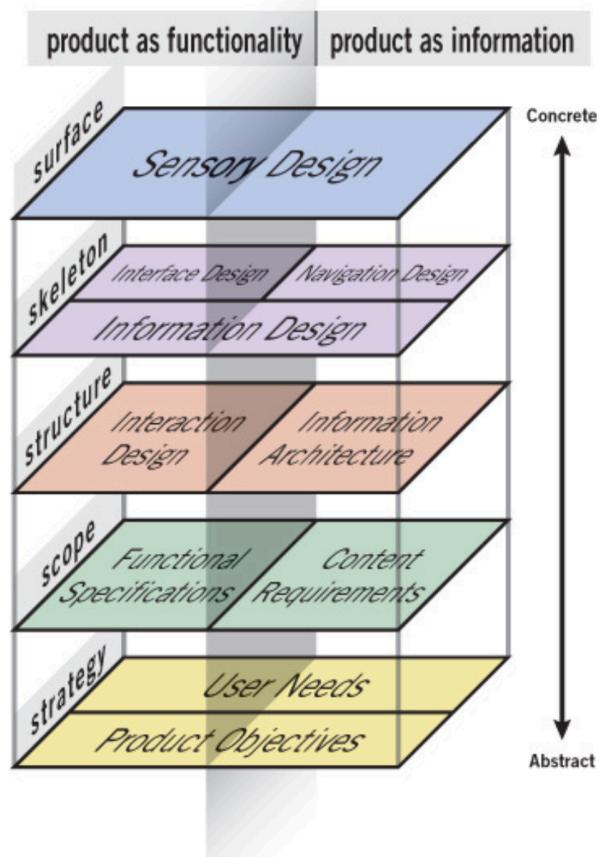


図 1.1: Garret の UX 五段階モデル [3]

職業としている人の中でも Garret が述べている本来のデザインを忘れ、表層のデザインに注力する場合が多く本来のデザインが一般に普及しているとは言えない。また、デザイナよりもアプリに向き合っている時間が長いエンジニアの方がアプリのデザインについて専門的知識を持ち開発をおこなっている場合も多数ある。エンジニアの場合 Garret の UX 五段階モデルもさることながら、UI が実際に動く仕組みを理解している分、内部構造という面での専門知識を有する場合も多い。実際に日本経済新聞電子版のメンバーによるデザインに関する登壇では 14 個中全ての登壇がエンジニア、又はエンジニア経験のある人の登壇となっている。[4]

個人開発アプリは大規模開発するほど収益が見込まれないものの、ニッチな消費者のニーズにあうさまざまなアプリケーションがアプリストアにリリースされている。しかしながら資金、時間、人的リソースが十分にない個人開発では開発者がデザインの重要性について理解した専門家がいないことにより使い勝手が悪いものや、表層のデザインにのみ注力したアプリケーションがあり、消費者のニーズに答えられずにいる。

以上のことから、(1) 現在は大規模な開発でしか行われていない専門性の高いユーザインタフェースデザインのハードルを下げる、(2) 表層のデザインにとらわれずに奥深くのデザインに注力することができる手法を開発することが求められる。

## 1.2 目的

本研究では(1) 現在は大規模な開発でしか行われていない専門性の高いユーザインタフェースデザインのハードルを下げ、(2) 表層のデザインにとらわれずに奥深くのデザインに注力することができるようとするシステムの開発を目指す。

現在の UI デザインには確立された設計手法はなく、デザイナの経験やセンスに頼っているものの、既存の UI を分析することにより UI 設計の手法を確立すれば簡易な UI 自動生成システムになり得る。また、(2) 注力されがちであった表層のデザインを自動化することでデザイナ、個人開発者が本来のデザインに向き合うことにも目を自然と向けられるようになることが可能になる。

本研究ではこの手法になりうる仮説として、1画面に必要な要素一覧、各要素の相対的な優先度、要素同士のグルーピングから UI を自動で生成する手法を提案する。

そして提案した手法をもとに1画面に必要な要素一覧、各要素の相対的な優先度、要素同士のグルーピングの情報から iOS ネイティブアプリケーションのソースコード (SwiftUI) を自動で生成するシステムを開発した。そして、開発したシステムの有用性を示した。

## 1.3 問題の所在

- 質の高い UI を設計するには専門知識が求められる。
- デザインを表層のグラフィックデザインだけだと誤解している場合が多く、専門知識の欠如に関する自覚が薄い。

- 既存のノーコードではデザインは自分で行う又はテンプレートから選んだデザインを使用する必要がある
- ノーコードでは生成される成果物がアプリケーションのソースコードではないため。ノーコードでできる範囲を超えた機能の実装は困難である。
- 機械学習を用いた自動生成では一意性が担保できない。
- GANで生成したUIは画像なのでアプリとして機能しない。

また、スマートフォンのUIは2007年の初代iPhoneから、パソコンについては1984年の初代Macintoshから基本的なUIは全く変わっていないにもかかわらず、UIの生成の法則性の研究はあまり行われてこなかった。以上のことから表層のデザインを自動化し、専門知識のない人でも質の高いUIを構築できるシステムの開発が求められる。

## 1.4 本論文の構成

本論文の構成を示す。

第1章では本研究の背景について述べた。第2章では要素、優先度、グルーピングからUIを自動生成する手法を提案し、第3章では関連研究と関連プロダクトを整理する。第4章ではその手法をもとにiOS向けネイティブアプリケーションのコードを生成するシステムを提案する。そして、それらの有効性を示す。第5章の結論では本研究を総括し、考察と展望を述べる。付録として、本研究で行った実験で得られたデータを添付する。

# 第2章 ノーデザイン: 要素, 優先度, グルーピングの情報からのUI自動生成手法の提案

本章では、後述の iOS ネイティブアプリケーションのインターフェース自動生成システムの開発に先立ち、既存のインターフェースの分析から導きさした要素、優先度、グルーピングによりインターフェースの構築が行える法則について検討した。自動生成を行うアルゴリズムの提案を行う。

## 2.1 既存インターフェースの分析

既存の iOS ネイティブアプリケーションの UI のレイアウトの分析をおこなった。数多くの UI を分析した一部として Instagram(図 2.4)、Facebook(図 2.5)、日経電子版(図 2.6)を例に解説する。分析の結果、画面内での UI は画面遷移を司り、画面内の体験には直接影響を与えない UI(図 2.4、図 2.5、図 2.6 左側スクリーンショット赤枠部分)、画面のコンテンツを表示する UI に分類した。

### 2.1.1 画面遷移を司る UI

画面遷移を司る UI では画面やアプリのタイトル、アプリ内での別機能への動線が配置されている。Apple が iOS アプリのデザインについて定めているガイドラインである Human Interface Guidelines[5] では、主要な画面遷移の構造として Hierarchical Navigation(図 2.1)、Flat Navigation(図 2.2)、Content-Driven or Experience-Driven Navigation(図 2.3)、Content-Driven or Experience-Driven Navigation(図 2.3) が挙げられており、本分析のサンプルとなったアプリケーションでは Flat Navigation の中に Hierarchical Navigation があるような構造になっている。

実際の SwiftUI における画面遷移を司る UI である NavigationView[6]、TabView[7] でも画面遷移を司る UI、画面のタイトルとコンテンツは別で実装され、画面遷移を司る UI の上にコンテンツが乗る形式で実装されているため本研究ではそれに則り、自動生成手法の提案ではこのような画面遷移を司る UI は切り離し、画面のコンテンツの UI に注目した。

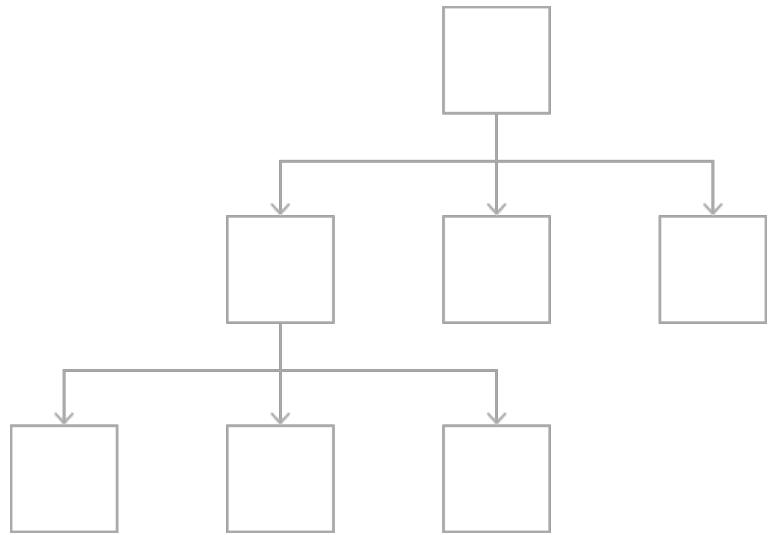


図 2.1: Hierarchical Navigation[5]

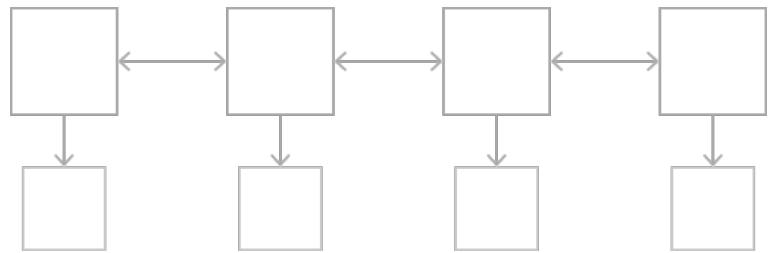


図 2.2: Flat Navigation[5]

