Processing によるアンドロイドのプログラミング

―携帯アプリの作り方―

Processing for Android

―How to develop mobile phone applications ―

ネットワーク情報学部　石原秀男

School of Network and Information　Hideo Ishihara

Keywords: processing, android, mobile phone, programming

Abstract

Processing is an open source programming language and one of the most convenient languages to create drawings, animations, and interactive applications. Recently Processing has been revised to let programmers develop applications for Android. Ben Fry, one of the inventors of Processing, says “Processing for Android makes it foolishly easy to create Android apps”. This is easy instructions to how to install Processing on your PC and how to develop Android applications with it.

１．はじめに

プログラムと言えばコンピュータの画面の中で動くものというイメージがあるかも知れないが，自分で書いたプログラムが現実世界の「モノ」の中で動くのは楽しいものだ。ユビキタス応用演習の授業だと,これに続けて「マイコンボードで回路を組んで…」という話になるのだが,「そこまでは…」という人も，スマートフォンならどうだろう。タッチパネル，GPS，加速度センサー，カメラなどの面白そうな機能が付いていて，当然のことながらネットワークにも接続できる。OS があってプログラムが動くという仕組みはコンピュータそのものなので,プログラムさえ書けばそれらを自由に操れるのだ。

というわけで，スマートフォンのプログラミングは魅力的な題材だと思うのだが，いざ始めようとすると結構大変だ。アンドロイド[1]の場合なら，まともなプログラムを作ろうとすると，普通はJava JDK[2]，Eclipse[3]，Android SDK，EclipseプラグインのADT（Android Development Tools）の四つを組み合わせて使うことになる。開発言語であるJavaに加えて，アンドロイド特有のアプリケーションフレームワークについても学ばなければならない。脅すわけではないが，一年次のプログラミング入門程度では厳しくて，ソフトウェア開発基礎演習くらいは履修していないと歯が立たないだろう。

それでは，iPhoneならどうかと言うと，こちらもユーザーにはフレンドリーだが，開発者にとってはそうでもない。EclipseがXCode[4]に，Android SDKがiPhone SDKに替わるだけで，やるべきことは何も違わない。むしろ開発言語が，Objective C[5]になるぶんだけ厄介かも知れない。

要するに,これらの開発環境はかなり本格的なもので,使いこなすためにはそれなりの準備が必要なのだ。しかし楽しみで作る程度のプログラムなら，これほど大げさな道具を持ち出すこともない。ここでは，大聖堂[1]は建てられないかも知れないが，数時間で掘立小屋くらいは作れる方法[2]を紹介しよう。わかりやすく解説するつもりなので，プログラミング入門の内容が身に付いていれば，インストールも含めて半日くらいで簡単なアプリケーションを作れるようになるはずだ。アンドロイドを所有している人はもちろん，持っていなくても PC 上のエミュレータで試してみることは可能なので，時間を見つけてチャレンジしてもらいたい。

---

[1] Google が開発した Linux ベースの携帯機器用 OS.

[2] Java のプログラムを開発するためのソフトウェア.

[3] オープンソースの IDE．IDE とは，エディタ，コンパイラ，デバッガなどの開発環境を統合したもの.

[4] Apple 用ソフト開発のための IDE．ときどき有料になったりするが基本的には無料.

[5] Apple が C をオブジェクト指向に拡張した言語. Apple 用ソフトの開発以外には使い道がないのが残念.

２． Processing for Androidとは

Processing は，画像，アニメーション，インタラクティブアプリケーションなどの作成に適した[3]オープンソースのプログラミング言語である。グラフィカルプログラミングの入門用として開発されたため非常に扱いやすい。特別なインストールは不要で，ダウンロードして解凍すればすぐに使い始めることができる。IDE はコンパクトでシンプルな操作性を持つエディタを中心に作られており，インタプリタなので作成したプログラムの実行・停止はボタン一つだ。対応するプラットホームも，Windows，Mac OS，Linux と幅広い。

文法的には Java ベースなので，制御文などは Java や C と変わらず，それに描画機能やマウス，キーボードなどのイベント処理機能が付け加えられている。基礎的なプログラミングの経験があれば，後で紹介するサンプルなどを参考に，実行結果を画面上で確かめながら自習していくのは容易だろう。

