2011/01/22

吉岡梅(yoshioka@ciruelo.jp)

AIR for Android をプレリリース時からテストしてきた体験から、AIR for Android に関す るよもやま話を集めてみました。基本的な仕組みや、スマートフォンのデバイスにアクセ スするための方法などは他の方が解説してくださっているかと思いますので、その他の話 題や、個人的にハマって苦労した点に的を絞ってご紹介します。まとまりもとりとめもな く、裏をきちんと取ってないので間違いなどもあるかもしれないレポートですが、悪しか らずご了承ください。

1. AIR for Android の歩みと現状

AIR for Android(以下、「AFA」)の開発はどのような経緯で進んできたのでしょうか。以下、 ざっとまとめてみました。

1	iPhone 発売、大人気!当時は iOS3.0
2	Adobe の AIR チームさん、iPhone 用に Flash で作成したアプリを書き出せる、 「Packager for iPhone(PFI)」の開発スタート。
3	PFIのベータテスト始まる。 ←私はこのへんから参加
4	PFI が目玉機能の FlashCS5 の発売決定!PFI は AIR2.0 相当の機能。
5	AIR チーム、Android 向けの AIR「AIR for Android(AFA)」の開発にも着手。 確か当時の Android の OS は 2.0 (Éclair)、対応 AIR バージョンは 2.0。
6	Apple 社、CS5 発表1日前にサードパーティ製のコンパイラで作成した iOS 向けアプリの登録を 禁止。事実上、PFI が死に機能に。本当にびっくりしました。
7	CS5 発売。
8	AIR チームさん、iOS 向けの開発をストップし、AFA へ注力。
9	Google、Android 向けの OS2.2(Froyo)を発表(この辺時系列あいまい)。
9 10	Google、Android 向けの OS2.2(Froyo)を発表(この辺時系列あいまい)。 AIR チーム、AFA を Froyo 向けに変更、さらに対応 AIR バージョンを 2.5 に。パフォーマンスが 一気に上がる。本当に一気に。「これはいけるわー」という感触。
9 10 11	Google、Android 向けの OS2.2(Froyo)を発表(この辺時系列あいまい)。 AIR チーム、AFA を Froyo 向けに変更、さらに対応 AIR バージョンを 2.5 に。パフォーマンスが 一気に上がる。本当に一気に。「これはいけるわー」という感触。 AFA リリース !
9 10 11 12	Google、Android 向けの OS2.2(Froyo)を発表(この辺時系列あいまい)。AIR チーム、AFA を Froyo 向けに変更、さらに対応 AIR バージョンを 2.5 に。パフォーマンスが 一気に上がる。本当に一気に。「これはいけるわ一」という感触。AFA リリース !Apple 社、アプリ登録の規約を変更。PFI のアプリでも登録可能に。 が、この時点で PFI は AIR2.0 ベースのままストップ中。Android の 2.5 ベースと差が生まれる。
9 10 11 12 13	Google、Android 向けの OS2.2(Froyo)を発表(この辺時系列あいまい)。AIR チーム、AFA を Froyo 向けに変更、さらに対応 AIR バージョンを 2.5 に。パフォーマンスが 一気に上がる。本当に一気に。「これはいけるわー」という感触。AFA リリース !Apple 社、アプリ登録の規約を変更。PFI のアプリでも登録可能に。 が、この時点で PFI は AIR2.0 ベースのままストップ中。Android の 2.5 ベースと差が生まれる。Apple 社、iOS を 4.0 に更新。同時にアプリのアップロードには専用のアップローダー(MAC の Cocoa 環境で動作)となる。事実上、MAC がないと登録不可に。
9 10 11 12 13 14	Google、Android 向けの OS2.2(Froyo)を発表(この辺時系列あいまい)。 AIR チーム、AFA を Froyo 向けに変更、さらに対応 AIR バージョンを 2.5 に。パフォーマンスが 一気に上がる。本当に一気に。「これはいけるわー」という感触。 AFA リリース! Apple 社、アプリ登録の規約を変更。PFI のアプリでも登録可能に。 が、この時点で PFI は AIR2.0 ベースのままストップ中。Android の 2.5 ベースと差が生まれる。 Apple 社、iOS を 4.0 に更新。同時にアプリのアップロードには専用のアップローダー(MAC の Cocoa 環境で動作)となる。事実上、MAC がないと登録不可に。 CS5 の PFI で書き出した iOS 向けファイルは Application Loader ではアップロードできないことが判明。