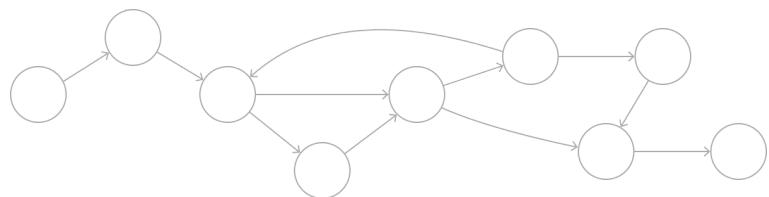


図 2.3: Content-Driven or Experience-Driven Navigation[5]

### 2.1.2 画面の内容を表示する UI

前述では画面内の UI について画面の内容を表示する UI を画面遷移を司る UI が内包していると述べた。本セクションでは画面の内容を表示する UI のレイアウトは主に画面内に表示する要素 (UI パーツ), その要素同士の優先度, そして要素同士のグルーピングの 3 つのルールに則って構成されており, そのルールさえ示せば画面を構成できるのではないかという仮説を立てた。要素の種類が少ない限定的な機能を持つアプリケーションを開発する環境では, グルーピングと優先度から UI を自動生成する研究 [8] がされている。

次にレイアウトを構成する 3 つのルールについて述べる。

#### (1) 要素

画面内にどの要素がいくつ存在するのかを定める必要がある。UI 分析 (図 2.4, 図 2.5, 図 2.6) 右図の灰色の長方形で要素の種類と数を表した。

#### (2) 優先度

画面内の要素の種類と個数がわかったら次に画面内でその要素の重要度を明確にする必要がある。例えば, 同じ SNS サービスであっても写真をメインとする Instagram と文字をメインとする Twitter では文字と画像の優先度の比重が変化する。

#### (3) グルーピング

要素, 優先度に加えて各要素同士でグループがある場合, グループを明確にした。UI 分析 (図 2.4, 図 2.5, 図 2.6) では緑枠, 青枠がグルーピングを表した。またグルーピングの入れ子構造では要素の表示の方向を縦方向, 横方向と順に切り替えることで多くの UI が実装されていた。UI 分析 (図 2.4, 図 2.5, 図 2.6) では横方向配置のグループを青枠, 縦方向配置のグループを緑枠で示した。多くの UI でグループの入れ子の場合, 縦横が交互に入れ替わっていることがわかる。

例外として Facebook(図 2.5) 上部のユーザが投稿を行う要素ではライブ, 写真, Room のボタンのグループが横に並んでいるが, 写真を投稿するボタンとライブや Room を始めるボタンが隣にあるのは機能がかけ離れているため, ユーザの誤解を招く。しかし使いやすさよりも Facebook が Live 機能や Room 機能へのリーチを優先したためにこのような配置になっていると推測した。

## 2.2 アルゴリズムの構築

次にこの要素, 要素同士の優先度, 要素のグルーピングから 1 画面の UI を構成するアルゴリズムを検討した。要素を優先度の高い順に画面内に縦に並べることを基本とし, グル

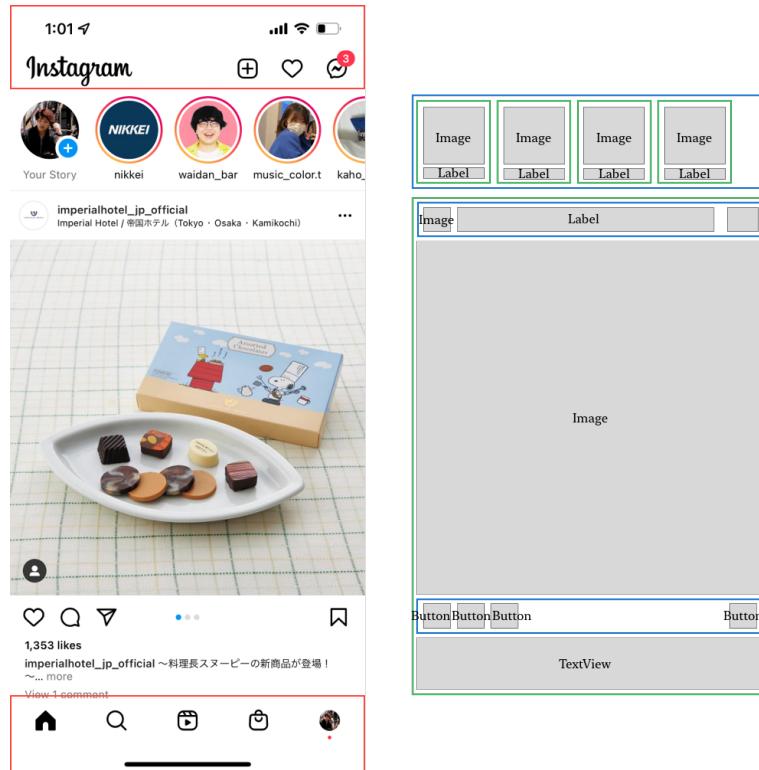


図 2.4: InstagramiOS アプリの UI の分析

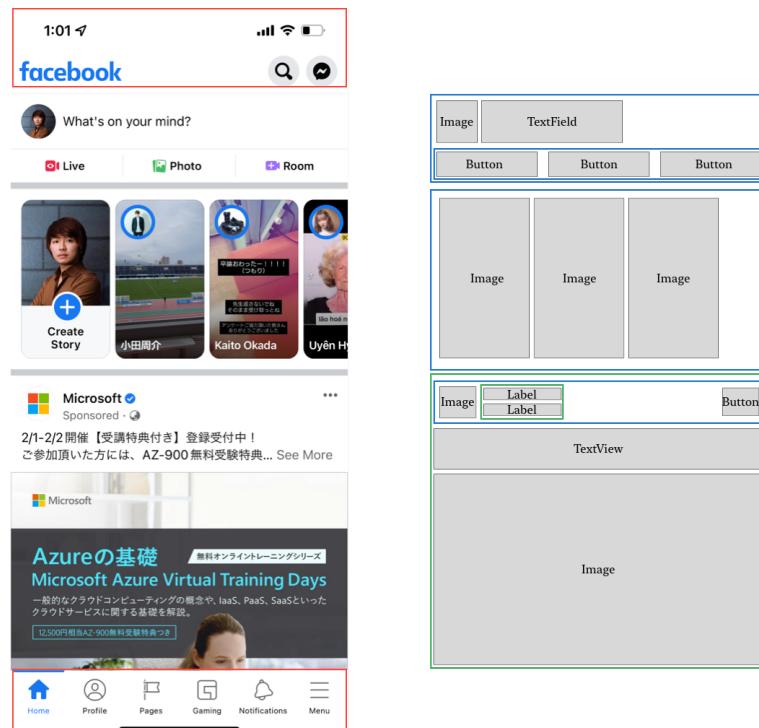


図 2.5: facebookiOS アプリの UI の分析



図 2.6: 日経電子版 iOS アプリの UI の分析

は一つの要素として配置する。また、グループの優先度はグループ内の要素の優先度の合計とした。しかしながら、要素の中にはユーザが操作するもの、見るだけのものがあり、操作するものは指の届きやすい画面下部に配置するのが適切である。各要素の種類に応じて別途配置係数を定め、配置係数と優先度を組み合わせて画面内に配置する場所を決定する必要がある。Group が入れ子になる場合は Hstack, VStack を交互にすることによって UI を構成する。画面内の要素の優先度の基準を 10 とし、それを超える場合はスクロールするようとする。優先度が上がる度に文字は大きく、太く、ボタンもただの青文字ではなく、border がついたり、塗り領域が増えたりと視覚的にも目立たせることで優先度をより効果的に扱う。



# 第3章 関連研究と関連プロダクトの整理

## 3.1 ノーコードでのシステム開発

大規模な開発を行うリソースがない企業や専門知識をあまり持たない個人でもアプリケーションやウェブサイトを作れるように近年、ノーコードと呼ばれるサービスが登場した。ノーコードとは文字通り、コーディングを行わずにアプリケーションやウェブサイトを作れるサービスである。ここでは既存のノーコードのサービスの例をいくつかあげ、特徴と課題点を挙げていく。

### 3.1.1 MIT App Inventor

MIT App Inventor[9] は Google が開発し、現在は MIT Media Lab が運営する、スマートフォンアプリケーションのノーコードシステムである。MIT AppInventor では専門性のない人でも android アプリをコードを書かずに開発することができる。コードは書かなくていいものの、コードを書く作業を GUI 化しているためとても複雑で使いこなすのは難しい上、UI 設計に関しては完全にユーザに委ねられており、質の高い UI を構築するには UI に関する専門知識が求められる。また、android アプリのソースコードとして書き出すことができないため、App Inventor でできることの範囲を超えてしまった場合、アプリケーションを android studio で作り直す必要がある。

### 3.1.2 STUDIO

STUDIO は STUDIO 株式会社が運営するノーコードで Web デザインを行い、リリースできるツールである。デザインの自由度が高いのが特徴で、web front-end の専門知識はなくても Web デザインが行えるが、デザインの専門知識は求められる。自由度の高いビジュアルデザインを売りにしているが、細かなインタラクションなどを追加でソースコードを書くことで補えない STUDIO のできる範囲内で行わなければならない。素人ではなくある程度知識のある人が使うものであるにもかかわらず、STUDIO で制作した web サイトは一目でわかつてしまうため、サイトの製作者はコードを書けないと閲覧者に印象付けてしまう。

### 3.1.3 Glide

Google Spread Sheetsなどをデータベースとして扱い、テンプレートベースの簡易なアプリを作れる。社内ツールの自動化や小規模向けで Glide で作ったアプリケーションをプロダクトとしてリリースすることは難しい。

