Android 携帯
アプリケーション製作

国際文化学部　国際文化学科

07g0504　４年 F 組　　藤田　尚美

● 目次

1、Android 概要

１、Android とは

「Android」は、アメリカ合衆国のソフトウェア会社、インターネット検索サービス大手である Google が 2007 年 11 月に発表した、携帯情報端末用ソフトウェアプラットフォームである。この Android の発表と同時に、Google は「Open Handset Alliance」（※以下 OHA と表記）を発足。OHA には Acer やモトローラ、東芝などの端末メーカー、NTT ドコモ、KDDI、ソフトバンクモバイルといった通信事業者や、インテル、NVIDIA などの半導体メーカー合わせて 34 社が加盟し、各社が連携して関連技術の開発や普及を推進している。

２、Android の特徴と利点

Android の最大の強みは、なんといっても完全にオープンなライセンスで公開されていることである。なかでも OS からオープンソースなのは非常に珍しい。iPhone とよく比較されるが、SDK や開発ツールが、こちらは有料であるのに対し、Android はすべて無料で入手・制作できる。また、使用言語が Java、開発ツールが Eclipse といった Windows、Mac などといった PC の OS に制限されないのが嬉しい。これに対して iPhone では、使用言語が Object-C といった少々マイナーな言語であり、OS が Mac でないと容易に制作できない。（Windows 上でも開発できるような開発環境がでているようだが、こちらも有料・もしくは公式サポート外である）
また、Android の特徴としてマルチタスクが挙げられる。iPhone のアプリは音楽を除けばほとんどがシングルタスクで、同時に複数のアプリを開くことができないが、Android はバックグラウンドで動作させることが可能である。
例えば、インターネット検索を楽しみながらチャットをする、といったことができるのだ。これはシングルタスクである iPhone にはできない芸当である。
また、Android にはインテントという仕様が存在する。
このインテントについて、分かりやすく説明してあるブログ記事を発見したの

で引用したい。（※以下、下線引用部）
インテントは平たく言ってしまうと、アプリごとに連携するのではなく、アプリに割り当てた「役割」を指定して連携できる仕組み。もうちょっと具体的に言うと、例えば写真を撮ってメール投稿するアプリの場合、写真を撮るところだけを開発し、あとは「メール投稿してくれるアプリに投げる」という仕組みにしておくと、好きなメール投稿アプリを選択して使えるということです。
このあたり複雑なので、よく例に挙げている「読んだ 4！」というアプリで説明。
twitter で読書記録。読んだ 4!
http://yonda4.com/
このアプリは「本のバーコードを読み取るとその本のデータを取得し、Twitter に投稿することで読書記録できる」という仕組みなのですが、このうち「読んだ 4！」のアプリがやっているのは本のデータ取得だけ。バーコード読み取りと Twitter 投稿は、ユーザーがインストールしている Android アプリを呼び出してそのまま使うのです。
開発者にとってはインテントで指定さえすれば余計なアプリを開発する必要はない。ユーザーにとってもいつも使っているアプリをそのまま使えるというメリットがあります。また、後発のアプリであってもインテントを使えば、ユーザーがいつも使っているアプリと簡単に連携することができる。アプリ間の横の広がりという点は Android を使っていて一番感じる魅力です。
（※http://bloggingfrom.tv/wp/2010/01/12/3237：カイ士伝より引用）
ただ、端末が 1 機種しかない iPhone とは違い、端末の多様性による弊害がアプリケーション開発者にとっては頭を悩ませる点である。ディスプレイの解像度（HVGA・VGA）などが機種によって違い、縦長（ポートレート）形式なのか横長（ランドスケープ）形式かで画面の仕様は大きく変わるだろう。アプリケーションの互換性という問題では、マイナス点になるかもしれない。開発する際には、最初にどの機種で動かすか、的をしぼって制作、その後他の各端末に対応させるしか方法はないのである。

３、Android の可能性

先ほど端末の多様性による開発の際の問題点を挙げたが、これが利点となるケースもある。Androido はスマートフォン用のプラットフォームとして知られているが、Android を搭載できるのは携帯電話だけではない。
Andorid 搭載カーナビ、Android 搭載電子ブックリーダーなど、続々と開発がすすめられ、Android 搭載インターネット対応テレビが既に発売されている。
今はまだその能力が最大限発揮されているとは言い難いが、今後 Android 搭載端末による、多岐にわたる連携が期待される。
また、Android 専用ゲーム機なるものも発売されていた。

http://www.redstar.co.jp/gp2xcaanoo.htm（CAANOO）
by 株式会社レッドスター：http://www.redstar.co.jp/index.htm

