留学生のための
ひらがな学習アプリの開発

豊橋技術科学大学　情報・知能工学課程
河合研究室
学籍番号：111301　櫛原 大地
指導教員：河合 和久

平成 26 年 12 月 26 日

留学生のためのひらがな学習アプリの開発

学籍番号:111301　櫛原　大地　　指導教員:河合和久

Abstract:It is the program for the foreign students who have begun to just learn Japanese. I ported the hiragana learning application of the Yamamura of this laboratory for Android application. In addition, I perform more useful hiragana learning support by adding a function. This application is developed by Eclipse4.4.0 that installed AndroidSDK. As an actual machine, I used Nexus7Android4.3. I implemented function to work in the same way on Android which was implemented by Yamamura. As for the function added to learn mode, 1 stroke of the hiragana is displayed whenever I push the button. A function added to an exercise mode is a function to apply stroke order. In addition, I added the puzzle function using the Japanese syllabary list. I cannot be transfused into iOS now without being able to port only an Android version. Thus, I transplant it to an iOS version near future.

1.　はじめに

日本語を学び始めたばかりの留学生を対象とした学習支援プログラムである。本プログラムでは、ひらがなの形、発音、書き順を理解することが目的である。前年度、本研究室の山村が作製したひらがな学習を行うWWWプログラムをAndroid端末に移植し、機能を追加することで、より有用なひらがな学習支援を目指す。

2.　目的

2.1　Android 移植の検討

山村が開発したひらがな学習プログラム[1]では、HTML方式を用いているために、学習しようと思うと毎回ブラウザを介して利用しなければならない。そのため、手軽に学習しようと思うと少し手間がかかってしまう。

そこで、Androidアプリにすることで、アプリケーションを立ち上げればすぐに学習できるようにすれば、より学習頻度があがるのではないかと考えた。

また、スマートフォンのタッチ機能を用いれば指を動かしながら覚えられるので学習に有効ではないかと考えた。外出した際も、空いている僅かな時間を有効に活用しアプリケーションを立ち上げることで学習することができる。

サーブレット上のJavaとAndroidアプリでのJavaでは同じ言語だが使用方法が異なるため、1から開発していかなければならなかった。

2.2　先行研究

日本語学習者数は、毎年増加していっている。そのような増加する日本語を学習し始める人に向けて開発が行われた。

まず、漢字等を学ぶ前に日本語の基礎であるひらがなを覚えることが必要になる。学ぶにあたって単語カードのような手軽に学習できることを目指し、急激にスマートフォンやタブレット端末で実行可能なアプリ形式の「ひらがななぞり」[2]のような仕様に加えPC上でも動作可能なようなプラットフォーム非依存なプログラムを研究、開発が行われた。

2.3　新機能の検討

山村のひらがな学習アプリケーションでは学習モードと演習モードがある。学習モードでは、ひらがな一つひとつの形、書き順、発音を学ぶことができる。また、演習モードでは書き順とひらがなを覚えられるようになぞり書き演習、発音からひらがなを当てる演習、ひらがなから発音を当てる演習が実装されていた。

学習モードで、ひらがなの書き順を覚える機能では一気にすべてを描いてしまう仕様になっていた。そこで、ボタンを押すごとに1画1画が表示されるようにすることでより、書き順、書き方が学べるようにした。演習モードでは、書き順をあてる演習を追加した。これは、2つの異なる1画が黒くなっている画像を見て、書き順が正しい方を選ぶ演習である。

また、演習に五十音表を虫食いの状態にして、虫食いの箇所に当てはまるピースをはめるパズル機能も実装するようにした。五十音表でどこにどのひらがなが当てはまるのかを考えることによって、ひらがなの母音と子音の関係を意識することでひらがなを覚えやすくなると考えた。

3.　開発環境

Eclipse4.4.0にAndroidSDKをインストールすることによってコンパイルとデバッグを行うことができる。また、実機として、Nexus7Android4.3を用いた。

4.　プログラム詳細

学習モードでの、ボタンを押すことによってひらがなの1画1画を表示する機能では、コマ送りのひらがな画像を1画ごと配列に格納し、0.1sごとにアニメーションすることで実現している。

演習モードでの、書き順演習では、2画以上のひらがなを対象として、ひらがなそれぞれ1画ごとに線を黒くした画像を用意し、その画像をランダムで二枚表示し、1画目から順に画像を選択させる。

また、五十音表を用いたパズル演習については、画像を自由に動かせることができるFramelayoutに当てはめる画像を配置することによって、画像を動かしている。タッチイベントを取ることによって指の動きに追随して画像が動くように実装している。

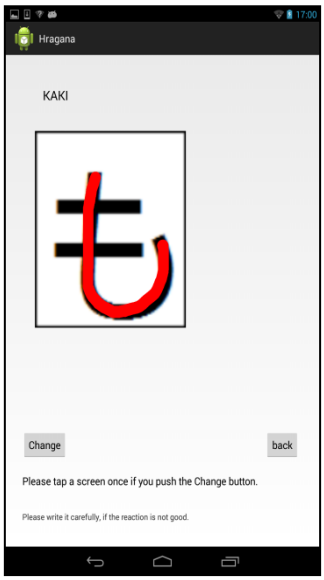

図1. 実行結果（書き演習）

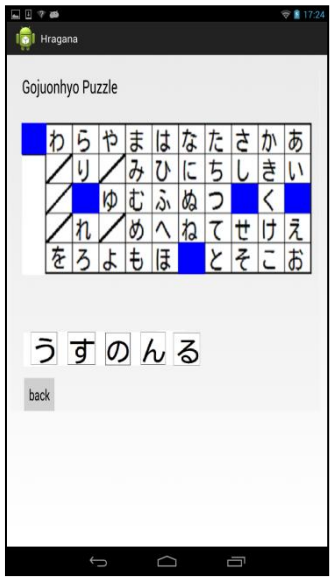

図2. 実行結果（パズル演習）

5.　結果

サーブレット版のひらがな学習アプリケーションにおける学習機能、演習機能の基本機能の移植が完了した。

作成したアプリケーションはEcripce上のエミュレータとNexus7上での動作を確認した。

6.　考察

アプリケーションサイズは25.93MBとなっており、実際に操作する上で問題のない処理速度を実現できている。

また、パズル機能、書き順機能等追加したことで空いた時間に効率よく学習できるアプリが開発できたと考える。

7.　まとめ

今後、iOSへの移植も予定している。また、機能拡張として各演習のデータをサーバで管理し、ランキング機能等実装する。

参考文献

(1)　山村樹:初級日本語学習者のためのひらがな学習プログラムの開発　豊橋技術科学大学　情報知能工学課程　(2013)
(2)　株式会社スタティクス　「ひらがななぞり」 http://play.google.com/store/apps/details?id=jp.co.statix.hiragana

目次

第1章　序論

本研究は日本語を学び始めた人を対象とした日本語の基礎であるひらがなの学習を支援するプログラムである。昨年度、本研究室の山村が作製したWWWプログラム[1]をAndroidに移植し、新機能を追加することでより有用な学習支援を目指して開発した。

ひらがなは、日本語の表記に用いられる音節文字の一種で、万葉仮名を起源として成立した（図1）。ひらがなが作り出されたのは、まず書きやすさが求められたからで、手軽に使えるという実用性によるものであろう。[2]

図1　漢字からひらがなのうつり変わり（Wikipedia-ひらがな-より）

ひらがなは、漢字に比べると書きやすい。しかし、日本語を学び始めた人にとってはひらがなも漢字も片仮名も初めて見るものであり、その習熟は難しい。

本学には現在約200人の留学生が在学しており、他の講義と並行して日本語を学んでいる。また海外の日本語学習者数は2012年のデータで136ヶ国・地域で約398万人となっており、学習者数は毎年増加している。[3]

日本の小学校でも小さい子ども達に最初に教えるのはひらがなであり、日本語を学び始める人には日本語の基礎であるひらがなを学ぶ必要がある。このような留学生を対象として開発された山村のWWWプログラムは、WWWプログラムであるため、利用しようとするたびにブラウザを介して利用しなくてはならず、手間がかかる。そこで私はAndroidアプリならばアプリケーションを立ち上げるだけですぐに利用できると考え、Android移植を行った。