と、いうわけで、現在は Flash では iOS 向け、Android 向けの両方のアプリが書き出せる ようになっていますが、一時期 iOS 向けの開発がストップしていたため、iOS 向けと Android 向けでは、使用できる機能に差があります。

■ Flash でアプリの作成できるスマートフォン向け OS

OS	AIR	説明
iOS3.0以上	2.0	CS5 に Packager for iPhone のアップデートを適用すると、
	相当	AIR2.0 相当の機能を使ったアプリの開発が可能
		また、Windows でもアプリ開発自体はできるが、appStore
		に登録する場合には、MAC が必要
Android2.2 以上	2.5	CS5 に AIR for Android エクステンションを組み込むと、
	相当	AIR2.5 相当の機能を使ったアプリの開発が可能

iOS 向けのアプリ書き出しは、AIR2.0 ベースです。それに対して Android 向けのアプリ書 き出しは AIR2.5 ベースです。AIR2.0 と AIR2.5 の機能の差を抜粋すると、以下のようなも のが挙げられます。

■ AIR2.5 から使用できる機能

機能	カメラへのアクセス
	マイクへのアクセス
	StageWebView の利用
	SD カードへのアクセス ··等

つまりは、現状では iOS ではカメラ機能を使ったアプリ作成等ができないわけですね。 ただ、この差はすぐに埋められると思われます。

ちなみに、Android OS のバージョンは次のような感じでバージョンアップしています。コ ードネームはアルファベット順だそうですが、このままだとすぐZまで行くんじゃないか、 という恐ろしい勢いです。バージョンアップの速度が超早いので、いろいろ検証が大変に なりそうですね。

コードネーム	バージョン	<i>ک</i> ر
Donut	1.6	ドーナツ
Eclair	2.1	エクレア
Frave	0.0	フローズンヨーグルト
Froyo	Ζ.Ζ	AIR for Android が動作するのはここから
	0.0	ジンジャーブレッド
Gingerbread	2.3	2011 年はこれが主流になるのかな?
11	10	ハニカム
Honeycomb	3?	タブレット用と言われているが、詳細不明

Android	のバー	ジョ	ン

2. Android アプリを作成する際にそろえておきたい環境

Flashを使ってAndroidのアプリを開発する際には、いろいろと準備が必要です。単にFlash から Android 向けのアプリを書き出すだけであれば、エクステンションひとつをダウンロ ードして組み込めば済むのですが、同じ PC 上でいろいろなテストを行いたい場合には、そ れだけではちょっと足りないのです。

そもそも、Android アプリというものは、Java で作成されてきました。そのため、Java 環境ではいろいろな便利なツールが揃っているのです。これらのツールの一部は、Flash を 使ってアプリを作成する際にも有用なのです。

と、いうわけで、「Java 環境を作る」「Java 環境で使える Android アプリ作成用のツール を用意する」という2段構えで環境を作成することとなります。

環境	目的
JDK	Java を使った開発行うための環境
(Java Development Kit)	Android SDK をインストールして実行する際に必要
	※win の場合は 64 ビットマシンでも 32 ビット用の物をイン
	ストールしておいた方が無難だそうです
Android SDK	Android 開発者向けの様々なツール

■ 準備する2段階の環境

とりあえず、以下に win の場合の設定をざっとメモしておきます。また、このあたりの環 境づくりに関しては、web等で詳しい方法を検索していただいた方が確実かもしれません。

Java 開発環境を準備する

まずは Java 開発環境を準備します。Oracle さんから JDK をダウンロードして指示に従っ てインストールします。

 \Rightarrow <u>http://www.oracle.com/technetwork/java/javase/downloads/index.html</u>

インストール後に、JDK の bin フォルダへのパスを、Windows の環境変数「Path」へと 設定します。win のメニューより、[コンピュータ]を選択し、[システムのプロパティ]を選 択し、さらに左側のメニューから「システムの詳細設定」を選択します。