このように、大いなる可能性を秘めた Android の普及と発展による今後の展開が、非常に楽しみである。

2、 環境

１、必要な手順

・Eclipse のインストール

・Java 開発キットのイストール

・Android 用プラグインのインストール

★今回の開発環境

Eclipse 3.5　・　JDK 6（Windows）　・　Android SDK（Windows）

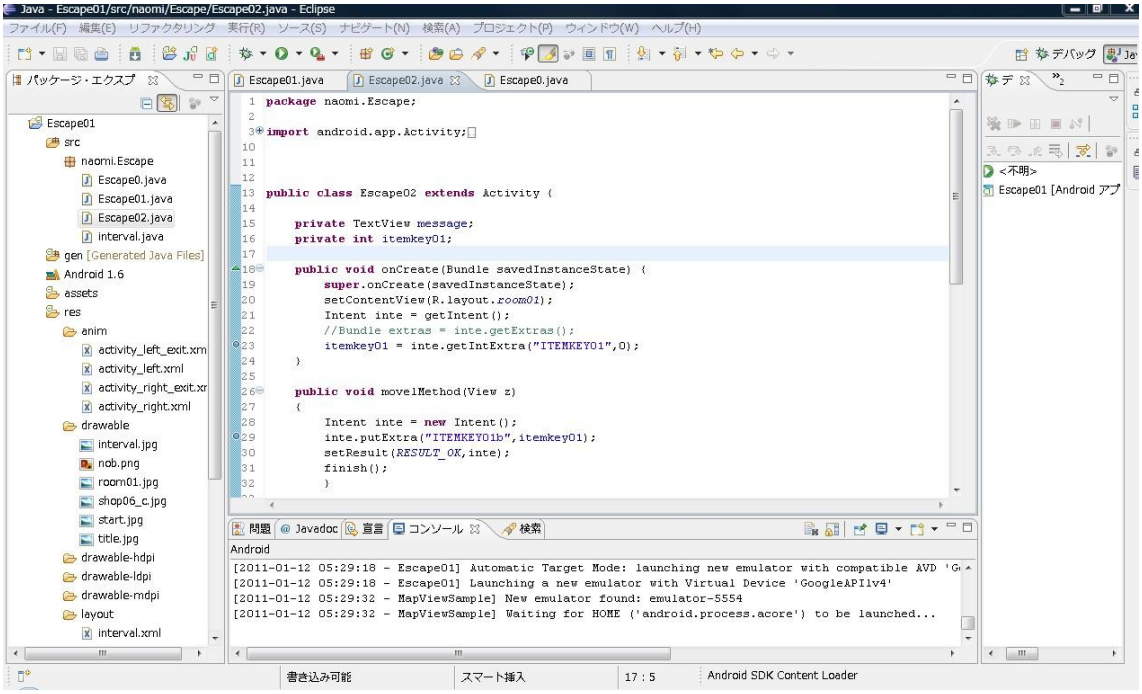
２、JAVA とは

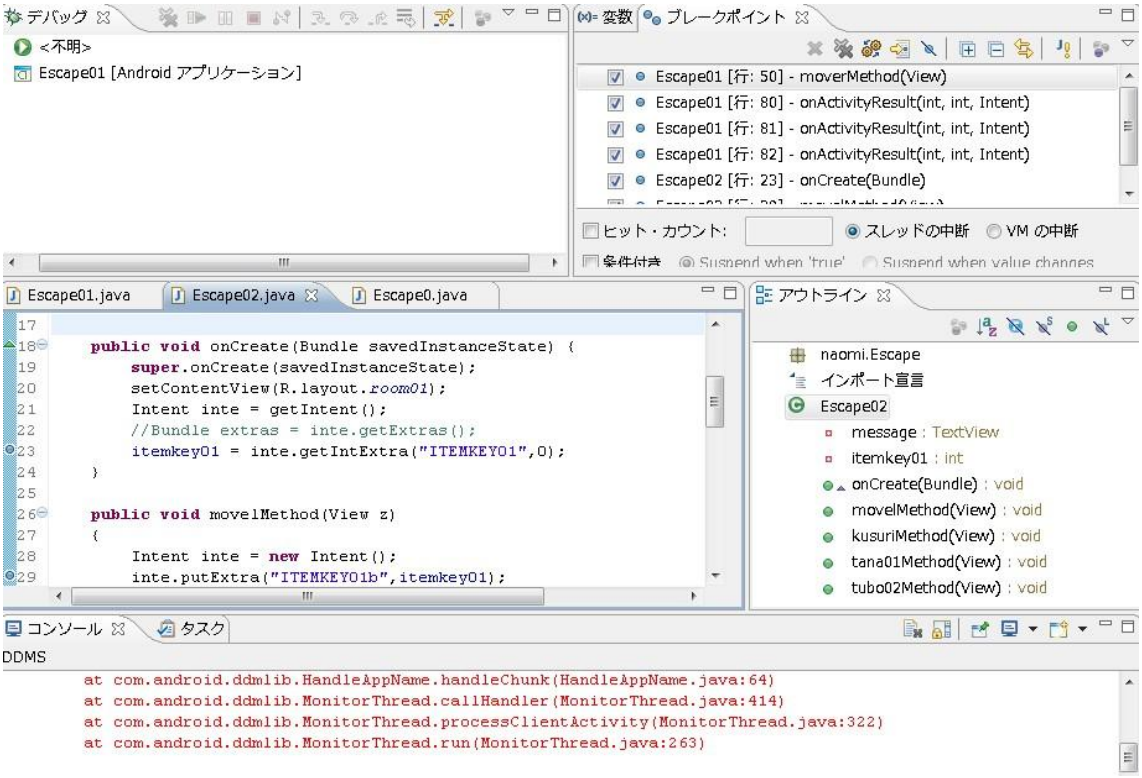
Java は、1995 年に Sun Microsystems から初めてリリースされたプログラミング言語およびコンピューティングプラットフォーム。
C 言語に似た表記法を採用しているが、既存の言語の欠点を踏まえて一から設計された言語であり、最初からオブジェクト指向性を備えている点が大きな特徴。強力なセキュリティ機構や豊富なネットワーク関連の機能が標準で用意されており、ネットワーク環境で利用されることを強く意識した仕様になっている。（※http://e-words.jp/w/Java.html：IT 用語辞典 e-Words より引用）
ちなみに JavaScript とはまったく別物である。
世界中で開発者が最も多いことを理由に、Android に採用された。

３、Eclipse とは

Eclipse は、IBM が開発・提供しているオープンソースの統合ソフトウェア開発環境(IDE)の一つ。今回はこの Eclipse を使用してアプリケーションを制作する。日本語化パッチも制作されているので、非常に使いやすい。

※アプリケーション開発中の画像





※Eclipse デバック画面　↑

3、 脱出型アドベンチャーゲーム制作

～タッチイベントとアニメーションを最大限利用した
スタイリッシュな脱出ゲームの提案～

１、構想

１）アドベンチャーパート（シナリオ読み進め）と探索パートの
インターフェイスを分ける。
２）BGM と効果音の挿入

※タイトル画面→アドベンチャーパート→探索パート→アドベンチャーパート
以下、アドベンチャーパートと探索パートの繰り返し

★アドベンチャーパート

１）画面上に表示されているボタンをタッチすることにより、シナリオを読み進める仕様にする。
２）シナリオ管理を容易にするため、文章は別の xml ファイルに書きこみ、呼び出すようにする。
３）読みやすい行間・フォントサイズに指定する。