## 3.2 デザインの自動化

デザインの専門知識がない人でもテンプレートに頼らず適切の UI を制作できるように UI の自動生成を行う研究が行われている

### 3.2.1 機械学習を用いた UI の自動生成

#### (1) GAN を用いた自動生成

Tianming Zhao らの GUIGAN[10] では敵対生成ネットワークを用いたアプローチで UI の自動生成の手法が提案されている。機械学習を用いたシステムでは出力される UI を一意的に決定できないほか、利用者がシステムのアルゴリズムを推測して微調整を行うことが難しく、自由度が下がってしまう。

#### (2) GPT-3 を用いた自動生成

Masum Hasan らが開発した Text2App[11] では GPT-3 を用いて自然言語から MIT App Inventor のコードを生成するシステムである。自然言語でアプリの詳細を伝えるのは難しく、複雑なアプリケーションの開発には向いていない。

# 第4章 iOSネイティブアプリケーションのインターフェース自動生成システムの開発

前述の既存 UI の分析と分析から導き出したアルゴリズムをもとに iOS のネイティブアプリケーションのインターフェース自動生成システムの開発を行った。

また、本自動生成システムを用いて既存 UI の再生成をおこなった他、普段から iOS のネイティブアプリケーションを個人開発している 14-17 歳の男女を被験者とした本システムの有用性をはかる実験を行い、有用性を示すことができた。

本システムの設計は以下のとおりである。

## 4.1 システムの設計・開発

本システムでは Swift プログラミング言語を用いて開発を行った。要素、優先度、グルーピング情報からの画面のコンテンツ UI 自動生成システムのプロトタイプとして主要な部分のみを実装した。本研究において画面のコンテンツ UI とは実際画面に表示される UI のうち、画面遷移を司る UI、画面のタイトル表示を除いたものを指す。

本研究の UI 自動生成において重要なのは画面内の要素、優先度、グルーピング、の情報のみで UI を自動生成できることである。この点を実現するためのシステムを開発した。

本システムへの入力データは要素、優先度をもつ構造体の配列とし、出力は SwiftUI のソースコードとし、画面一枚分の SwiftUI View を出力する。後述の図 4.10 の要素、優先度、グルーピングのインプットデータは以下となる。

```
GroupElement( viewElements: [
    GroupElement( viewElements: [
        ViewElement( ui: .Image, priority: 1),
        ViewElement( ui: .Label, priority: 1),
        ViewElement( ui: .Button, priority: 1),
    ]),
    ViewElement( ui: .Image, priority: 4),
    GroupElement( viewElements: [
        ViewElement( ui: .Button, priority: 1),
        ViewElement( ui: .Button, priority: 1),
        ViewElement( ui: .Button, priority: 1),
    ])
])
```

```
    ViewElement(ui: .Button, priority: 1),  
]) ,  
ViewElement(ui: .TextView, priority: 2)  
] , direction: .vertical)
```

本システムで利用できる要素(UI パーツ)の名称と機能は以下(表 4.1)とする。名称の命名に関しては基本的に SwiftUI での名称を使用するが、SwiftUI ではテキスト表示の行数による名称の違いがないため独自に一行表示を Label, 複数行表示を TextView とした。

本システムでの名称	UIKit での名称	SwiftUI での名称	SwiftUI の操作の有無	ユーザーの操作の有無	主な使用用途
Label	UILabel	Text	無	場合によっては有	一行の文字列を表示するために使用する
TextField	UITextView	Text	有	複数行の文字列を表示、入力、編集するのに使用する	複数行の文字列を表示、入力、編集するのに使用する
TextField	UITextField	TextField	有	一行の文字を入力するために使用する	一行の文字を入力するために使用する
Button	UIButton	Button	有	タップアクションを取得するために使用する	タップアクションを取得するために使用する
Toggle	UISwitch	Toggle	有	ON/OFF の切り替えのために使用する	ON/OFF の切り替えのために使用する
Image	UIImageView	Image	無	画像表示のために使用する	画像表示のために使用する
Map	MKMapView	Map	有	地図の表示のために使用する	地図の表示のために使用する
DatePicker	UIDatePicker	DatePicker	有	日付選択のために使用する	日付選択のために使用する
Slider	UISlider	Slider	有	特定の数値を調整するために使用する	特定の数値を調整するために使用する

表 4.1: 利用可能な要素一覧

### 4.1.1 基本的な配置アルゴリズム

前述のアルゴリズムより、基本的には要素を優先度順に縦に配置する。グループも一要素として扱い表示を行う。グループ内のレイアウトについては後述のグルーピングセクションで行う。

基本的な優先度に加え、UI の要素にはユーザからの操作を受け付けるものと受け付けないものがある。ユーザからの操作を受け付けるものに関しては指の届きやすい画面下に配置するのが適切であり、以下の表 B.1 のように各要素の種類ごとに配置係数を定め、後述の優先度の値と掛け合わせた値の昇順に配置する。

名称	配置係数
Label	1.0
TextView	0.9
TextField	0.8
Button	0.5
Toggle	0.5
Image	2.0
Map	1.5
DatePicker	0.7
Slider	0.7

表 4.2: 各要素の配置係数

### 4.1.2 優先度

本システムにおいて優先度からは要素の視覚的な目立たせ具合、表示順序を決定する。優先度の値は一画面あたりの合計を 10 とし、それを超えた場合はスクロールする画面が提供される。優先度による各要素の目立たせ具合は以下の通りである。以後登場するスクリーンショットは全て本システムによって自動生成された UI である。

#### (1) Label

図 4.1 のように優先度ごとに文字の大きさ、太さを調整して実装した。優先度 5 以上でもこれ以上の大きさになることはなく、それ以上の優先度は配置にのみ反映されるようにした。

#### (2) TextView

図 4.2 のように実装した。前述の Label から文字の太さ、大きさを少し調整すると共に、最大行の指定を優先度 1 の場合は 2 行まで、優先度 2 の場合は 4 行まで、優先度 3 の場合は 5

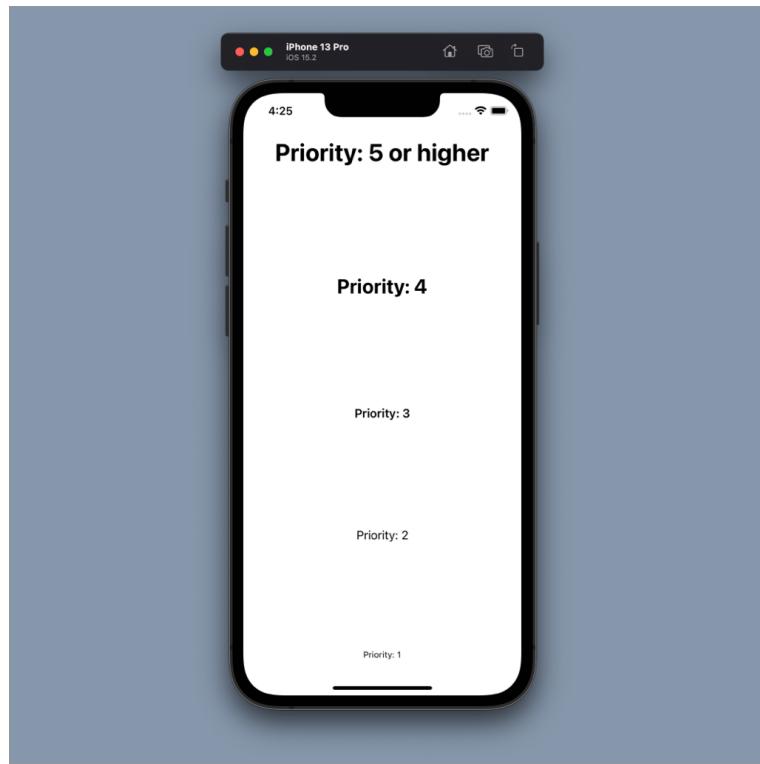


図 4.1: Label の優先度による視覚的目立たせ具合

行まで, 優先度 4 の場合は 8 行まで, 優先度 5 以上の場合は 15 行を最大とした. 優先度 5 以上でもこれ以上の大きさになることはなく, 配置のみに優先度を利用する.

### (3) Button

図 4.3 のように実装した. 優先度によって文字の大きさ, 文字の太さ, タップエリアの大きさ, 色等が変更されている. 優先度 5 以上でもこれ以上の大きさになることはなく, 配置のみに優先度を利用する.

### (4) Toggle

図 4.3 のように実装した. Toggle に関しては優先度で大きさや目立たせ具合が変わることなく, 優先度は配置のみに利用する.

### (5) Image

図 4.5 のように実装した. Image は優先度/ 10 の高さを持つ. 表 4.5 で使用しているサンプル画像が切れずに全てが映るようにしているため. 6 以上でも大きさが変化していない.