言語の簡単さは底の浅さに直結することも多いが，Processing に限ってはそうではない。オブジェクト指向的なこともできるし，ネットワークや画像処理などのライブラリも充実している。もちろん，大規模なビジネスアプリケーションの開発効率では Java に太刀打ちできないだろうし，グラフィックのスピードは C++に追いつけないだろう。しかし扱いやすさではこれらの言語の比ではない。たとえアンドロイドのプログラミングに興味がなくても，グラフィカルでインタラクティブなプログラムを作ってみたいと思うのなら，開発者によるわかりやすい解説書の翻訳[4]も出たことだし，一度試してみる価値はある。

この Processing で作成したアプリケーションを，ほとんどそのままアンドロイド上でも動作させることを目指しているのが，Processing for Android というプロジェクトである。文献[2]には，

> ただ一連のコードを書いて，Run（Crtl-Rもしくは Cmd-R）を押すだけだ。そうすれば，スケッチ[6]は，エミュレータの中でアンドロイドのアプリケーションになる。Run on Device（Ctrl-Shift-R もしくは Cmd-Shift-R）を選べば，PCにつないだアンドロイドの実機で動作する。（筆者訳）

と書かれているが，本当にそのとおりなのである。現在も開発途上であり，エミュレータとの連携などには若干不安定なところもあるが，Eclipseを使った正攻法の開発と比べれば”foolishly easy” [7]にアプリケーションを作ることができる。

[6] Processing ではソースコードのことをスケッチと呼ぶ.
[7] Ben Fry の言葉.

３． インストール

Processing を使ったアンドロイドアプリケーション作成には，Java，Android SDK そして Processing 本体が必要になる。以下では，それらのインストール法を紹介するが，リンク先などに関する情報は 2012 年 1 月中旬のものなので注意してほしい。主に使用した環境は

CPU：Core i7　2720QM(2.2GHz)
RAM：4.00GB
OS：Windows7 Service Pack1

だが，適切なものをダウンロードすれば Mac OS，Linux でも同様なはずである。

3.1. Javaのインストール

Javaには，Javaで作られたアプリケーションを動かすための実行環境JREと，Javaのアプリケーションを作成するための開発キットJDK（もしくはSDK[8]）がある。今回はJDKを使用するが，JREはJDKに付属しているので別途ダウンロードする必要はない。

JDK には，標準的な開発環境の SE，組み込みシステム開発用の ME，サーバーシステム開発用の EE の各エディションがある。通常のプログラム作成に使用するのは SE でその最新バージョンは 7(1.7)であるが，Android SDK が公式に対応しているのは JDK5(1.5)と 6(1.6)であるため，今回は Java SE SDK 6 update29 を使用した。

ダウンロードとインストールの具体的な手順は以下のとおりである。

(1)http://www.oracle.com/technetwork/jp/java/javase/downloads/index.html 中段 Java SE 6 Update 29 の JDK Downloads のリンクをクリック。
(2)Oracle Binary Code License Agreement のリンクからライセンスを確認し，同意できるのなら Accept License Agreement にチェック。
(3)64bit WindowsなのでWindows x64 (32bit[9]ならWindows x86)のリンクからダウンロード開始。
(4)ダウンロードされた jdk-6u29-windows-x64.exe をダブルクリックしインストーラ起動。
(5)インストール先をC:¥Program Files¥java¥jdk1.6.0_29¥からC:¥java¥jdk1.6.0_29¥へ変更[10]。
(6)JRE のインストール先を C:¥java¥jre6¥へ変更。

[8] JDK が Java2(1.2)以降では SDK と呼ばれるようになり，5(1.5)以降では再び JDK に戻った.
[9]32bit であるか 64bt であるかは，”コントロールパネル”の”システムとセキュリティ”の”システム”にある”システムの種類”から確認できる.
[10] 空白を含むフォルダ名はトラブルを起こすことがあるので避けることにしている。

インストールが終了すると製品登録になるが，登録は任意であり行わなくても使用には差し支えない。JDKと同時にJREもインストールされるため，JREが元々入っていた場合[11]には複数のバージョンが共存する可能性がある。http://java.com/ja/download/uninstall.jspを参考に旧バージョンのJREの削除も検討すべきだろう。なお，最近のMacにはJDK6 がプリインストールされているので，JDKのインストールは不要なはずだ。