★探索パート

１）画面をタッチして、気になるところを調べられるようにする。
探索した場所によってはキーアイテムの入手、それによって
ストーリが進んだり、分岐したりする。(フラグ・変数の管理)
２）部屋の中を移動する際は、タッチした方向にスクロールアニメーションをすることにより次の部屋の画像を表示、自然に視点変更をしてゲームに臨場感をだす。

２、実際の動作

１）タイトル画面　※「続きから」は未実装

・タイトル用 BGM を流す

・「はじめから」をタッチ、ゲームスタート。

2）アドベンチャーパート

・タイトルの BGM とは別の BGM を鳴らす

・NEXT ボタンをタッチで読み進める

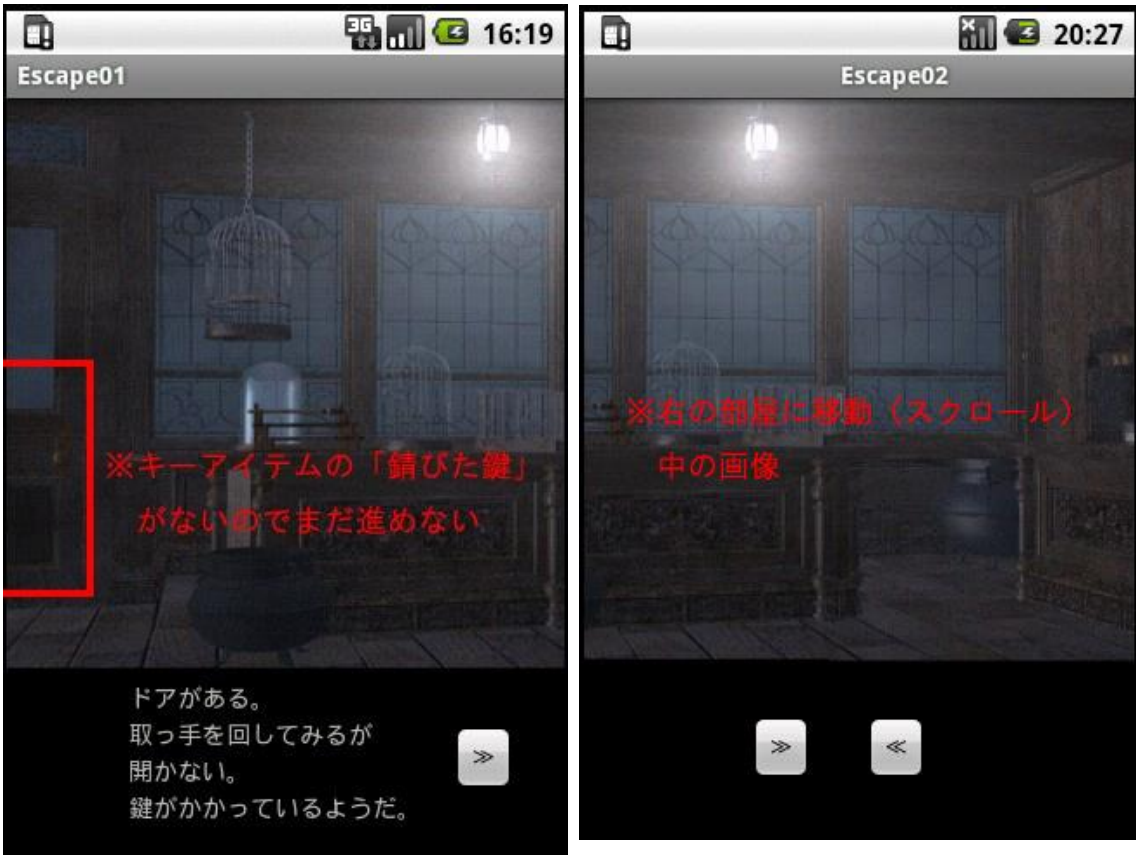
・この後、探索パートへ移行

３）探索パート

・画面をタッチして探索、下部に対応メッセージを表示

・[>>]ボタンタッチで部屋移動

※この部屋には＜鳥籠・窓・机の上・壺・ドア＞の５カ所に
探索場所を設置



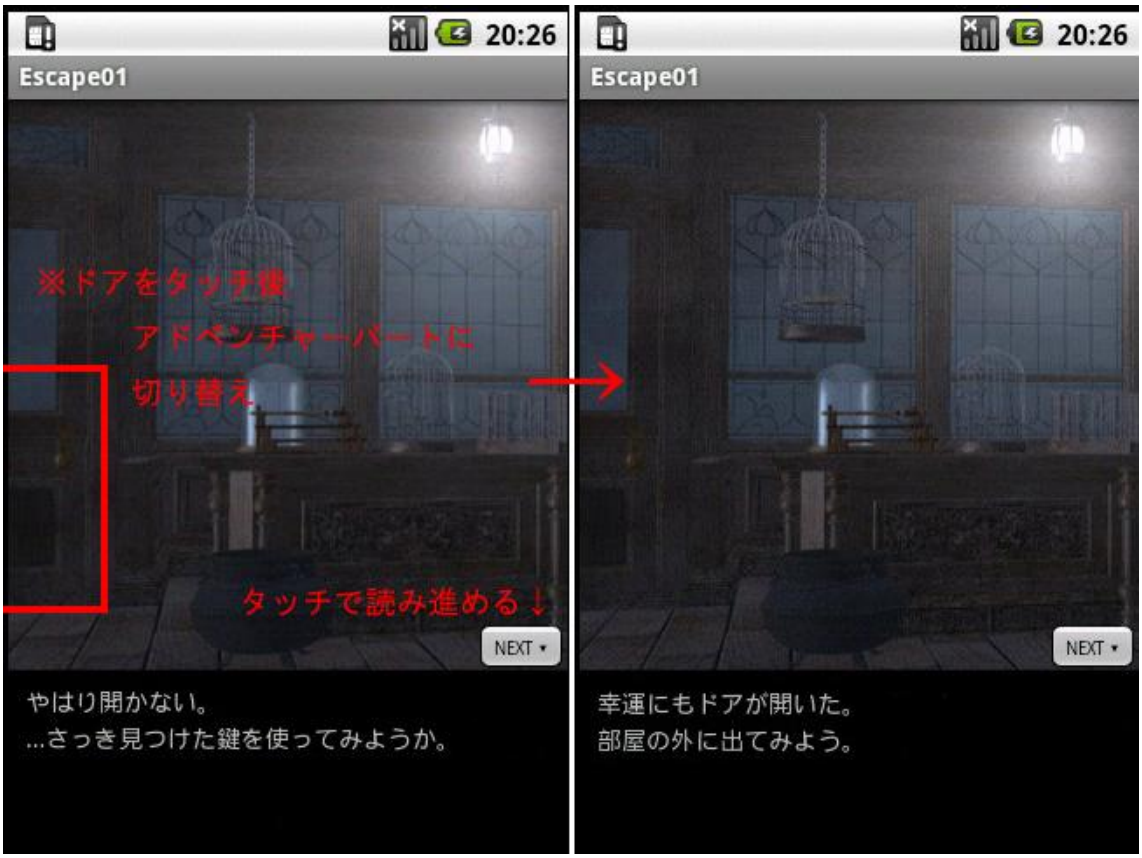
※[>>]ボタンタッチで移動中の画像

・ドアを調べてもキーアイテム未入手のため、先に進めない。
進むためには右の部屋にある＜錆びた鍵＞が必要なので移動する。

・右の部屋にて、ドアを開けるためのキーアイテムを入手。
　入手の際には効果音とトースター｛錆びた鍵を入手した｝を表示。

・もう一度調べても再度入手できないようにする。　※右図参照

※この部屋には、＜壺、棚、棚の左＞の３カ所に
　探索場所を設置



・左の部屋に戻り、開かなかったドアを調べると先ほどとは違った展開に。
この際、探索パートからアドベンチャーパートに切り替えて
文章を読み進められるようにする。

・ドア解錠の効果音を鳴らす。

・右図の後、また探索パートに以降。

4、 基礎知識として

１、はじめに

Android は、Linux のカーネル（バージョン 2.6）を採用している。