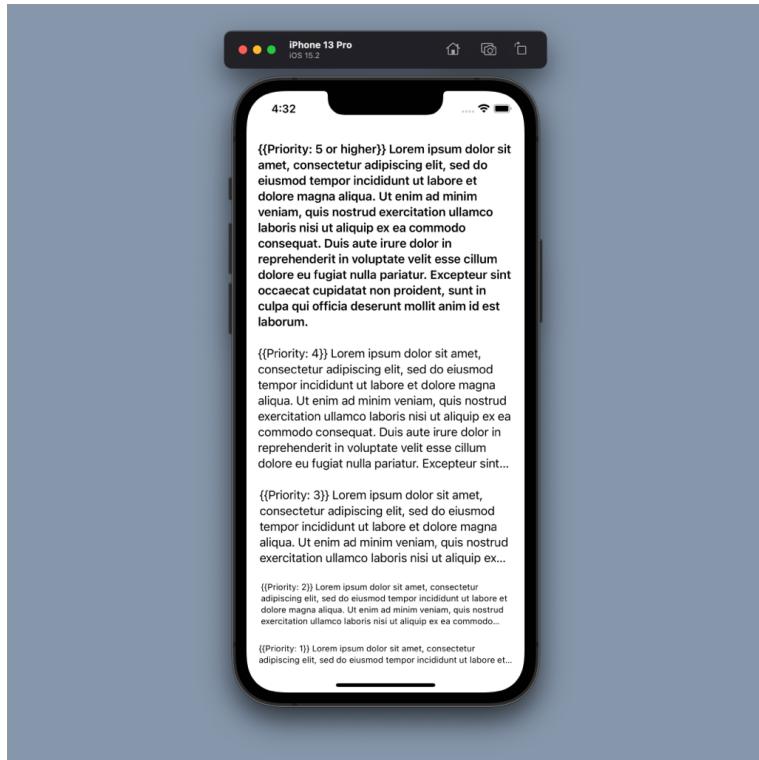


図 4.2: TextView の優先度による視覚的目立たせ具合

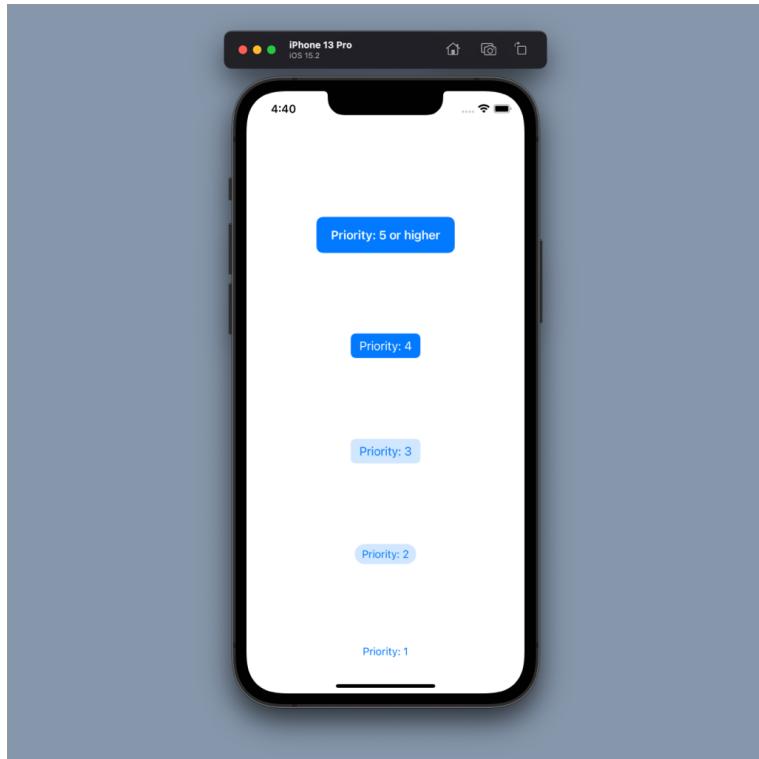


図 4.3: Button の優先度による視覚的目立たせ具合

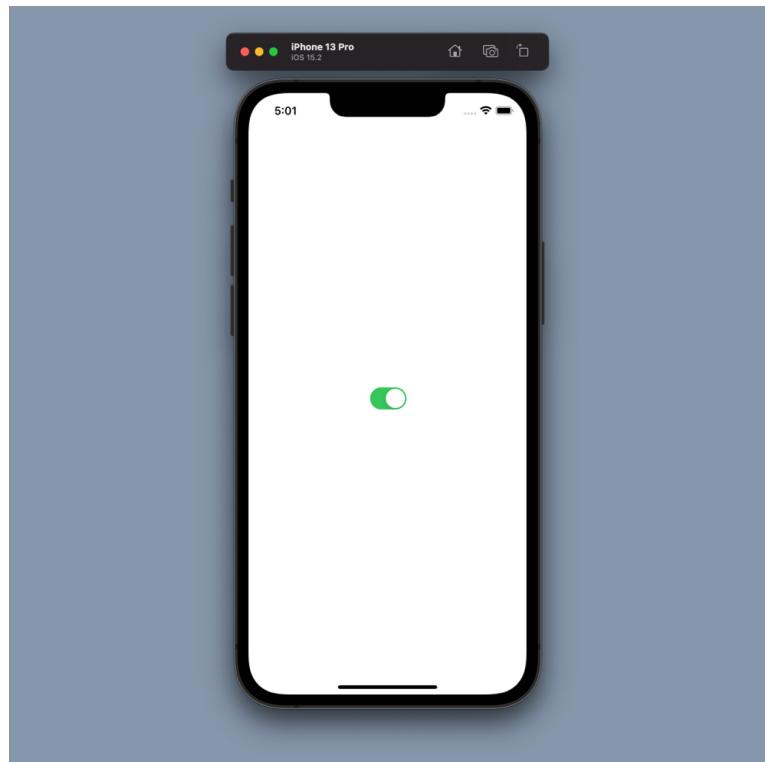


図 4.4: Toggle の優先度による視覚的目立たせ具合

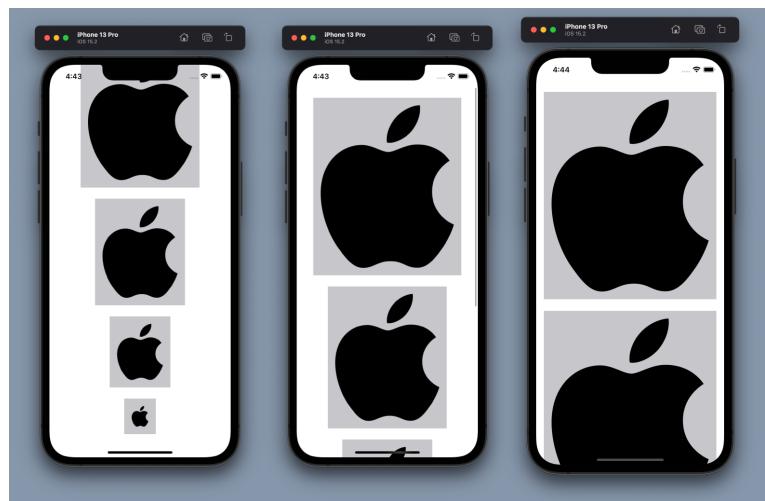


図 4.5: Image の優先度による視覚的目立たせ具合

## (6) Map

図 4.6 のように実装した。本要素も優先度は配置のみに利用し、目立たせ具合は変化させない。また、本要素はユーザの操作を受け付けるが、特例として画面下に持ってはいかない。

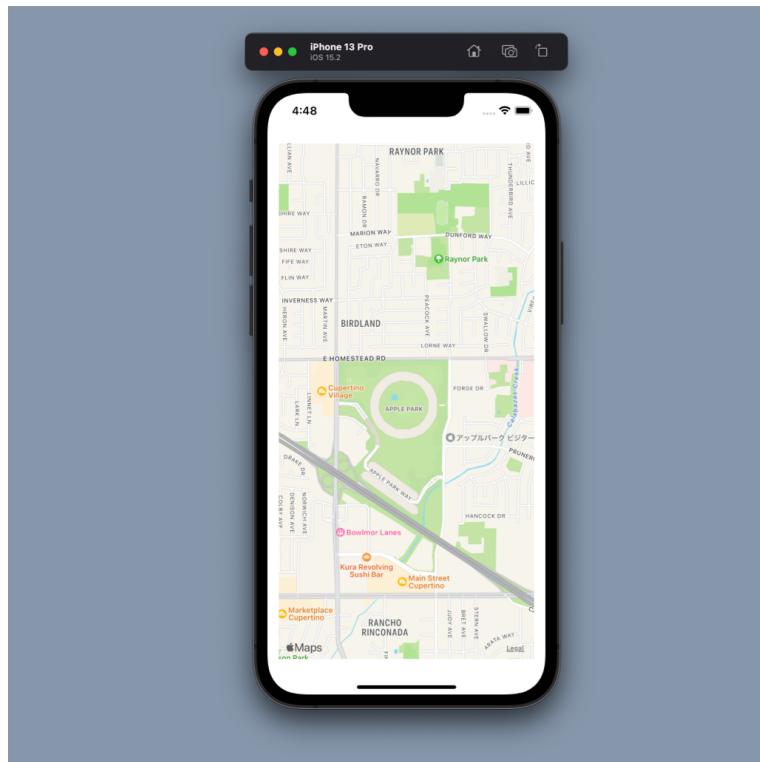


図 4.6: Map の優先度による視覚的目立たせ具合

## (7) DatePicker

図 4.7 のように実装した。優先度 1, 2 では画面右下のコンパクトスタイル、優先度 3 ではドラムロールスタイル、4 以上ではカレンダースタイルで日付を選ぶようにした。でもこれ以上の大きさになることはなく、余白として扱われる。