3.2. Android SDKのインストール

Android SDK の導入は以下のように行った。

(1)http://developer.android.com/sdk/index.html から Windows 用 Android SDK をダウンロード。
(2)ダウンロードした installer_r16-windows.exe を起動し Next をクリック。
(3)自動的に JDK6（version 1.6）を見つけるので再び Next をクリック。
(4)ディレクトリを C:¥Android¥android-sdk へ変更。
(5)Android SDK Tools をスタートボタンに登録。

インストールが終了すると Android SDK Manager が起動し，SDK の Platform をダウンロードする画面（Fig.1）が表示される。

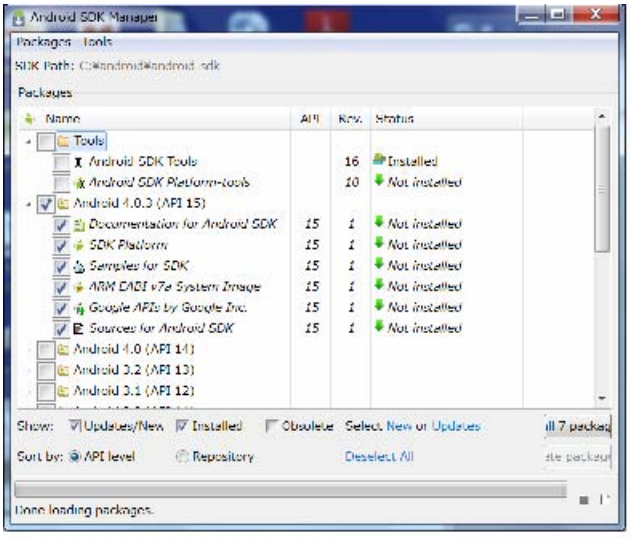

Fig.1　Android SDK Manager

デフォルトでは Android 4.03(API15)用のパッケージが選択された状態になっているが，Processing は API8 を必須としている。そこで Android2.2 の項目を展開し

SDK Platform
Samples for SDK
Google APIs by Google Inc.

にチェックを付けておく。また Windows では Extra の下にある Google USB Driver package も必要なので，チェックが付いていることを確認しておく。

以上の後に Install と表示されたボタンをクリックし，次の場面で Accept ではなく Accept All にチェックを入れれば，数分でダウンロードとインストールは終了する。

3.3. Processing on Androidのインストール

現在のProcessingのバージョンは 1.51 であるが，プレリリースのバージョン 2.0a4 が

http://www.processing.org/download/

からダウンロードできるので，それを使用した。ダウンロードしたら適当な場所[12]に置いてダブルクリックすれば，フォルダprocessing-2.0a4 が作られ，中にはIDEの起動プログラムprocessing.exeが入っているはずだ。

3.4. デバッグ用ドライバのインストール

作成したアプリケーションを実機で動作させるためには，各社から提供されるデバッグ用ドライバが必要になる。筆者のアンドロイドは LG 製の L-01D なので

http://www.lg.com/jp/mobile-phones/download-page/L-01D/product-info-driver.jsp

からダウンロードし，以下の手順でインストールした。

(1)ダウンロードした LGUnitedMobileDriver_S4981CAN35AP22_ML_WHQL_Ver_3.5.exe を PC にインストール（ダブルクリックするのみ）。
(2)L-01D の”設定…開発…USB デバッグ”にチェック。
(3)PCとL-01Dを付属のUSBで接続（ドライバはPNPで自動的[13]にインストールされる）。

インストール後はデバイスマネージャーで，？マークが出ていないことを確認しておく。

4．　アプリケーションの作成

4.1. PC上のアプリケーション

最初にProcessingによるPC用アプリケーションの作成について述べよう。Processing-2.0a4 のフォルダ内のprocessing.exeを起動するとFig.2 のIDEが表示される。

