平成２７年度

卒業論文

題目	Androidアプリ「Select-Color」の 企画と開発

担当教員 （自署）		印

学籍番号　201214058

氏　　名　木村　正紀

広島経済大学

Android アプリ『Select Color』の企画・開発

201214058　木村　　正紀

担当教員　　田中　　章司郎　　教授

要旨

このアプリケーションを作成した目的は、Android を用いた男性向けファッションアプリケーションが少なかったので、その基盤として画像に対して色を付けることが出来るようにする「メイキング系アプリケーション」を作成しました。

第 1 章のはじめには、簡単な概要説明です Google Play の Android アプリケーション作成に用いた開発環境などの簡単な説明です。

第 2 章から第 3 章にかけては、近年のモバイル産業についてふれ、スマートフォンがどのようにモバイル産業との関わりを持っているか簡単にまとめました。

第 4 章から第 6 章までは今回作成した「 Select Color 」についての概要とソースコードの説明、今後の追加したい機能について述べました。

第 7 章は、全体を通してのまとめと、「 Select Color 」を作成して Android アプリケーション作成に抱いた感想です。

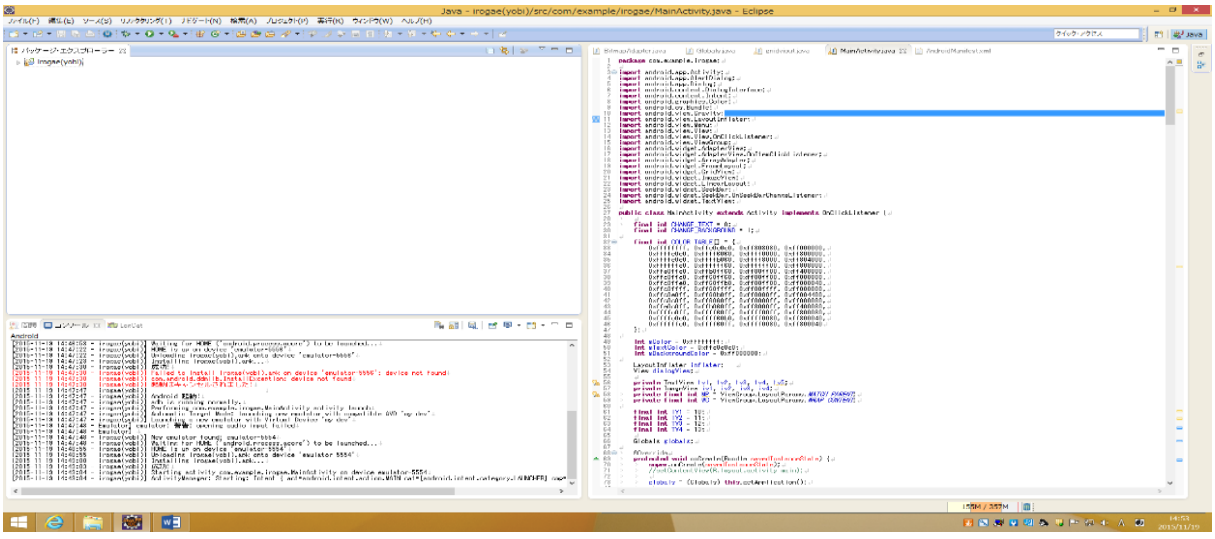
広島経済大学

第１章　はじめに

近年モバイル産業の上昇にともない、アプリケーション市場にて App Store と Google play が認知度も高くスマートフォンを使用する上では、欠かせない存在の１つと言えます。そこで Android に注目をおき、Google playでのアプリケーションがどのように開発されているか考え、 Android アプリケーションを制作します。

そして、私が今回開発した「 Select -Color 」という Android アプリケーションはオープンソースとして公開されている scarlet氏 (引用文献 : a)のソースコードを基盤に開発しています。

開発動機は、当初はカラーパレット系のアプリケーションを作成していたのですが、様々な意見があったのでファッション向けに開発を進め様々な服装（テンプレ）に対し、自分好みに色を配置してみて、配色の目安にして欲しいなと考え開発しました。開発環境は、 Eclipse （図１）を採用しております。今回のアプリケーションは作成途中の過にありますが、その開発過程と共に、このアプリケーションの将来的に実現できそうな可能性を本論文で、提唱します。



（図１　開発環境　 Eclipse ）

第 2 章　モバイル産業とスマートフォン業界の関わりについて

2.1 近年のモバイル産業について

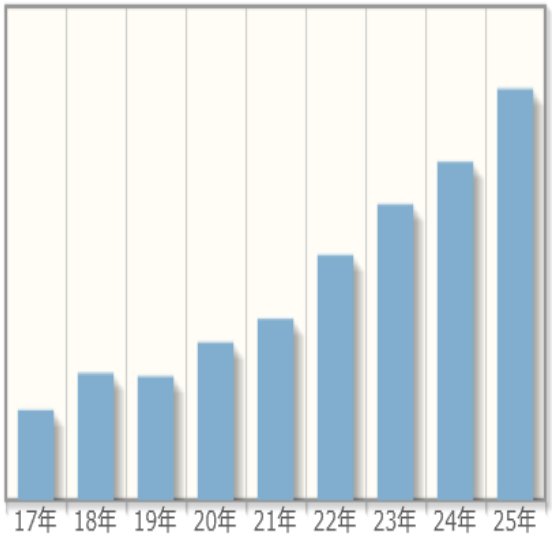


図 2　引用：業界動向 SEARCH . COM

近年のモバイル業界の現状と動向

近年、モバイル産業の業界規模は図 2 から読み取れるように年々右上がりにあります。平成 25 年から 26 年のモバイル産業の業界規模（主要対象企業 43 社の売上高計）は 1 兆 0,613 億円となっています。急速な発展を遂げているモバイル産業ですが、ゲーム業界のスマホ向けアプリケーションの力によってここ数年の順位変動は激しくなっています。例として 2 つ上げると、1 つの例は順位が上がった例で、平成 25 年から 26 年売上高で 2 位に浮上した「ガンホー・オンライン・エンターテイメント」（以降ガンホー）。ガンホーはスマートフォン向けアプリ「パズル＆ドラゴンズ」が大ヒットし、平成 25 年 12 月決算では、1,630 億円を計上。平成 24 年の 258 億円から +531.8% の大幅増を記録しています。もう 1 つの例は逆に順位が下がった例で、平成 24 年売上高では 2 位を記録していたグリーですが、5 位に大きく転落しました。理由としては、平成 24 年に社会問題化した「コンプリートガチャ問題」以降ヒットに恵まれず、売り上げは伸びない状況にあります。グリーの業績は急上昇し一転、急転落を記録しました。

以上のように、モバイル産業は「若年層」を中心として回っており、今の「トレンド」の変化は速いのです。普及する時の速度は爆発的に増えますが、同時に飽きられるのも速く、モバイル産業業績は急上昇し急降下する傾向にあるのです。以上がモバイル産業の現状ですが、これはスマートフォンが急速に普及されたためです。「パズル＆ドラゴンズ」は、スマートフォンアプリケーション向けに開発されています。モバイル産業の中でもスマートフォン業界は、近年での重要なビジネスとして注目されています（引用文献：b)。

2.2 スマートフォン業界の近況

モバイル産業の中でもスマートフォン業界の競争が激化するのと同時に、著しい成長を遂げています。スマートフォン業界では、携帯電話用のプラットフォームには、 Apple 社の iOS 、Google社の Android OS 、 Microsoft 社の Windows Phone 、 Mozilla の Firefox OS など様々なプラットフォームあります。

スマートフォン販売された初期は、iPhone先陣切って発売されていたので、 2009 年時点では iOS が Android を上回っていました。 2011 年頃には、 Android も注目されるようになり 2013 年中頃には、 Android が 50%程度でiPhoneが 45%程度で半分半分と均衡するといったかたちになりました。ですが近年では Android が勢いに乗っておりシェアを広げています。例を挙げると、 2013 年ころスマートフォンが定着しつつあるときに Apple 社はiPhone、iPadなどで販売拡大に成功し着々とユーザーを伸ばしていきました。

それに対し、Google社はあまりにもシェアを拡大されすぎ iOS が市場独占をする一歩手前まで上り詰められましたので阻止するべく 2005 年ごろに買収した Android 社が開発した Android OS の開発環境を、一般公開した事により、市場独占を阻止することに成功し、更に様々なメーカーが Android OS を採用し、結果的に Android OS 普及が急速に上がりました。そして 2014 年 3 月時点でのスマートフォンシェア率は、一般公開したことにより様々なメーカーが Android OS を採用しているので Android OS が優勢にたっています。 Android OS を採用しスマートフォン業界に参戦したメーカーサイドも生産・販売で成功し、特に中国や韓国、台湾などのアジア系企業が飛躍しています。中でも、「Samsung」（大韓民国　水原市　本社）は Android OS を採用しスマートフォン業界参戦し、その結果右の図 3 から解るようスマートフォンベンダー別世界出荷台数 2 年の連続で、 70 %以上キープすることに成功しています（引用文献：c）。

2014年4～6月のスマートフォンベンダー別世界出荷台数（単位：百万台）

ベンダー名	2013年4～6月	2014年4～6月
Samsung	76.0	74.5
Apple	31.2	35.2
Huawei	11.1	20.1
Lenovo	11.3	15.8
Xiaomi	4.1	15.1
LG	12.1	14.5
その他	87.2	120.0
Total	233.0	295.2