これは、ディスプレイ・カメラ・オーディオ・キーパッドなど、様々なハードウェアの機能を利用するための各種ドライバーが含まれている。
そしてその上に、一般的な機能（動画・音声の扱い・グラフィックエンジンなど）を提供するライブラリ、Google 独自の Java 仮想マシンを含むランタイムが搭載され、更にその上に Android 独自の機能を提供するフレームワークがある。

２、Android アプリケーションを構成する４種類のコンポーネント（部品）

「アクティビティ」

・UI（ユーザーインターフェース）を持った実行単位
・１つのウィンドウの中で動作するもの
・多くはユーザーのメニュー操作で個別起動する

「サービス」

・UI を持たない
・バックグラウンドで継続的に動作
（ex.音楽再生、内臓タイマー監視、センサーで状態をモニター）

「ブロードキャストレシーバ」

・そのものは UI をもたない
・受信したブロードキャスト（情報）によってアクティビティを起動など
・通知マネージャで情報を画面に表示
・システムが発信するブロードキャスト
（＝バッテリー残量・タイムゾーンの変更など）

「コンテンツプロバイダ」
・アプリケーションが管理しているデータ（コンテンツ）を
他のアプリケーションに提供

３、アクティビティのサイクル

onCreate()：初期化（初回起動）、ビュー作成・データ準備
↓
onStart()：ユーザーから見えるようになる直前に呼ばれる
↓
フォアグラウンド—onResume()：ユーザーとのやりとりをする直前
↓
onPause()：他のアクティビティが実行しようとしている場合に呼ばれる
未保存のデータ保存、重い動作（アニメーションなど）
の停止
↓
このアクティビティが前面に来る—onResume()に戻る
or
ユーザーから隠される—onStop()
or
バックグラウンド—onStop()：ユーザーから見えなくなる際呼ばれる
（破棄もしくは他のアクティビティが実行）
↓
破棄—onDestroy()：アクティビティの破棄
or
再実行—onRestart()：停止中だったアクティビティの再実行
↓
onStart()

