平成２７年度

卒業論文

題目	Androidアプリ「絵しりとりメモ」の企画と開発

担当教員 （自署）		印

学籍番号　201214020

氏　名　長濱　弘樹

広島経済大学

要旨

第 1 章では、「絵しりとりメモ」を開発するに至った経緯を述べた。

第 2 章では、「絵しりとりメモ」を開発するにあたっての目的、どのようなアプリであるかを計画した開発計画について述べた。

第 3 章では、開発計画に基づいた実際の仕様を述べた。また、使用方法や実際に制作したプログラムについても段階を追って説明した。

第 4 章では完成した「絵しりとりメモ」の評価方法とその結果、評価に対する総評を述べた。

第 5 章では、第 1 章で述べた経緯に対する総評と、本論文を執筆しての考察や総評を述べた。

目次

第1章 はじめに

Android アプリ「絵しりとりメモ」を制作することになった経緯は 3 つある。1 つ目は、新規アプリを開発するにあたって、オリジナル性があるものを作りたかったからだ。Android のアプリ市場である Google Play で絵しりとりアプリを検索したところヒット数が非常に少なかった。一方、お絵描きアプリを検索した場合にはたくさんのアプリがヒットした。アプリそのものが少ない絵しりとりというジャンルはオリジナル性があると考えたのが 1 つ目の経緯だ。今回は過去に制作したサンプルアプリの中から白い画面に絵を描くというシンプルなお絵描きアプリを選択し、派生させて絵しりとりのアプリにしている。2 つ目の経緯は、1 つのアプリを複数の用途に使えるようにしたかったからだ。1 つのアプリで 1 つの事しかできないというのは少し面白みに欠けるのではないかと考えたため、機能は少なくてもできることを増やすことにした。3 つ目の経緯は、誰にでも使えるアプリを作りたいという点だ。老若男女に関係なく気軽に使えることは強いメリットになると考えた。ターゲットを絞らずに広い視野でのアプリ制作を心がけている。以上 3 つの点が Android アプリ「絵しりとりメモ」を制作するに至った経緯だ。

第2章 Android アプリ「絵しりとりメモ」の企画

第1節 目的

Android アプリ「絵しりとりメモ」を制作した目的は、第 1 章はじめにの中で述べたオリジナル性、ひとつのアプリで複数の機能、誰にでも使えるという 3 点の条件を満たした上で、1 人でも複数人でも使う事の出来るアプリを作ることだ。基本的に絵しりとりアプリというと、あくまでもしりとりというゲームをするアプリであるため複数人で使うことを想定しているものが多い。しかし今回の Android アプリ「絵しりとりメモ」では、描いたものを画像として保存するというメモの機能を追加することにより、絵しりとり以外での使用用途も生まれるため、1 人で使うこともできる。絵しりとりという複数人向けのゲームに、機能を追加するというアレンジをすることで、1 人でも複数人でも使う事の出来るアプリを作るという目的をクリアすることができると考えた。

第2節 開発計画

Android アプリ「絵しりとりメモ」の開発を計画するにあたって、次の 3 つの制作段階に分けて案を作成した。1 つ目は、何のアプリを開発するのかという点を決めた。最初はゼミナールで作成したお絵描きアプリを派生させ、シンプルな絵しりとりのアプリを作ろうと考えていた。内容は、飲み会などで盛り上がるためのツールとして、複数人で絵しりとりをするアプリとなっていた。しかし、類似した絵しりとりのアプリがすでに Google Play に存在していたため、何かアレンジを加えて作成する方向性になった。2 つ目は、絵しりとりアプリに実装したい機能を決めた。当然、絵しりとりアプリであるため描いた絵を使ってしりとりをする機能は第一の機能として考えた。また、ただの絵しりとりアプリとしての違いを出すために描いた絵を保存できる機能を追加することを考えた。3 つ目は、どのようなレイアウトで作成するかという点を決めた。具体的な画面のイメージや操作方法を考え、出たアイデアを絵として描いた。画面は絵を表示する画面と絵を描く画面の 2 種類あり、できるだけシンプルで見やすい画面にすることとした。

第3章 Android アプリ「絵しりとりメモ」の制作

第1節 仕様

第 2 章第 2 節の開発計画において 3 つの点について計画をしたが、実際作成するにあたっての仕様は変更されたものもある。ここでは確定した実際の仕様を説明し、実装に取り掛かるまでの過程を書いていく。

第1項 作成するアプリ

はじめに、何のアプリを作成するのかについてだが、アプリ名を「絵しりとりメモ」に確定し、概要は「複数人で盛り上がるためのツールとしての機能はもちろん、1 人でも使える機能を加え多様性を持たせたアプリ」とした。次に、「絵しりとりメモ」に実装する機能についてだが、絵や文字を描く機能、描いた絵を表示する機能、描いた絵を消す機能、描く線の色や太さを変える機能、描いた絵を画像として保存する機能の 5 つが実装されている。絵や文字を描く機能とは、実際に白いキャンバス画面の範囲内に自由に絵や文字を描くことができる機能のことだ。今回のアプリにおける描いた絵を表示する機能とは、キャンバス画面に描いた絵や文字を一時的に保存し、別途用意されている領域に表示させる機能のことだ。描いた絵を消す機能とは、自由に描いた絵を一度に全部消去する機能のことだ。描く線の色や太さを変える機能とは、キャンバス画面に絵や文字を描く際に数種類の色や線の太さを選べるようにしている機能のことだ。描いた絵を画像として保存する機能とは、キャンバス画面に描いて完成した絵や文字を使用している端末内のフォルダに画像ファイルとして保存する機能の事だ。この 5 つの機能の中の描いた絵を画像として保存する機能を利用する事によって、後から描いた絵を振り返るとともに、メモとしての機能を実質的に追加した。ここまでに説明した 5 つの機能を組み合わせることで、絵しりとりとしての機能とメモとしての機能を持ち合わせた「絵しりとりメモ」のアプリとなった。

第2項　レイアウト・機能

ここからは、実際に 5 つの機能を実装したレイアウトについて説明する。画面の種類は大きく分けて 2 種類あり、「描いた絵を表示する画面」と「絵を描く画面」だ。「描いた絵を表示する画面」では、画面上に 2 枚の画像が表示されており上の画像にはテキストで「絵しりとり」と仮のタイトルが書かれている。下の画像にはテキストで「下画面をタッチで絵を描くことができます」と説明文が書かれている。仮想の Android デバイスで起動した時の画面のスクリーンショットは以下のようになっている。

図 1

「下画面をタッチで絵を描くことができます」とあるように、2 つ並んでいるうちの下の枠をタッチする事で「描いた絵を表示する画面」から次の「絵を描くためのキャンバス画面」に移動するようになっている。「絵を描くためのキャンバス画面」のレイアウトだが、白いキャンバス部分と 4 つのボタンで構成されている。白いキャンバス部分には、Color ボタンと Width(1)ボタンで設定されたオプションに基づいて、自由にタッチ・スワイプすることで絵を描くことができる。4 つのボタンは画面の下部に横に並んでいて、それぞれ「Clear」、「Color」、「Width(1)」、「Save」となっている。仮想の Android デバイスで起動した時のスクリーンショットは以下のようになっている。