[11] ほとんどの場合そうである．

[12] パス名に日本語が入っていると動作しないことがある．そういう場合は C:¥などに置くとよい．

[13] 携帯側で USB 接続モードの設定画面が出た場合は PC 同期にチェックを入れる．

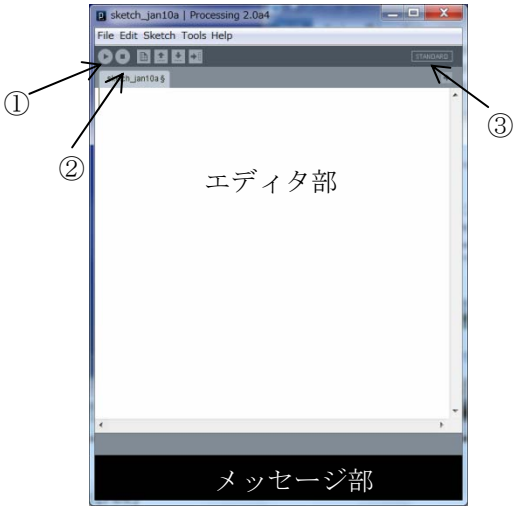

Fig.2　Processing Development Environment

使い方は簡単で，エディタ部にスケッチを入力し，実行するには①のボタンをクリックする。停止させたいときは②をクリックする。プログラムに文法的なエラーがあればメッセージ部に内容が表示されるので，それを参考に修正すればよい。なお③の実行モードは初期状態ではSTANDARDと表示されているはずだが，もしANDROIDとなっていたらこの部分をクリックしてSTANDARDに変更しておく。

エディタ部に以下のスケッチを入力[14]する。

```
float x, y;                                (1)
float targetX, targetY;                    (2)
float easing = 0.05;                       (3)

void setup() {                             (4)
  size(640, 480);                          (5)
  x = 320.0;                               (6)
  y = 240.0;                               (7)
  mouseX = 320;                            (8)
  mouseY = 240;                            (9)
  fill(111, 231, 234);                     (10)
}                                          (11)

void draw(){                               (12)
  background(255);                         (13)
  targetX = (float)mouseX;                 (14)
  targetY = (float)mouseY;                 (15)
  x = x+(targetX-x)*easing;                (16)
  y = y+(targetY-y)*easing;                (17)
  ellipse((int)x,(int)y,100,100);          (18)
}                                          (19)

void mousePressed(){                       (20)
  if(mouseX<10 && mouseY<10)               (21)
    exit();                                (22)
}                                          (23)
```

Sketch1. Easing

入力が終わったら，Fig.1の②をクリックすると，下のようなウィンドウが開く。

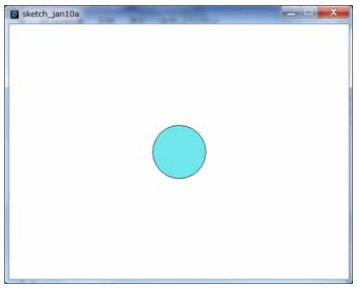

Fig.3 Easing

画面上でマウスを動かすと，少し遅れて円が追いかけていくはずだ。このスケッチは，IDEのFileメニューにあるExamplesから参照できるサンプルBasics/Input/Easingをベースに作成したものである。Processingの文法も含めて内容を説明しておこう。

最初に知らなくてはならないことは，スケッチが次のような形式をしていることである。

```
外部変数の宣言　//必要に応じて

void setup(){
    //起動直後に一度だけ実行される部分
}

void draw(){
    //無限に繰り返される部分
}

その他の関数の記述　//必要に応じて
```

まず，最初に外部変数[15]を宣言する。次のsetup関数とdraw関数が中核となる部分で，前者で初期設定を行い，後者の内容を無限に繰り返す構造をしている。つまりプログラムの開始直後に一度だけsetupが実行され，その後はdrawの内容が繰り返される。したがってプログラムには明示的な終わりはなく，停止するにはユーザーの手でウィンドウを閉じるのが普通なのだが，この例ではmousePressedで，画面上のある範囲内でマウスをクリックすると終了するようにしている。

[14] 各行末の(1)～(23)までの番号は説明のために付加した行番号なので，入力してはならない。

[15] プログラム中のすべての関数で使用できる変数.