出典：Strategy Analytics

図 3

その一方、 Apple 社の iOS は 32.1% と出荷台数が少なく劣勢かというとそんなことはなく、 Apple ユーザーが多い国を見てみると米国、イギリス、カナダ、日本など主要な西側市場では首位をキープしています。具体的に数値で比較すると Cult of Android での調査では、 101 ヶ国での調査にてGoogle社の Android OS が 67 ヶ国優勢で、 Apple 社の iOS が 34 ヶ国で優勢である Apple 社の iOS とGoogle社 Android OS の話題は持ち上がりましたが、などほかのプラットフォームはどうなったのかというと Apple 社とGoogle社が市場拡大に成功しユーザーもしっかり押さえているので，状況としてはあまりよろしくありません。 Apple 社とGoogle社以外でプラットフォームのシェアが優勢にたっているのが 101 ヶ国中 2 ヶ国しかなく、 1 ヶ国はカナダの通信機器メーカーのBlackBerry社（旧社名： Research In Motion Limited ）が優勢にたっている南アフリカで39%をシェアしていて。もう 1 ヶ国は、フィンランンドの電気通信機器メーカーの NOKIA 社が優勢にたっているバングラデシュで26%をシェアしています。

以上が、 2014 年 3 月時点の世界のスマートフォン業界の状況である。ちなみに 2014 年 3 月時点の日本のプラットフォームのシェア率は iOS が約65% で Android OS が約30% で、ウェブブラウジングの約95%を占めているそうだ（引用文献：d）。

2.3 Android についての簡単な説明

Android とは、 2003 年にアメリカのカリフォルニア州パロアルト市に携帯電話向けソフトウェアプラットフォームを開発している「 Android 社」という会社のことでした。 2005 年に、モバイル検索や携帯電話からのインターネット強化を進めていたGoogle社（米国）は、創設２２ヶ月という創設したての Android 社を買収しました。その後 2007 年11月頃、無償で利用できる携帯電話用のプラットフォームとして Android を発表したのです。 Android は、「 Linux カーネル」（複数のアプリケーション起動した時、メモリーやハードディスクなどの制御を上手く連携するため補助する役割）をベースとした携帯電話用の OS やミドルウェア、アプリケーションをすべて含んだ環境です。 Android のメリ

ットは、やはり Android OS を搭載するスマートフォンを作る上で、ライセンス料金がかからないことと、 Google Play や Google Earth などのサービスとの連携が確立しているところです。更に、開発環境も無償で公開されているので、一般の開発者も無料でアプリケーション作成ができるメリットもあります。メーカー側にとっても端末を製作するコストを安く抑えることができます（引用文献： e ）。

第 3 章　既存のデザイン系に関りのある様々アプリケーションの状況

ここでは、初めに有用性の高いアプリケーションの紹介と第２章も踏まえアンケート結果の考察を行います。

3.1 写真加工・画像編集系アプリケーションの状況

カメラアプリと同様に、Playストア上には無数に存在しますがその中でも安定して高評価を得ている凡庸アプリケーションは、「 Snapseed 」と「 Autodesk Pixlr 」、「Aviary」です。特殊な画像加工に特化したアプリケーションからは、「 TouchRetouch Free 」と「スケットグル」となります。ここで紹介するのは安定して高評価を得ている凡庸アプリケーションの内「 Autodesk Pixlr 」（図４）を紹介します。硬軟バランスの取れた画像加工アプリケーション。多数の調節項目とエフェクト、オーバーレイなどがポイント。コラージュ機能も充実しています。更に画像の変更とかサイズなどを変更できますので自分好みのレイアウトができるようになっています。

図４　Autodesk Pixlr 提供元　Autodesk Inc.

コラージュ操作中の画面

「引用元　Autodesk Pixlr　提供元：Autodesk Inc.

http://appllio.com/the-best-android-apps-100　」

特殊な画像加工に特化したアプリケーションからは「スケットグル」を紹介します。撮影した写真を、アプリケーションにインポートすると、ワンタッチでカラースケッチや水彩画、ペンシルスケッチなどに加工しちょっとした芸術家気分が味わえるようなアプリケーションとなっています。このアプリケーションで画像を編集し SNS　なのでアップロ

ードするのが、評価から読み取れる流れとなっております。 Google Play 内の評価もよくインストール数も 1000 万人を超えているGoogle play内でも注目を集めています。

3.2　メイキングデザイン系アプリケーションの状況

これまでに、 Google Play で配信されているデザイン系のアプリは、幅広く無料とは思えないほどに完成されているものもあります。カラーパレット系でもっとも多いのは、カラーコードについての参考アプリケーションが多いです。これはパソコンのイラストソフトなので使用される RGB 16 進数や RGB 10 進数の数値コードを表示し、アプリケーションによってはWebページ作成時に使用するHTMLや、ゲーム内メイキング時に出てくるCMYKの数値コードも対応しています。そのようなアプリケーションは複数のカラーコードを、その色の和名・英名の正式名称と共に同時に表示することによって、解りやすさと扱いやすさを向上させています。図５と図６を比較して見ても図５も操作性は良いのですが図６のほうが解りやすい印象です。

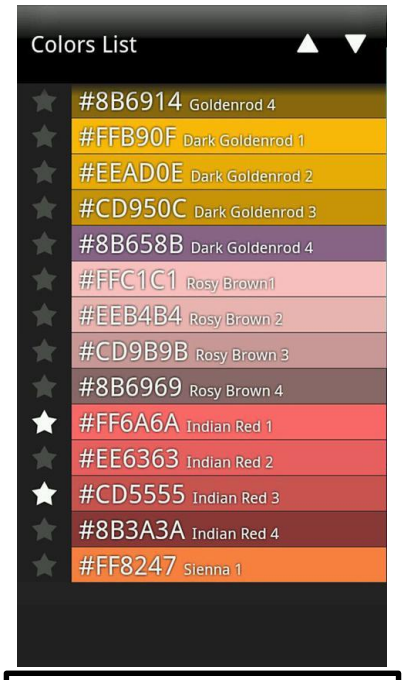


図 5　Color Reference

提供元　D9d9

図 6　色コードブック

提供元　darkdrive.net

「引用元 Color Reference　提供元： D9d9

https://play.google.com/store/apps/details?id=com.dmena.colorreference 」

「引用元　色コードブック　提供元：darkdrive.net

https://play.google.com/store/apps/details?id=net.darkdrive.android.color 」

イラスト系のアプリケーションでは、イラストソフト（絵描きアプリケーションなど）も多種多様です。最近ではイラスト素材を配布するアプリケーションなど配信されているくらいです。イラスト系アプリケーションはここ近年で急増しています。理由としては、イラストアプリケーションを起動するスペックに対してスマートフォンの性能が追い付いてきたことにあります。スマートフォン普及初期はフューチャーフォンに比べると高性能ですが、それでもイラストアプリケーションも大量のデータと処理、更に画面に同時に表示される画像が増えると動作が遅くなるなど色々と問題が多く、必然と性能が制限されてきました。ですが、近年技術が向上傾向でスペックは年々上がり続けるのでスペックが追いつきイラストアプリケーションの自由度が広がりました。例として、「 LINE Brush 」は絵を描くのは勿論、画像の編集もできます。このアプリケーションでの注目点としてなんといってもインターフェイスの多さです。自分で描いてもよし、インポートした写真を加工しそこからフォトブラシやペン入れして加工などバリエーション豊富です。

メイキング系では、部屋のメイキングやインテリアのメイキングなどホームインテリアメイキングデザインが多く、中にはそのメイキングデザインの簡易的ではありますが、総費用見積もりなど出してくれるアプリケーションも存在します。次に、ファッション系ですが多くはネイルアート系やメイクアップ系・着せ替え系が多くターゲット層としては女性向けのアプリケーションが多く見受けられます。ですが、写真加工や画像編集、イラスト系アプリケーションに比べ全体的にメイキング系アプリケーションは少ないのが現状です。

3.3　市場アンケート調査によるアプリケーション所有傾向と考察

最初に、本来は最低 50 人以上にアンケートを取らないと、アンケートとして有意義なデータは得られなのですが、今回は２０代しかアンケートが取れず、大学生の 34 人といった少人数アンケートになっています。

3.3.1　アンケート対象者と実施形式

対象者は、スマートフォン（機種やメーカーは問いません）を所有する経済学部・芸術学部・情報学部の学生にアンケートを取りました。実施理由は、所有数や所有する種類、アプリケーションを選ぶ際で重視している点などの傾向が知りたく調査しました。実施期間は 3 日間でアンケートはチェック形式で以下の図 7 のように質問しました。

問 1 ：年齢を、教えてください

□ 20 代　□ 30 代　□ 40 代　□ 50 代　□ 60 代以上

問 2 ：あなたの、アプリケーション所有数を教えてください

□ 10 個～ 20 個　□ 21 ～ 30 個　□ 31 ～ 40 個　□ 41 個以上

問 3 ：あなたの使っているアプリケーションの種類を、教えてください (複数可)