## (8) Slider

図 4.8 のように実装した。優先度による見た目の変化はなく、配置のみに利用される。

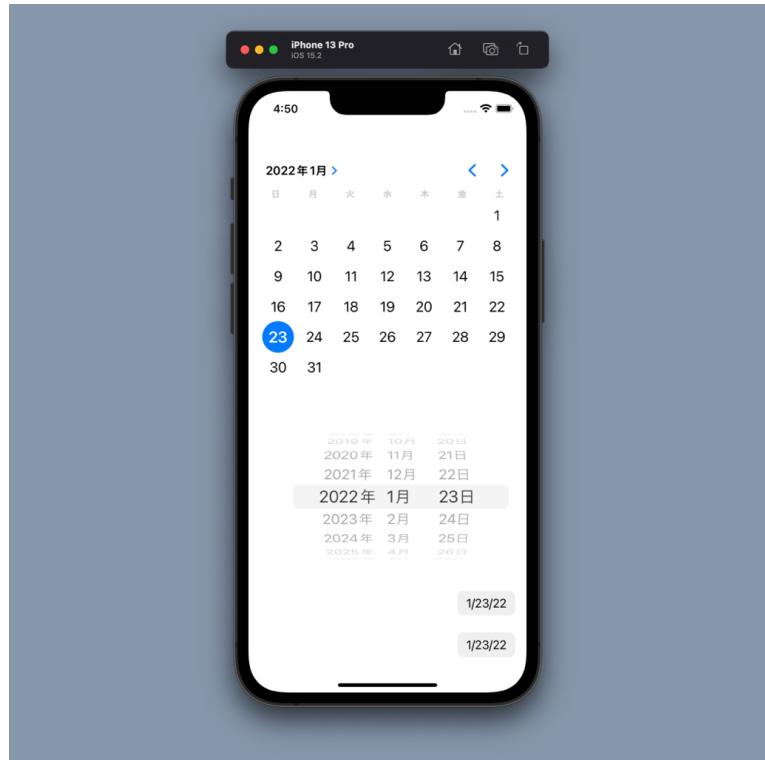


図 4.7: DatePicker の優先度による視覚的目立たせ具合

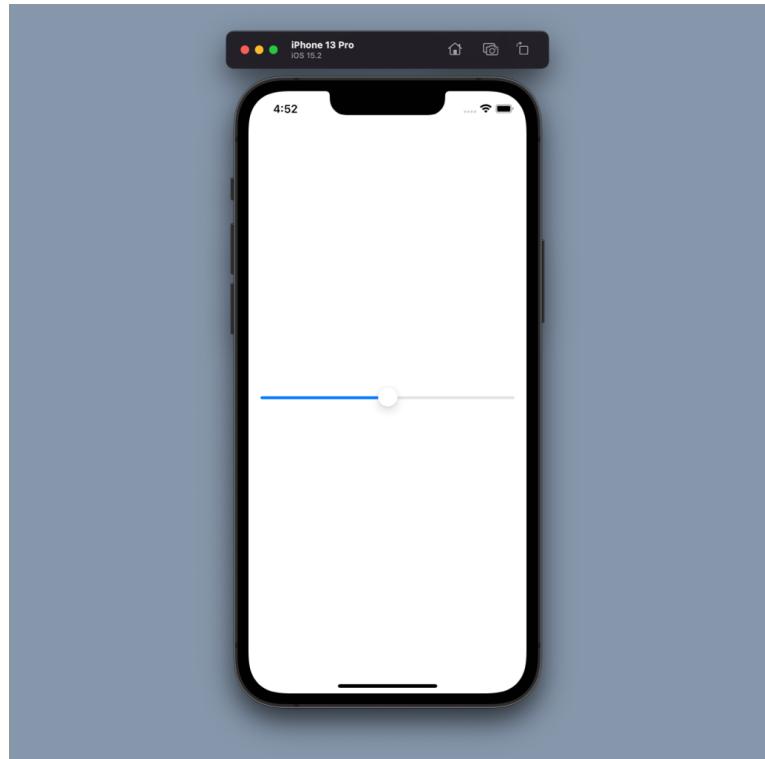


図 4.8: Slider の優先度による視覚的目立たせ具合

### 4.1.3 グルーピング

グルーピングでは前述の要素をグループに入れて表示する。グループ内の要素の優先度から自己の優先度を決定する実装にした。グループの表示は基本的には root Group(自分が他のグループの中の要素ではないグループ)が横方向に配置している。しかしながらそこには横方向には適さない UI の場合もあるので縦方向に指定するオプションも用意した。

#### (1) グルーピングの入れ子構造

グルーピングの入れ子構造では親グループの方向が縦ならば横、横なら縦と、縦方向の配置と横方向の表示を交互で表示するように実装した。前述同様、強制的にグループの方向を決めることもできる。

## 4.2 システムの有効性についての実験

### 4.2.1 既存 UI の再現

前セクションでは本システムの表示アルゴリズムの具体的なシステムの実装を説明した。本セクションでは既存 UI を要素、優先度、グルーピングに落とし込み、本システムを利用して再度 UI 構築を行う実験をおこなった。本実験では SNS フィード画面 (Instagram), ログイン画面 (NIKKEI Wave), リスト画面 (設定), 地図画面 (Maps) についておこなった。

#### (1) SNS フィード画面 (Instagram)

図 4.9 を解析し、図 4.10 の要素、優先度グルーピングを得ることができた。図 4.10 の構造を複数個本システムにインプットしたところ図 4.11を得ることができた。

#### (2) ログイン画面 (NIKKEI Wave)

図 4.12 を解析し、図 4.13 の要素、優先度グルーピングを得ることができた。図 4.13 の構造を複数個本システムにインプットしたところ図 4.14を得ることができた。

#### (3) リスト画面 (設定)

図 4.15 を解析し、図 4.16 の要素、優先度グルーピングを得ることができた。図 4.16 の構造を複数個本システムにインプットしたところ図 4.17を得ることができた。



図 4.9: 既存の SNS フィード画面 (Instagram)

## Group

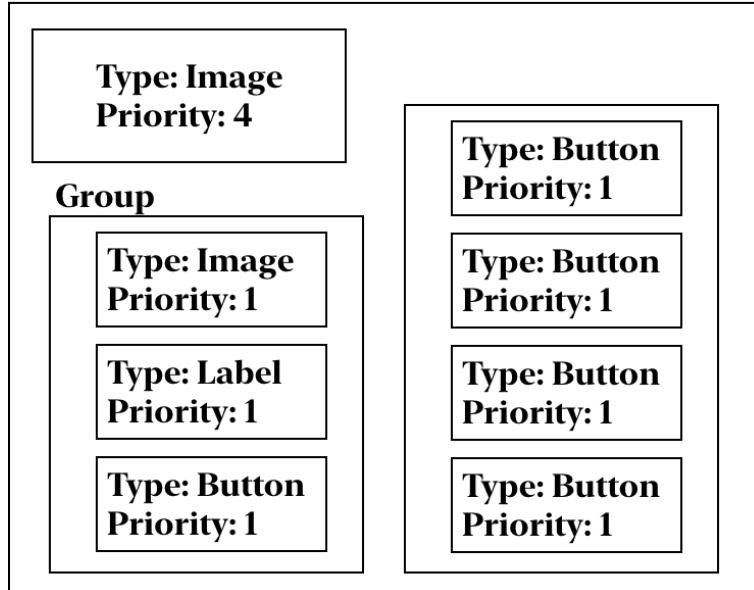


図 4.10: 既存の SNS フィード画面 (Instagram) の要素, 優先度, グルーピング



図 4.11: 本システムで再構築された SNS フィード画面



図 4.12: 既存のログイン画面 (NIKKEI Wave)

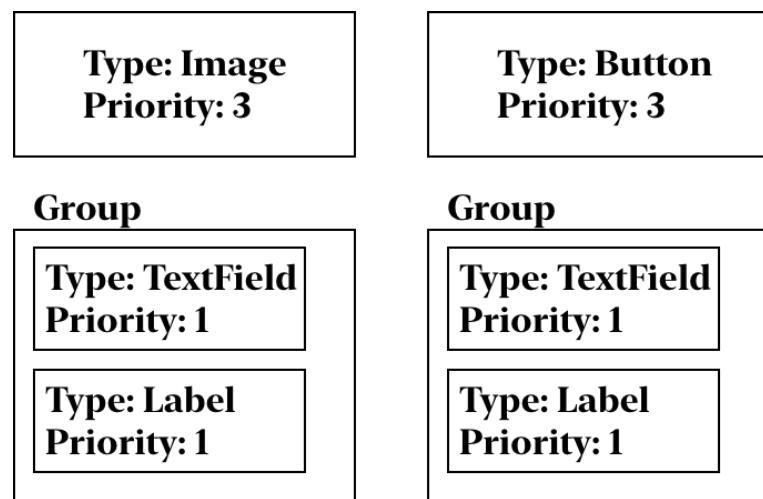


図 4.13: 既存のログイン画面 (NIKKEI Wave) の要素, 優先度, グルーピング

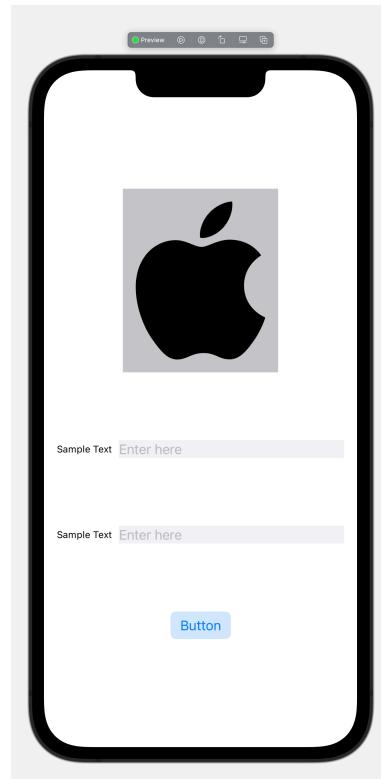


図 4.14: 本システムで再構築されたログイン画面

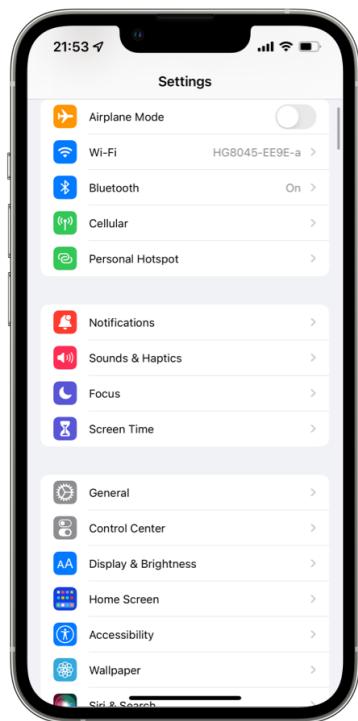


図 4.15: リスト画面 (Settings)