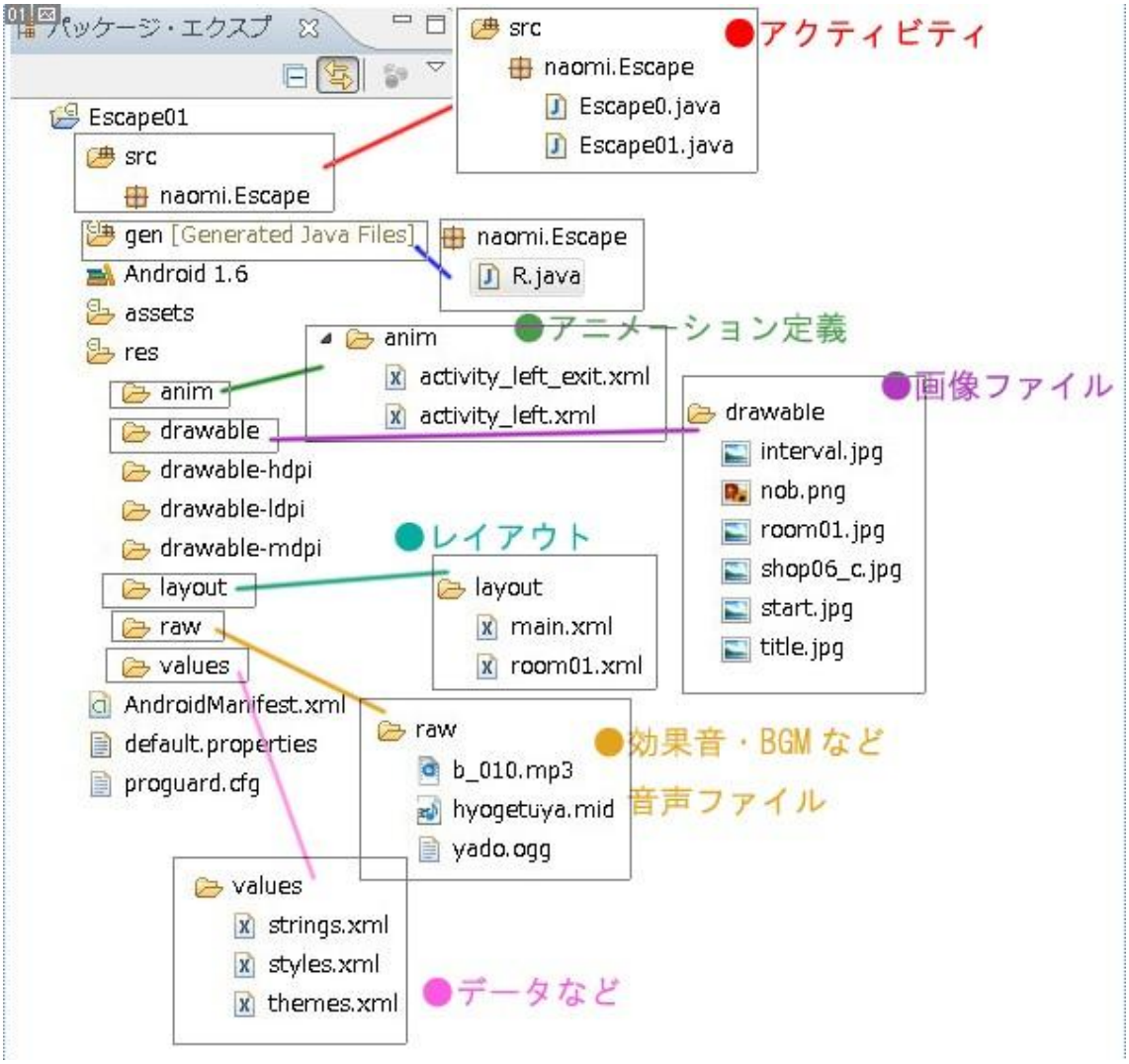
全体　：onCreate～onDestroy
アクティビティが見える間　：onStart～onStop
アクティビティが操作可能な間　：onResume～onPause

※アクティビティスタック
　アクティビティからアクティビティへ移動すると履歴が保存され「戻る」ボタンで１つ前のアクティビティに戻れる
　つまり表示させたい”元の画面”はアクティビティとして作らなければならない

タスク：一連のアクティビティのこと

5、プログラムの中身

★Android アプリケーションの構成（in Eclipse）



★Escape0.java

```
1  package naomi.Escape;
2
3  import android.app.Activity;
4  import android.content.Intent;
5
6  import android.media.MediaPlayer;
7  import android.os.Bundle;
8  import android.view.View;
9
10 public class Escape0 extends Activity
11 {
12   //MediaPlayer
13     private MediaPlayer mediaPlayer;
14
15   //起動時に呼び出し
16     public void onCreate(Bundle savedInstanceState)
17     {
18         super.onCreate(savedInstanceState);
19         //レイアウトリセット
20         setContentView(R.layout.main);
21
22         //MediaPlayer生成
23         mediaPlayer = MediaPlayer.create(this,R.raw.hyogetuya);
24         //ループ再生有効
25         mediaPlayer.setLooping(true);
26         //再生開始
27         mediaPlayer.start();
28     }
```

・import　：　使用するライブラリをインポートする
※Ctrl＋Shift＋O で使用したライブラリを自動インポート

・setContentView で main.xml を表示

・Media Player で BGM（ファイル名：hyogetuya）を再生

```
30
31         //「初めから」が押された時
32         public void moverMethod(View z)
33         {
34             mediaPlayer.stop();
35             Intent inte = new Intent(getApplicationContext(),Escape01.class);
36             startActivity(inte);
37         }
```

・ボタン　[はじめから]　を押すと、BGM を止め、Escape01 へ移動する

```
39          //一時停止からの再開;
40          protected void onResume()
41          {
42              super.onResume();
43
44              mediaPlayer.start();    //サウンド再生
45          }
46
47          //停止状態からの再開
48          protected void onRestart()
49          {
50              super.onRestart();
51
52              mediaPlayer.start();
53          }
54
55          //一時停止時にサウンド停止
56          protected void onPause()
57          {
58              super.onPause();
59              mediaPlayer.pause();
60          }
61          //停止時にサウンド停止
62          protected void onStop()
63          {
64              super.onStop();
65              mediaPlayer.pause();
66          }
```

・アクティビティが停止・一時停止した際、BGM を止める
これを記述しないと、例えば実行中に電話がかかってきた際にも BGM を再生し続けてしまうということになりかねない
※これは停止・一時停止しても、アクティビティの状態が保持されるという Android アクティビティの特徴である
また、アクティビティ再開の際には再度鳴らすように、忘れずに記述する

```
68          //アクティビティ終了時
69          protected void onDestroy()
70          {
71              super.onDestroy();
72              mediaPlayer.stop();
73              mediaPlayer.release();  //メディプレイヤーを解放
74
75          }
```

・アクティビティが完全に終了した際には、Media Player を解放する

★Escape01.java

```
12 public class Escape01 extends Activity
13 {
14     //MediaPlayer
15     private MediaPlayer mediaPlayer;
16
17     SoundPool soundPool;
18     int[] sounds = new int[1];
19
20     private TextView message;
21     private int i;
22     private int ss = 0;
23     private int itemkey01;
24
25     private int dd = 0;
26     private static final int SUBACTIVITY = 1; //定数
27
28     @Override
29     //onCreateメソッド＝アクティビティが最初に作成された時などに呼び出されるメソッド
30     public void onCreate(Bundle savedInstanceState) {
31         super.onCreate(savedInstanceState);
32         setContentView(R.layout.start);
33         mediaPlayer = MediaPlayer.create(this,R.raw.yume);
34         mediaPlayer.setLooping(true);
35         mediaPlayer.start();
36     }
39     //文章(シナリオ)表示
40     public void nextMethod(View x)
41     {
42             if (ss ==0){
43                 i = naomi.Escape.R.string.zyosyo0;
44                 ss =+1;
45                 }else{
46                     i +=1;
47                     }
48             message = (TextView)this.findViewById(R.id.TextView01);
49             message.setText(this.getString(i));
50
51
52                 if (i==2131099652){
53                     setContentView(R.layout.tansaku);
54                     }
55     }
```