図 2

Clear ボタンは、描いた絵を消す機能を持っており、このボタンを押すことで白いキャンバス画面に描いていた絵がリセットされ白紙になる。キャンバス画面に描いたものが全て消える仕様になっているので、今後の改良が必要な項目だ。Color ボタンと Width(1)ボタンは、描く線の色や太さを変える機能を持っている。Color ボタンは、タッチする事で Color の文字の色が変化し、キャンバス画面に描画される線の色が Color の文字の色と同じ色に変更する。例えば、Color の文字が黒なら黒い線、黄色なら黄色い線になる。色は 5 種類あり、黒、青、赤、緑、黄色となっているので、組み合わせる事で様々な種類の絵を描けるようになっている。Width(1)ボタンは、描画する線の太さを変えるボタンであり、()内の数字によって太さが変わる仕組みになっている。タッチするごとに()内の数字が 1 ずつ増えていき、最大で 5 まである。数字が大きい状態ほど描画する線の太さは太くなる。Save ボタンは、キャンバス画面に描いた絵を画像として保存する機能を持っている。保存されるフォルダは自動的にルートディレクトリに作成され、名前は「ImageSaveDir」となっている。画像は jpg 形式で保存される。また、決定や確定ボタンとしての意味合いもあり、このボタンを押すことによってキャンバス画面を終了し、起動時の画面に移動し且つ 2 つ並んでいるうちの上の枠にキャンバス画面に描かれていた絵が表示されるという機能も持っている。以上が、上図のボタンについての機能説明である。

第3項 使用方法

ここでは「絵しりとりメモ」の使用方法を解説していく。解説には実際に Android 端末で動かして撮影したスクリーンショットを使用する。このアプリは大きく分けてゲームとして使用する場合とメモとして使う場合の 2 種類の使用方法がある。ゲームとしての使用には現在「絵しりとり」と「お絵描き伝言」の 2 パターンを考えている。

絵しりとり

「絵しりとり」として遊ぶ場合の人数は 1~5 人程度であり、複数人で遊ぶ場合には 1 つの端末を回しながら使用する。今回は例として人数は 2 人と仮定し、文字の太さを 5 ポイントにしてヒヨコとコップの絵を描くこととする。まずタイトル画面で、上下に 2 つある枠のうちの下の枠をタッチする。すると絵を描く画面に移動する。

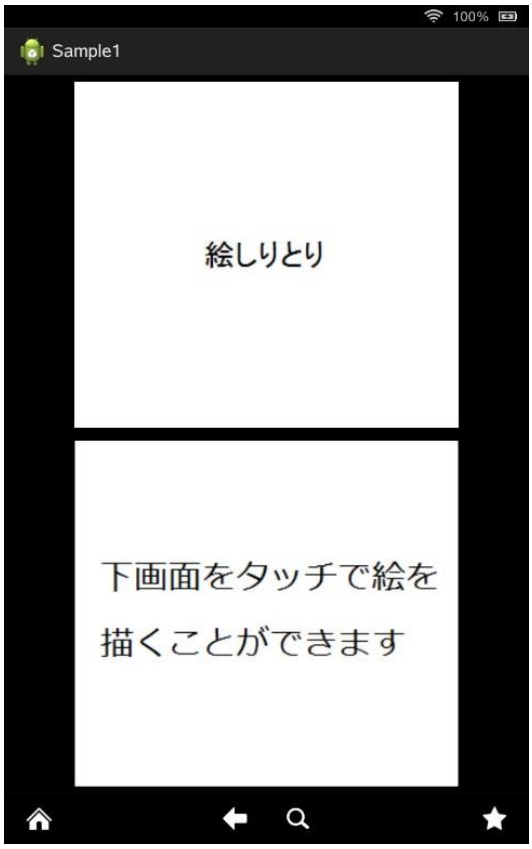


図 3

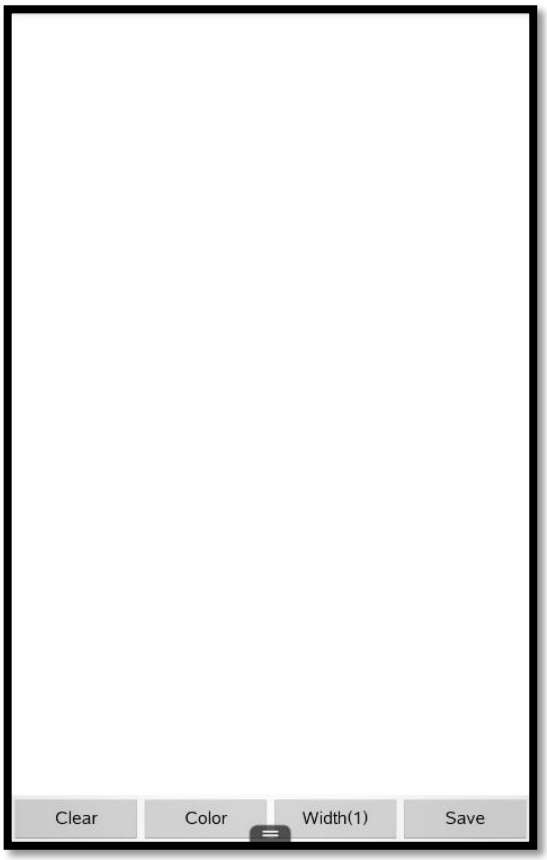


図 4

絵を描く画面に移動したら、Color ボタンで文字の色を選び、Width(1)ボタンで文字の太さを選ぶ。文字の色と太さを選んだら、白いキャンバスの画面に自由にタッチやスワイプで絵を描く。絵を描いているときにミスをしてしまい消したい場合には Clear ボタンを押すことで、キャンバス画面がリセットされ最初から絵を描きなおすことができる。絵を描き終えたら、Save ボタンを押し画像を保存する。するとタイトル画面に戻り、2 つあったうちの上の枠には下の枠の中の画像が、下の枠には先程描いた絵が表示される。

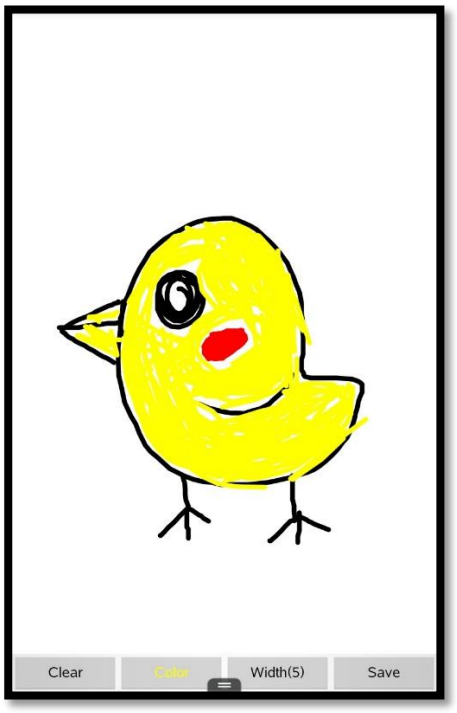


図 5

図 6

次に絵を描く人は、下の画面をタッチすることで絵を描く画面に移動することができる。絵を描く画面に移動したら、Color ボタンと Width(1)ボタンで文字の色と太さを選び、前の絵の最後の言葉（今回の場合はヒヨコの「コ」）から始まるものの絵を描き Save ボタンを押して保存する。すると、タイトル画面に戻り先程のヒヨコの絵は上の枠へ、新たに描いたコップの絵は下の枠へ表示される。次の人は表示されたコップの絵をタッチすることで新たな絵を描く画面に移動し、コップの「プ」で始まる絵を描く。これを繰り返し、最終的に絵を描くことができなくなった人の負けという遊び方だ。