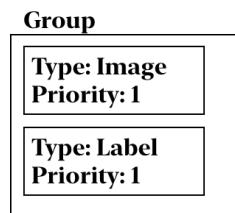


図 4.16: リスト画面 (Settings) の要素, 優先度, グルーピング

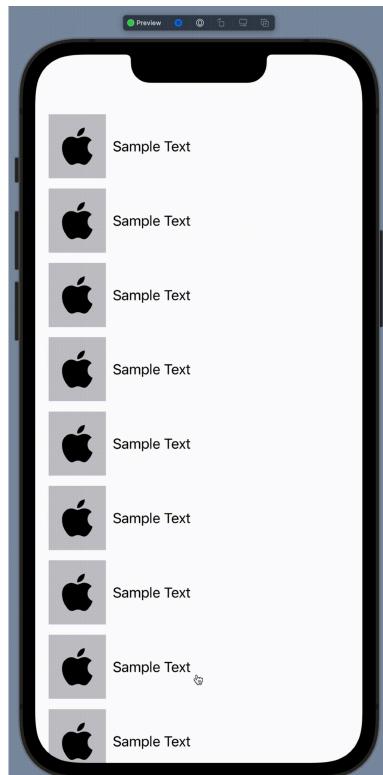


図 4.17: リスト画面

#### (4) 地図画面 (Maps)

図 4.18 を解析し、図 4.19 の要素、優先度グルーピングを得ることができた。図 4.19 の構造を複数個本システムにインプットしたところ図 4.20を得ることができた。

#### 4.2.2 ユーザテスト

普段から iOS ネイティブアプリケーションの企画、設計、開発を行う 14-17 歳の中高生男女を対象に設計段階のアプリにおいて本システムを利用し UI の自動生成の有用性を測る実験とアンケートを実施した。本ユーザテストでは従来の UI デザイン手法ではなく、画面内の要素、優先度、グルーピングを抽出することの難易度調査、期待する UI が自動生成されるかどうか、また優先度を編集することでの本システムとのインタラクションによる UI 設計があるかを調査した。

#### 4.2.3 対象と手続き

本ユーザテストではコンピュータの基本操作は理解しており、日頃から iOS ネイティブアプリケーションの企画、設計、開発をおこなっている中高生を対象とした。



図 4.18: 地図画面 (Maps)

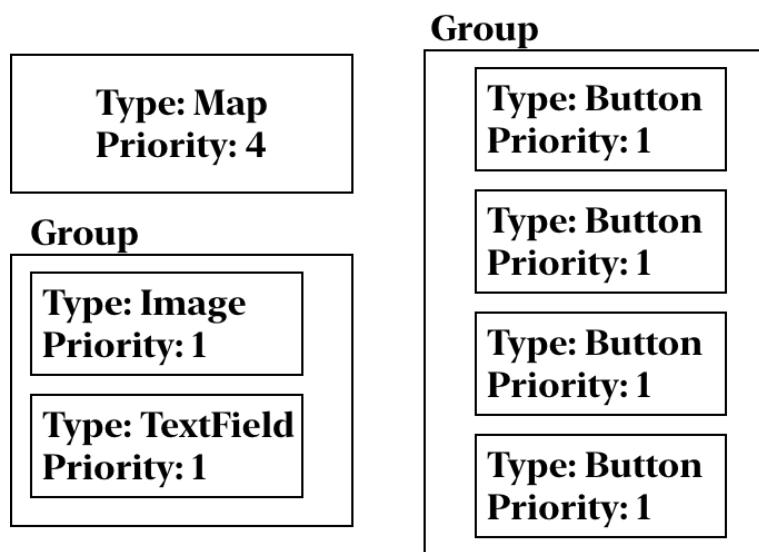


図 4.19: 地図画面 (Maps) の要素, 優先度, グルーピング

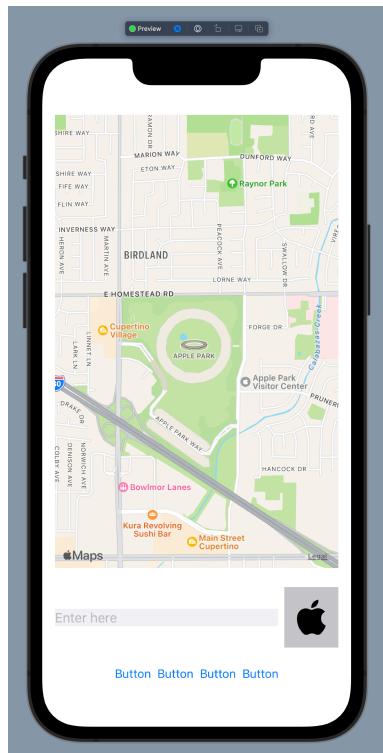


図 4.20: 地図画面

被験者には事前説明として画面内の要素、優先度、グルーピングで構成できる旨の説明を行い、使用できる要素の種類一覧(表 4.1)を示した。

実験では白紙一枚を渡し、白紙に図 4.10, 図 4.13, 図 4.16, 図 4.19, にならった要素、優先度、グルーピングの図示を行わせ、自動生成システムにかけ実際に UI を出力させた。

#### 4.2.4 結果と考察

本実験を通して、既存 UI を本システムで再現すること、ある程度アプリ開発について知識のある人なら本システムを利用して UI を生成することが可能であることが明らかになった。またアンケート結果によると(付録 A, B を参照)UI の生成を自動化することで効率化が図れることがわかった。

また普段デザインの見栄えを考えるあまり UI としての本来の目的を見失いがちになるとの意見もあった。本システムはそのような場合にでも本来の目的に向かって進むことができるツールになり得る。

そして UI の設計開発ではダークモード対応、多画面サイズ対応、アクセシビリティ対応等工数はかかるものの、単純な作業が必要である。この部分についてもアンケートで実装する上で大変だと思うことで回答されており、本自動生成システムを使用することで自動で実装されるため本システムはとても有用であるといえる。

o Map 4  
 o Text 2  
 o Date picker 2  
o Button 1  
 o Text 1

図 4.21: 被験者 A 記入用紙

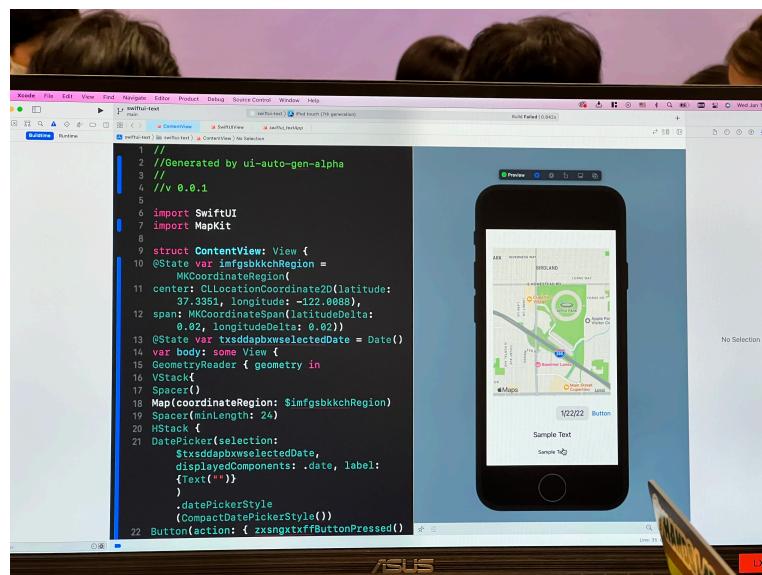


図 4.22: 被験者 A 自動生成 UI

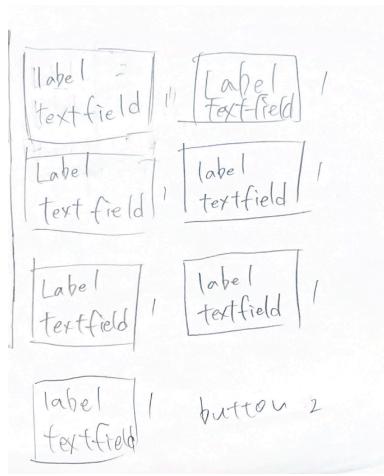


図 4.23: 被験者 B 記入用紙

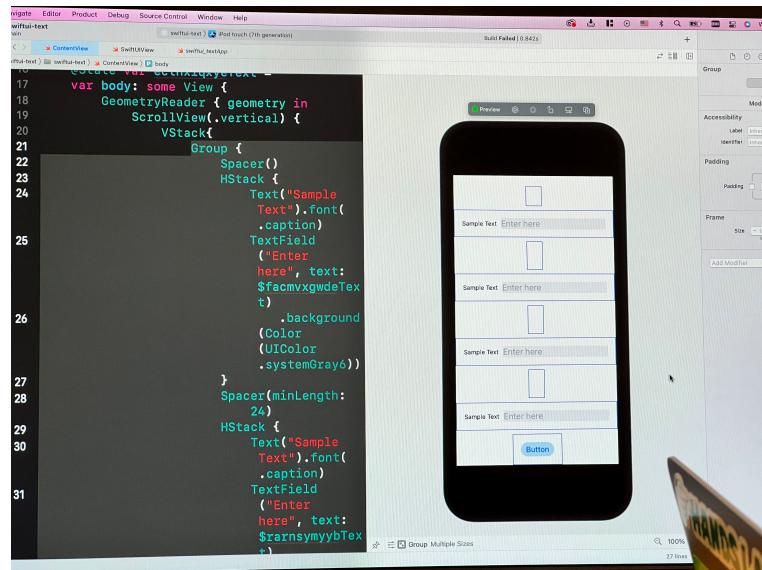


図 4.24: 被験者 B 自動生成 UI

!Image x1 (1)  
 !Label (3).  
 Button x2 → Group (2)  
 TextField x1. (4)