・[NEXT▼]　ボタンを押したら、文章を下部に表示する
最初は zyosyo0 を strings.xml から読み込み、表示
そして変数 i に zyosyo0 の id を代入
その後は i に 1 を足した id 番地の string（文章）が読み込まれ、表示される
id が指定のものになったら、tansaku.xml を表示、探索パートへ移行

```
57      //錆びた鍵を持ってる際の処理
58      public void next02Method(View x){
59          dd += 1;
60          if (dd == 1){
61              soundPool.play(sounds[0],1.0F, 1.0F, 0, 0, 1.0F);
62              message = (TextView)this.findViewById(R.id.TextView01);
63              message.setText(this.getString(R.string.door02b2));
64              itemkey01 = 3;
65              }
66          if (dd ==2){
67              setContentView(R.layout.tansaku);
68          }
```

・キーアイテム「錆びた鍵」を所持していて、ドアを調べた時の処理

start02.xml から呼び出される

効果音を鳴らし、メッセージを表示

その後 tansaku.xml へ画面を変え、再び探索パートへ移行

```
71      //右の部屋へ移動、アイテムの取得情報を渡す
72      public void moverMethod(View z)
73      {
74          Intent inte = new Intent(this,Escape02.class);
75          inte.putExtra("ITEMKEY01",itemkey01);
76          //第二引数はリクエストコード(どのアクティビティから戻ってきたか)
77          startActivityForResult(inte, SUBACTIVITY);
78      }
79
```

・[>>]　ボタンを押した時の処理

アクティビティ Escape02 へ移動、その際にキーアイテムを持っているか否かのデータを受け渡したいので、変数 itemkey01 の値を“ITEMKEY01”というキーワードに置き換え、Escape02 へ渡す

```
80      //以下探索画面
81      public void torikagoMethod(View z)
82      {
83          message = (TextView)this.findViewById(R.id.TextView02);
84          message.setText(this.getString(R.string.torikago));
85      }
86      public void mado01Method(View z)
87      {
88          message = (TextView)this.findViewById(R.id.TextView02);
89          message.setText(this.getString(R.string.mado01));
90      }
91      public void door01Method(View z)
92      {
93          if(itemkey01==0)
94          {
95              message = (TextView)this.findViewById(R.id.TextView02);
96              message.setText(this.getString(R.string.door01));
97          }
98          if(itemkey01==1)
99          {
00              setContentView(R.layout.start02);
01
02          }
03          if(itemkey01==3)
04          {
05              message = (TextView)this.findViewById(R.id.TextView02);
06              message.setText(this.getString(R.string.door01c));
07          }
08
09      }
```

・探索パート

各位置に透明で設置されているボタンをタッチすると

その場所に対応した文章が下部に表示される

Door01Method は、キーアイテム「錆びた鍵」入手前、入手後に初めて調べた時（ここで解錠）、解錠後の処理を変数 itemkey01 の値によって分岐させている

入手後に初めて調べた時は、前述した start01.xml でアドベンチャーパートへ移行、next02Method が呼び出され、読み進めた後は再び探索パートへ移行する

```
//他のアクティビティから戻ってきた際に実行(onActivityResultメソッド)
protected void onActivityResult(int requestCode, int resultCode, Intent inte) {
    super.onActivityResult(requestCode, resultCode, inte);
    //効果音読み込み
    soundPool = new SoundPool(1, AudioManager.STREAM_MUSIC, 0);
    sounds[0] = soundPool.load(this, R.raw.b_005, 1);
    //メッセージリセット
    message = (TextView)this.findViewById(R.id.TextView02);
    message.setText(this.getString(R.string.none));
    //アイテムの取得情報を読み込む
    if (requestCode == SUBACTIVITY){
        if (resultCode == RESULT_OK){
            itemkey01 = inte.getIntExtra("ITEMKEY01b",0);
        }
    }
}
```

・Escape02 が終了し、元の Escape01 に戻ってきた際の処理

ドアを解錠する際に鳴らす効果音は、ここで前もって読み込んでおく

※Sound Pool 用意、鳴らすファイルの指定と読み込み　→　鳴らす

という風に、鳴らす直前に記述すると、ファイルを読み込むのに少し時間がかかるため、ファイルが読み込まれずに鳴らすことになり、結果的には鳴らなくなる

・戻ってきた際、前に表示していた文章は消しておく

・Escape02 での、キーアイテム取得情報”ITEMKEY01b”を受け取る。

★Escape02.java

```
15 public class Escape02 extends Activity {
16
17     private TextView message;
18     private int itemkey01;
19
20     SoundPool soundPool;
21     int[] sounds = new int[1];
22
23     public void onCreate(Bundle savedInstanceState) {
24         super.onCreate(savedInstanceState);
25         setContentView(R.layout.room01);
26         //アイテムの取得情報を読み込む
27         Intent inte = getIntent();
28         itemkey01 = inte.getIntExtra("ITEMKEY01",0);
29         //効果音読み込み
30         soundPool = new SoundPool(1, AudioManager.STREAM_MUSIC, 0);
31         sounds[0] = soundPool.load(this, R.raw.b_003, 1);
32     }
```