次に，各命令について説明しよう。

(1)～(3)では浮動小数点型の変数を宣言している。これらは外部変数[16]であるから，プログラム内のすべての関数で使用可能である。宣言したx， yは描画する円の中心のX座標とY座標, targetX, targetYは円の移動目標（マウスカーソルの位置）のX座標とY座標である。また，easingは円が目標へと近づく比率（≒速度）に相当し，初期値 0.05 は目標と円の距離が 1 ループごとに 5%ずつ近づいていくことを意味している。

(4)はスケッチの実行が開始される setup 関数の始まりで，(11)までがその範囲である。

(5)はプログラムのメインウィンドウの大きさの設定で，X 軸（横軸）方向に 640 ドット，Y 軸（縦軸）方向に 480 ドットを指定している。

(6)，(7)は表示する円の初期位置で，X座標を 320，Y 座標を 240 としている。通常，コンピュータ画面上の座標は左上隅を原点（0，0）とし，X座標の増加方向は右向き，Y座標の増加方向は下向き[17]であるが，Processing もその原則に従っている。したがって，ウィンドウサイズが 640×480 ドットの場合，（320，240）は画面中央に相当する。なお(1)を

```
float x=320.0;
float y=240.0;
```

としておけば(6)，(7)は不要なのだが，後述するアンドロイド版との関係上あえてこのようにしている。

(8)，(9)のmouseX, mouseYは，整数型のシステム変数[18]で最新のマウスの座標位置を保持している。このスケッチでは円をマウスに追従させているが，そのためのマウス位置はこれらの値から取得する。なお, 起動直後は，マウスがウィンドウ上にないためmouseX, mouseYの値は 0 となっており，プログラムは（0，0）をマウスの位置と認識してしまう。そのため円は左上へ移動を開始してしまうのだが，それを防ぐため，円の初期位置と同じ（320，240）に設定しているのである。もちろん値を代入していても，実際にマウスカーソルがウィンドウ上で移動すれば値は正しい位置へと更新される。

(10)は塗りつぶし色の設定で, 別の設定を行うまでは，描かれる図形はすべてこの色で塗りつぶされる。色の指定は RGB 方式により行い，Red，Green，Blue の各色素を 0～255 の範囲で定める。ここでは Red を 111，Green を 231，Blue を 234 としているため，エメラルドグリーンに近い色合いとなっている。なお，任意の色に対する RGB の値を調べたいときは，IDE メニューの Tools の下にある Color Selector が役に立つ。

(12)は draw 関数の始まりで(19)までがその範囲である。この部分は無限ループとして実行され，プログラムの実質的な処理内容となる。特に指定しなければ 1 秒間に 30 回このループが実行されることになっているが，PC の速度が追いつかない場合はその限りではない。

(13)は背景の設定で，ウィンドウ全体を指定された色で塗りつぶす。ここではモノクロ指定を行っており 255 は白色での塗りつぶしに相当するが，0 ならば黒色，1～254 ならば値に応じた灰色となる。また(10)の fill と同様に RGB 指定も可能で，(255,255,255)とすれば同様に白色が指定される。このように無限ループの内部で画面の塗りつぶしを行えば，毎回ウィンドウ全体が書き直されるため，円が動いているように見えるわけである。

(14)，(15)はマウス座標の取得だが，得られるのが整数型のデータなので(float)によって浮動小数点型へと変換[19]している。

(16)，(17)は円の位置をマウスに追従するように定めている部分である。円の中心 x，y は easing で決められた割合でマウスの位置 targetX，targetY へと近づけている。たとえば x, y が(320, 240)のときにマウスを(100, 100)へ移動させると，x，y は(309，233)，(298，226)，(288，219)，(278，213)，・・・と徐々に(100，100)へと近づいていくことになる。

(18)は楕円を描く命令である。パラメータとして与えるのは，中心の X 座標，Y 座標，楕円の X 軸長，Y 軸長であるが，X 軸長，Y 軸長を共に 100 とすることによって直径 100 の円を描いている。なお Processing には点 point，線 line，長方形 rect，三角形 triangle，四角形 quad などの描画命令もある。各パラメータの与え方などについては，IDE の HELP の下にある Reference を参照してもらいたい。