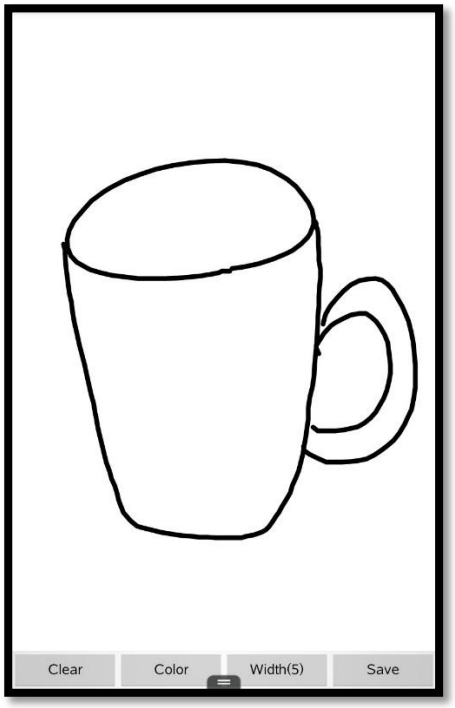


図 7

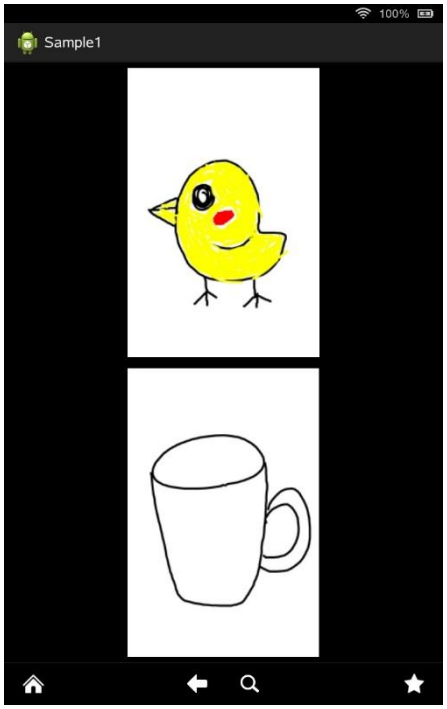


図 8

このように、タイトル画面で下の枠をタッチし、文字の色と太さを選び、自由に絵を描き Save ボタンを押して絵を保存しタイトル画面に戻るという一連の流れを繰り返すことで絵しりとりゲームとしての機能を達成している。以上が「絵しりとりメモ」における絵しりとりとしての使用方法である。

お絵描き伝言

ゲームとして使用する場合の 2 つ目はお絵描き伝言だ。お絵描き伝言とは複数人で遊ぶゲームで、お題を用意し、最初の人がお題の絵を描いて次の人に渡す。次に渡された人はその絵を見てお題を想像してまた絵を描く。これを最後の人まで繰り返し、最後の人はお題が何だったのかを答えるというゲームだ。「絵しりとりメモ」を使ってお絵描き伝言をする場合には、1 つの端末を複数人で回しながら遊ぶ。基本的には絵しりとりと同じで、タイトル画面から下の枠をタッチし絵を描く画面に移動してお題の絵を描く。絵しりとりと違う点は、1 つのお題から様々な種類の絵が生まれるため想像の幅が広がる。人数が多い場合は絵しりとりよりもこちらの方が盛り上がることが期待される。

メモ

「絵しりとりメモ」にはゲームとしての機能の他にメモとしての機能も備わっている。

このアプリにおけるメモとは、描いた絵を保存する機能を利用する「画像メモ」ということだ。使用方法は単純で、タイトル画面から下の枠をタッチしキャンバス画面へ移動する。ここまでは絵しりとりをする要領と同じで、メモの場合には絵の代わりに文字を書き Save ボタンを押して保存する。

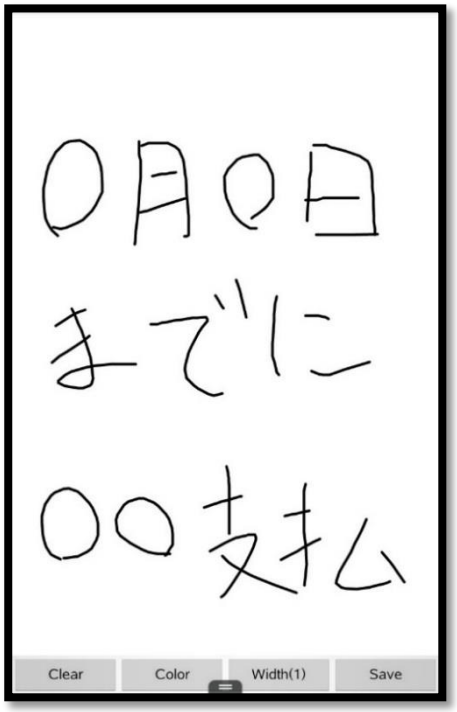


図 9

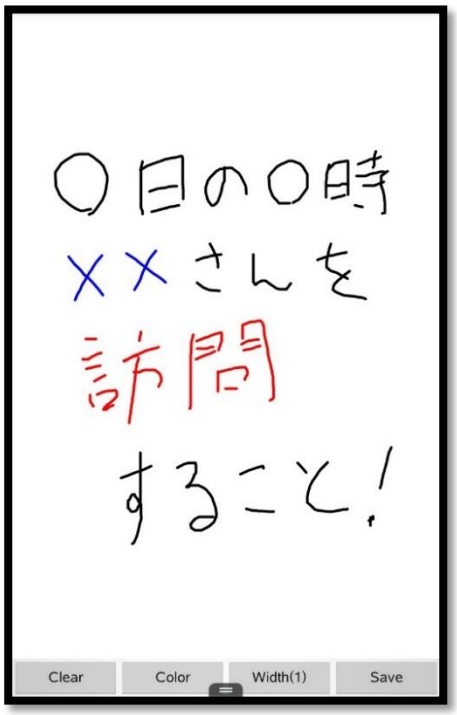


図 10

書いたメモを見返したいと思った時には、画像が保存されている「ImageSaveDir」フォルダを開くことで今までに保存された画像が一覧で表示される。保存されている画像ファイルの名前は年/日/時間となっているため、時系列順に振り返ることができる。