□書籍　□ニュース / 天気　□写真 / 画像加工　□デザイン系

□動画 / 音楽　□交通 / 地図　□仕事効率化　□ゲーム

問 4 ：１日に起動するアプリの数を、教えてください

□ 1 個～ 9 個　□ 10 個～ 19 個　□ 20 個以上

問 5 ：普段、アプリを利用している時間を、教えてください

□ 2 時間以内　□ 2 時間～ 4 時間　□ 4 時間以上

問 6 ：あなたがアプリを選ぶ上で重視していることはどこですか？（複数可）

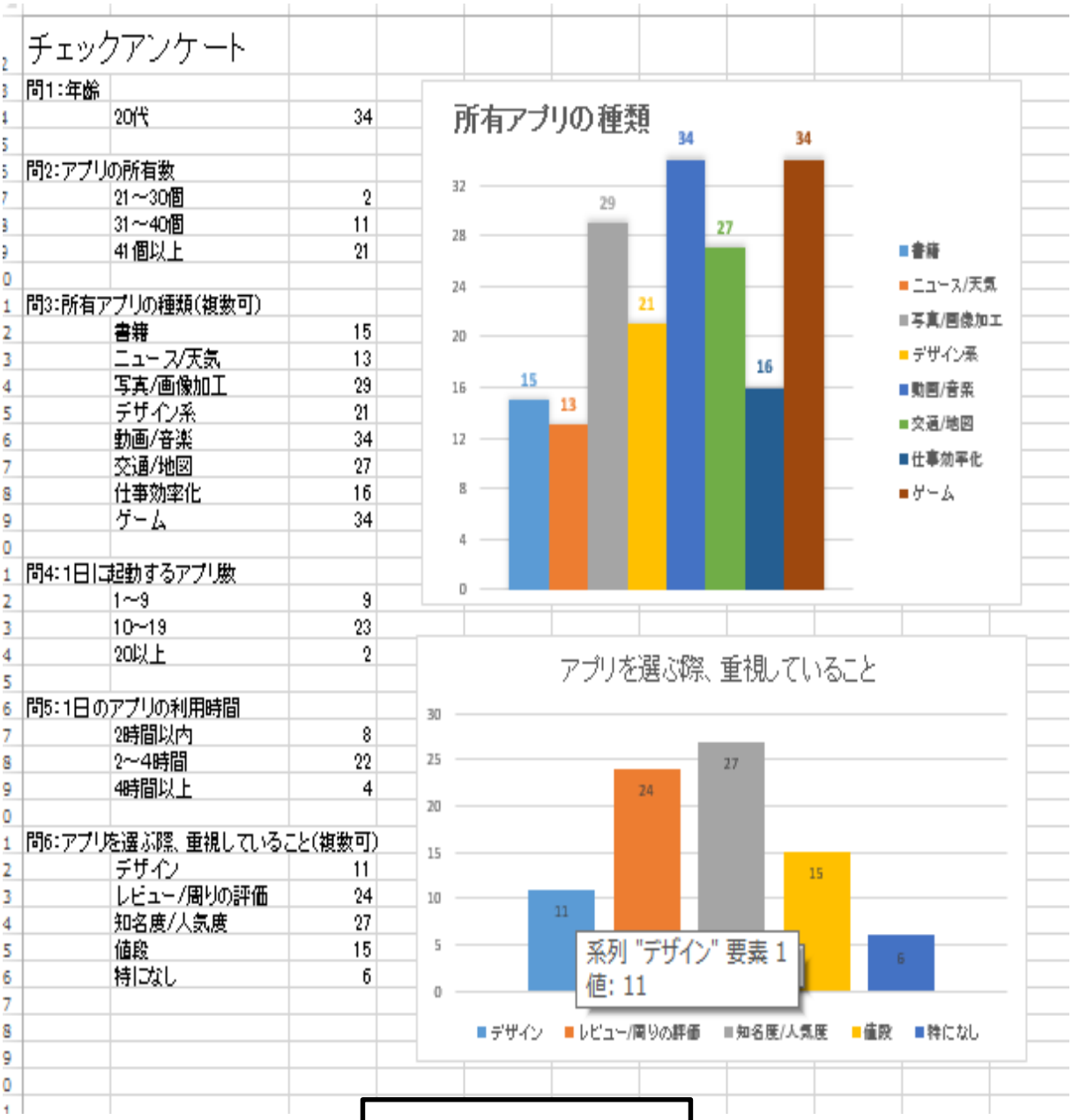
□デザイン　□レビュー / 周りの評価　□知名度 / 人気度

□値段（有料 or 無料含む）□特になし

図 7　アンケート用紙

3.3.2　アンケート結果と考察

チェック式で具体的な数値結果をまとめたデータが以下の図８です。



チェックアンケート

問1:年齢		
	20代	34
問2:アプリの所有数		
	21～30個	2
	31～40個	11
	41個以上	21
問3:所有アプリの種類(複数可)		
	書籍	15
	ニュース/天気	13
	写真/画像加工	29
	デザイン系	21
	動画/音楽	34
	交通/地図	27
	仕事効率化	16
	ゲーム	34
問4:1日に起動するアプリ数		
	1～9	9
	10~19	23
	20以上	2
問5:1日のアプリの利用時間		
	2時間以内	8
	2～4時間	22
	4時間以上	4
問6:アプリを選ぶ際、重視していること(複数可)		
	デザイン	11
	レビュー/周りの評価	24
	知名度/人気度	27
	値段	15
	特になし	6

図８　アンケート結果

所有アプリケーションの種類は、「ゲーム」、「動画 / 音楽」のアプリケーション所有者がやはり多くおられ最多人数を記録しています。次いで「写真 / 画像加工」、「交通 / 地図」といった順位なっております。最多人数を出している「ゲーム」、「動画 / 音楽」の集計アンケート結果からの考察ですが、第 2 章でモバイル産業について説明した際にゲーム業界にも触れましたが、男性は「パズル＆ドラゴンズ」や「モンスターストライク」

など様々なアプリケーションを所有していますが、今回のアンケートではゲームに関心を持たない女性の方でも、「ツムツム」を筆頭とした「 LINE ゲーム」といって、 LINE 内にて関連アプリケーションゲームをプレイするとフレンド同士でスコアを競うことができるサービスがあり、更にディズニーといった人気キャラクターが拍車をかけるといった流れができ、女性は今回のアンケートに関してヘビィーユーザーもいる中ライトユーザーも多い結果となりました。

「動画 / 音楽」に関しては、「 YouTube 」や「ニコニコ動画」などの動画系アプリケーションの利用率は多かったのですが、 2015 年注目を集めた「Vine」の所有率が高く「動画 / 音楽」を選択されたユーザーの 9 割以上が所有していました。このアプリケーションは友達と簡単にビデオメッセージを共有できるということで注目を集めました。更に、面白おかしく撮影した「 6 秒動画」というものを作成し友達同士でシェアすることで人気に拍車をかけました。逆に同じ括りで調査した音楽アプリケーション自体は所有しているのですが利用率が少なくいといった結果になりました。

次に半数以下の結果となっていしまった「書籍 」、「ニュース / 天気」のアンケート結果からの考察なのですが、書籍に関しては漫画アプリケーションの所有が多く小説などは、画面サイズが大きいタブレットでないとアプリケーション内は購入せず、書籍を直接購入するといった方が多い結果になりました。更に、漫画アプリケーションはフェイクアプリケーションも多く有害サイトへリンクで飛んでしまうといった事例も少なくなく、総じて半数を切ってしまった結果になったと考えられます。「ニュース / 天気」に関しても、今回のアンケートではニュース系のアプリケーション所有者は、ほぼ 0 に近い人数となりました。更に、天気系アプリケーションに関しても、最初からスマートフォンに入っているアプリケーションでない限り、インストールせずネットで検索し情報を得るといった流れができているので、こちらも総じて半数を切ってしまった結果になったと考えられます。

3.3.3　アプリケーションを選ぶ際、重視していること

まず、「レビュー / 周りの評価」と「知名度 / 人気度」が頭 1 つ抜けています。これは、「知名度が高い開発グループが開発したアプリケーションを試しにしてみる」→「有用性が高い、面白い」→「人気が出て、周りの評価が上がる」といった流れができ、逆に周りの評価が上がることによって知名度が上がるといったこともあるのです。知名度が上がると様々なメディアなどから特集されさらに注目を集めるといった流れになるのです。

次に、「値段」と続きましたがここで読み取れるのは、有料アプリケーションもちゃんと選択肢に入っているという点です。多種多様な無料のアプリケーションが溢れ返っている中、有料アプリケーションに求めるものは「信頼性」です。 Google Play や App Store などアプリケーションを、配信する際にそれぞれ独自の審査基準に伴った審査し不審な点がないかチェックはされていますが、それでも抜け穴を見つけ審査を通過する害悪アプリケーション（ウイルス感染や情報の抜き取りなど）が存在します。そのような、害悪アプリケーションをインストールしないために有料アプリケーションを選択肢にいれ、提供元や開発元が公的機関かどうか確認し購入するといった流れです。これは、情報社会となりつつある現代では、重要な点と言えます。

デザインは全体の 3 割程度しか気にかけないといった結果になりました。これは、ユーザーにとってデザインがいいのは当たり前のことであり、意識して選択するようなことではないといった結果です。

3.4　モバイル産業についてのまとめ

第 2 章ではモバイル産業の急速な成長の理由やこれまでの経緯についてふれ、第 3 章では評価の高い評価を受けるアプリケーションの紹介し、実際にユーザーがどのようなアプリケーションを所有している調査しました。 2 つの章で共通していえることは、スマートフォンの普及に伴いアプリケーションでの有料サービスが、今まで以上に提供しやすい時代になってきているということです。

第 2 章のでも話が上がり、アンケートからもわかるように「ゲーム」系アプリケーション所有率が高く有料コンテンツ内でもかなり利益が出しやすい部類です。具体的にコンセプトは「基本無料」です。ですが、ゲーム内で他のユーザーと差をつけるために 1 つ当たり高くない有料コンテンツを購入させ利益を出すといった流れを確立しており、その結果が「 2.1 」でも取り上げた、「ガンホー」の「パズル＆ドラゴンズ」に関わっています。

他に芸術系統の中では写真加工や画像編集系アプリケーションが人気で利益も出しやすく何よりもターゲット層が広いのです。具体的どうゆうことなのかといいますと、イラスト系やカラーパレット系アプリケーションは芸術系の方が主なターゲットとなり、そのターゲット層からアプリケーション内有料コンテンツを購入させるのが難しいのです。それこそ、無料提供できる範囲でのアプリケーションなのです。もし、アプリケーション内で購入するのならば芸術系の方は、パソコン用イラストソフト内での購入を行うからです。それ対し写真加工や画像編集は、スマートフォンで写真を撮ることがある方全てが対象となります。写真撮影しSNSに、アップロードする時に画像に一手間加えたいとき、フレームをつけたいなど要望は様々です。そこに、追加要素として有料コンテンツを 50 円から 100 円程度で配信すると、それがユーザーの目に留まれば購入されます。更に、「 3.3.3 」でもふれた知名度が上昇することで、アプリケーションダウンロード数も増える流れにより、有料コンテンツの購入者が増えます。一人当たりの利益は低いですが、それが何千何万もの数字になれば利益は一目瞭然となります。

第 4 章　「 Select -Color 」の設計と仕様

4.1 開発したアプリケーションの要求事項

第 3 章で「メイキング系アプリケーションは全体を見ても少ない」と触れましたが、これはファッション系アプリケーションに着目点を持ち、ファッション系のアプリケーションの中で更に男性向けのものがどのくらい存在するのか調査し少ないことがわかりました。（ 2013 年 7 月時点 Google Play 内調査 ）。 そこで私は、完成形は男性向けファッション系着せ替えアプリケーションをイメージして作成しようと考えました。