さらに、本研究室の笹岡の英単語学習アプリの研究[4]では、学習アプリの起動から終了までの1回の利用時間は、20分未満の利用が全体の約7割を占めていることが示されてい

る。(図 2) この結果は英単語学習アプリの結果であるが、同じ学習アプリであるひらがな学習アプリにも同じことが言えると考える。20 分未満の利用ということはいわゆる「すきま時間」に利用されていることがわかる。

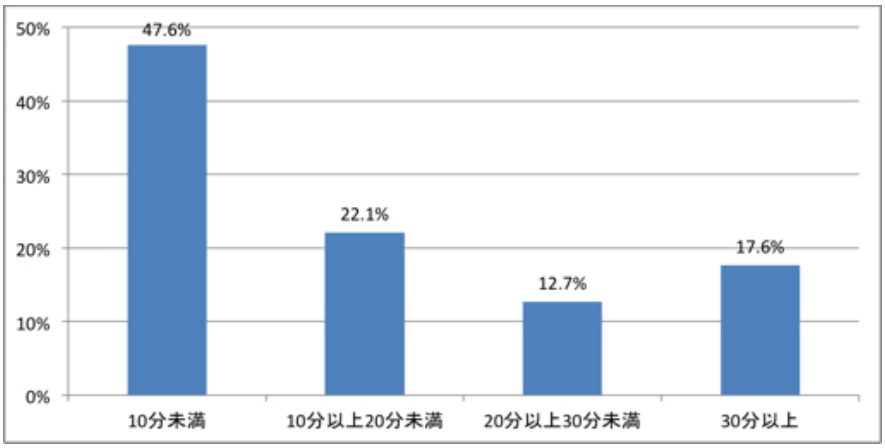

図 2　1 回の利用時間長の分布(笹岡 [4] より)

これより、利用しようとするたびにブラウザを介して利用する仕様では手間がかかり学習時間が少なくなってしまうと考えた。また、ひらがなへの理解を深めるために新機能を加えた。

第2章　先行研究

ここでは、本研究の基盤である山村のシステムについて述べる。

2.1　研究背景

日本語を学び始めた人というのは、「あ」を「あ」と認識できない人である。逆に日本人から見た外国語で考えると図 3 のようなタイ文字の違いを見分けられない、ということである。図 3 の例のように小さい違いだが文章中や、文字サイズが小さい場合など十分に理解していないと見分けがつかなくなる文字がひらがなにもある。このような違いを理解できるようになるための学習支援システムとして山村の WWW プログラムが開発された。

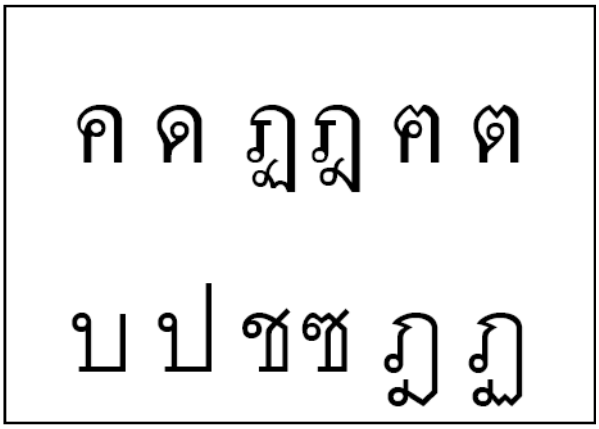

図 3　タイ文字の違い（山村[1]より）

山村は、ひらがなを学ぶにあたって単語カードのように手軽に学習できることを目指し、「ひらがななぞり」[5]や「Hiragana-Learn Japanese」[6]のようななぞり書きの利用に加え、プラットフォーム非依存なプログラムを目指した、としている。

山村の WWW プログラムでは大きく分けて学習モードと演習モードの 2 つのモードがある。学習モードではひらがなの形、書き順、発音を学習できる。演習モードではひらがなのなぞり書きを行う問題、発音から形を選択する問題、形から発音を選択する問題の 3 つがある。

2.2　学習モード

このモードは Servlet による HTML 形式で出力される。図 4 のように画面は 2 つに分割されており左にひらがなのリスト、右に選択したひらがなの詳細が表示される。発音の再

生には wav 形式の音声ファイルを使用し、ひらがなのアニメーションには html5 の機能を使ったアニメーションを使用している。

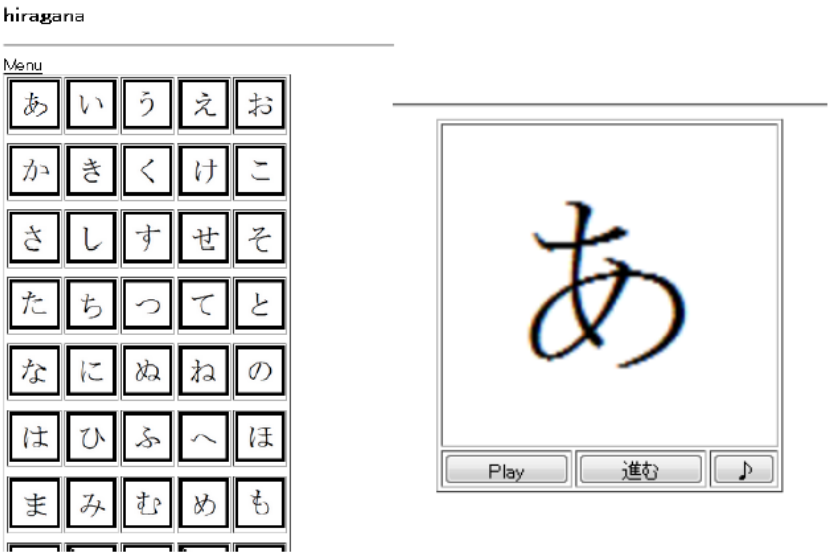

図 4 学習モード

Play のボタンを押すことによってアニメーションが始まる。進むボタンを押すと次の文字に切り替わる。音符のボタンで発音を確かめることができる。

2.3 演習モード

はじめに問題形式を選択する。

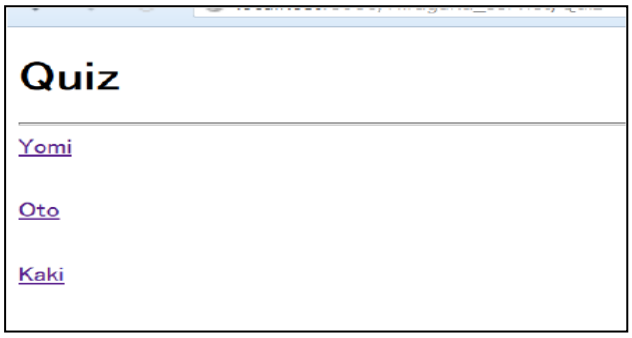

図 5 演習モード選択画面

図 5 のように Yomi と Oto と Kaki から選択するようになっている。Yomi はひらがなを見て正しい発音を選択する問題、Oto は発音を聞いてひらがなを答える問題、Kaki はひらがなのなぞり書きを行う問題である。

2.3.1 発音から形を答える問題

問題は図 6 のように出力される。問題となっているのは♪マークで、その下にあるひらがなの並びから解答を選ぶ。♪マークのボタンをクリックことによって発音が聞ける。

図 6 発音から形を答える問題

2.3.2 形から発音を答える問題

問題は図 7 のように出力される。画面下に並ぶ♪マークをクリックすることで再生される発音から解答を選び、その下にある数字の書かれたボタンを選ぶ。

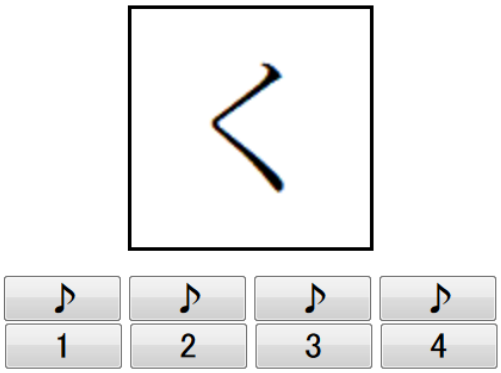

図 7 形から発音を答える問題

2.3.3 なぞり書きの問題

問題は図 8 のように初めに半透明なひらがなが出力されている。そしてこの半透明なひらがなの 1 画をなぞることにより、半透明で無くなり表示される。すべてなぞり書きを終えると利用者がなぞった書き順と向きの正誤判断をする。