(20)～(23)の mousePressed はマウスボタンがクリックされたときに実行されるイベント処理関数である。この例では，プログラムを終わらせる方法として画面左上の(0，0)から(10．10)の範囲内でマウスをクリックしたら，exit を実行するようにしている。もちろん，実際の終了はウィンドウを閉じてしまえば良いので，この処理が必要というわけではなく，あくまでもイベント処理の例として提示したものである。

4.2. エミュレータでの実行

次は，スケッチを PC 上のプログラムであるアンドロイドエミュレータで実行しよう。

まずFig.2③のSTANDARDと表示されている部分をクリックしAndroidモードへと変更する。ファイルを保存していなければ，するように指示されるので従うと

[16]これに対して各関数内で宣言した変数は内部変数と呼ばれ，その関数の内部でしか使用できない.
[17] Y 軸の向きが数学の座標軸とは逆向きになる.
[18] 使用する前に宣言する必要はない.
[19] このケースでは自動的に型変換されるので，必須というわけではない.

Android SDKがインストールされていることを確認[20]される。「はい」をクリックするとインストール場所を求められるので，3.2(4)で指定したC:¥Android¥android-sdkを入力する。するとIDE画面が薄緑色となりAndroidモードへ変更されたことが認識できるはずだ。なおSDKのインストール先の入力が必要なのは，最初にAndroidモードを実行したときのみである。

驚くべきことにPC用に作成したProcessingのスケッチはほとんどそのままアンドロイドでも実行できるが，多少は変更しなければならない部分もある。今回は，Sketch1のsetup関数を以下のように変更する。

```
void setup() {
  x = (float)width/2;
  y = (float)height/2;
  mouseX = (int)x;
  mouseY = (int)y;
  fill(111, 231, 234);
}
```

Sketch2. Modified setup function

アンドロイドでは，個々の機器によって液晶サイズや解像度に大きな差があるため，size関数でウィンドウサイズを指定するのは難しい。そのためsize関数を書かなければ，ディスプレイ全体にメインウィンドウが描画されることになっている。描かれた実際の画面のサイズについては，システム変数widthとheightを通じて入手可能であり，Sketch2ではこれらを利用して画面の中央を求めている。

プログラムの変更が終わったら，先ほどと同様にFig.2の①をクリックして実行を試みると，いくつかのウィンドウが閉じたり開いたりしながらアンドロイドのエミュレータの起動が開始されるはずだ。初回の起動時には，GoogleからAndroid SDKの使用状況を収集したいというメッセージが表示されるかも知れないが，それはProceedで受け入れてしまえば良い。とにかく①をクリックして数十秒後[21]にはエミュレータが完全に起動し，Fig.4のような画面が表示される。

立ち上がったエミュレータでスケッチの実行が開始されれば万々歳なのだが，筆者の環境[22]ではいくら待っても動作しなかった。ちなみに，起動時のエミュレータはロック状態なのだが，画面左下の緑色の鍵マークをドラッグして解除しても状況は変わらない。IDEのメッセージ部にはLost connection with device while launching. Try again.と表示されるが，Fig.2の①を何度押してみても同じだった。

Fig.4　Android　Emulator

試行錯誤的にいろいろと試してみた結果，実行を成功させるためには，下の手順によってエミュレータを立ち上げることが必要だということが判明した。

(1)エミュレータが開いていたら閉じる。
(2)IDE のメニューから”Android…Android SDK＆AVD Manager”を選び SDK Manager を呼び出す。
(3)SDK ManagerはAPIのリポジトリチェックを行うが，それが終了するのを待つ。
(4)SDK Managerメニューの“Tools…Manage AVDs[23]”でAndroid Virtual Device Managerを呼び出す。
(5)デフォルトで登録されている API Level 8 の AVD を選択し，右側の Start ボタンをクリック。
(6)表示されたエミュレータの設定画面でLaunchのボタンをクリック。

体感的には，先ほどよりもさらに長く待たされるが，再びエミュレータが起動し Fig.4 と同様の画面が表示されるはずである。なお，数分間待ってもエミュレータが起動しない場合には，起動途上のエミュレータのウィンドウを閉じて再び上の手順を繰り返す。それでも駄目な場合は，PC を再起動する。こうして起動したエミュレータを E1 と呼ぶことにする。