調査を行う前は作成時始めの頃はカラーパッレト系のアプリケーションを考えていました。ですがGoogle play内の検索にカラーパレット系アプリケーションが多いので何か工夫を加えたいなと考え、様々な方（特に芸術系学生）に話をしてみてからもファッション系のアプリケーションの中に、男性向けファッションデザインアプリケーションが少ないという意見が多かったこともあり、カラーパレットのアプリケーションの基盤はある程度作成できていたのですが、着せ替えアプリケーションになるように考え取り組ました。類似のアプリケーションに、まだ先の話ではあるのですが 2013 年時点でも一部情報としてアパレルショップに大型スクリーンを設置し、自身を撮影しその場でモジュール化することで画面内フィッティングといった案もあるのです。それと、近いものは要求事項としてはありますが、このアプリケーションは携帯できるアプリケーションという方向で作成していました。

要求事項としては、男性向け着せ替えファッションアプリケーションです。画像（モデル）データがあればアプリケーションに参照し、カラーコーディネートしファッションの参考にしようといった流れになります。具体的な色の名前を解りやすくするために日本工業規格（ JIS ）が規定している慣用色名を、表示できるようしたいです。更に、 RGB 16 進数や RGB 10 進数の数値コードを表示、Webページ作成時に使用するHTMLや、ゲーム内メイキング時に出てくるCMYKの数値コードのパーセンテージの表示など搭載し、芸術系の方がパソコンで作業をしている際に、アプリケーション一つでそのモデルに合わせる

服装や色の参考にしやすくするといった案もあります。

つまり、着せ替え系アプリケーションと一緒に Photoshop やイラストソフトなどで、一部カラーパレットとのような機能も搭載してあれば便利程度で考えています。

4.2 「 Select -Color 」の概要

4.2.1 「 Select -Color 」の全体

本研究で作成したアプリケーションは、 Android Ver.2.2 の Java 言語で作成しました。機能としては、画像（テンプレート）に対して、 16 進数のシークバーの上から赤、緑、青を調整、色の調整が難しいようであれば見本からイメージに近い色を選択しシークバーに戻って色の調整し、画像に色を付けるといった着せ替え系アプリケーションです。

アプリケーションの処理を図 9 で示します。

アプリケーションを実行すると，初期画面の左側に画像（仮作成なのでここでは「頭 」、「胴 」、 「膝 」、 「足」の 4 部位に分けました）を選択するか、右側の Text View を選択すると色の選択ができるようになります。

まず、色を選ぶ場合（右のルート）の解説です。 Text View を選択すると、上側に現段階で選択されている色が表示されています。この状態でシークバーを操作すれば、その表示されている色が変更されます。もし、色の調整が難しいようであれば表示されている色の Text View を選択すると、色の見本が表示され選択すればシークバーの画面に戻り、そこで微調整し終えたら、決定すると反映されます。

次に、画像を変更する場合（左のルート）の解説です。今回は、仮組なので「頭 」、「胴 」、 「膝 」、 「足」の 4 部位に別けて色をそれぞれに区切ることはできているか、グローバル（プログラムの中のもっとも外側のブロックで宣言された変数）から、画像を反映し変更できるかを確認しました。変更したい部位を選択したらグローバルで作成した仮想空間に画面が変わり、そこで変えたい画像を選択すれば、画像に反映されるといった流れです。このソースコードでは、各部位の画像を、一つの仮想空間から持ってきているで、

部位ごとに仮想空間を作成すれば部位ごとにリンクができるので、頭なら頭だけの画像を反映した仮想空間ができます。

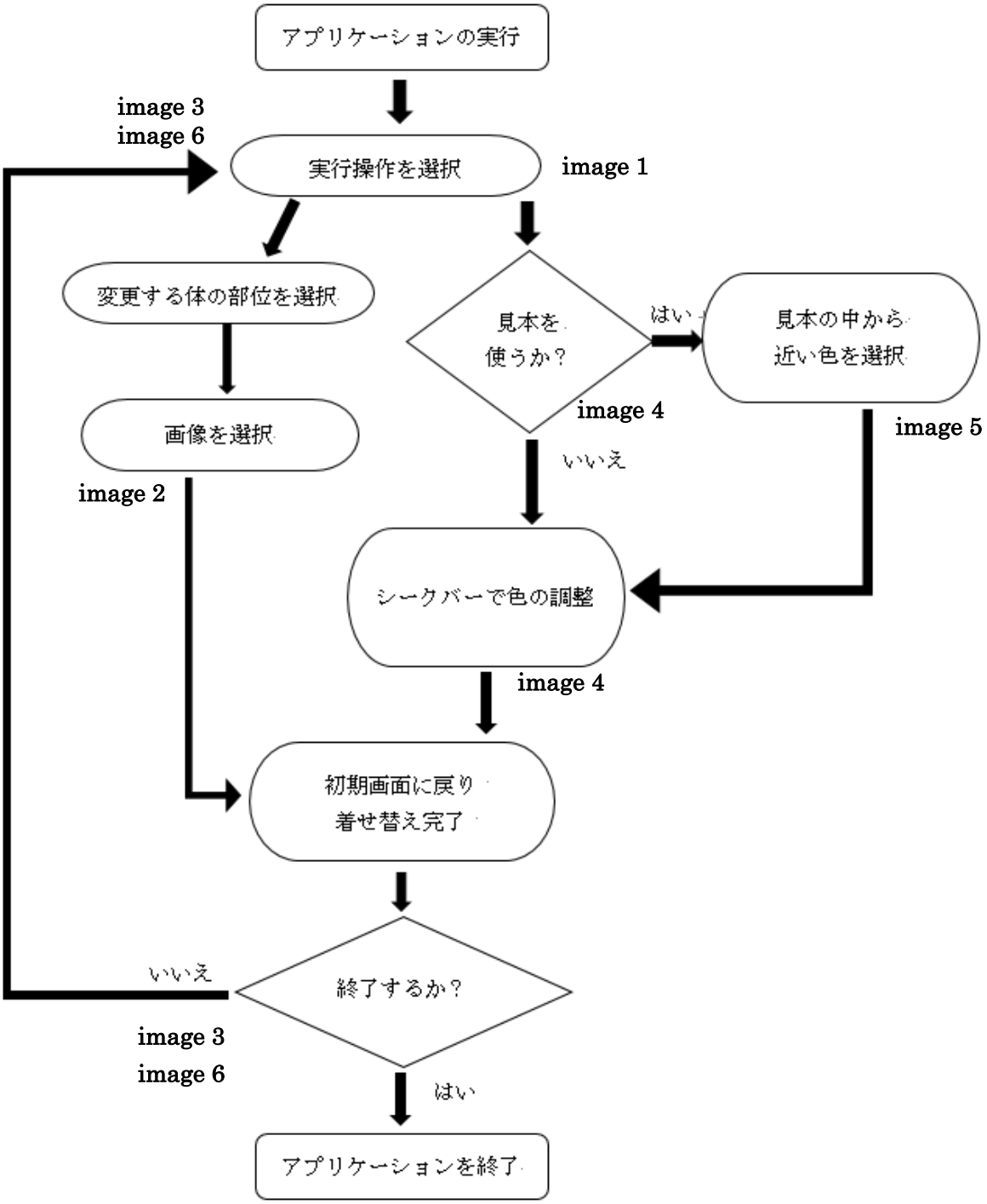
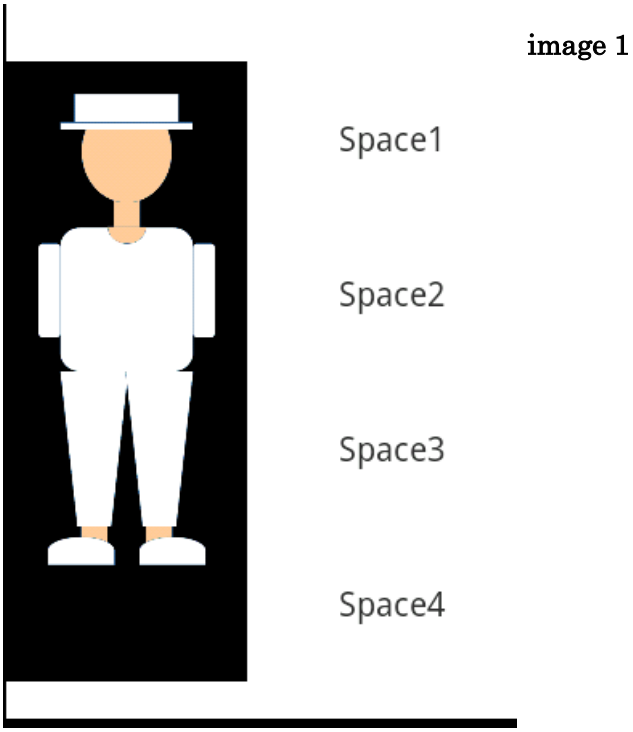


図９　フローチャート

4.2.2　「 Select -Color 」の実行の流れ

ここでは、イメージ画像とともに作成した「 Select -Color 」の流れを説明していきます。尚、「 4.2.1 」のフローチャートと合わせて説明しますので、参照は「image ？？」と説明します。

初期画面は、以下の image1 のようになります。



初期画面では、これではちょっと解りづらいですが上から頭、胴、腰、足と別れています。ここでは、アプリケーション起動時の初期指定カラーは白に設定されています。テキストボックスも初期表示は「 Space 」と、初期設定で表示しています。

（ここまでの説明は、「 4.3.2 」から「 4.3.3 」が対象となっています）

先に画像を選択した場合の流れを説明します。

image2 が、画像選択時の「 Globals 」変数を用いて別領域から参照され選択画面に表示されている画像です。ここでは、動作確認のため初期設定で使用している画像を流用していますので頭をどの部位を選択してもこの状態では全て表示されるようになっています。ここで選択すれば。変えたい部位の画像を変更することができます。