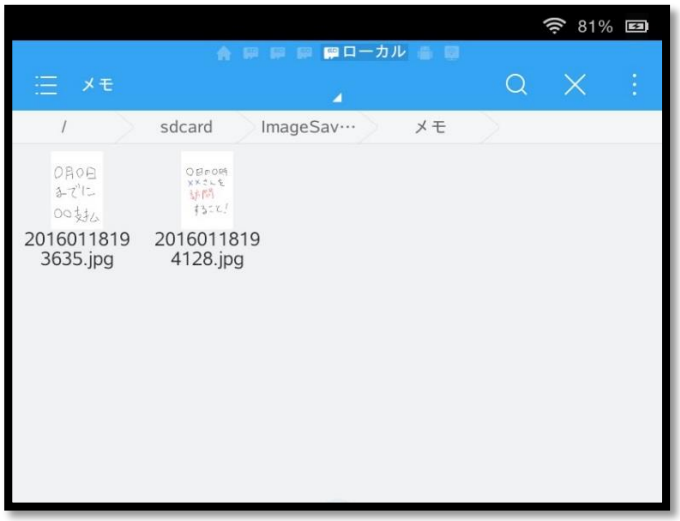


図 11

以上が、「絵しりとりメモ」におけるメモとして使用する場合の使用方法である。

第2節　プログラムの概要

ここでは、「絵しりとりメモ」のプログラムについて解説する。まず、処理の順番をフローチャートに表した。

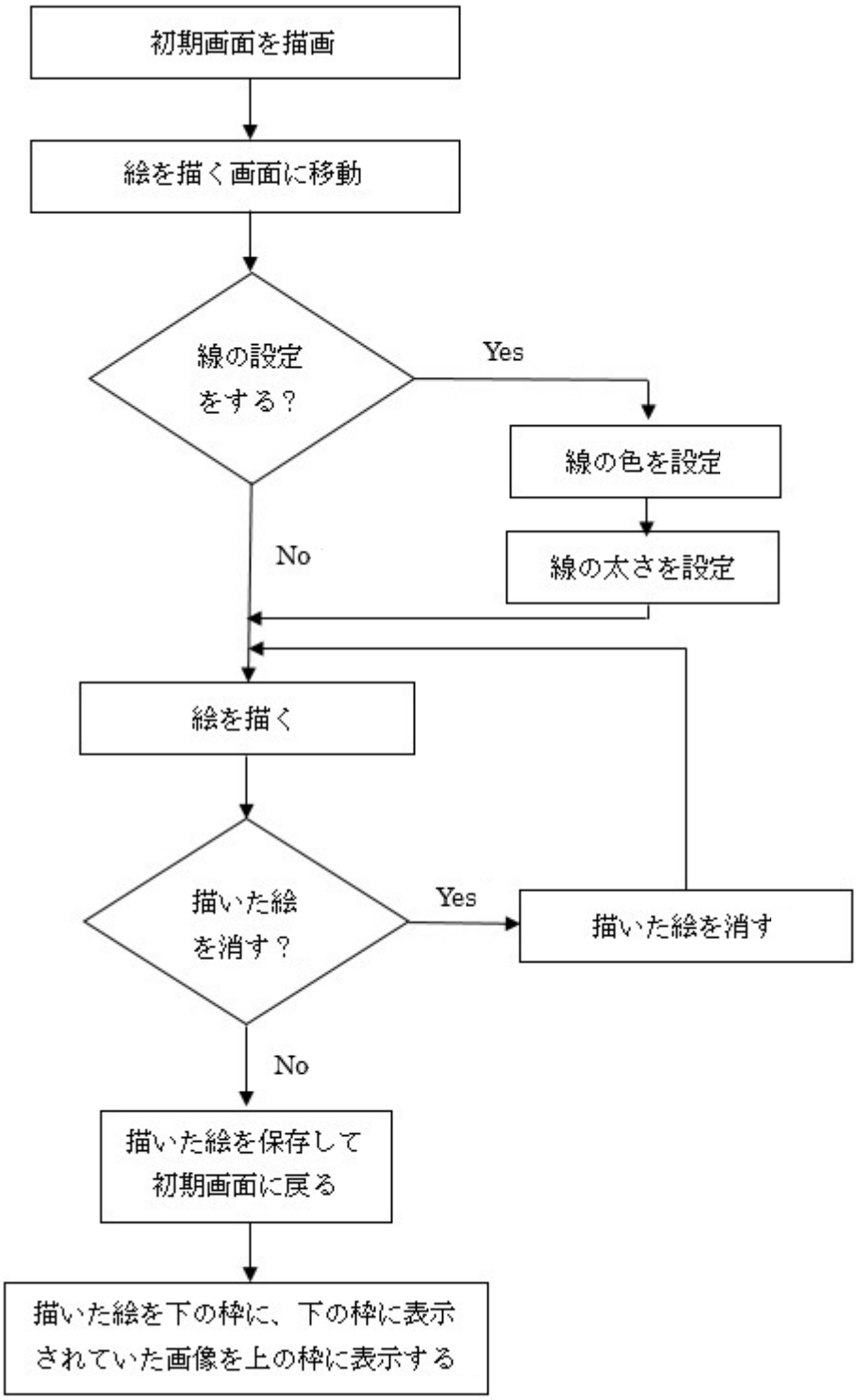


図 12

このフローチャートをもとに解説を行う。プログラム開発には統合開発環境である Eclipse を Android 開発用にカスタマイズして使用している。プログラム言語は Java である。

第1項 初期画面を描画

```
27  public class MainActivity extends Activity implements OnClickListener {
28
29      private ImageView iv1, iv2;
30      private Bitmap bmp;
31      private Uri bitmapUri;
32      private final int REQUEST_CODE_1 = 1;
33      private final int MP = ViewGroup.LayoutParams.MATCH_PARENT;
34      private final int WC = ViewGroup.LayoutParams.WRAP_CONTENT;
35      Globals globals;
36
37      /** Called when the activity is first created. */
38      @Override
39      public void onCreate(Bundle savedInstanceState) {
40          super.onCreate(savedInstanceState);
41          // setContentView(R.layout.main);
42
43          globals = (Globals) this.getApplication();
44
45          LinearLayout layout = new LinearLayout(this);
46          layout.setOrientation(LinearLayout.VERTICAL);
47          layout.setGravity(Gravity.CENTER_HORIZONTAL);
48          layout.setBackgroundColor(Color.BLACK);
49          setContentView(layout);
50
51          LinearLayout.LayoutParams param1 = new LinearLayout.LayoutParams(MP, MP);
52          param1.weight = 1.0f;
53
54          iv1 = new ImageView(this);
55          Resources r1 = getResources();
56          bmp = BitmapFactory.decodeResource(r1, R.drawable.im1);
57          iv1.setImageBitmap(bmp);
58          iv1.setLayoutParams(param1);
59          iv1.setPadding(10, 10, 10, 10);
60          layout.addView(iv1);
61
62          iv2 = new ImageView(this);
63          Resources r2 = getResources();
64          bmp = BitmapFactory.decodeResource(r2, R.drawable.im2);
65          iv2.setImageBitmap(bmp);
66          iv2.setLayoutParams(param1);
67          iv2.setPadding(10, 10, 10, 10);
68          iv2.setOnClickListener(this);
69          layout.addView(iv2);
```

図 13

この図は初期画面を描画するためのコードである。名前は MainActivity となっており、この中に描かれているコードで 1 つの画面を表す。「絵しりとりメモ」の初期画面は黒い背景の画面に 2 つの画像が表示されているものになっている。画面のレイアウトを作成するためのコードが 45 行目からの LinearLayout クラスであり、setContentView メソッド

によって縦、横、背景のオブジェクトを指定している。黒い背景を指定しているコードは 48 行目の layout.setBackgroundColor である。この中の色の部分を変更することによって背景を好きな色に変更できる。次に、設定した黒い背景画面に 2 つの画像を表示するためのコードだ。画像を表示するには、54 行目と 62 行目にある ImageView クラスを使用する。今回の場合は、上に表示される画像を iv1、下に表示される画像を iv2 としている。それぞれ im1.png と im2.png と名前を付けた画像を参照して表示する。画像の参照には BitmapFactory を使用している。画像の大きさは setPadding で指定しており、前後左右とも 10 ポイントとなっている。

第2項 絵を描く画面に移動

初期のタイトル画面から実際に絵を描くための画面に移動するには、タイトル画面に表示されている 2 つの画像のうち、下の画像をタッチしなければならない。画像をタッチした際にアクションを起こすためのコードは図 13 の 68 行目にある setOnClickListener である。今回は iv2 に this と変数を設定している。

```
73⊖    @Override
74     public void onClick(View v) {
75         // TODO 自動生成されたメソッド・スタブ
76         Intent intent = new Intent(this, Idraw.class);
77         startActivityForResult(intent, REQUEST_CODE_1);
```

図 14

このコードは、setOnClickListener で設定した対象に起こすアクションを設定しているコードである。画面の移動には Intent を使用する。Intent は複数の Activity を繋ぐための架け橋の役割を担っている。Intent(this, Idraw.class)とあるように、this と設定してある iv2 をタッチすることによって、絵を描く画面のプログラムをしている Idraw.class へと移動することができる。