この問題では 1 画の向きが正しくなくても 1 画が半透明でなくなり表示されるようになっており、より正しいひらがなの理解度を確認しやすくなっている。

図 8 なぞり書きの問題

2.4 ほかのひらがな学習システム

この章の冒頭でも触れたが、他のひらがな学習システムとして「ひらがななぞり」や「Hiragana-Learn Japanese」等がある。

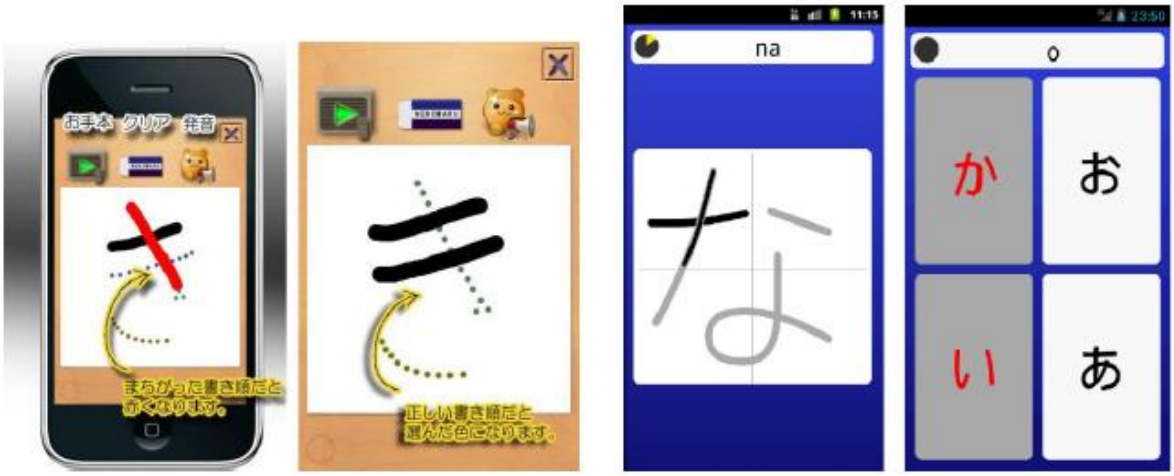

図 9「ひらがななぞり」[5] と「Hiragana-Learn Japanese」[6]

「ひらがななぞり」では、名前の通りなぞり書きの機能が実装されている。教科書通りの筆順でなぞらなければ字が赤くなりすぐに気づくことができる。

また綺麗に書けたら花丸で表示され、そうでなくても最後まで書けたら丸がつくようになっている。フォントは教科書体に近い、楷書体となっている。練習した文字数だけスタンプが貯まるようになっているため、ひとめで練習量がわかるようになっている。

「ひらがななぞり」では小さい子どもを対象としているため、親の立場からの機能として、選択機能がある。重点的に学ばせたい文字を選択することができる。短い時間に繰り返し遊ぶことで苦手な文字を減らすことができる。

「Hiragana-Learn Japanese」ではなぞり書きの機能と発音からひらがなの形を答える機能が実装されている。

このシステムは、海外の日本語を学び始めた人を対象としている。発音を聞きながらなぞり書きを行うことができる。このほか、ひらがなを見て、それにあうローマ字を入力する機能が実装されている。フォントは公開されていないが、「さ」や「き」が 3 画や 4 画としてではなく 2 画、3 画のフォントを使用している。

第3章　システムの設計と実現

3.1　システムの概要

上述のとおり、本プログラムでも学習、演習モードの 2 つのモードを実装した。

学習モードでは、我々が外国語の学習をする際の教科書の使用を意識して構成している。この構成に加えて、紙の教科書では実現することのできない書き順のアニメーションや音声の再生機能を持たせることで、本プログラムの利用価値を高めている。さらに新機能として、書き順のアニメーションに 1 画 1 画を表示できる機能を追加した。山村の WWW プログラムではアニメーションはボタンを押すとアニメーションですべてが表示される仕様だった。この仕様に加えて新機能として追加した。これは 1 画 1 画をアニメーションで表示できるようにすることで書き順を更に意識し、学習効率を高める目的がある。また、演習モードを実装することによって学習モードで覚えきれないひらがなを確認し、反復練習を可能とすると同時にクイズ形式にすることによってゲーム性をもたせることで学習意欲を高める目的がある。

演習モードには、山村が作製した WWW プログラムの肝でもあったなぞり書きの機能を Android に移植することによって、タッチ機能で実装した。タッチ機能で実装したことにより、書き順を理解するだけでなく、実際に紙にひらがなを書く為の高度な学習を可能とした。また、新機能として書き順を答える機能と五十音表でのパズル機能を追加した。書き順を答える機能ではなぞり書きの前の段階として視覚的にどちらの書き順が正しいのかとわかるようになると考えて実装した。また、パズル機能では五十音表にひらがなを当てはめていくことによってひらがなの母音と子音の関係を意識して覚えられると考えて実装した。

3.2　実現方法

開発には統合開発環境 Eclipse IDE for Java Developers を用いた。バージョンは Luna Release (4.4.0)を用いた。また、実行環境として AndroidSDK をインストールすることでコンパイルとデバッグをおこなった。エミュレータの他に実機として Nexus7 Android4.3 を用いて動作を確認した。プログラムの動作部分は Java を用いてコーディングを行い、表示部分には xml を用いる。

図 10 が開発環境の画面である。左が Eclipse で右がエミュレータ Android4.3 になる。

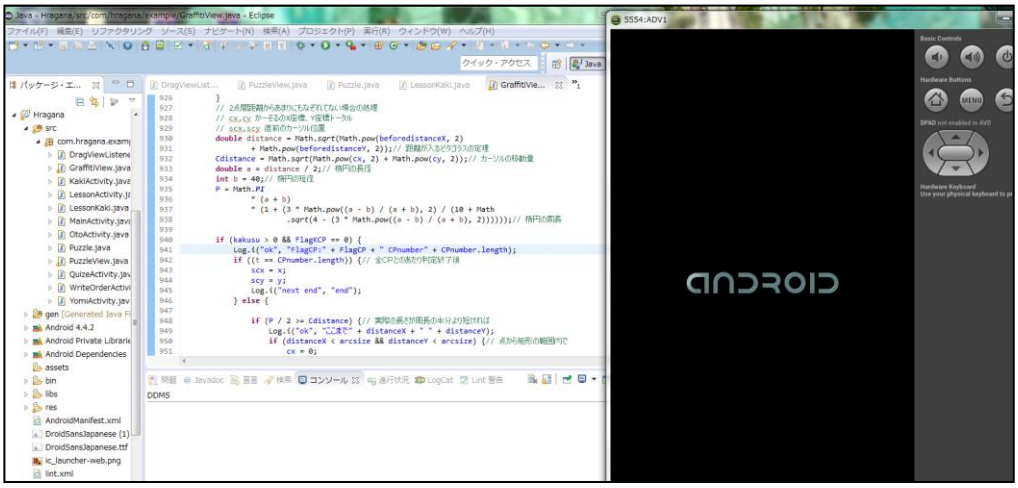

図 10　開発環境

3.3　学習モード

音声の再生には山村の wav 形式の音声ファイルを使用している。またひらがなのアニメーションには Android のライブラリである AnimationDrawable を用いて配列に入っている画像をパラパラ漫画のようにアニメーションしている。また、画像は Bitmap で表示している。これは演習モードをする前にひらがなの形、発音、書き順を学ぶためのものである。また、山村の学習モードでは、ボタンを押すとすべて表示されていた。そこで、1 画 1 画ボタンで表示されたほうが書き順やひらがなの形を覚えるにはいいと考えて 1 画 1 画表示する機能を追加した。