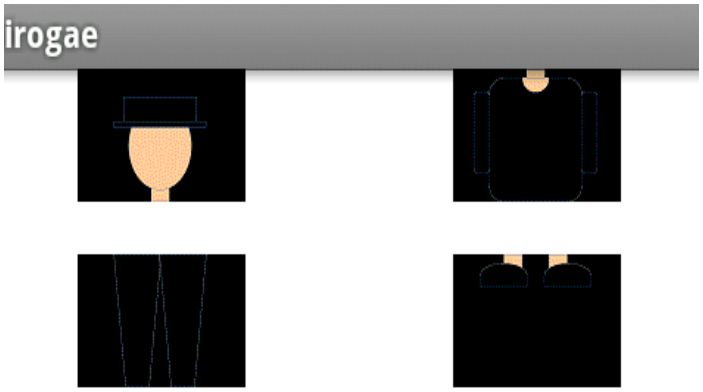


image 2

（ここまでの説明は「4.3.5.1」から「4.3.5.2」までが対象となっています）

選択したらメイン画面に戻ります。下の image3 は画像がちゃんと変わっていることを説明するために、上から足、腰、胴、頭と部位を逆さまにしたものとなります。

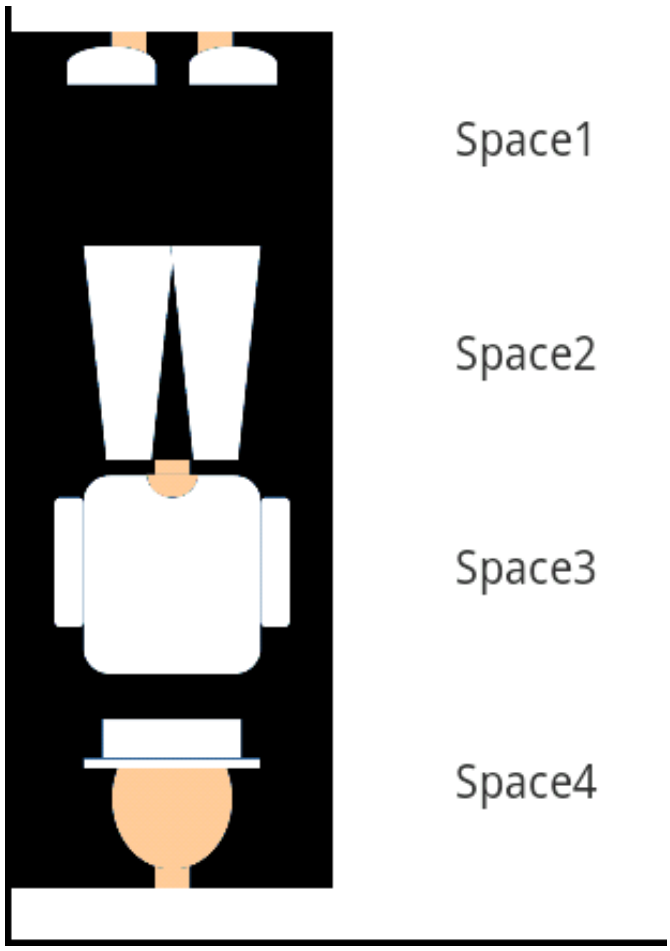


image 3

では、次に色の選択の流れについて説明します。上の image3 の状態で色の変えたい部位を選択します。テキストボックス（ Text View ）をタッチした際、表示される画面が下の image4 のようになります。

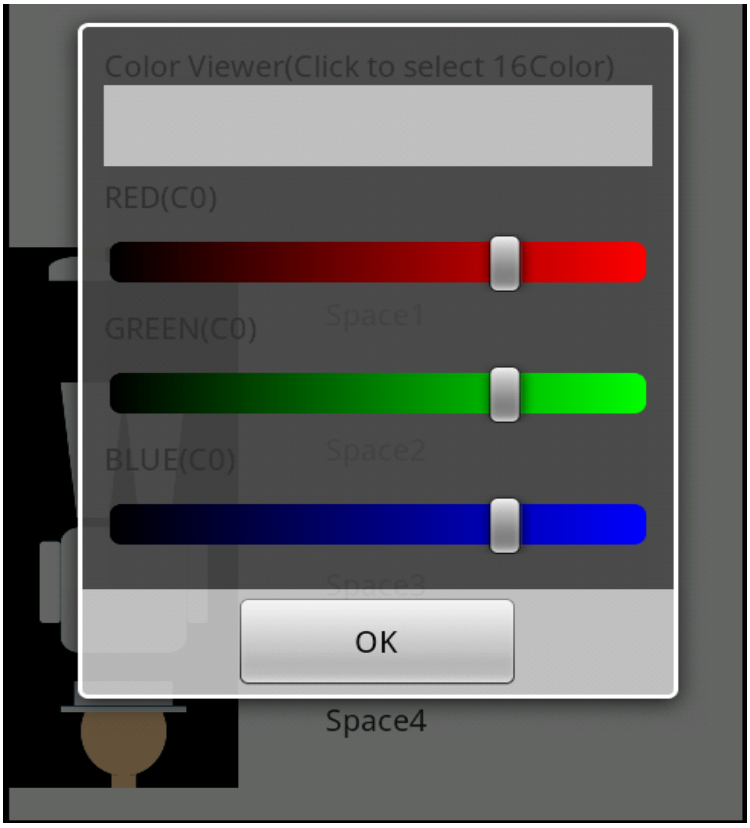


image 4

ここから、更に色が表示されている（シークバーよりも上に表示されている色の部分）を、タッチすることにより image5 画面が表示されます。

この状態で自分のイメージに近い色を選択してもらえれば image4 の画面に戻り、シークバーで微調整することができるといった流れになっています。すでに RGB 16 進数のカラーナンバーが把握し終えているのであれば、ちょっと解りづらいですが image4 のシークバーの「 RED 」「 GREEN 」「BLUE」の隣にカラーナンバーが表示されていますので、そこを見ながら調整ができます。

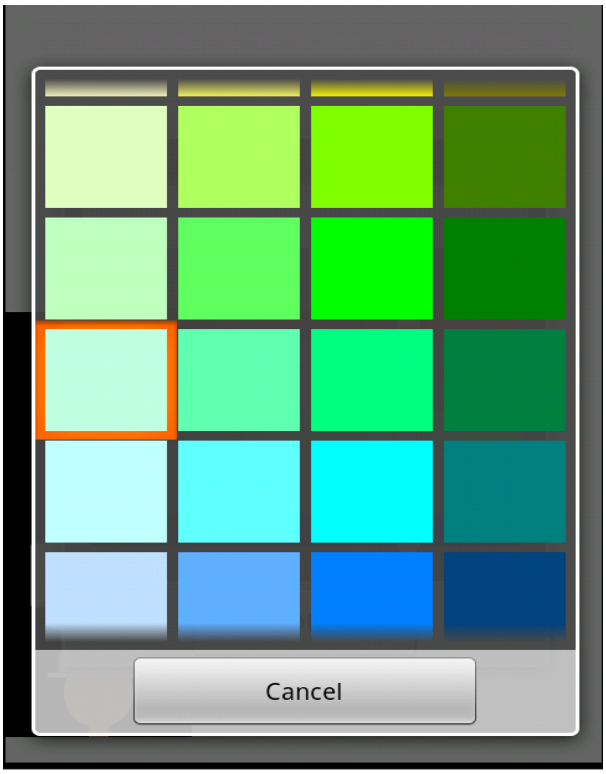


image 5

以上で操作を終えた画面が下の image6 となります。

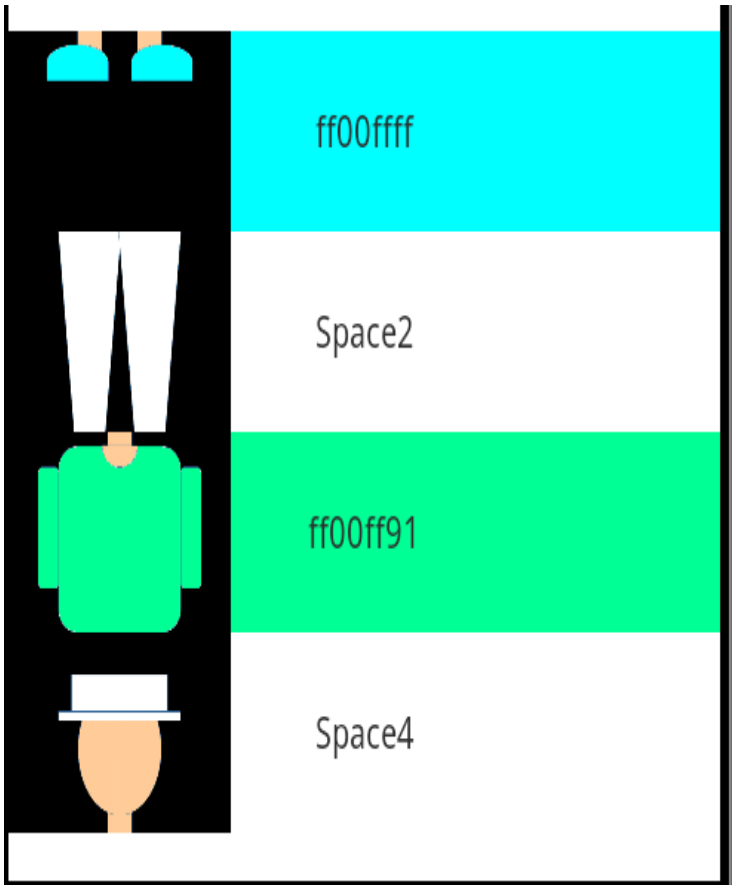


image 6

部位ごと分けられていることを解りやすくするために腰の部位だった胴に色を加えています。以上が、「 Select -Color 」の実装の実際の流れとなります。

4.3　画像の見せ方について

本節では画像がどうやって色を反映しているのか、説明していきます。

イメージとしまして、「 Text Veiw 」の上から、「Image Veiw」を重ねて画面に表示するイメージとなります。イメージは、以下の図１０ようになります。