第3項 線の設定

絵を描く画面に移動したらまず線の色と太さを設定する。線の色と太さを設定するには、それぞれ Color ボタンと Width(1)ボタンを使用する。

```
81          paint = new Paint();
82          paint.setColor(Color.BLACK);
83          paint.setStyle(Paint.Style.STROKE);
84          paint.setStrokeJoin(Paint.Join.ROUND);
85          paint.setStrokeCap(Paint.Cap.ROUND);
86          paint.setAntiAlias(true);
87          width = 5;
88          paint.setStrokeWidth(width);
89  // 線の初期設定をするコード
```

図 15

このコードは、線の初期設定のコードである。paint という名前をつけた変数に、82 行目で線の色は黒、83 行目で図形は線のみで描画をする、84 行目で線のつなぎ目を丸くする、85 行目で線の端を丸く、86 行目でジャギを軽減するためのアンチエイリアスを設定、87 行目～88 行目で初期の線の太さは 5 ポイントということを設定している。

```
129         case R.id.button2:
130             // paint.setColor(Color.BLACK);
131             color++;
132             color = color % 5;
133             if (color == 0) {
134                 bt2.setTextColor(Color.BLACK);
135             } else if (color == 1) {
136                 bt2.setTextColor(Color.BLUE);
137             } else if (color == 2) {
138                 bt2.setTextColor(Color.RED);
139             } else if (color == 3) {
140                 bt2.setTextColor(Color.GREEN);
141             } else if (color == 4) {
142                 bt2.setTextColor(Color.YELLOW);
143 // ボタン2を押すと色を変更するコード
```

図 16

色を変更するためのコードだが、Color ボタンをプログラム上では button2 と定義している。色の変更には case 文を使い、button2 を 1 回タッチするごとに変数 color の値 1 ずつ増えていく。color の値は 0~4 までの 5 通りあり、4 の次は 0 に戻るようになっている。また、if 文を使って color の値を判定し、color が 0 の場合は文字の色が黒に、1 の場合は青に、2 の場合は赤に、3 の場合は緑に、4 の場合は黄色になるように設定されている。

```
147         case R.id.button3:
148             // paint.setColor(Color.BLUE);
149             width++;
150             if (width == 10) {
151                 width = 5;
152             }
153             int wn = width - 4;
154             bt3.setText("Width(" + Integer.toString(wn) + ")");
155             break;
156 //ボタン3を押すと太さを変更するコード
```

図 17

線の太さを変えるコードも色と同様に、プログラム上では Width(1)ボタンを button3 と定義している。case 文により Button3 を 1 回押すごとに変数 width の値が 1 ずつ増えて

いく。初期の線の太さは 5 ポイントに設定してあるため、button3 を 5 回押して変数 width の値が 10 になったら変数 width の値に 5 を入れる。この時、button3 を押す度に実際の画面での Width(1)ボタンの()の中に表示される数字が、変数 width の値から 4 をマイナスした数字になるよう 153 行目で設定されている。つまり、Width(1)の時変数 width の値は 5 であり、Width(5)の時変数 width の値は 9 ということになる。Width(1)ボタンの()内の数字は文字として表示するため、154 行目の Integer.toString にて数字を文字に変換して表示している。

第4項 絵を描く

線の色と太さを設定したら、実際に絵を描く過程に移る。

```
232    @Override
233    public boolean onTouchEvent(MotionEvent event) {
234        // TODO 自動生成されたメソッド・スタブ
235        touch = 1;
236        paint.setStrokeWidth(width);
237        x = (int) event.getX();
238        y = (int) event.getY();
239        switch (color) {
240        case 0:
241            path = path1;
242            break;
243        case 1:
244            path = path2;
245            break;
246        case 2:
247            path = path3;
248            break;
249        case 3:
250            path = path4;
251            break;
252        case 4:
253            path = path5;
254            break;
255        default:
256            break;
257        }
```

図 18.1

```
258         switch (event.getAction()) {
259         case MotionEvent.ACTION_DOWN:
260             path.moveTo(x, y);
261             break;
262         case MotionEvent.ACTION_MOVE:
263             path.lineTo(x, y);
264             break;
265         case MotionEvent.ACTION_UP:
266             path.lineTo(x, y);
267
268             break;
269         default:
270             break;
271         }
272
273         canvas = holder.lockCanvas();
274         canvas.drawColor(Color.WHITE);
275         paint.setColor(Color.BLACK);
276         canvas.drawPath(path1, paint);
277         paint.setColor(Color.BLUE);
278         canvas.drawPath(path2, paint);
279         paint.setColor(Color.RED);
280         canvas.drawPath(path3, paint);
281         paint.setColor(Color.GREEN);
282         canvas.drawPath(path4, paint);
283         paint.setColor(Color.YELLOW);
284         canvas.drawPath(path5, paint);
285         holder.unlockCanvasAndPost(canvas);
286
287         return super.onTouchEvent(event);
```

図 18.2

236 行目の setStrokeWidth で設定した線の太さをセットする。また、239 行目の switch で設定した色の変数 color をセットする。258 行目の switch(event.getAction)によってタッチしたときに起こるイベントを設定する。259 行目の ACTION_DOWN は、指で画面を押しているということを示している。262 行目の ACTION_MOVE は、画面を押している間にスワイプして指を動かしたときのことを表している。265 行目の ACTION_UP は、画面から指を離したときのことを表している。この一連のプログラムによってタッチしてから離すまでの間、線を描画することができる。描画をするためには canvas のインスタンスが必要になっている。273 行目で canvas のインスタンスを holder.lockCanvas によってロックする。これは 1 つの canvas スレッドで描画を行っているときには他のスレッドとの競合を避けるために行う。つまり、今回の場合 5 種類ある色のうちいずれかの描画の最中には他の色で描くことができないようにするということだ。初期設定で canvas には白の色を描画するように設定している。色の判断は path によって判断され、設定した色と太さに設定される。なお描画が終わった時に、285 行目の holder.unlockCanvasAndPost

によってロックした canvas を解除する。以上がタッチして線を 1 回書くときの処理である。287 行目の return super.onTouchEvent(event)を行うことで 233 行目へと戻ってまた次の線を描くことができるようになっている。

第5項　描いた絵を消す

「絵しりとりメモ」では描いた絵を消す場合、キャンバス画面に描いた線がすべてリセットされる仕組みになっている。

```
107        case R.id.button1:
108            canvas = holder.lockCanvas();
109            canvas.drawColor(Color.WHITE);
110            path1.reset();
111            canvas.drawPath(path1, paint);
112            path2.reset();
113            canvas.drawPath(path2, paint);
114            path3.reset();
115            canvas.drawPath(path3, paint);
116            path4.reset();
117            canvas.drawPath(path4, paint);
118            path5.reset();
119            canvas.drawPath(path5, paint);
120            holder.unlockCanvasAndPost(canvas);
121            break;
```

図 19

絵を消す機能があるボタンは Clear ボタンであるが、プログラム上では button1 という名前になっている。この一連のプログラムでは、変数 canvas に入っている path と paint をリセットし、何も入力されていない状態に戻すことでこれまでに描いた線をすべて消すことができるようになっている。