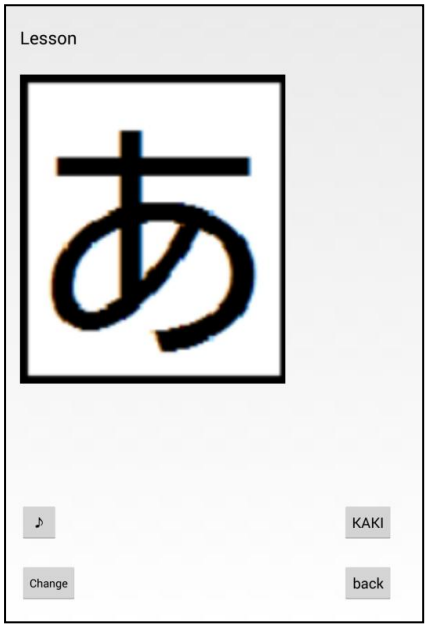

図 11 学習モード画面

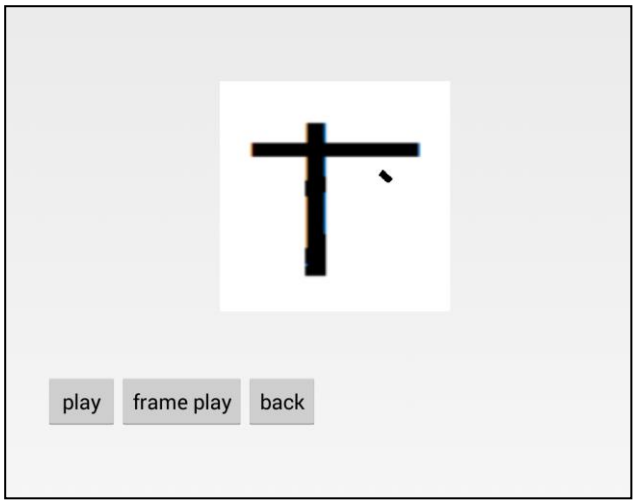

図 12 学習モード画面（書き順）

出力画面は図 11 のようになっている。図 11 の♪のボタンを押すと発音が確認できる。また、KAKI と書かれたボタンを押すと画面が遷移し、書き順を確認することができる画面に変わる。(図 12)

図 12 では play のボタンですべてをアニメーションする。frameplay のボタンで 1 画 1 画をアニメーションする。

3.4 演習モード

はじめに問題形式を選択する。

図 13 のように KAKI、YOMI、OTO、PUZZLE、ORDER から選択できるようになっている。

それぞれ KAKI がなぞり書きの問題、YOMI がひらがなから発音を答える問題、OTO が発音からひらがなの形を答える問題、PUZZLE が五十音表でのパズル問題、ORDER がひらがなの書き順を答える問題となっている。

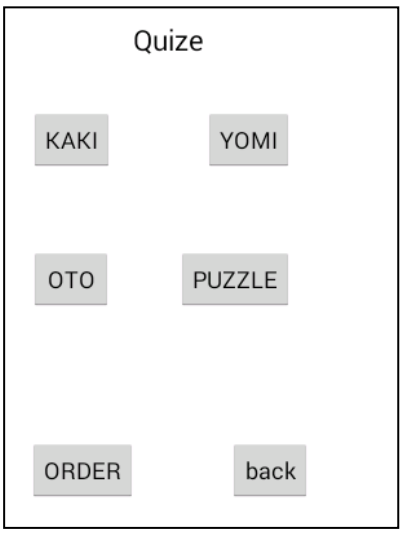

図 13　演習モード選択画面

3.4.1　なぞり書きの問題

問題は図 14 のように表示される。なぞり書きを行った箇所は赤く表示される。Android のライブラリの Canvas を用いることによってタッチに反応して線が描けるようになっている。また、書き順のとおりのなぞり書きを行わない場合は描けないようになっている。また、なぞり書きなので大幅にひらがなからずれたものも書けないようになっている。これは、2 点間距離を取ることによって実現している。

1 画 1 画に判定の為のポイントを複数作っておりそこでポイントを順番のとおりに通っているか、大きく外れていないかを判定している。

画像は濁音、半濁音含めた 71 文字からランダムに表示される。

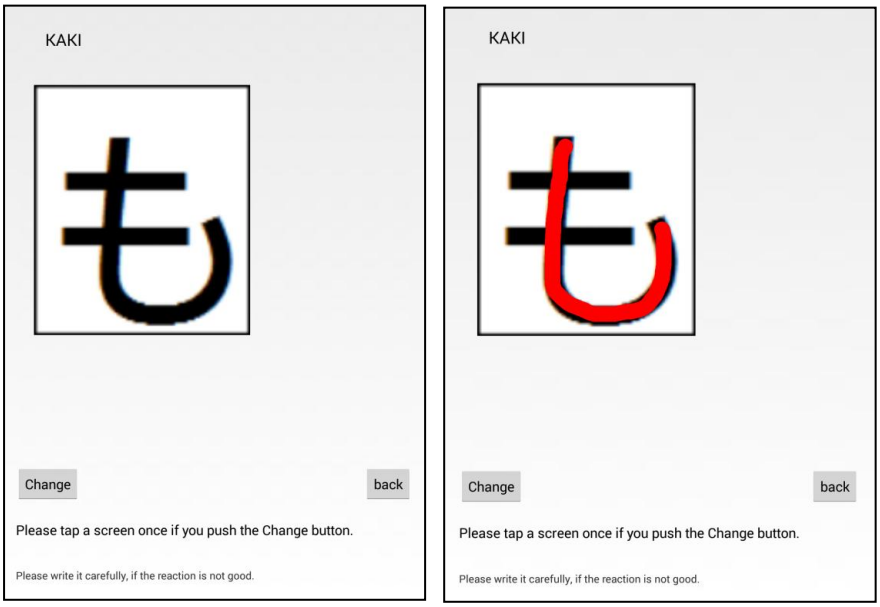

図 14　なぞり書きの問題

3.4.2　ひらがなから発音を答える問題

問題は図 15 のように表示される。ひらがなを見て発音を答える問題である。♪のボタンを押すと発音が聞ける。解答はその下の番号のボタンを押す。正解だった場合「Collect」と表示され、間違った場合は「False」と表示される。

画像と発音はなぞり書きと同じく 71 文字からランダムで表示される。

ひらがなの形と発音が一致していなければ、ひらがなを覚えたとは言えないと考える。そこで、ひらがなの形から発音を答える問題を実装した。

3.4.3　発音からひらがなの形を答える問題

問題は図 16 のように表示される。発音を聞いてひらがなの形を答える問題である。♪のボタンを押すと発音が聞ける。下のひらがなの画像ボタンから選択する。正解だった場合「Collect」と表示され、間違った場合は「False」と表示される。

画像と発音はなぞり書きと同じく 71 文字からランダムで表示される。

上述のとおり、今度はひらがなの形からだけでなく発音からも一致させられるように発音からひらがなの形を答える問題を実装した。

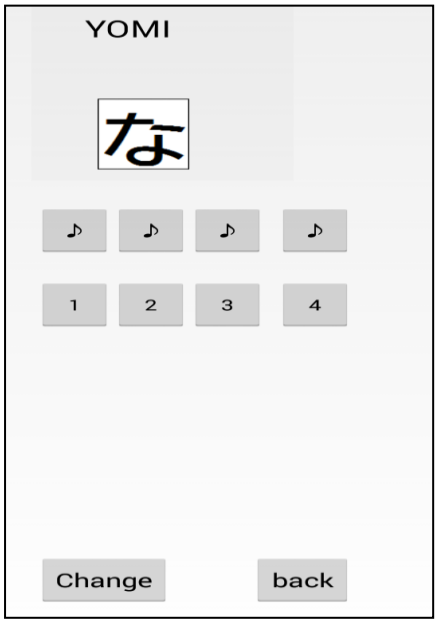

図 15 ひらがなから発音を答える問題

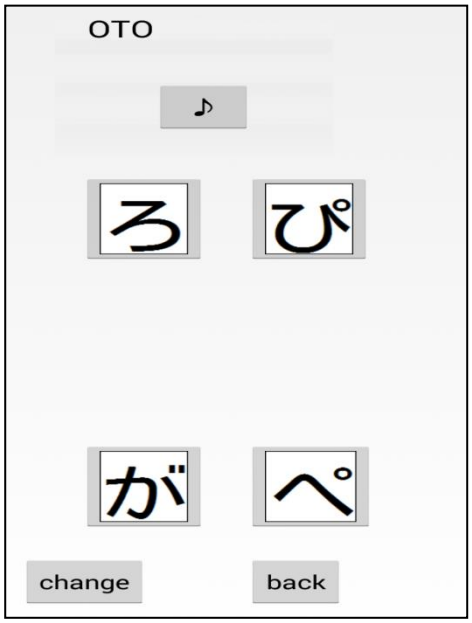

図 16 発音からひらがなの形を答える問題

3.4.4　書き順を答える問題

図 17 のように表示される。2 つのボタンから書き順が正しい方を選択する。1 画目から正しい書き順を選択していく。表示される文字は濁音、半濁音を除く 2 画以上の文字を対象としている。