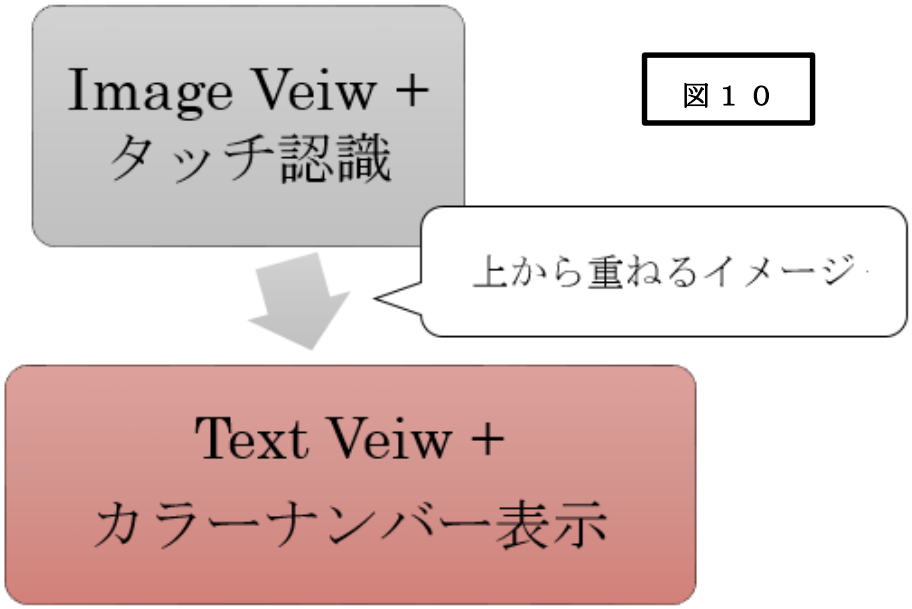


図１０

具体的に、「 Text Veiw 」と「Image Veiw」を別々で作成し、最終的にレイアウトで調整し「 Text Veiw 」と「Image Veiw」が重なるように配置することによって、画像に色が反映されているように見せるということです。そうすることによって、画像を選択したいのであれば画像を、色を選択したいようであれば色を選択するといった、分断化しました。

補足として、上から重ねる画像はGIF形式で保存してください。色々な形式を試してみたのですが、 Android Ver2.2 では、GIF形式しか適応できませんでした。

第 5 章　「 Select -Color 」の実装

5.1　ソースコードについて

本節では、コード全体の中で、必要な部分のみ説明していきます。

図の参照は「コード表？？」と説明します。

5.1.1　Android Manifest について

「 Android Manifest.xml 」の全体は、以下のコード表 1 のようになります。

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3         package="com.example.irogae"   android:versionCode="1"   android:versionName="1.0" >
4 <uses-sdk    android:minSdkVersion="8"    android:targetSdkVersion="18" />
5 <uses-permission android:name="android.permission.CAMERA" />
6 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
7 <application     android:allowBackup="true"     android:name="Globals"
8         android:icon="@drawable/ic_launcher"    android:label="@string/app_name"
9         android:theme="@style/AppTheme" >
10 <activity   android:name="com.example.irogae.MainActivity"
11         android:label="@string/app_name" > <intent-filter>
12 <action android:name="android.intent.action.MAIN" />
13 <category android:name="android.intent.category.LAUNCHER" />    </intent-filter>
14 </activity>   <activity android:name="gnidviout">    </activity>    </application>
15 </manifest>
```

5.1.1　コード表 1

「 Android Manifest.xml 」での注意点としましては、 7 行目から11行目まで続いている「 Globals 」変数を宣言していることをここで入力し忘れないことです。他に関しては特に変わった点はありません。

5.1.2　MainActivity 内で使った変数について

まず、最初にこのソースコードでの Text View と Image View はその他必要なコンテナ（色や画像を反映し表示する場所など）や変数を、以下のコード表 2 のように変数を宣言しました。

```
53 LayoutInflater inflater;
54 View dialogView;
56 private TextView tv1, tv2, tv3, tv4;
57 private Image View iv1, iv2, iv3, iv4;
58 private final int MP = ViewGroup.LayoutParams.MATCH_PARENT;
59 private final int WC = ViewGroup.LayoutParams.WRAP_CONTENT;
～中略～
66 Globals globals;
～中略～
61 final int TV1 = 10;
62 final int TV2 = 11;
63 final int TV3 = 12;
64 final int TV4 = 13;
```

5.2.1　コード表 2

53 行目の「LayoutInflater」は、他の xml リソースの「 View 」（さまざまな情報を表示し、それらに対して操作を行うことができる）を取り扱えます。

Android でのプログラミングでは、画面の「 View 」を構成する場合、レイアウト用 xml ファイルを用いて、レイアウトを作成することが推奨されています。しかし、 xml ファイルを用いた場合、静的にレイアウトが決定してしますのでそれを回避するために「LayoutInflater」があります。このクラスを使えば、プログラムの実行時のレイアウトを変更できるようになります（引用文献： f ）。

「 View 」は、さまざまな情報を表示し、それらに対して操作を行うことができる画面のことです。

色を反映しカラーコードを表示させるコンテナは Text View で定義しています。頭に「tv」とつけ、 56 行目に「 private TextView tv1, ～中略～」と定義しました。勿論、ソースコードの続きに、「 tv5, tv6, …」と続けて定義すれば、後ほど説明する表示する

グローバルから持ってきた画像を表示するためにコンテナは Image View で定義しています。頭に「iv」とつけ、 57 行目に「 private Image View iv1, ～中略～」と定義しました。

66 行目の「 Global 」は、プログラムの中のもっとも外側のブロックで宣言された変数で、すべてのブロックで有効なもの。「大域変数」とも呼ばれます。プログラム全体に対して定義されている変数です。ここでの「 Globals globals; 」は、グローバルとの関連付けを宣言しています。

61 行目から 64 行目の「TV1」から「TV4」は、タッチ操作の際の中間地点として利用します。

5.2　メイン画面の画像の表示について

最初に、メイン画面の画像のそれぞれの部位に対して、タッチ操作があった場合の処理操作を定義しておきます。ソースコードは以下コード表 3 の箇所になります。

```
79 LinearLayout layout2 = new LinearLayout(this);
80 layout2.setOrientation(LinearLayout.VERTICAL);
```

5.2 コード表 3

このソースコードは、下準備のようなものです。ここでの注意点としまして、 79 行目の「 layout2 」の全体の配置の順序を、垂直方向（上から下へ）にするために「VERTICAL」（ 80 行目）と宣言をすることくらいです。これを、宣言しなければ配置がバラバラになり調節が難しくなります。

続いて、 Text View と重ね合わせる画像の為を水平方向（左から右へ）にするために、

「 HORIZONTAL 」を宣言するためのそれぞれのソースコードです。

先に画像の方の宣言するソースコードです。本論文では、「 ll21 」と「 ll22 」は同じ宣言ですので割愛させていただきます。以下のコード表 4 の箇所のようになります。

```
87 LinearLayout ll21 = new LinearLayout(this);
88 ll21.setOrientation(LinearLayout.HORIZONTAL);
89 ll21.setGravity(Gravity.CENTER);
～ 中 略 ～
95 iv1.setScaleType(ImageView.ScaleType.FIT_XY);
```

5.2　コード表 4

このコードの変数「 iv1 」を（頭）とし、「 iv2 」を（胴）と仮定し、「 ll21 」として一括りにグループ化し配置しやすくし、参照した画像の大きさを指定しています。

ここでの注意点は、 87 行目の「 ll21 」に対して 88 行目で「 HORIZONTAL 」を宣言するのと、 89 行目に「 Gravity . CENTER 」を宣言すること。更に 95 行目で画像に対し「 setScaleType 」で「 ScaleType . FIT_XY 」を、宣言することです。まず、 89 行目の「 Gravity . CENTER 」の、「 Gravity 」はレイアウトの中の配置を決める確認です。「 CENTER 」は対象のコンテナをフィールド内の中央に配置する宣言です。ここでは、私が作成する際に解りやすくするために中央設定にしていますので、「 top 」（コンテナの一番上に配置される）や「bottom」（コンテナの一番下に配置される）など宣言すれば自由に配置できます。

次に、 95 行目の「 ScaleType . FIT_XY 」ですが、「 ScaleType 」は画像を表示する「 Image View 」クラスのオブジェクトの大きさと、表示される画像の大きさが異なる場合に「 Image View 」のエリア内での画像をどのように表示するかを設定の確認です。「 FIT_XY 」は、画像のサイズが縦横それぞれ独立しているのをリサイズし、「 Image View 」に合わせて全てが表示されるようにフル画面にする宣言です。

この宣言をしなければ「 Globals 」から、参照した画像が正しく表示されません。

残りの「 iv3 」を（腰 ）、 「 iv4 」を（足）と仮定し、「 ll22 」として一括りにグループ化すればここでの入力は完了です。

次に「 Text View 」の説明です。基本的に、「 Image View 」とソースコードは同じです。「 tv1 」を頭の下に、「 tv2 」を胴の下に、「 tv3 」を腰の下に、「 tv4 」を足の下に配置するようになっています。注意点は「 tv1 」の場合を例に説明します。

重要なコードソースは、以下のコード表 5 の箇所となります。

```
144 tv1.setOnClickListener(new OnClickListener() { @Override
146  public void onClick(View v) {
147    mColor = mTextColor;
148       colorDialog(TV1);            }
```

5.2　コード表 5

特に注意するべき点としては、 144 行目の「 setOnClickListener 」を、それぞれに別々に宣言することです。この宣言をしなければ、画像をクリックした時にアクションを起こせなくなります

5.3　シークバーと見本の表示方法について

5.3.1　シークバーを使用にする為の下準備

このシークバーの表示方法、見本の表示方法のソースコードはscarlet氏のコードを参考にさせていただきました。

本節では赤色（R）のシークバーの説明のみ行い、緑色（G）と青色（B）に関しては、変更点のみ説明していき行きます。

最初にシークバーについて説明していきます。

シークバーの表示形式については以下のコード表 6 のようになります。