図 4.25: 被験者 C 記入用紙

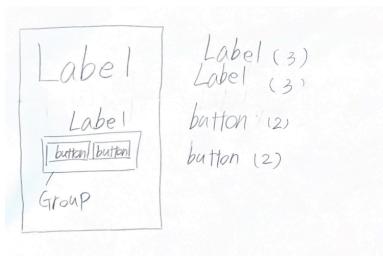


図 4.26: 被験者 D 記入用紙

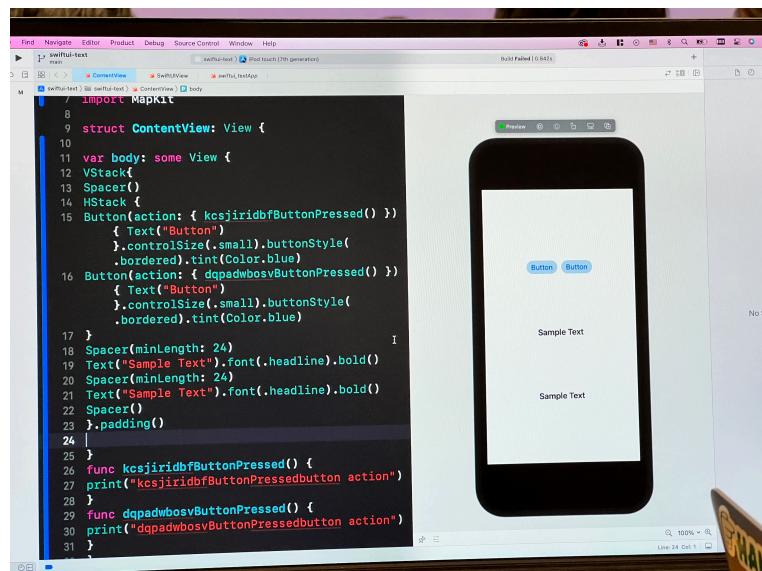


図 4.27: 被験者 D 自動生成 UI

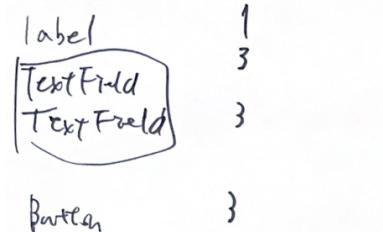


図 4.28: 被験者 E 記入用紙

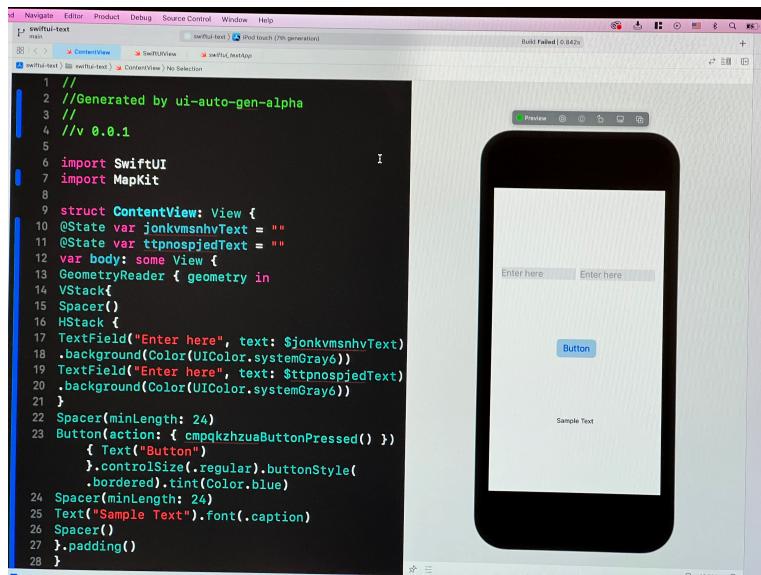


図 4.29: 被験者 E 自動生成 UI

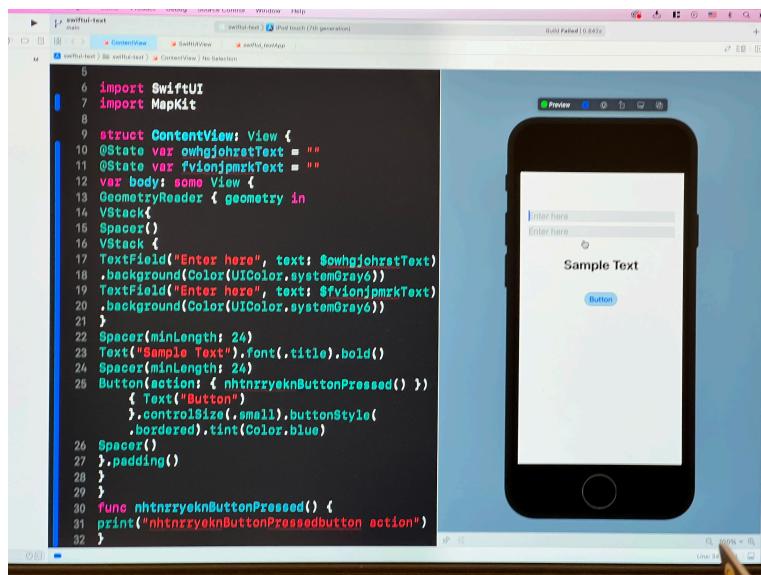


図 4.30: 被験者 E 自動生成 UI-2



# 第5章 結論

前章までで UI の表層的なデザインは画面内の要素、その優先度、グルーピングで行うことができるのではないかという仮説を立てた。そして仮説をもとにシステムを実装し、既存 UI の再現、ユーザテストを経て本システムの有用性を示すことができた。本研究により現在では大規模な開発でしか行われていない専門性の高いユーザインタフェースデザインのハードルを下げ、表層のビジュアルデザインに捉われずに、奥深くのデザインに注力する手法を示すことができた。

## 5.1 今後の課題

要素、優先度、グルーピングから UI を自動生成するシステムの開発に成功したものの、現状の本システムはアプリケーションとしての完成度はとても低く、今後の統合的なシステムとして完成させる必要がある。以下に今後の課題と展望を述べる。

### 5.1.1 要素、優先度、グルーピングの入力 UI の検討と開発

本研究における UI 自動生成システムではユーザが直接システムを使うことはできず、用紙に要素、優先度、グルーピングを記述してもらい、筆者がシステムが受け付けるコードに変換することで実現している。

今後は要素、優先度、グルーピングを効果的に直感的に入力できるインターフェースの研究開発が必要であり今後の修士課程で、研究を進めていきたい。

### 5.1.2 画面遷移を司る UI の自動生成

本研究では画面の UI を画面遷移をつかさどる UI と画面のコンテンツを表示する UI の二種類に分類して自動生成をおこなった。画面遷移を司る UI に関しても法則性を導き出し自動生成を行えるようにしていきたい。これについても今後の修士課程で研究を進めていきたい。

### 5.1.3 配色の自動生成

実際の UI 設計開発では配色はとても重要であり、色によってアプリケーションの見た目や、サービスのプランディングにも大きく影響してくる。

本研究の自動生成システムでは iOS アプリでは最もベーシックな白を基調としブルーをアクセントカラーとするデザインが採用されている。ここに関してもサービスのプランディング、ユーザビリティに適した色を提案するシステムを提案したい。これについても今後の修士課程で研究を進めていきたい。

### 5.1.4 スライダー UI による表層デザインの調整

本研究における自動生成システムでは UI の表層的な見た目は一意に決定されてしまい、自由度がない。プロトタイプやβ版のアプリケーションであれば問題はないが、リリースするプロダクトに本システムを用いるためには UI をある程度調整できることが不可欠である。これを Adobe Lightroom に習い、スライダーでアプリケーション全ての画面を一括に直感的に変更するシステムも現在研究中であり、本研究のシステムと統合することでより自由度の高く、使いやすいデザインをデザインの専門知識なくとも実現することができる。こちらに関しても修士課程で研究を進めていきたい。

### 5.1.5 統合的なシステムとしての展望

本研究で開発した UI 自動生成システムを基盤とし、前述の課題点を解決するアルゴリズムを開発し、macOS 向けのネイティブアプリケーションでの開発を予定している。以下の図 5.1、図 5.2 は現状の統合的なシステムのモックアップである。