E1を起動させたままで，再度Fig.2の①のボタンを押して実行を試みると，E1とは別のエミュレータE2が起動を開始するが，途中で再びLost connection…が発生してしまう。E2が完全に起動するのを待ってから，E2のウィンドウを閉じて終了させ，もう一度①のボタンを押して実行を試みる。すると別のエミュレータE3 の起動が始まり，これまで以上に時間がかかるかも知れないが，そのまま待っていればE1かE3のどちらかでスケッチが

[20] インストールされていない場合，この段階でインストールすることもできるようだ．
[21] 結構長い時間を必要とする．
[22] いずれも Windows7 を搭載した VAIO と MacBook 自作 PC の三台でテストした．
[23] AVD は Android Virtual Device の略で，エミュレータを意味する．

動作[24]する。妙な手順であるが，異なるマシンで何度試しても同じ結果だったので，それなりの再現性はあると思われる。また，一度実行できてしまえば，二度目以降の実行は容易であった。

さてエミュレータでの実行であるが，動作速度が非常に遅い。しかし，画面上でマウスをクリックすれば，円はその場所へとゆっくり移動するし，画面左上[25]でクリックすればプログラムが停止するので機能については完全に再現されている。率直に言って，現状でのエミュレータの使い勝手は良いとは言えないが，アンドロイドの実機を所有していなくてもテストを行えることのメリットは大きい。

4.3. 実機での実行

最後は，実機 L-01D での実行である。実機のディスプレイでは左上隅 10 ドットの範囲は非常に狭く，指先でタップするのは難しい。そこで Sketch.1 の(21)を

```
if(mouseX<100 && mouseY<100)
```

とし，左上 100×100 ドットの領域をタップしたら停止するように変更した。

実機での実行手順は以下のとおりである。

(1)スケッチが実行状態なら Fig.2 の②の停止ボタンをクリックして停止。
(2)実機側で「設定…アプリケーション…開発」から USB デバッグにチェックが入っていることを確認。
(3)PC と実機を USB ケーブルで接続。
(4)実機側で USB 接続モードが表示されたら，”PC 同期”にチェックを入れ OK をタップ。
(5)IDE の Sketch から Run on Device を選択。

ビルドに多少時間を要するが，エミュレータでの実行のようなトラブルもなく，あっけなく実行が開始されるはずだ。動作速度もエミュレータより遥かに高速で，簡単なアプリケーションではあるが，指先でボールを動かしているような不思議な感覚を味わえるだろう。

なおスケッチはデバイス上にアップロードされた状態になるので，アンドロイドマーケットからダウンロードしたソフトと同様に使い続けることができる。

4.4. GPSの利用

アンドロイドの魅力の一つは，さまざまなセンサーが搭載されていることである。ここではセンサー利用の例として，文献[2]にあるサンプル How to read the GPS をベースに作成した現在地の緯度，経度，高度，速度を表示するプログラムを紹介しよう。なお，Sketch.1 と同様に(1)～(44)は説明のための行番号である。

```
import android.content.Context;      (1)
import android.location.Location;    (2)
import
  android.location.LocationListener; (3)
import
  android.location.LocationManager;  (4)
import android.os.Bundle;            (5)

LocationManager lctManager;          (6)
MyLctListener lctListener;           (7)

float crtLatitude = 0.0;             (8)
float crtLongitude = 0.0;            (9)
float crtAltitude = 0.0;            (10)
float crtSpeed = 0.0;               (11)

String[] fList;                     (12)
PFont myFont;                       (13)

void setup (){                      (14)
  fList = PFont.list();             (15)
  myFont =
    createFont(fList[0], 48, true); (16)
  textFont(myFont);                 (17)
}                                   (18)

void draw(){                        (19)
  background(0);
  text("緯度:"+crtLatitude,20,50);   (20)
  text("経度:"+crtLongitude, 20,100);(21)
  text("高度:"+crtAltitude,20,150);  (22)
  text("速度:"+crtSpeed,20,200);     (23)
}                                   (24)

void onResume(){                    (25)
  super.onResume();                 (26)

  lctListener = new MyLctListener();(27)
  lctManager
  = (LocationManager)getSystemService
    (Context.LOCATION_SERVICE);     (28)

  lctManager.requestLocationUpdates
  (LocationManager.GPS_PROVIDER, 0, 0
  ,lctListener);                    (29)
}                                   (30)
```