```
233 final String rString = getResources().getString(R.string.color_picker_red);
～中略～
240 final TextView textR = (TextView)dialogView.findViewById(R.id.colorPickerTextR);
241 textR.setText(String.format("%s(%02X)", rString, (mColor & 0x00ff0000) >> 16));
～中略～
247 viewer.setBackgroundColor(mColor);
248 viewer.setOnClickListener(new OnClickListener() {   @Override
250         public void onClick(View v) {
251   }});
```

5.3.1.1　コード表 6

このコードは、予め用意したシークバーのコードをインポートしたものを、画面に表示するための下準備です。中略部分の緑色（G）と青色（B）のコードの変更点は、241 行目の「 mColor & 0x00ff0000) >> 16 」のカラーコードと数値を、変更する点です。緑色（G）のカラーコードを「0x0000ff00」と変更し、数値を「 16 」から「 8 」に変更してください。青色（B）のカラーコードを「0x000000ff」と変更し、数値は入力しなくて大丈夫です。

5.3.2　シークバーのソースコード

シークバーについてですがシークバーのデザインやソースコードは、別所で xml ファイルを作成しなければなりません。本論文では、このアプリケーションで使用したシークバーのソースコードとなります。赤色のソースコードは以下のコード表 7 のようになります

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <layer-list xmlns:android="http://schemas.android.com/apk/res/android">
3 <item android:id="@android:id/background"> <shape> <corners android:radius="5dip" />
4 <gradient android:startColor="#ff000000"
5          android:centerY="0.5"
6           android:endColor="#ffff0000"
7           android:angle="0"  />   </shape>    </item>
8 <item android:id="@android:id/progress">    <clip>    <shape>
9 <corners android:radius="5dip" />
10 <gradient
11          android:startColor="#ff000000"
12          android:centerY="0.5"
13          android:endColor="#ffff0000"
14          android:angle="0"    />    </shape>    </clip>    </item>    </layer-list>
```

5.3.2　コード表 7

緑色（G）と青色（B）の変更点は 6 行目と 13 行目の「 android:endColor="#ffff0000" 」の 2 か所ともに緑色（G）の場合は、「 android:endColor="#ff00ff00" 」に変更し、青色（B）の場合は、「 android:endColor="#ff0000ff" 」に変更し、それぞれ個別に xml ファイルで入力してください。後は、エクスプローラー内の作成中のファイル内の「 res 」ファイル内の「drawable」ファイルにインポートすればシークバーの作成は完了します。

5.3.3　見本の表示の仕方

見本での表示方法です。以下のコード表 8 の箇所が対象です。

```
27 public class MainActivity extends Activity implements OnClickListener {
 29 final int CHANGE_TEXT = 0;
30 final int CHANGE_BACKGROUND = 1;
31 final int COLOR_TABLE[] = {
33          0xffffffff, 0xffc0c0c0, 0xff808080, 0xff000000,
          ～中略～
46          0xfffffe0, 0xffff60ff, 0xffff0080, 0xff800040  };
 49 int mColor = 0xffffffff;
 50 int mTextColor = 0xffc0c0c0;
51 int mBackgroundColor = 0xff000000;
```

5.3.3　コード表 8

まず見本で表示する色の指定する場所は「 final int COLOR_TABLE[] = {0xffffffff ～ 0xff800040} 」の中に、16 進数で表示したいコードを入力します。入力する際には「 0xffffffff, 0xffc0c0c0, 0xff808080, 0xff000000, 」と、カラーコードごとにコンマで区切ることを忘れなしでください。そして一列に表示できる色の個数は Android Ver.2.2 では 4 個がきれいに見えます。

その他の、バックグラウンドやテキストカラーの初期画面の色の設定などについては任意の設定になります。

5.4　メイン画面での画像の表示する方法

5.4.1. 画像の引用方法

まず、「protected void onCreate(Bundle savedInstanceState)」に以下のコード表 9 のように宣言します。

```
69 protected void onCreate(Bundle savedInstanceState) {
70        super.onCreate(savedInstanceState);
73        globals = (Globals) this.getApplication();
```

5.4.1　コード表 9

最初に、「globals = (Globals) this.getApplication();」（ 73 行目）と宣言し、別にグローバル変数を用いたjavaコードとの関連付けをここで宣言します。

では、先に別所で用意したグローバル変数を用いたjavaコードやその他からのソースコードに対する関連付けににについて説明していきます。

タイトルに関しては、自身が作成する際に解りやすいように設定してください。ここでは、私が作成した際につけたタイトルで説明します。

メインとは別に作成したjavaのソースコードは以下のコード表 10 のようになります。

```
1 package com.example.irogae;
3 import android.app.Application;
5 public class Globals extends Application {
7         int doko;
8         int position;     }
```

5.3.4.1　コード表10

ソースコード事態は少なく、ここでの「 Globals 」の役割はコンテナの変数を宣言しています。

では、どのように反映しているのかとい、また別のjavaコードを作成します。タイトルは、任意で設定してください。

まず「 private 」使用したクラスと変数の説明です。コードは以下のコード表11のようになります。

```
20 private ArrayList<Bitmap> list;
21 private ArrayList<String> list_path;
22 private GridView gridView;
23 private File[] files;
24 private Uri bitmapUri;
25 private Bitmap bmp;
27 Globals globals;
```

5.3.4.1　コード表11

20 行目と 21 行目の「ArrayList＜＞」は、「 ArrayList 」の後ろについている「＜＞」（以降、「型」と説明します）は、これは J2SE5.0 から導入されている Generics 機能とよばれるもので、この「 ArrayList 」にどのような型の値を格納するか指定しています。この型の部分にはクラス名を指定します。そして要素として格納できる値は、ここで指定したクラスのオブジェクトになります。ここでは「＜Bitmap＞」と「＜String＞」を指定しています。「 Bitmap 」クラスは、画像などを利用場合に指定するクラスです（引用文献：g ）。

23 行目の「File」クラスは、ファイルやディレクトリーの作成、 Read 権・ Write 権の調査などファイル・ディレクトリーに対しての動作する様々なメソッドが用意されています。また、「File」クラスのオブジェクトは「File Reader」クラス、「File Writer」クラスなどのファイル入出力関連のオブジェクトを作成する際に引数としても使用されます（引用文献： h ）。

続いて、「 Activity 」にまたがっているオブジェクトの共有に対して紐づけしていきます。

ソースコードは以下のコード表 12 のようになります。

```
53    private ArrayList<Bitmap> load() {
55      list = new ArrayList<Bitmap>();
56      list_path = new ArrayList<String>();
57      for(int i = 1; i <= 4; i++){
59        Resources r = getResources();
60              if (i == 1) { bmp = BitmapFactory.decodeResource(r, R.drawable.h1);}
62               else if (i == 2) { bmp = BitmapFactory.decodeResource(r, R.drawable.h2);}
64              else if (i == 3) { bmp = BitmapFactory.decodeResource(r, R.drawable.h3);}
66              else if (i == 4) { bmp = BitmapFactory.decodeResource(r, R.drawable.h4);}
70      list.add(bmp);                        }
```

59 行目の「 Resources 」は、リソース ID を引数に指定すると対応するリソースを習得できるメソッドがいくつか用意するクラスです（引用文献： i ）。

60 行目から 66 行目にある「 BirmapFactory 」は、外部ファイルやリソース、ストリームなどから「 Bitmap 」クラスのオブジェクトを作成するクラス。

57 行目から 70 行目にかけて入力されているソースコードは「 Globals 」変数を経由してメイン画面表示させるために画像一つ一つに if 文で定義しました（引用文献： j ）。

5.4.2 画像選択の指示の指定

ここまで、「 Globals 」変数との関連付けを説明しましたが、最後にメイン画面の画像の選択された際の処理を入力さなければなりません。「Main Activity」に戻り、以下のコード表 13 のように入力します。

```
452 public void onClick(View v) {
454     if (v == iv1) {
455             globals.doko = 1;
456     } else if (v == iv2) {
457             globals.doko = 2;
  ～中略～
461             globals.doko = 4;    }
463     Intent intent = new Intent(this, gnidviout.class);
464     startActivityForResult(intent, 1);}
```

5.3.4.2　コード表13

このソースコードは、「もし頭が選択されたら」や「もし足が選択されたら」など if 文で入力しました。これは、どこの部分でアクションがあったか確認するための文です。注意点としまして、それぞれの「iv」に対して数値が一致しているかどうかです。具体的に、454 行目に「(v == iv1)」とありますが、それに対し「 globals.doko 」に対し「 1 」を指定しているということで。「(v == iv2)」に対して「 2 」を指定しているかどうかということです。

続いて、いかに以下のようにコード表 14 のように入力します。

```
468 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
470       super.onActivityResult(requestCode, resultCode, data);
471       if (globals.doko == 1) {
472                 if (globals.position == 0) {
473                           iv1.setBackgroundResource(R.drawable.image1);
474                 } else if (globals.position == 1) {
475                           iv1.setBackgroundResource(R.drawable.image2);
  ～中略～
481       } else if (globals.doko == 2) {
482                 if (globals.position == 0) {
483                           iv2.setBackgroundResource(R.drawable.image1);
  ～以下略～
```

5.3.4.2　コード表14

ここでも、if 文で入力しました。注意点は、if 文の中に更に if 文を指定しているので数値が「0」から始まる点です。もう一つは「iv」のコンテナの数値を間違えないことです。欠点としては、4.3.5.1 でも取り上げたのですが画像をインポートするたびにここの if 文が増えるという点です。

以上が、本論文での「Select -Color」の重要な部分となります。

第 6 章　「 Select -Color 」に対するアンケート結果

6.1　アンケート対象者と実施方法

対象者は、スマートフォン（機種やメーカーは問いません）を所有する経済学部・芸術学部・情報学部の学生にアンケートを取りました。実施理由は、未完成ではあるのですが現時点でどのくらいの評価がもらえて、更に作成中では思いつかなかった面白く今後の参考にしたいと考え実施しました。質問内容は、以下の図 10 の 2 問となります。