第6項　描いた絵を保存して初期画面に戻る

絵が一通り完成したら、Save ボタンを押して保存する。Save ボタンはプログラム上では button4 という名前になっている。

```
151         case R.id.button4:
152             String seq = new SimpleDateFormat("yyyMMddHHmmss")
153                     .format(new Date());
154             globals.seq = seq;
155             Bitmap image = Bitmap.createBitmap(sv.getWidth(), sv.getHeight(),
156                     Config.ARGB_8888);
157             Canvas canvas_t = new Canvas(image);
158             canvas_t.drawColor(Color.WHITE);
159             paint.setColor(Color.BLACK);
160             canvas_t.drawPath(path1, paint);
161             paint.setColor(Color.BLUE);
162             canvas_t.drawPath(path2, paint);
163             paint.setColor(Color.RED);
164             canvas_t.drawPath(path3, paint);
165             paint.setColor(Color.GREEN);
166             canvas_t.drawPath(path4, paint);
167             paint.setColor(Color.YELLOW);
168             canvas_t.drawPath(path5, paint);
169             File file = new File(Environment.getExternalStorageDirectory()
170                     .getPath() + "/ImageSaveDir/");
171             if (!file.exists()) {
172                 file.mkdir();
173             }
174             String imgName = file.getAbsolutePath() + File.separator + seq
175                     + ".jpg";
176             File saveFile = new File(imgName);
177             try {
178                 FileOutputStream out = new FileOutputStream(imgName);
179                 image.compress(CompressFormat.JPEG, 100, out);
180                 out.flush();
181                 out.close();
182             } catch (Exception e) {
183             }
184             setResult(RESULT_OK);
185             finish();
186             break;
187         default:
188             break;
189 //描いた絵を画像として保存するコード
```

図 20

保存する画像を作成するのは 155 行目の createBitmap である。しかし、キャンバス画面に描いた絵を Bitmap、つまり画像として作成する前に、156 行目の Config.ARGB_8888 によって画像のサイズを 4 バイトまでの大きさに縮小して読み込むように設定する。これは画像のサイズを落とすための処理である。画像が作成されたら、169 行目の File で画像を保存するフォルダを指定する。同じく 169 行目の Enviroment.getExternalStorageDiretory は内部ストレージのパスを取得するメソッドで、今回の場合は内部ストレージ内に「ImageSaveDir」という名前のフォルダを指定している。なお、最初はフォルダが存在していないため 171 行目の file.exists でフォルダが存在しているかを確認し、存在していなかった場合 172 行目の file.mkdir によってフォルダを作成する。画像を作成して保存するフォルダを指定したら次にファイル名を決める。174 行目の file.getAbsolutePath は、ファイル名に”.”や”..”が入っても問題がないようにする

コードである。また、file.separator は、ファイル名に”/”や”¥”などの区切り文字が入っても問題がないようにするコードである。今回の場合ファイル名は変数 seq + ”.jpg”となっている。変数 seq には 152 行目で設定してある日付と時刻を取得する SimpleDateFormat で取得した日付データが入力される。今回は yMdHms の 6 種類のパターン文字を使っており、y は年、M は月、d は月における日、H は 1 日を 0~23 時と定義したときの時、m は分、s は秒を表している。例として 2016 年の 1 月 1 日の 10 時 30 分 45 秒に画像が作成された場合には 20160101103045.jpg という名前になる。名前が決まったら、157 行目で作成された画像データが入っている変数 image を 178 行目の FileOutputStream で読み込む。読み込んだ画像を 179 行目の image.compress で形式を指定する。今回は JPEG 形式で保存する。以上が描いた絵を画像として保存するためのプログラムである。Save ボタンには、画像を保存する機能に加えて、絵を描く画面を終了する機能も備わっている。184 行目の setResult(RESULT_OK)によって絵を描く画面である Idraw のアクティビティから初期画面である MainActivity のアクティビティへと移動するようになっている。

第7項 描いた絵を下の枠に、下の枠に表示されていた画像を上の枠に表示する～

```
80  @Override
81  protected void onActivityResult(int requestCode, int resultCode, Intent data) {
82
83      if (requestCode == REQUEST_CODE_1) {
84          if (resultCode == RESULT_OK) {
85              File file = new File(Environment.getExternalStorageDirectory()
86                      .getPath() + "/ImageSaveDir/");
87              File mediaFile = new File(file.getPath() + File.separator
88                      + globals.seq + ".jpg");
89              if (mediaFile.exists()) {
90                  bitmapUri = Uri.fromFile(mediaFile);
91                  BitmapFactory.Options options = new BitmapFactory.Options();
92                  options.inSampleSize = 4;
93                  bmp = BitmapFactory
94                          .decodeFile(bitmapUri.getPath(), options);
95                  Bitmap wk = ((BitmapDrawable) iv2.getDrawable())
96                          .getBitmap();
97                  iv1.setImageBitmap(wk);
98                  iv2.setImageBitmap(bmp);
99                  if (bmp != null) {
100                     // bmp.recycle();
101                 }
102             } else {
103                 Toast.makeText(this, "Draw does not exist any more.",
104                         Toast.LENGTH_LONG).show();
```

図 21

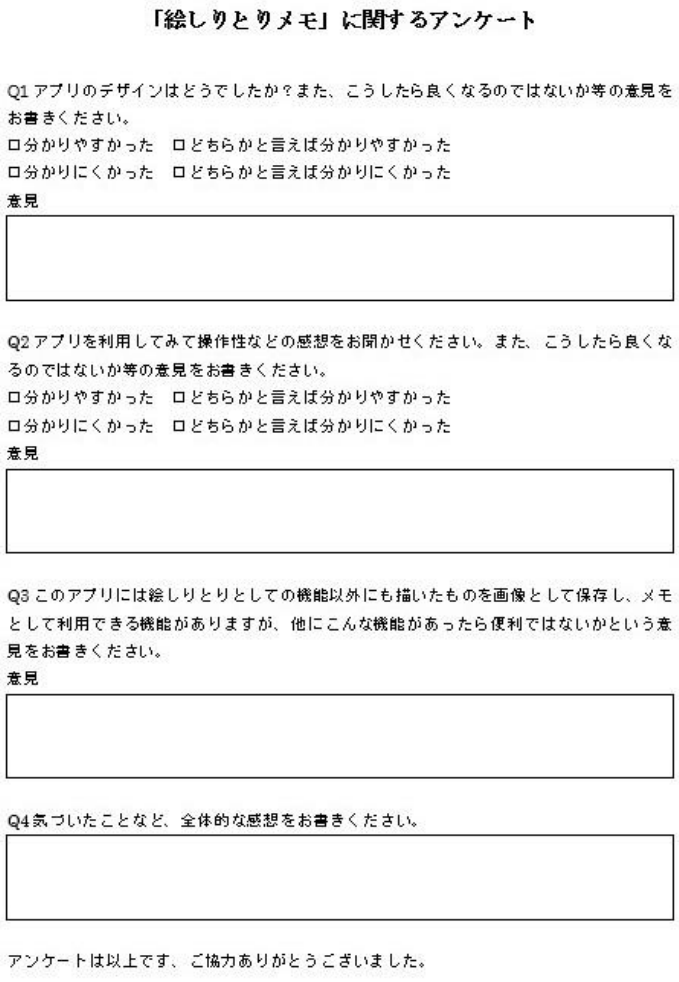
画像を保存する際に setResult(RESULT_OK)の信号を送ることによってキャンバス画面の処理を終了し、初期画面に移動した。枠の中の画像を変化させるコードは、84 行目の if 文で RESULT_OK の信号を受け取った場合に実行される。85 行目で画像が保存されたフォルダを探し、87 行目でフォルダの中のファイルを探し出して取得する。90 行目の Uri,File で取得した画像を呼び出し、91 行目から 92 行目で 4 ポイントの Bitmap を作成しておく。93 行目で変数 bmp にフォルダから取得した画像を保存する。95 行目で、変数 wk に iv2、つまり下の枠に表示されていた画像を保存する。97 行目で上の枠である iv1 に先ほど下の枠の画像が保存された変数 wk を表示する。98 行目で下の枠である iv2 にフォルダから取得された画像が入った変数 bmp を表示する。この一連の動作をすることによって初期画面において、下の枠に表示されていた画像を上の枠に、描いて保存された絵を下の枠に表示させることができる。