1 画 1 画正解するごとに「Collect」と表示され、全画正解すると「Congratulations」と表示される。間違うと「False」と表示されるようになっている。

なぞり書きの問題を行う前の段階として、正しく書き順を覚えられているかどうかを確かめる問題として実装した。やはり、ひらがなが書き順のとおり正しく書けていないとひらがなの形としては合っていてもひらがなを覚えたとは言えない。

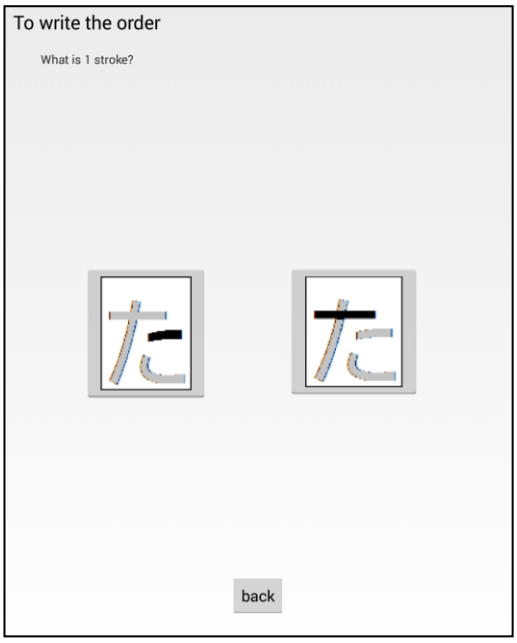

図 17 書き順を答える問題

3.4.5 五十音表でのパズル問題

五十音表での位置を確かめる問題として実装した。五十音表を用いることでひらがなの母音と子音の関係を意識することができれば、ひらがなをより覚えられると考えた。

図 18 のように表示される。図 16 のように五十音表からランダムに 5 つ虫食いにされる。五十音表の下のひらがなのピース（以下ピース）を当てはまるところに当てはめていく。

図 18　パズルの問題

五十音表は縦 5、横 11 の座標で管理されており、ランダムで座標を出したあとそこを青で塗りつぶすことによって虫食い部分を作っている。

ピースはタッチ機能と紐ついており、ピースをタッチすることで指の動きに合わせてピースが動く。ピースは当てはまる虫食いの座標で指を離すとそれ以上動かなくなるようにしてある。

5 つすべてのピースを当てはめ終えると「Congratulations」と表示される。

第4章　結果と考察

本システムは実際に操作する上で問題のない処理速度を実現することができた。また、アプリケーションサイズは 25.93MB に収まっており、アプリケーションとして特段問題のない大きさになっている。

山村の WWW プログラムで実装されていた機能はすべて Android へ移植することができた。また、新機能を加えた。空いた時間に効率よく学習できるアプリが開発できたと考える。

スマートフォンのタッチ機能を活かすことができるなぞり書きの問題を実装できた。

しかし、画面の大きさを Nexus7 で合わせているため、パズル問題で画面サイズが違う場合、虫食い部分がずれてしまう。どの画面サイズでも対応できるように今後の課題としたい。

Android2.2 以上で動作することを、エミュレータを用いて確認した。ただし、推奨は Android4.2 以上となっている。

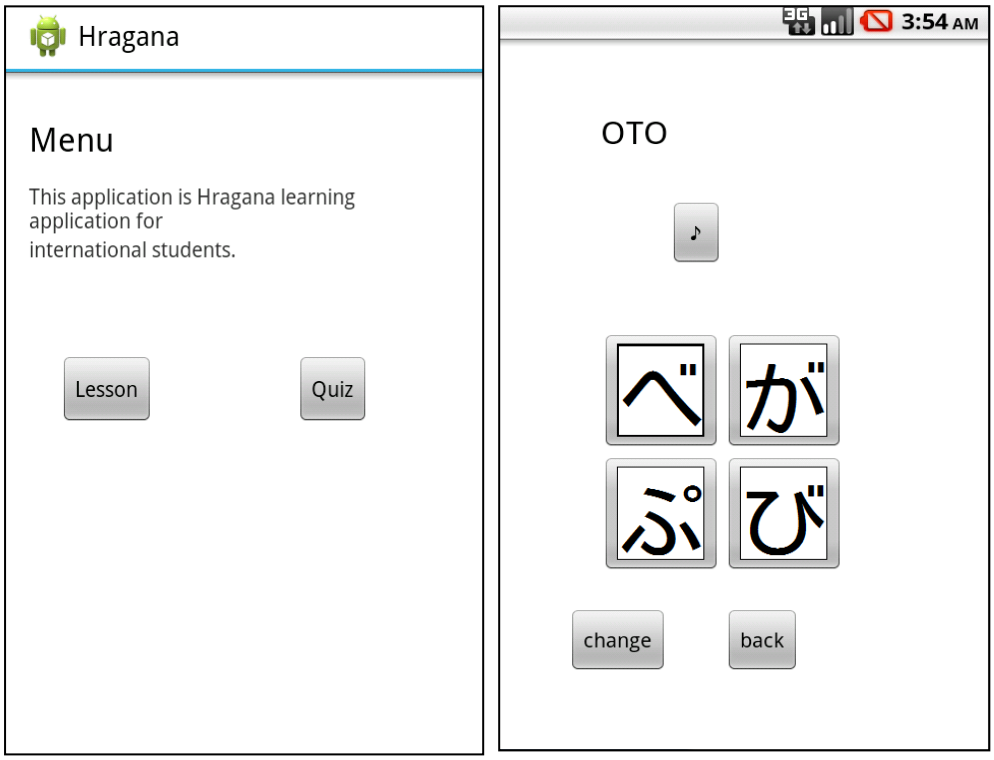

図 19　Android2.2 での実行画面

第5章　結論

本研究では、山村が作製したひらがな学習 WWW プログラムを Android へ移植し、新機能を加えることでより有用な学習支援アプリを開発した。日本語の基礎であるひらがなの形、書き順、発音を学ぶことが Android 端末でできるようになった。これにより、どこでもすぐにひらがなを学習できるようになった。

しかし、Android にしか移植できていないため、今後は iOS の方にも移植し、学習機能を追加することでより使いやすいアプリケーションにしていきたい。

参考文献

［1］ 山村樹：初級日本語学習者のためのひらがな学習プログラムの開発　豊橋技術科学大学　情報・知能工学課程　　卒業論文(2013)

［2］ 大辞林 特別ページ 日本語の世界 3 平仮名
http://daijirin.dual-d.net/extra/hiragana.html

［3］ 国際交流基金　2012 年海外日本語教育機関調査結果
http://www.bunka.go.jp/bunkashingikai/kondankaitou/nihongo_suishin/04/pdf/siryou_11.pdf

［4］ 笹岡勇佑：携帯端末を活用した高専生むけ英単語学習支援システム　豊橋技術科学大学　情報・知能工学専攻　修士学位論文(2013)

［5］ 株式会社スタティクス 「ひらがななぞり」
https://play.google.com/store/apps/details?id=jp.co.statix.hiragana

［6］ Legendarya/Imaginactiva 「Hiragana – Learn Japanese」
https://play.google.com/store/apps/details?id=com.legendarya.helloandroid&hl=ja