[24] 動作するエミュレータは E1 であることもあったし，E3 であることもあった.
[25] (0，0)～(10，10)の範囲内.

```
void onPause(){                          (31)
  super.onPause();                       (32)
}                                        (33)

class MyLctListener implements
   LocationListener{                     (34)

  void onLocationChanged
    (Location lct){                      (35)
    crtLatitude
      = (float)lct.getLatitude();        (36)
    crtLongitude
      = (float)lct.getLongitude();       (37)
    crtAltitude
      = (float)lct.getAltitude();        (38)
    crtSpeed
      = (float)lct.getSpeed()*3.6;       (39)
  }                                      (40)

  void onProviderDisabled
    (String provider){}                  (41)

  void onProviderEnabled
    (String provider){}                  (42)

  void onStatusChanged (String provider,
    int status, Bundle extras){}         (43)
}                                        (44)
```

Sketch.3　GPS

プログラムについて簡単に説明しておこう。

(1)～(5)は使用するライブラリの取り込みである。アンドロイドにおけるセンサー情報は，マネージャーに登録したリスナーを通じて取得するのだが，(6)，(7)はそれらの宣言である。

(8)～(11)は GPS から取得した情報を保管するための変数で，それぞれ緯度，経度，高度（m），速度（m/sec）に相当する。

(12)，(13)はフォントを格納するための変数である。

(14)～(18)のsetupでは，フォントの設定をしている。(15)で使用可能なフォントのリストを取得し，(16)でリストの[0]番目[26]を使って 48 ドットサイズのフォントを作成し，(17)でそれを選択している。

(19)～(24)の draw では，取得した緯度，経度，高度，速度を表示している。

(25)～(30)はアプリケーションの開始直後に実行される関数で通常の処理(26)を行った後に，(27)でリスナー，(28)ではマネージャーを作成している。(29)ではリスナーにマネージャーを登録しているが，同時に位置情報の更新法を設定している。なお，ネットワーク（無線 LAN）による位置検出を行う場合は(29)の GPS_PROVIDER は， NETWORK_PROVIDER とする。続く 2 つのパラメータは，位置情報通知のための時間・距離の間隔であり，0，0 とすれば時間，距離ともに最小間隔での更新を要求することになる。

(31)～(33)はアプリケーション停止時の処理だが，通常の処理(32)以外は何もしていない。

(34)～(44)はリスナーの実装で，4 つの関数が必要となるが，内容があるのは(35)～(40)の onLocationChanged だけである。この関数は(29)で定められた条件で呼び出され，最新の情報が代入される。なお，速度は(39)で m/sec から Km/hour へと変換している。

スケッチの実行法は 4.2，4.3 と同様だが，事前に IDE の Android メニューから Sketch Permissions を選び，各機能のパーミッションを設定する必要がある。GPS なら， ACCESS_FINE_LOCATION， ネットワークは ACCESS_COARSE_LOCASION， エミュレータは ACCESS_MOCK_LOCATION にチェックを入れなければならない。なお，エミュレータでテストする場合は android-sdk/tools にある ddms.bat からデバッグモニタを呼び出し，右上にある Emulator Control タブの Manual で仮想的に位置情報を設定することができる。

実機で実行してみると，高度や速度にはある程度の誤差があるようだが，経度，緯度に関してはかなり正確に測定できるようである。

５． 終わりに

ここでは，プログラム言語 Processing によるアンドロイドアプリケーションの開発法を紹介した。エミュレータには若干の問題があるし，そもそもエミュレータで動かしても特に面白くもないのだが，もしアンドロイドの実機を持っているなら，是非一度試してもらいたい。これまで体験したことのない，新たなプログラミングの面白さを味わえるはずだ。

参考文献

[1]レイモンド，伽藍とバザール
http://cruel.org/freeware/cathedral.html(1999)
[2]Ben Fry，About Processing for Android
http://wiki.processing.org/w/Android(2011)
[3] Processing　http://processing.org/
[4] Casey Reas, Ben Fry，Processing をはじめよう，O'Reilly(2011)
[5] Google，Location and Maps，http://developer.android.com/guide/topics/location/index.html

[26] 0 番目を選ぶことに特別の意味はない.