・Escape01 で指定したキーワード”ITEMKEY01”の値を itemkey01 という変数に読み込む

・ここで鳴らしたい効果音を事前に登録

```
34     //←の部屋へ移動、アイテムの取得情報を渡す
35     public void movelMethod(View z)
36     {
37         soundPool.release();
38         Intent inte = new Intent();
39         inte.putExtra("ITEMKEY01b",itemkey01);
40         setResult(RESULT_OK,inte);
41         finish();
42         }
```

・[<<]　ボタンを押して、前の部屋に戻る際の処理
キーアイテム取得情報を”ITEMKEY01b”に代入後に
アクティビティを終了させている

```
55      public void tubo02Method(View z)
56      {
57          //アイテムの取得情報により、内容を変化させる
58          if(itemkey01==0){
59              message = (TextView)this.findViewById(R.id.TextView02);
60              message.setText(this.getString(R.string.tubo02));
61              itemkey01 = 1;
62              soundPool.play(sounds[0],1.0F, 1.0F, 0, 0, 1.0F);
63
64              /**********************引数について
65               * loadしたときに返るいんと
66               * , float左音量
67               * , float右音量
68               *  優先度？
69               * , いんとループ再生するか。0でしない。-1で無限にする。整数でその回数追加再生する
70               *  , floatレート（速度？） 範囲は0.5-2.0。
71               * **********************/
72
73              Toast myToast = Toast.makeText(this,"錆びた鍵を入手した",Toast.LENGTH_LONG);
74              myToast.setGravity(Gravity.CENTER, 0, 0);
75              myToast.show();
76          }else{
77              message = (TextView)this.findViewById(R.id.TextView02);
78              message.setText(this.getString(R.string.tubo03));
79
80          }
81
82      }
```

・右の部屋の探索パートの内、キーアイテム入手にかかわる壺を調べた時の処理のみ抜粋
もし未入手（itemkey の値が 0）だった場合、効果音を鳴らして
トースト（※）でキーアイテムの
入手を告知、変数 itemkey には 1 を代入して、再度同じ場所（壺）を調べた際には別の文章を表示し、再度アイテムの入手をしないようにしている

※トースト　：　短いメッセージを一時的に表示するもの
一定時間表示した後自動的に画面から消える
表示時間は LONG と SHORT の２種類あるが
秒数は指定不可能
ここでは LONG（およそ４秒）を指定

★tansaku.xml（rayout ファイルの中の xml の中で抜粋）

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
3    xmlns:android="http://schemas.android.com/apk/res/android"
4    android:orientation="vertical"
5    android:layout_width="fill_parent"
6    android:layout_height="fill_parent"
7    android:background="@drawable/start">
8    <LinearLayout android:id="@+id/LinearLayout01"
9                  android:layout_width="wrap_content"
10                 android:layout_height="wrap_content"
11                 android:layout_marginBottom="0dp">
12     <Button android:text=""
13         android:id="@+id/ob01"
14         android:layout_width="35dp"
15         android:layout_height="65dp"
16         android:layout_marginTop="50dp"
17         android:layout_marginLeft="120dp"
18         android:onClick="torikagoMethod"
19         android:background="@drawable/nob"
20         android:textSize="17sp">
21         </Button>
```

・ボタンやテキストエリアなどを表示するレイアウトについて指定している
　表示する位置、表示する要素の大きさなどを細かく指定できる
　また、ボタンはクリックした際に呼び出すメソッドも指定している

・ボタンを透明にするために、透明な画像（nob.png）をボタンの背景に指定

・fill_parent　：その部品がおける領域いっぱいに広げて配置
・wrap_content　：中身がちょうどはいる大きさに調整

※orientation　：　vertical＝下へ要素が追加される）
　　horizontal＝横へ要素が追加される

・ここでは最初に vertical を指定しているが
　再度<LinearLayout>　</LinearLayout >で囲ったものは horizontal 要素に
　なり横に並べて表示されていく

★strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:android="http://schemas.android.com/apk/res/android">
    <string name="zyosyo0">頭が痛い。\n夜の寒気が身を包み、肌を容赦なく冷やしていた。\n…ここは、どこだろう。
                           </string>
    <string name="zyosyo1">薄ぼんやりとしか覚醒していない頭は、未だ夢と現を彷徨っているようだ。
                           </string>
```

・ここに文章を記述していき、各アクティビティから呼び出す
　/n　は改行を指定

```
    <string name="app_name">ActivityAnimation</string>
    <string name="app_name0">Escape0</string>
    <string name="app_name1">Escape01</string>
    <string name="app_name2">Escape02</string>
    <string name="app_name3">interval</string>
    <string name="activity_animation_duration">2000</string>
</resources>
```

・その他アプリケーション名や、アニメーションの速度の値など
　様々なデータをここに記述し、他から読み込む

★styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="Animation" parent="android:Animation" />
    <style name="Animation.Activity" parent="android:Animation.Activity">
        <item name="android:activityOpenEnterAnimation">@anim/activity_right</item>
        <item name="android:activityOpenExitAnimation">@anim/activity_right_exit</item>
        <item name="android:activityCloseEnterAnimation">@anim/activity_left</item>
        <item name="android:activityCloseExitAnimation">@anim/activity_left_exit</item>
    </style>
</resources>
```

・アクティビティを開いたとき、閉じるときのアニメーションファイルを指定

★themes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="MyTheme" parent="android:Theme">
        <item name="android:windowAnimationStyle">@style/Animation.Activity</item>
    </style>
</resources>
```

・呼び出し元と呼び出し先のアクティビティにアニメーションを設定

★AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="naomi.Escape"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
            <activity android:name=".Escape0"
                      android:label="@string/app_name0"
                      android:theme="@style/MyTheme">
                              <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
                    </activity>
        <activity android:name=".Escape01"
                  android:label="@string/app_name1"
                  android:theme="@style/MyTheme">
            </activity>
        <activity android:name=".Escape02"
                  android:label="@string/app_name2"
                  android:theme="@style/MyTheme">
        </activity>
                <activity android:name=".interval"
                  android:label="@string/app_name3"
                  android:theme="@style/MyTheme">
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="4" />