謝辞

本研究を進めるにあたり、親切にご指導頂きました河合和久先生、様々なアドバイスを下さった河合研究室の皆様に感謝致します。

付録　ソースコード

GraffitiView.java の一部（なぞり書きの問題）

```
        // 順番通りにCPがなぞられているかの処理
        public void CPlist() {
                if (Integer.parseInt(arrayK1[f][0]) <= x + 30
                                && Integer.parseInt(arrayK1[f][1]) <= y + 30
                                && Integer.parseInt(arrayK1[f][0]) >= x - 30
                                && Integer.parseInt(arrayK1[f][1]) >= y - 30) {
                        if (arrayK1[f][2].equals("end")) {
                        } else if (CPnumber[f - 1] == f) {
                                f++;  Flagf = 1;}}
                if (kakusu > 1 && FlagK1 == 1) {
                        if (Integer.parseInt(arrayK2[f2][0]) <= x + 30
                                        && Integer.parseInt(arrayK2[f2][1]) <= y + 30
                                        && Integer.parseInt(arrayK2[f2][0]) >= x - 30
                                        && Integer.parseInt(arrayK2[f2][1]) >= y - 30) {
                                if (arrayK2[f2][2].equals("end")) {
                                } else if (CPnumber2[f2 - 1] == f2) {
                                        f2++;  Flagf = 1;}}}
                if (kakusu > 2 && FlagK2 == 1) {
                        if (Integer.parseInt(arrayK3[f3][0]) <= x + 30
                                        && Integer.parseInt(arrayK3[f3][1]) <= y + 30
                                        && Integer.parseInt(arrayK3[f3][0]) >= x - 30
                                        && Integer.parseInt(arrayK3[f3][1]) >= y - 30) {
                                if (arrayK3[f3][2].equals("end")) {
                                } else if (CPnumber3[f3 - 1] == f3) {
                                        f3++;  Flagf = 1;}}}
                if (kakusu > 3 && FlagK3 == 1) {
                        if (Integer.parseInt(arrayK4[f4][0]) <= x + 30
                                        && Integer.parseInt(arrayK4[f4][1]) <= y + 30
                                        && Integer.parseInt(arrayK4[f4][0]) >= x - 30
                                        && Integer.parseInt(arrayK4[f4][1]) >= y - 30) {
                                if (arrayK4[f4][2].equals("end")) {
                                } else if (CPnumber4[f4 - 1] == f4) {
                                        f4++;  Flagf = 1;}}}
                if (kakusu > 4 && FlagK4 == 1) {
```

```
                    if (Integer.parseInt(arrayK5[f5][0]) <= x + 30
                                    && Integer.parseInt(arrayK5[f5][1]) <= y + 30
                                    && Integer.parseInt(arrayK5[f5][0]) >= x - 30
                                    && Integer.parseInt(arrayK5[f5][1]) >= y - 30) {
                            if (arrayK5[f5][2].equals("end")) {
                            } else if (CPnumber5[f5 - 1] == f5) {
                                    f5++;  Flagf = 1;}}}
            if (kakusu > 5 && FlagK5 == 1) {
                    if (Integer.parseInt(arrayK6[f6][0]) <= x + 30
                                    && Integer.parseInt(arrayK6[f6][1]) <= y + 30
                                    && Integer.parseInt(arrayK6[f6][0]) >= x - 30
                                    && Integer.parseInt(arrayK6[f6][1]) >= y - 30) {
                            if (arrayK6[f6][2].equals("end")) {
                            } else if (CPnumber6[f6 - 1] == f6) {
                                    f6++;  Flagf = 1;}}}}
    // 2点間の当たり判定
    public void distance(int t) {
            distanceX = Math.abs(x - Integer.parseInt(arrayK1[1][0]));// x座標の距離
            distanceY = Math.abs(y - Integer.parseInt(arrayK1[1][1]));// y座標の距離
            if (StartFlag == 1) {
                    beforedistanceX = Math.abs(Integer.parseInt(arrayK1[1][0])
                                    - Integer.parseInt(arrayK1[0][0]));
                    beforedistanceY = Math.abs(Integer.parseInt(arrayK1[1][1])
                                    - Integer.parseInt(arrayK1[0][1]));
            } else if (StartFlag == 2) {
            beforedistanceX=Math.abs(Integer.parseInt(arrayK1[arrayK1.length-1][0])
                                    - Integer.parseInt(arrayK1[arrayK1.length][0]));
            beforedistanceY = Math.abs(Integer.parseInt(arrayK1[arrayK1.length-1][1])
                                    - Integer.parseInt(arrayK1[arrayK1.length][1]));
            } else { Log.i("error", "error1");}
            if (kakusu > 1 && FlagK1 == 1) {
            distanceX = Math.abs(x - Integer.parseInt(arrayK2[1][0]));// x座標の距離
            distanceY = Math.abs(y - Integer.parseInt(arrayK2[1][1]));// y座標の距離
            if (FlagST == 2) {beforedistanceX = Math.abs(Integer.parseInt(arrayK2[1][0])
                                            - Integer.parseInt(arrayK2[0][0]));
                            beforedistanceY = Math.abs(Integer.parseInt(arrayK2[1][1])
```

```
                                        - Integer.parseInt(arrayK2[0][1]));
                } else if (f2 - 1 == CPnumber2.length) {
                beforedistanceX = Math.abs(Integer.parseInt(arrayK2[1][0])
                                - Integer.parseInt(arrayK2[arrayK2.length - 1][0]));
                beforedistanceY = Math.abs(Integer.parseInt(arrayK2[1][1])
                                - Integer.parseInt(arrayK2[arrayK2.length - 1][1]));
                } else { Log.i("error", "error2");}}
            if (kakusu > 2 && FlagK2 == 1) {
            distanceX = Math.abs(x - Integer.parseInt(arrayK3[1][0]));// x座標の距離
            distanceY = Math.abs(y - Integer.parseInt(arrayK3[1][1]));// y座標の距離
            if (FlagST == 3) {beforedistanceX = Math.abs(Integer.parseInt(arrayK3[1][0])
                                        - Integer.parseInt(arrayK3[0][0]));
                        beforedistanceY = Math.abs(Integer.parseInt(arrayK3[1][1])
                                        - Integer.parseInt(arrayK3[0][1]));
            } else if (f3 - 1 == CPnumber3.length) {
                        beforedistanceX = Math.abs(Integer.parseInt(arrayK3[1][0])
                                - Integer.parseInt(arrayK3[arrayK3.length - 1][0]));
                        beforedistanceY = Math.abs(Integer.parseInt(arrayK3[1][1])
                                - Integer.parseInt(arrayK3[arrayK3.length - 1][1]));
                } else { Log.i("error", "error3");}}
            if (kakusu > 3 && FlagK3 == 1) {
            distanceX = Math.abs(x - Integer.parseInt(arrayK4[1][0]));// x座標の距離
            distanceY = Math.abs(y - Integer.parseInt(arrayK4[1][1]));// y座標の距離
            if (FlagST == 4) {beforedistanceX = Math.abs(Integer.parseInt(arrayK4[1][0])
                                        - Integer.parseInt(arrayK4[0][0]));
                        beforedistanceY = Math.abs(Integer.parseInt(arrayK4[1][1])
                                        - Integer.parseInt(arrayK4[0][1]));
                } else if (f4 - 1 == CPnumber4.length) {
                        beforedistanceX = Math.abs(Integer.parseInt(arrayK4[1][0])
                                - Integer.parseInt(arrayK4[arrayK4.length - 1][0]));
                        beforedistanceY = Math.abs(Integer.parseInt(arrayK4[1][1])
                                - Integer.parseInt(arrayK4[arrayK4.length - 1][1]));
                } else { Log.i("error", "error4"); }}
            if (kakusu > 4 && FlagK4 == 1) {
            distanceX = Math.abs(x - Integer.parseInt(arrayK5[1][0]));// x座標の距離
            distanceY = Math.abs(y - Integer.parseInt(arrayK5[1][1]));// y座標の距離
```

```
                if (FlagST == 5) {beforedistanceX = Math.abs(Integer.parseInt(arrayK5[1][0])
                                                - Integer.parseInt(arrayK5[0][0]));
                        beforedistanceY = Math.abs(Integer.parseInt(arrayK5[1][1])
                                                - Integer.parseInt(arrayK5[0][1]));
                } else if (f5 - 1 == CPnumber5.length) {
                        beforedistanceX = Math.abs(Integer.parseInt(arrayK5[1][0])
                                        - Integer.parseInt(arrayK5[arrayK5.length - 1][0]));
                        beforedistanceY = Math.abs(Integer.parseInt(arrayK5[1][1])
                                        - Integer.parseInt(arrayK5[arrayK5.length - 1][1]));
                } else { Log.i("error", "error5"); }}
                if (kakusu > 5 && FlagK5 == 1) {
                distanceX = Math.abs(x - Integer.parseInt(arrayK6[1][0]));// x座標の距離
                distanceY = Math.abs(y - Integer.parseInt(arrayK6[1][1]));// y座標の距離
                if (FlagST == 6) {beforedistanceX = Math.abs(Integer.parseInt(arrayK6[1][0])
                                                - Integer.parseInt(arrayK6[0][0]));
                        beforedistanceY = Math.abs(Integer.parseInt(arrayK6[1][1])
                                                - Integer.parseInt(arrayK6[0][1]));
                } else if (f6 - 1 == CPnumber6.length) {
                        beforedistanceX = Math.abs(Integer.parseInt(arrayK6[1][0])
                                        - Integer.parseInt(arrayK6[arrayK6.length - 1][0]));
                        beforedistanceY = Math.abs(Integer.parseInt(arrayK6[1][1])
                                        - Integer.parseInt(arrayK6[arrayK6.length - 1][1]));
                } else { Log.i("error", "error6");}}
                // 2点間距離からあまりにもなぞれてない場合の処理
                // cx,cy カーソルのX座標、Y座標トータル
                // scx,scy  直前のカーソル位置
                double distance = Math.sqrt(Math.pow(beforedistanceX, 2)
                                + Math.pow(beforedistanceY, 2));// 距離が入るピタゴラスの定理
                Cdistance = Math.sqrt(Math.pow(cx, 2) + Math.pow(cy, 2));// カーソルの移動量
                double a = distance / 2;// 楕円の長径
                int b = 40;// 楕円の短径
                   P = Math.PI* (a + b)* (1 + (3 * Math.pow((a - b) / (a + b), 2) / (10 + Math.sqrt(4
                   - (3 * Math.pow((a - b) / (a + b), 2))))));// 楕円の周長
                if (kakusu > 0 && FlagKCP == 0) {
                        if ((t == CPnumber.length)) {// 全ＣＰとのあたり判定終了後
                                scx = x;         scy = y;
```

```
                } else { if (P / 2 >= Cdistance) {// 実際の長さが周長の半分より短ければ
            if (distanceX < arcsize && distanceY < arcsize) {// 点から矩形の範囲内で
                cx = 0;  cy = 0;  scx = x;  scy = y;  FlagCP++;}
                    } else { delete();}}}
            if (kakusu > 1 && FlagK1 == 1) {
                if ((FlagCP == CPnumber2.length)) {// 全ＣＰとのあたり判定終了後
                    scx = x;  scy = y;  Log.i("next end", "end");
                } else { if (P / 2 >= Cdistance) {// 実際の長さが周長の半分より短ければ
            if (distanceX < arcsize && distanceY < arcsize) {// 点から矩形の範囲内で
                        cx = 0;  cy = 0;  scx = x;  scy = y;  FlagCP++;}
                    } else { delete();}}}
            if (kakusu > 2 && FlagK2 == 1) {
                if ((FlagCP == CPnumber3.length)) {// 全ＣＰとのあたり判定終了後
                    scx = x;  scy = y;  Log.i("next end", "end");
                } else { if (P / 2 >= Cdistance) {// 実際の長さが周長の半分より短ければ
            if (distanceX < arcsize && distanceY < arcsize) {// 点から矩形の範囲内で
                        cx = 0;  cy = 0;  scx = x;  scy = y;  FlagCP++;}
                    } else {delete();}}}
            if (kakusu > 3 && FlagK3 == 1) {
                if ((FlagCP == CPnumber4.length)) {// 全ＣＰとのあたり判定終了後
                    scx = x;  scy = y;  Log.i("next end", "end");
        } else { if (P / 2 >= Cdistance) {// 実際の長さが周長の半分より短ければ
            if (distanceX < arcsize && distanceY < arcsize) {// 点から矩形の範囲内で
                        cx = 0;  cy = 0;  scx = x;  scy = y;  FlagCP++;}
                    } else { delete();}}}
            if (kakusu > 4 && FlagK4 == 1) {
                if ((FlagCP == CPnumber5.length)) {// 全ＣＰとのあたり判定終了後
                    scx = x;  scy = y;
                    Log.i("next end", "end");
                } else { if (P / 2 >= Cdistance) {// 実際の長さが周長の半分より短ければ
            if (distanceX < arcsize && distanceY < arcsize) {// 点から矩形の範囲内で
                        cx = 0;  cy = 0;  scx = x;  scy = y;  FlagCP++;}
                    } else { delete();}}}
            if (kakusu > 5 && FlagK5 == 1) {
                if ((FlagCP == CPnumber6.length)) {// 全ＣＰとのあたり判定終了後
                    scx = x;  scy = y;
```

```
                        Log.i("next end", "end");
                    } else { if (P / 2 >= Cdistance) {// 実際の長さが周長の半分より短ければ
            if (distanceX < arcsize && distanceY < arcsize) {// 点から矩形の範囲内で
                                cx = 0;  cy = 0;  scx = x;  scy = y;  FlagCP++;}
                        } else { delete();}}}}
    public void startHit() {
            for (int i = 0; i < arrayK1.length; i++) {
                    int distanceX = (int) Math.abs(x - Integer.parseInt(arrayK1[i][0]));
                    int distanceY = (int) Math.abs(y - Integer.parseInt(arrayK1[i][1]));
                    if (distanceX < arcsize && distanceY < arcsize) {
                            if (arrayK1[i][2].equals("start")) {
                                    StartFlag = 1;
                            }else if(arrayK1[i][2].equals("end")){
                                    StartFlag = 2;}
                            this.mPath.moveTo(x, y);
                            this.drawPath(this.getmWallCanvas(), this.mPath);}}}
    public void endHit(int i) {
            StartFlag = 0;
            EndFlag = 0;
            if (arrayK1[i][2].equals("end")) {
                    float distanceX = Math.abs(scx - Integer.parseInt(arrayK1[i][0]));
                    float distanceY = Math.abs(scy - Integer.parseInt(arrayK1[i][1]));
                    if (P / 2 >= Cdistance) {
                            if (distanceX < arcsize && distanceY < arcsize) {
                                    this.mPath.moveTo(x, y);
                                    this.drawPath(this.getmWallCanvas(), this.mPath);}}}}
```