以上が「絵しりとりメモ」における主要プログラムの概要である。

第4章 Android アプリ「絵しりとりメモ」の評価

第1節 評価方法

アンケート用紙を作成し、複数人に回答してもらい、結果を総合的にまとめることで評価する。アンケートの内容は「デザイン」、「操作性」、「機能について」、「気づいたことや感想」の 4 つの項目についてであり、「デザイン」と「操作性」については 4 択のチェックボックスを利用し、「分かりやすかった」、「どちらかと言えば分かりやすかった」、「分かりにくかった」、「どちらかと言えば分かりにくかった」の中から 1 つを選んでチェックをしてもらう形になっている。更に 4 つの質問項目のそれぞれにテキストボックスによる記述エリアを設けており、意見を書いてもらうようになっている。実際のアンケート用紙は以下のとおりになっている。



「絵しりとりメモ」に関するアンケート

Q1 アプリのデザインはどうでしたか？また、こうしたら良くなるのではないか等の意見をお書きください。
□分かりやすかった　□どちらかと言えば分かりやすかった
□分かりにくかった　□どちらかと言えば分かりにくかった
意見

Q2 アプリを利用してみて操作性などの感想をお聞かせください。また、こうしたら良くなるのではないか等の意見をお書きください。
□分かりやすかった　□どちらかと言えば分かりやすかった
□分かりにくかった　□どちらかと言えば分かりにくかった
意見

Q3 このアプリには絵しりとりとしての機能以外にも描いたものを画像として保存し、メモとして利用できる機能がありますが、他にこんな機能があったら便利ではないかという意見をお書きください。
意見

Q4 気づいたことなど、全体的な感想をお書きください。

アンケートは以上です、ご協力ありがとうございました。

図 22

第2節　評価結果

第 4 章第 1 節で作成したアンケート用紙を実際に配布し、ゼミナールのメンバー6 人にアンケートに回答してもらい集計をとった。ここでは、4 つの質問項目についての集計のまとめを行う。

質問項目 1：Q1 アプリのデザインはどうでしたか？また、こうしたら良くなるのではないか等の意見をお書きください。

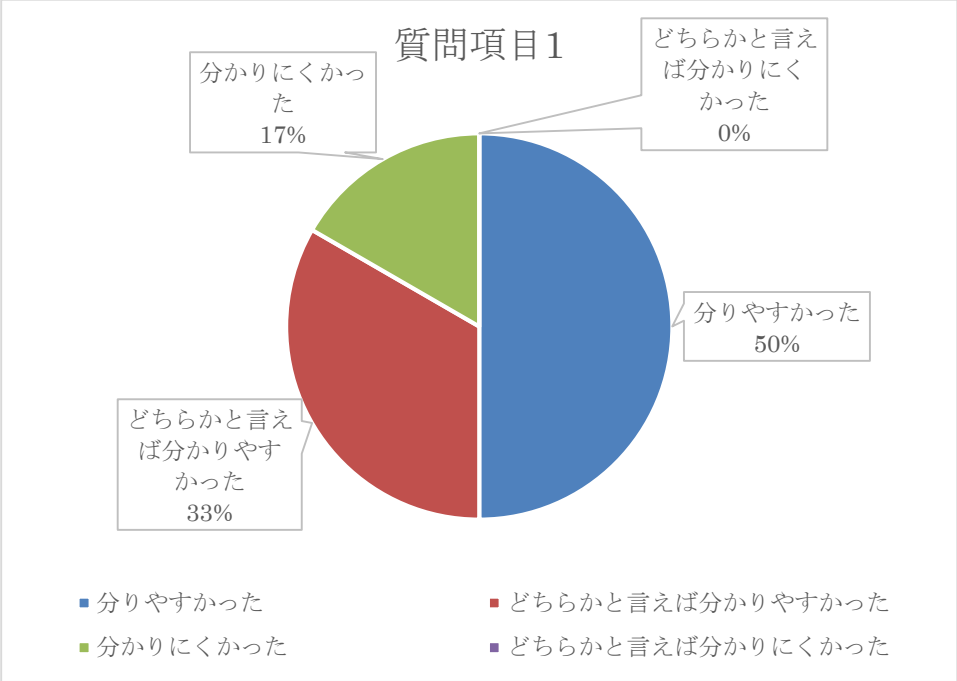


図 23

上図は、質問項目 1 に対するチェックボックスの集計結果の円グラフである。それぞれ得票数は割合の多い順に 3 票、2 票、1 票、0 票という結果であった。別途用意していたテキストボックスに書いてもらった意見は次のようになっている。「分かりやすかった」と答えた人の意見には、「見やすい」「わかりやすい」「デザインもシンプルで使いやすかった」とあり、「どちらかと言えば分かりやすかった」と答えた人の意見には、「進行するための説明がほしい」とあり、「分かりにくかった」と答えた人の意見には、「もう少し説明がほしい」とあった。

質問項目 2：Q2 アプリを利用してみて操作性などの感想をお聞かせください。また、こうしたら良くなるのではないか等の意見をお書きください。

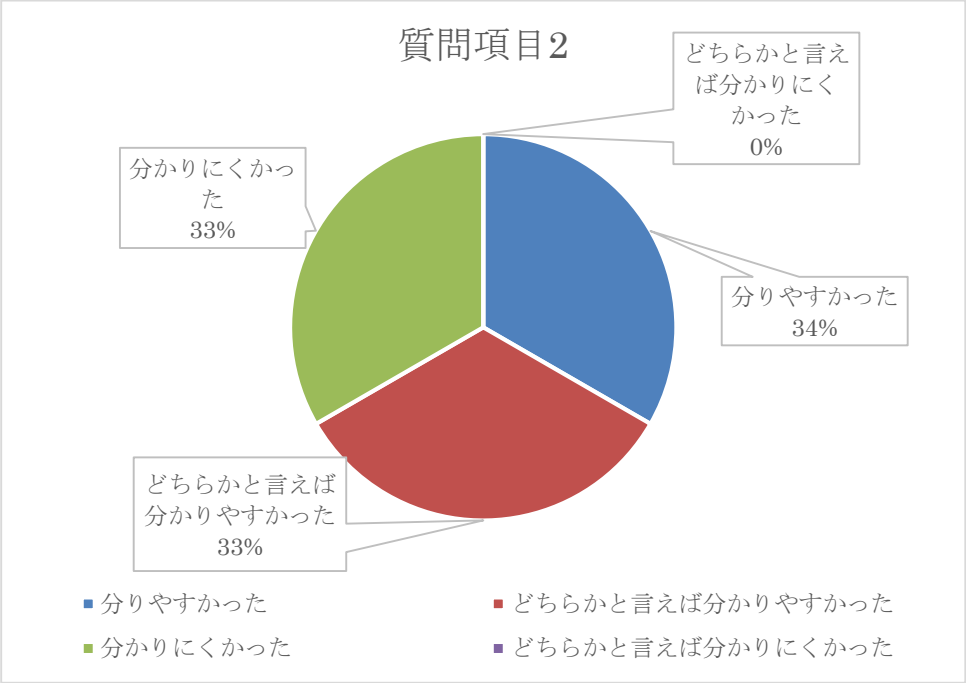


図 24

上図は、質問項目 2 に対するチェックボックスの集計結果の円グラフである。それぞれ得票数は 2 票、2 票、2 票、0 票という結果であった。別途用意していたテキストボックスに書いてもらった意見は次のようになっている。「分かりやすかった」と答えた人の意見には、「絵を描く際はペンなどを使って描いたほうがいいかも」「操作も簡単で、わかりやすかった」とあり、「どちらかと言えば分かりやすかった」と答えた人の意見には、「Save ボタンを押さないと次に進めないのは分からなかった」とあり、「分かりにくかった」と答えた人の意見には、「消しゴムがほしい」「絵を描いている時も前の画像を表示してほしい」「画像が 2 枚並んでいるとき、どちらを押せば絵を描く画面に移動するのかがわからない」とあった。