</manifest>
```

・パッケージネーム、SDK バージョンなどを指定
新しくアクティビティを追加する場合は、ここに記述することを
忘れてはならない

★R.java

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.

package naomi.Escape;

public final class R {
    public static final class anim {
        public static final int activity_left=0x7f040000;
        public static final int activity_left_exit=0x7f040001;
        public static final int activity_right=0x7f040002;
        public static final int activity_right_exit=0x7f040003;
    }
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
        public static final int interval=0x7f020001;
        public static final int nob=0x7f020002;
        public static final int room01=0x7f020003;
        public static final int shop06_c=0x7f020004;
        public static final int start=0x7f020005;
        public static final int title=0x7f020006;
```

・リソース ID の管理をしている
R.java ファイルは初めてプロジェクトをビルドした時に作成され
その後ビルドが行われるたびに自動的に更新されていくので
手を入れる必要はない

6、 課題・今後の改良点

１）画面遷移について

今回、タイトルからゲーム開始画面への遷移、部屋の移動などの画面転換は全て別のアクティビティへの移動で制作したが、これは画面転換の際にスクロールアニメーションを使用したいという理由によるものだった。しかし、これでは今後制作を進めていくに連れてアクティビティの数が多くなってしまい、このままの仕様で進めることは困難になる。そこで、SurfaceView というクラスを使って、あらかじめスクロール表示したい画像同士をくっつけた大きな画像を読み込み表示し、それをスクロールするようにすることで代替可能になることがわかった。これにより、部屋を移動するたびに新規のアクティビティを制作・破壊を繰り返さずに、ひとつの java ファイルにまとめることが可能になり、キーアイテム入手の有無など、フラグについての変数操作も容易になる。(今回はアクティビティ移動の都度、必要な値をすべて受け渡ししていた)

２）センサーを使ったアクション要素の取り込み

例えば、箱を壊して中身のアイテムを取り出すために、プレイヤーに端末を振ってもらい、それを加速度センサーで感知して結果を返す、などの Android ならではのアクション要素を取り入れたい。
制作初期から構想に練り込まれてはいたのだが、筆者が Android 搭載端末を所持していないため、今回は見送った。(加速度センサーや傾きセンサーについてはエミュレーターで動作しない)

３）セーブ・ロード機能の実装

探索パート時のみ、いつでもセーブ可能。ロードはタイトル画面から、という仕様を考えていたが、今回は実装に至らなかった。
可能ならばセーブは複数スロットを用意したい。

４）文字表示の仕様

文章が一気に表示されてしまうと味気ないので、一文字ずつなめらかに表示することを試みたのだが、実力不足により実装に至らなかった。
Canvas クラスの drawPosText メソッドで一文字ずつ座標を指定・表示する方法で可能かもしれないが、熟考の余地がある。
可能ならば、描画スピードも変更できるようにしたい。

５）「読み進める」ボタン（ [NEXT▼] ボタン）の改良

次の文章を表示するために押してもらう[NEXT▼]ボタンだが、指を置くと文章が隠れてしまう位置にあるので、変更したい。
これは実際にはマウスクリックで操作していたエミュレーター上でのデバック作業では気づきにくかった点であった。
代替案としては、[NEXT▼]ボタンをメッセージ表示スペースの上に被せるように透明なボタンとして設置、メッセージ欄（どこでも OK）をタッチして読み進めてもらう様式を考えている。

６）所持キーアイテム一覧欄の導入

入手したキーアイテムを閲覧できるようにしたい。
試作版ではキーアイテム「錆びた鍵」を所持していた場合、自動で使用されたが、ここから「(アイテムを）調べる」「使用する」などのコマンドを用意してどのアイテムを使うのかプレイヤーに考えてもらう仕様にするのも良いのではないかと思った。
また、これは新規アクティビティで制作する方が利便性が高まるので、実装の際にはそのようにする。

７）BGM のフェードイン・フェードアウト

BGM 切り替えの際、不自然にならないようにフェードを取り入れたい。
しかし、今のところどうやって実装すればよいのか不明である。
更に、アニメーションと Media Player の操作を同期させようとすると、エラーがでたので、その点も一考せざるを得ない。

7、感想 ・ 謝辞

Android は発表からすでに３年経っていますが、日本では特に発展初期段階のプラットフォームなのだな、という印象を受けました。バージョンも早いペースで更新され、次々と技術が新しくなっていくので、対応するのが本当に大変です。サンプルコードなどがまだ広く出回っておらず、リファレンスも全て英語で、解析・理解にとても苦労しました。

それでも、日本語に翻訳してまとめてくださった数々の Android アプリケーション制作支援サイト運営の協力者様方や、拙いゲームに彩りを添えて下さった数々の素材サイト運営者様のお力添えより、未熟ながらもなんとか形にすることができました。ここで改めて感謝の意を示したいと思います。

JAVA は授業で基本を触った程度で、特に何か新しいアプリケーションなどを制作したことはなく、更にほとんど忘れているような状態だったので本当にできるのかと不安でしたが、ゼミで培った今までの経験全てが役に立ち、少しは成長したのだな、と感慨深く思っています。

今までご指導くださった山本教授・重定教授には深く感謝申し上げます。

また、今まで苦楽を共にしましたゼミメンバー、大熊氏・畑部氏にも敬意と親愛の念を込めてここに記します。

２年間、本当にありがとうございました。

8、 使用素材

★BGM

時の迷い人

http://timelessberry.com/

★効果音

01SoundEarth

http://www.01earth.net/sound/index.html

★背景画像

誰そ彼亭

http://may.force.mepage.jp/

9、 参考文献

Android Wiki
http://wikiwiki.jp/android/

逆引き Android 入門
http://www.adakoda.com/android/

Java Drive
http://www.javadrive.jp/

KAB-studio
http://www.kab-studio.biz/

Techfirm(テックファーム)
http://www.techfirm.co.jp/

初歩からわかる Android 最新プログラミング
安生 真 (著, 監修), 柴田 文彦 (著), 藤枝 崇史 (著)