Puzzle.java の一部（五十音表でのパズル問題）

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.practice);
            this.randam();
            // リソースからbitmapを作成
            Bitmap image = BitmapFactory.decodeResource(getResources(),
                            R.drawable.gojuon);
            Button backbtn = (Button) findViewById(R.id.PuzzleButton);
```

```
                backbtn.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                                // TODO Auto-generated method stub
                                // 次画面のアクティビティ終了
                                finish();}});
                // 画像サイズ取得
                width = image.getWidth();
                height = image.getHeight();
                // 五十音表の表示
                ImageView gojuon = (ImageView) findViewById(R.id.imageView_Board);
                gojuon.setImageBitmap(image);
                top = gojuon.getTop();
                left = gojuon.getLeft();
                // ドラッグ対象Viewとイベント処理クラスを紐付ける
                ImageView dragView = (ImageView) findViewById(R.id.imageView1_Puzzle2);
                this.DragView(dragView,piece[0]);
                // ドラッグ対象Viewとイベント処理クラスを紐付ける
                ImageView dragView2 = (ImageView) findViewById(R.id.imageView1_Puzzle);
                this.DragView(dragView2,piece[1]);
                // ドラッグ対象Viewとイベント処理クラスを紐付ける
                ImageView dragView3 = (ImageView) findViewById(R.id.imageView1_Puzzle3);
                this.DragView(dragView3,piece[2]);
                // ドラッグ対象Viewとイベント処理クラスを紐付ける
                ImageView dragView4 = (ImageView) findViewById(R.id.imageView1_Puzzle4);
                this.DragView(dragView4,piece[3]);
                // ドラッグ対象Viewとイベント処理クラスを紐付ける
                ImageView dragView5 = (ImageView) findViewById(R.id.imageView1_Puzzle5);
                this.DragView(dragView5,piece[4]);
        }
        // ランダム数に応じてリソースを返す。
        public Drawable getRandomResource(int number) {
                return getResources().getDrawable(moji_gazo[number]);
        }
        // ドラッグ対象Viewとイベント処理クラスを紐つける
        public void DragView(ImageView dragView,int i) {
```