質問項目 3：Q3 このアプリには絵しりとりとしての機能以外にも描いたものを画像として保存し、メモとして利用できる機能がありますが、ほかにこんな機能があったら便利ではないかという意見をお書きください。

この質問項目に対する意見は、直接テキストボックスに書いてもらうようになっており次のような意見があった。「アプリ内から保存されたフォルダに一発で行ける機能がほしい」「タイマーを設定できればいいと思う」「消しゴムの機能をつけてほしい」「絵を描く際

に制限時間を設ける」「Google から画像を検索してくる CPU との対戦モード」とあった。

質問項目 4：気づいたことなど、全体的な感想をお書きください。

この質問項目に対する意見は、「数人でやると楽しい」「1 人よりも、複数人でやったほうが楽しい」「メモとしては必要になってくる機会があるかわからない」「塗りつぶしや円、図形のテンプレートがあると絵が描きやすいと思う」「カラーを変えるときに、ボタンの文字の色を変えるのではなく、ボタンの色を変えればいいと思う」とあった。

第3節 総評

アンケートを集計した結果、デザインに対する意見はシンプルで分かりやすいとの意見が多かったが、画面の見方や操作方法の説明がほしいとの意見もあった。操作性に対する意見は不満点が多く、Save ボタンを押さないと前の画面に戻れないのが不便という意見や、消しゴムを実装してほしいという意見が多かった。また、新機能に対する意見では時間制限を設けて絵しりとりにもっとゲーム性を出したほうが良いという意見が多かった。これらの意見をまとめ、「絵しりとりメモ」をより良いものにするために必要だと考えた機能は 5 つある。1 つ目は、使い方や機能を説明する取扱説明書の機能だ。アプリのデザインを見ただけではどうやって遊ぶのかがわからないという意見があり、実際に遊んでもらう際にも何度か質問された。快適に使用してもらうためにはまず使い方を理解してもらう必要があるということが分かった。取扱説明書の機能についての解決方法としてはヘルプボタンを作成し、それを押すことで「絵しりとりメモ」についての一通りの説明が記載された画像が表示されるようにすることで解決すると考える。2 つ目は、ボタンに関する機能だ。文字色を変えるときに、ボタンのテキストの色ではなくボタンそのものの色を変更したほうが分かりやすいのではないかという意見をもとに、ボタンごと色を変更する仕様にするとともに、現在 5 種類しかない色のバリエーションを、256 色のカラーパレットから選べるようにして見やすく且つ幅も広げることで解決すると考える。3 つ目は消しゴム機能だ。現在の仕様では Clear ボタンを押すことで、その時点で描かれている画面上の線がまとめて消去され、絵を描く前の真っ白な画面になる全消し機能となっている。ア

ンケートの結果、全てを消すのは面倒なので消したいところだけを消すことができる消しゴムの機能がほしいという意見が複数あった。これについては Clear ボタンとは別に Eraser ボタンを追加し、小・中・大の 3 種類の太さのポイントを用意し、なぞった箇所の線を消す機能を実装することで解決すると考える。4 つ目は Save ボタンを押して描いた絵が保存されているフォルダにアクセスできる機能だ。保存された画像を一覧として表示するか、保存されたフォルダに飛べるようにしてほしいという意見があった。これについては保存フォルダを開くボタンを設置し、そのボタンを押すことでそれまでに保存された画像がサムネイル表示される機能を実装することで解決すると考える。5 つ目は絵しりとりにもっとゲーム性を持たせるための機能だ。アンケートの結果、複数人でやる方が面白い、描く絵のお題や時間制限があると面白そうという意見をもらった。そこで、数種類のゲームモードを追加して用途に合わせて選択してもらう方式にすることによって解決すると考える。絵しりとりで遊ぶ際には 15 秒から 20 秒程度の時間制限のあるモードを実装することで、複数人で遊ぶとより盛り上がるゲームになるはずだ。また、絵しりとりモードの他に、ランダムで表示されるお題をもとに順番に絵を描いていき、最後の人がお題を当てるというお絵描き伝言ゲームモードも用意し、絵しりとりとは別の遊び方もできるようにすることで現在よりもゲーム性を持たせることができると考える。以上の 5 点の機能を実装することで、「絵しりとりメモ」はより良いアプリに近づき、アンケートに答えてもらった人達のニーズを満たすことができるだろう。

第5章 おわりに

第 1 章のはじめにの中で「絵しりとり」メモを制作する経緯を 3 点挙げた。ここではそれぞれの点について達成できたか、またその結果について考えることを書いていく。1 つ目は新規にアプリを制作するにあたり、オリジナル性のあるものを作りたかったという点だ。お絵描きアプリとは少し違う絵しりとりアプリを制作することによって、オリジナル性を出すことができたと考える。しかし、まだ完成品とは言えない状態であるため、唯一無二のアプリとは言えないのが事実だ。第 4 章第 3 節の総評で挙げた今後の改善点を実装することでよりオリジナル性のあるアプリになると考えている。2 つ目は 1 つのアプリを複数の用途で使えるようにしたいという点だ。これについては、絵しりとりのほかにメモという機能を実装したことで達成した。更に絵しりとりのほかにお絵描き伝言というゲームにも対応することができ、「絵しりとりメモ」は 3 つの用途で使用できるアプリとなった。3 つ目は誰にでも使えるアプリを制作するという点だ。これについては、第 4 章の評価の中でアンケートを取った結果シンプルで分かりやすいが、シンプルすぎて説明不足という評価を得た。誰にでも使えるということはただシンプルなだけではないということが分かった。以上が「絵しりとりメモ」を制作する経緯 3 点に対する結論である。知識が足りない中でのアプリ制作は苦労も多かったが、真摯に質問や相談に対応してくださった教授やゼミナールのメンバーのおかげで形にすることができた。実際に Android の端末で動かしてみた時に思い通りの動きをした時の喜びは大きな成果である。今後は、評価でもらった意見や課題を参考に、「絵しりとりメモ」をより良いアプリへと昇華させて行きたいと考える。

謝辞

最後に、本論文を執筆するにあたり、2 年次から 2 年間親身になり指導して頂いた伊藤則之教授、4 年次に本論文の指導を熱心にして下さった田中章司郎教授、いつも相談に乗って頂いたゼミナールの皆様には深く御礼申し上げます。

なお、本論文、本研究で作成したプログラム及びデータ、資料などの全ての知的財産権を本ゼミナールの指導教員である田中章司郎教授に譲渡致します。

参考文献

[1]http://www.javadrive.jp/android/button/index4.html

[2]http://techbooster.org/android/application/8346/

[3]http://techbooster.jpn.org/andriod/application/715/

[4]http://andante.in/i/%E6%8F%8F%E7%94%BB/%E3%81%8A%E7%B5%B5%E3%81%8B%E3%81%8D%E3%82%A2%E3%83%97%E3%83%AA%E3%82%92%E4%BD%9C%E3%82%8A%E3%81%9F%E3%81%84%E3%80%82%E6%8F%8F%E7%94%BB%E7%B7%A8/

[5]http://d.hatena.ne.jp/shuji_w6e/20090127/1233070723

[6]http://nobuo-create.net/sdcard-2/

[7]https://docs.oracle.com/javase/jp/6/api/java/text/SimpleDateFormat.html

[8]s083078.pdf