問 1 ：このアプリケーションに触れてみて、率直な感想をお願いします。

問 2 ：あったらいいなと感じた、要素はありますか？

図 10

6.2「Select -Color」に対する要望や課題

ここでは触れてみた率直な感想に対する対策と、アプリケーションに対する追加してほしい要望をまとめそこから見えてきた課題をまとめました。学生が考えた解決策に関してはイメージの参考程度くらいで見てください。

6.2.1　問 1 のアンケート結果と考察

問 1 に対する意見は以下の図 11 のような意見が多く上がりました。

・「第 1 印象として華やかさがない」

・「シークバーの数値（カラーコード）が解りにくい」

・「操作説明が欲しい」

図 11

問 1 の触れてみた感想で多くのコメントは、「第 1 印象として華やかさがない」「シークバーの数値（カラーコード）が解りにくい」といったコメントが多く見受けられました。１つ目の「第 1 印象として華やかさがない」という意見は在学期間中では作成するのに集中して見た目やバリエーションが不足してしまいユーザーにとっての最低水準となっている「デザインは良くて当たり前」という基準に到達できなかったのが原因です。具体的に、バックグラウンドの背景選択や、画像（男女やキャラクター）の選択肢を増やすことによって「華やかさ」は、少しずつでも改善されていきます。２つ目の「シークバーの数値（カ

ラーコード）が解りにくい」という意見は、色の調整をする画面が全体的に黒くシックに見せようとしたのですが、RGB 16 進数を表示するところを明るく白色ではっきり見えるように見せれば解りやすさが上がるなと考えました。

全ての意見をまとめてみても、作りこみが足りていないという点が上がった結果になりました。

6.2.2　問２のアンケート結果と考察

問２に対する意見は以下の図 13 のような意見が上がりました。

・「背景も選択できて、画像をより細かく選択したい」
・「作った色の保存機能が欲しい」
・「過去に使った色の履歴があったらいい」
・「カメラで撮った買った服を、反映できないか」

図 13

様々な貴重な意見の中でも、目に留まった要望をいくつか取り上げます。

要望その１:「背景も選択できて、画像をより細かく選択したい」

まず、開発段階ではメイン画面時に表示されている画像の黒い部分は消せないかなと色々と試してはみたのですが、Android 2.2 では画像上では黒い部分は消し、様々な画像形式を試しても黒い部分は消すことができなかったので、最新バージョンで画像をレイヤー形式が読み込めるのであれば、反映するだけで問題は解消されます。ですが、出来なかった場合の課題として開発したアプリケーションの画像優先順位は、上から「テンプレート＞Text View（色が反映される部分）＞バックグラウンド」となっているので、解決策として、テンプレートの黒い部分に重なるようにバックグラウンドの画像を反映するといったソースコードを作成するといった課題が上がります。定義として順にソースコード書き換えれば、黒い部分は消せて画像選択もできるのではないかと考えます。消せるようになれば、「Globals」変数を利用し、同じように作成すれば風景の選択も可能になるのではないかと考えます。

要望その 2：「作った色の保存機能が欲しい」

「過去に作った（使った）色の履歴があればいいな」

この意見に対しての解決策は、作った色を保存できる機能を搭載すれば解決だと考えます。具体的に、「Globals」変数を利用し、今までは仮想空間から画像（情報）を参照する形でしたが、記憶仮想を作成しコンテナに情報を保存し、指示があれば表示する定義を新しく作成しなければなりません。具体的に、情報の保存の指示がされた場合、仮想空間に Text View を作成し、その中に作成した色を反映したものを保存しておくといった流れのソースコードの作成となります。カラー選択画面に新しい選択ボタンを増やし、作った色を次回使う時まで保存もできコンテナを作成すれば問題は解決されます。間接的ではありますが保存機能となる形となります。アプリケーションの有用性も考えると削除機能も搭載すれば便利になります。記憶しておくのはカラーコードの数値なのでメモリーもあまり取らなくて済みますが搭載したほうが後々の整理する際に役立つでしょう。

要望その３：「カメラで撮った買った服を、反映できないか」

この意見は、今回のアンケートと中で一番面白いなと思った意見です。

ですが、具体的な解決策としては買った服装を認識し不要な部分を削除しそれをテンプレート化するといった作業をしなければならないので、この意見に対して開発技術としては 4.1 で軽く触れましたが現段階では大きなモニターの上にカメラを搭載し消費者を撮影して、データ上にテンプレート化しておいた商品（衣服）を、上にからレイヤーのように重ねて擬似試着を体験できるといった技術が、まだ公表はされていませんが開発されていますので、それと近いイメージとなると考えられます。具体的には、ポージングの選択の制限はかかりますが予めにフレームを作成しておき、フレーム内に収まるように撮影して後は仮想空間に反映されるように、画像を保存するソースコードを作成すれば反映が可能かもしれません。

6.3　アンケートに対しての感想

今回のアンケートを通して、少数ではありますが貴重な意見を収集できたと思います。アプリケーションの人気の傾向や、どのようなアプリケーションがユーザーの目に留まるのか、アンケートを実施して良かったと思います。今回テストプレイしてもったアプリケーションも、開発中では考えもしなかった意見など聴けて視野がさらに広がりました。

第７章　それぞれの章に対する感想

初めに第２章ではかけてモバイル産業に対して、どうスマートフォンが関わりを持ち、基本無料コンテンツの中の、有料コンテンツについても触れた感想ですが、まとめてみて最初に思ったことは「課金システムがこれほどの利益を、出しているのだな」と感じました。近年、ゲーム産業は衰退してきているといわれていますが、その反面モバイル産業に密接に関わりを持っているいわゆる「ソーシャルゲーム」や「ゲームアプリケーション」が主な利益元といえ企業が、急成長を遂げています。その背景には、スマートフォン業界の普及率と Google 社と Apple 社がどの様にスマートフォンを普及させたのか調査していてとても面白く楽しかったです。

第３章は、私が作成した「Select Color」と関わりがあるデザイン系のアプリケーションについて調査しましたが、まとめてみた感想は「ゲーム業界の売上高と比べてもデザイン系アプリケーションの売上高は圧倒的に少なく、利益を出せるアプリケーションの作成はとても難しい」と感じました。デザイン系アプリケーションの売上高を本論文にはまとめなかった理由としましてデータ不足です。更に本論文でまとめた、写真加工・画像編集系アプリケーションは、デザイン系アプリケーションの中でも需要があり利益を出しやすい部類です。調査しているうちに解ってきたことは、デザイン系アプリケーションの売上高はゲーム業界以外の業界とも比べても低いという結果です。基本的にデザイン系アプリケーションの需要はないという結果でした。利益を出すには、トレンドを抑えて需要を満たさなければならないのですが、その需要分析がとても難しいのであると再認識する形となりました。本論文で実施したアンケート結果にも出ているように、芸術系学部の方にも調査したので所有率は半分くらいとなりましたが、ゲーム系アプリケーションは全ての方が所有していると、結果が出ているのも事実だったということです。

第４章から第６章にかけての、「Select Color」について概要やコードについて説明しましたが、いざ作成したものを説明するとなるととても難しくどのような順番でどのように切り出して説明すればよいのか、感覚を掴むまで苦労したなと感じました。ですが、終え

てみて再度見直したときに、二度と体験できない体験ができたなとも感じました。具体的には、考えるのは簡単でも形にするのはとても難しくと感じました。更に、アプリケーションを作成するにあたって、「有用性」も大切なのですが「新規性」も大切にしなければならないと学びました。開発者の苦労が少しではありますが、解りました。

謝辞

就職活動や本論文を作成するにあたり熱心に御指導頂きました田中章司郎教授と、Android アプリケーション作成を熱心にご指導頂きました伊藤則之教授（現東北学院大学）には心より御礼申し上げます。また、同じ研究室の皆様や市場アンケート調査に御協力頂きました皆様には、御協力と御助言を頂きましたこと、深く感謝致します。

なお、本論文で作成したプログラム及び、データ、並びに関連する発表資料などの全ての知的財産権を本研究の指導教員の田中章司郎教授に謙譲致します。

コード表作成時引用文献

a：Scarlet 氏　「Android アプリを作ってみる『android カラーピッカー？カラーセレクター？　ダイアログ』」

「 http://scarlet711.blogspot.jp/2011/06/android.html 」

f：Androyer in Japan　「LayoutInflater『基礎知識』」

「　https://sites.google.com/site/androyerjapan/home/layoutinflater　」

g：JavaDrive　IT 技術全般の学習サイト「java 入門『コレクション〔ArrayList クラス〕』」

「 http://www.javadrive.jp/start/arraylist/index1.html 」

h：Java の道　「Java 基本『入出力　８.File クラス』」

「　http://www.javaroad.jp/java_io8.htm　」

i：JavaDrive　IT 技術全般の学習サイト「Android 入門『リソース管理〔Resources クラスを使ったリソース参照』」

「　http://www.javadrive.jp/android/xml_layout/index5.html　」

j：JavaDrive　IT 技術全般の学習サイト「Android 入門『Bitmap クラス〔Bitmap クラス及び BitmapFactory クラスの定義』」

「　http://www.javadrive.jp/android/bitmap/index1.html　」

引用文献

b：業界動向 SEARCH.COM　「モバイル業界　『基本情報（平成 25-26 年版）』」

「　http://gyokai-search.com/3-mobile.html　」

c：EE Times Japan　「ビジネスニュース『スマートフォン市場を席巻する“Android”と“中国勢”』」

「　http://eetimes.jp/ee/articles/1408/04/news063.html　」

d：TeachMe iPhone　「iPhone 最新情報『iOS 対アンドロイド：世界各国のシェア状況』」

「　http://www.teach-me.biz/iphone/news/android/140326-2.html　」

e ：All About　デジタル「Android とは何かをわかりやすく解説！『Android とは何か』」

「 http://allabout.co.jp/gm/gc/3588/ 」