```
                Drawable drawable2 = getRandomResource(i);
                dragView.setImageDrawable(drawable2);
                DragViewListener listener = new DragViewListener(this,dragView, width,
                                height,top,left,pointw,pointh,i,DragFlag);
                dragView.setOnTouchListener(listener);     }
```

PuzzleView.java の一部（五十音表のパズル問題の虫食い部分）

```
        // Viewを取り込んでいると自動で作成されるメソッド
        //五十音表虫食い部分
        @Override
        protected void onDraw(Canvas c) {
                super.onDraw(c);
                Paint paint = new Paint();
            paint.setColor(Color.argb(255, 0, 0, 255));//青色で色をセット
                 pointw=puzzle.getPontW();   pointh=puzzle.getPontH();
                 //塗りつぶす画像の大きさ
                 width = puzzle.getWidth()/11;//五十音表画像の横の長さを取り、子音数で割る
                 height = puzzle.getHeight()/5;//五十音表画像の縦の長さを取り、母音数で割る
                 top = puzzle.getTop()+height*2-13;    left = puzzle.getLeft();
         WindowManager wm = (WindowManager)getContext().getSystemService(Context.WINDOW_SERVICE);
                // ディスプレイのインスタンス生成
                Display disp = wm.getDefaultDisplay();
                Point size = new Point();  disp.getSize(size);
                int dispwidth = size.x/150;  int dispheight =size.y/260;
                rect1 = new Rect(left+(width*pointw[0]+pointw[0]*dispwidth),
top+(height*pointh[0]+pointh[0]*dispheight),left+width+(width*pointw[0]+pointw[0]*dispwidth),
top+height+(height*pointh[0]+pointh[0]*dispheight));
                c.drawRect(rect1, paint);
                rect2 = new Rect(left+(width*pointw[1]+pointw[1]*dispwidth),
top+(height*pointh[1]+pointh[1]*dispheight), left+width+(width*pointw[1]+pointw[1]*dispwidth),
top+height+(height*pointh[1]+pointh[1]*dispheight));
                c.drawRect(rect2, paint);
                rect3 = new Rect(left+(width*pointw[2]+pointw[2]*dispwidth),
top+(height*pointh[2]+pointh[2]*dispheight), left+width+(width*pointw[2]+pointw[2]*dispwidth),
top+height+(height*pointh[2]+pointh[2]*dispheight));
                c.drawRect(rect3, paint);
                rect4 = new Rect(left+(width*pointw[3]+pointw[3]*dispwidth),
```

```
top+(height*pointh[3]+pointh[3]*dispheight), left+width+(width*pointw[3]+pointw[3]*dispwidth),
top+height+(height*pointh[3]+pointh[3]*dispheight));
                c.drawRect(rect4, paint);
                rect5 = new Rect(left+(width*pointw[4]+pointw[4]*dispwidth),
top+(height*pointh[4]+pointh[4]*dispheight), left+width+(width*pointw[4]+pointw[4]*dispwidth),
top+height+(height*pointh[4]+pointh[4]*dispheight));
                c.drawRect(rect5, paint);
        }
```

DragViewListener.java の一部（五十音表でのパズル問題のピース部分）

```
        //タッチイベント
        @Override
        public boolean onTouch(View view, MotionEvent event) {
                // タッチしている位置取得
                int x = (int) event.getRawX();  int y = (int) event.getRawY();
                 WindowManager wm = indowManager)puzzle.getSystemService(Context.WINDOW_SERVICE);
                // ディスプレイのインスタンス生成
                Display disp = wm.getDefaultDisplay();
                Point size = new Point();  disp.getSize(size);
                int dispwidth = size.x/150;  int dispheight =size.y/260;
                switch (event.getAction()) {
                //虫食い部分で指を話したら判定Flagを建てる
                case MotionEvent.ACTION_UP:
                if (peace == number[0] && PeaceFlag==0) {
                if (x - 40 >= left2 + (width2 * pointw[0] + pointw[0] * dispwidth)
                  && y - 210 >= top2+ (height2 * pointh[0] + pointh[0] * dispheight)
                  && x - 40 <= left2 + width2+ (width2 * pointw[0] + pointw[0] * dispwidth)
                  && y - 210 <= top2 + height2+ (height2 * pointh[0] + pointh[0] * dispheight)) {
        puzzle.setDragFlag();  PeaceFlag = 1;  Log.d("PeaceFlag1", "PeaceFlag=" + PeaceFlag);}}
                if (peace == number[1] && PeaceFlag==0) {
                if (x - 40 >= left2 + (width2 * pointw[1] + pointw[1] * dispwidth)
                  && y - 210 >= top2+ (height2 * pointh[1] + pointh[1] * dispheight)
                  && x - 40 <= left2 + width2+ (width2 * pointw[1] + pointw[1] * dispwidth)
                  && y - 210 <= top2 + height2+ (height2 * pointh[1] + pointh[1] * dispheight)) {
        puzzle.setDragFlag();  PeaceFlag = 2;  Log.d("PeaceFlag2", "PeaceFlag=" + PeaceFlag);}}
                if (peace == number[2] && PeaceFlag==0) {
                if (x - 40 >= left2 + (width2 * pointw[2] + pointw[2] * dispwidth)
```

```
                && y - 210 >= top2+ (height2 * pointh[2] + pointh[2] * dispheight)
                && x - 40 <= left2 + width2+ (width2 * pointw[2] + pointw[2] * dispwidth)
                && y - 210 <= top2 + height2+ (height2 * pointh[2] + pointh[2] * dispheight)) {
        puzzle.setDragFlag();  PeaceFlag = 3;  Log.d("PeaceFlag3", "PeaceFlag=" + PeaceFlag);}}
            if (peace == number[3] && PeaceFlag==0) {
            if (x - 40 >= left2 + (width2 * pointw[3] + pointw[3] * dispwidth)
                && y - 210 >= top2+ (height2 * pointh[3] + pointh[3] * dispheight)
                && x - 40 <= left2 + width2+ (width2 * pointw[3] + pointw[3] * dispwidth)
                && y - 210 <= top2 + height2+ (height2 * pointh[3] + pointh[3] * dispheight)) {
        puzzle.setDragFlag();  PeaceFlag = 4;  Log.d("PeaceFlag4", "PeaceFlag=" + PeaceFlag);}}
            if (peace == number[4] && PeaceFlag==0) {
            if (x - 40 >= left2 + (width2 * pointw[4] + pointw[4] * dispwidth)
                && y - 210 >= top2+ (height2 * pointh[4] + pointh[4] * dispheight)
                && x - 40 <= left2 + width2+ (width2 * pointw[4] + pointw[4] *dispwidth)
                && y - 210 <= top2 + height2+ (height2 * pointh[4] + pointh[4] * dispheight)) {
        puzzle.setDragFlag();  PeaceFlag = 5;  Log.d("PeaceFlag5", "PeaceFlag=" + PeaceFlag);}}
            case MotionEvent.ACTION_MOVE:
                    // 今回イベントでのView移動先の位置
                    int left = dragView.getLeft() + (x - oldx);
                    int top = dragView.getTop() + (y - oldy);
                    // Viewを移動する
                    Log.d("PeaceFlag", "PeaceFlag=" + PeaceFlag);
                    //各ピースのフラグが建っていたらこれ以上動かさない
                    if (PeaceFlag == 0) {
                    dragView.layout(left, top, left + width2+10, top + height2+10);}
                    break;}
            // 今回のタッチ位置を保持
            oldx = x;  oldy = y;
            // イベント処理完了
            return true;  }
```