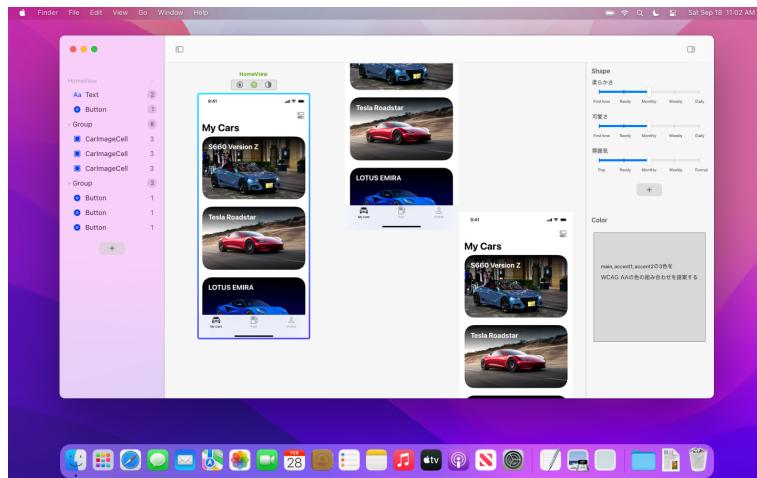


図 5.1: 統合デザインシステムのプロトタイプ

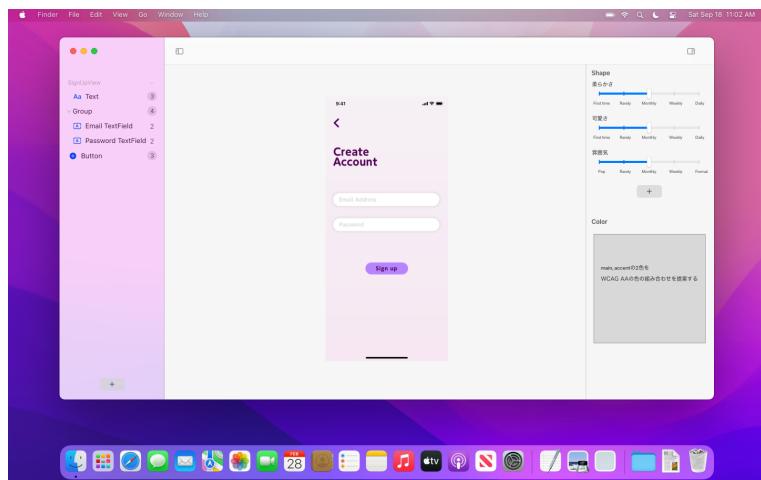


図 5.2: 統合デザインシステムのプロトタイプ 2



## 謝辞

本研究の遂行にあたり、研究室聴講時代から終始適切な助言を賜り、また丁寧に指導してくださった増井俊之教授に深く感謝申し上げます。増井俊之教授には本研究のみならず、日頃からプロダクト、アイディアに対しても NOTA CTO としてのプロダクト開発の側面のご意見、研究者としての学術的なご意見を多数賜りました。感謝申し上げます。増井研究会では、博士課程の田中優氏、修士課程 OB の左治木隆成氏には様々なご意見をいただき充実した研究会活動をさせていただきました。卒業生の皆様、研究会メンバーの民様にも多くのご支援をいただきました。

また同輩であり、孫正義育英財団 [?] 正財団生であり、Bridge UI 株式会社 [?] 代表取締役社長兼 CTO の佐々木雄司氏には常に気の置ける友人として、日々的確で新しい意見をいただいた他、TeX の環境構築にも多大なるご支援をいただきました。

そして1年次よりデザインに関する終始適切な助言を賜り、丁寧に指導してくださった鳴川肇准教授に感謝申し上げます。また鳴川研究会メンバーの皆様にも日々たくさんのご支援をいただきました。

学生時代公私に渡り終始お世話になりました、ライフィズテックの社員、メンターの方々、実験に協力してくださったモンスター班のメンバーの皆様にも日々たくさんのご支援をいただきました。感謝申し上げます。

公私に渡りご指導、ご支援いただきました皆様に心から感謝申し上げます。

2022年1月 吉日

尾崎 正和



## 参考文献

- [1] Brad A Myers and Mary Beth Rosson. Survey on user interface programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 195–202, 1992.
- [2] Goodpatch Inc. Goodpatch の原体験. <https://goodpatch.com/design>.
- [3] Jesse James Garrett. *The elements of user experience : user-centered design for the Web and beyond*. New Riders, Berkeley, CA, 2010.
- [4] 日本経済新聞社. 外部勉強会におけるデザインに関する登壇資料一覧. [https://hack.nikkei.com/slides/?categories\[\]=%E3%82%A4%E3%83%9A%E3%83%BC](https://hack.nikkei.com/slides/?categories[]=%E3%82%A4%E3%83%9A%E3%83%BC).
- [5] Apple. Navigation - App Architecture - iOS - Human Interface Guidelines - Apple Developer. <https://developer.apple.com/design/human-interface-guidelines/ios/app-architecture/navigation/>.
- [6] Apple. NavigationView — Apple Developer Documentation. <https://developer.apple.com/documentation/swiftui/navigationview>.
- [7] Apple. TabView — Apple Developer Documentation. <https://developer.apple.com/documentation/swiftui/tabview>.
- [8] 横沢邦一, 中所武司. エンドユーザ主導型アプリケーション開発環境 M-base におけるユーザインタフェースの自動抽出・生成技法の実現と評価. Technical Report 23(2001-SE-136), 明治大学大学院理工学研究科基礎理工学専攻情報科学系, 明治大学大学院理工学研究科基礎理工学専攻情報科学系, March 2002.
- [9] MIT Media Lab and Google. MIT App Inventor. <https://appinventor.mit.edu>.
- [10] Tianming Zhao, Chunyang Chen, Yuanning Liu, and Xiaodong Zhu. GUIGAN: Learning to Generate GUI Designs Using Generative Adversarial Networks. <https://arxiv.org/abs/2101.09978>, 2021.
- [11] Masum Hasan, Kazi Sajeed Mehrab, Wasi Uddin Ahmad, and Rifat Shahriyar. Text2App: A framework for creating Android apps from text descriptions. <https://arxiv.org/abs/2104.08301>, 2021.



# 付録A UI自動生成システムユーザテスト アンケート質問紙

1. 年齢
2. 性別 (Biological)  
選択肢
  - 男性
  - 女性
  - その他
  - 回答したくない
3. 普段、どれぐらいパソコンを使っていますか  
選択肢
  - 1. ほとんど触らない
  - 2. 得意とは言えないが、調べもの程度になら使える
  - 3. コンピュータは日頃から利用していて、プログラミングでものづくりもおこなっている
4. 普段アプリ開発において、デザインとはどのようなことをおこなっていますか。
5. 今回の自動システムを使った率直な感想をお願いします。
6. 優先度を調整して思い通りのUIに近づきましたか。
7. 普段UIを考える、実装する上で大変だと思うことは何ですか。
8. 全体を通しての感想や意見などなんでも記入してください。



## 付録B UI自動生成システムユーザテスト アンケート回答

### B.1 選択問題回答(問1-3)

年齢	性別	普段、どれぐらいパソコンを使っていますか
13	F	3
15	F	3
16	F	3
17	M	3
16	F	3

表 B.1: 各要素の配置係数

### B.2 普段アプリ開発において、デザインとはどのようなことをおこなっていますか。

- 色々なモチーフを組み合わせて Sketch やイラレに書き起こす
- テーマを決めて統一させる、パーツを置いてみる、色合いを見る
- 載せたい内容を一度整理する。
- Pinterest で UI のデザインを調べたり、友だちや家族に UI のデザインの案を見てもらい、どうしたら、もっと見やすくなるかなどを聞いて、より使いやすい UI のデザイン考えています。
- ラフスケッチ、ワイヤーフレームを作ってみたりして、ユーザーに見せたい順番で、文字の大きさや配置を考えてどの順番で追っていくかを考える

### B.3 今回の自動システムを使った率直な感想をお願いします。

- すげえ!! デザインするとき時短になりそう
- 自分でやらないといけないことが自動でできて嬉しかった
- 画期的で素晴らしいと思いました。
- UI のデザインを考えるとき、かなり悩むので、悩まなくてよくなるのは、とても便利だと思いました。

- UI パーツと優先順位の数の二つを決めるだけで実装できるようになるのが簡単で普段時間がかかる実装も試しやすかった

#### B.4 優先度を調整して思い通りの UI に近づきましたか.

- 近づいた
- 近づいた
- 近づいた。
- 近づいた !! 優先順位を決めるだけで大きさを勝手に調節してくれるから、デザインに悩む時間が減って実装しやすくなると思った !!
- 近づきました。

#### B.5 普段 UI を考える、実装する上で大変だと思うことは何ですか.

- バランス、押しやすいボタンの幅を考えたりすること
- パーツの場所を考える、コンセプト？を決める
- デザインの見栄えを考えるあまり、UI としての本来の目的を見失いがち。
- どのような UI が直感的に使えるか、どうしたら見やすくなるかを考えている。考えたデザインをどうやってダークモードとライトモードに対応させるかが大変だと思っている。
- AutoLayout を調節したり、配置したりすることが大変

#### B.6 全体を通しての感想や意見などなんでも記入してください.

- AutoLayout が楽になりそう！みんなが使ってハッカソンで同じ UI が揃うの想像したら面白い
- コードを書かなくて済んで嬉しい
- 情報の優先度と載せる情報を整理して、デザインを考える過程を楽にできるので、UI 構成の効率性を図ることができます。
- UI を作るときに、オートレイアウトを設定するのが面倒くさいので、優先順位をつけるだけで見やすい UI になるのはとても画期的だと思った。
- 簡単に実装できるようになるのが楽しくてこれから使えるようになったら、どんどん使ってみたい！