[システムのプロパティ]ダイアログが表示されますので、「詳細設定」タブを選択し、[環境 設定]ボタンを押します。[環境設定]ダイアログが表示されますので、「システム環境設定」 欄から「Path」という項目を探して選択し、[編集]ボタンを押します。

すると、やっと[システム変数の編集]ダイアログが表示されます。お疲れ様!もうちょいで す。おそらくこの時点で「編数値」欄には、既に何らかの文字列が入力されているかと思 います。この末尾に、区切り文字である「;」を加え、さらに JDK の bin フォルダへのパス を追加します。これで完成です。





さて、パスがきちんと通ったかどうかを確認してみましょう。[cmd]+[R]で「cmd」入力な どの方法でコマンドプロンプトを呼び出し、「javac -version」と入力して[Enter]キーを押 します。ここで、「javac 1.6.0_23」等のバージョン番号が表示されれば OK です。 ■ javac –version でパスが通っているかを確認

🧧 ファイル名を指定して実行	
定行するプログラム名、または開 ント名、インターネット リソース	くフォルダーやドキュメ 〈名を入力してください。
名前(<u>O</u>): cmd	- [command]+[R]で[ファイル名を指定し て実行]ダイアログを呼び出し、「cmd」 と入力して[OK]ボタンを押してコマン
OK \$77	ドプロンプトを表示 ンセル 参照(B)
■コマンドプロンプト Microsoft Windows [Version 6.1.7600] Copyright (c) 2009 Microsoft Corpora	tion. All rights reserved.
C:¥Users¥ume>javac -version javac 1.6.0_23 C:¥Users¥ume> ってい	ンドプロンプトで java のパスが通 いるかどうかを確認

Android SDK を準備する

Java の開発環境が整ったら、今度は Java を使った Android アプリの開発環境である 「Android SDK」をインストールします。

 \Rightarrow <u>http://developer.android.com/intl/ja/sdk/index.html</u>

自分で展開する ZIP 版と、インストーラ付きの exe 版がありますが、どちらをダウンロー ドしても構いません。

ダウンロードが済んだら、ZIP版の場合は展開し、展開された「android-sdk-windows」フ オルダを適当なディレクトリへと配置して下さい。さらにフォルダ内にある「SDK Manager.exe」を実行すると、Android SDKのうち、必要なパッケージを選択&ダウンロ ードするためのマネージャーが表示されます(exe版の場合には自動実行されます)。

■ 必要な物をダウンロード

Virtual devices	· · · · · · · · · · · · · · · · · · ·
Installed packages	SDK Location: C:¥Android¥android-sdk-windows
Available packages	Packages available for download
Settings	🔽 🎬 Android Repository
About	🔽 🙀 Android SDK Platform-tools, revision 1
	Documentation for Android SDK, API 9, revision 1
	🔽 🟺 SDK Platform Android 2.3, API 9, revision 1
	🔽 🏺 SDK Platform Android 2.2, API 8, revision 2
	🔽 🏺 SDK Platform Android 2.1, API 7, revision 2
	🔽 🏺 SDK Platform Android 1.6, API 4, revision 3
	🔽 🏺 SDK Platform Android 1.5, API 3, revision 4
	🔽 🎬 Third party Add-ons
	🔽 🦥 Google Inc. add-ons (dl-ssl.google.com)
	📝 🙀 Google APIs by Google Inc., Android API 9, revision 1
	📝 🐞 Google APIs by Google Inc., Android API 8, revision 2
	📝 🐞 Google APIs by Google Inc., Android API 7, revision 1
	📝 🙀 Google APIs by Google Inc., Android API 4, revision 2
	📝 👼 Google APIs by Google Inc., Android API 3, revision 3
	🔽 🛃 Google Usb Driver package, revision 4
	📝 🛃 Google Market Licensing package, revision 1
	🗹 🤮 Samsung Electronics add-ons (innovator.samsungmobile.com)
	🔽 🙀 GALAXY Tab by Samsung Electronics., Android API 8, revision 1
	Description
	Android Repository
	Add Add-on Site Delete Add-on Site I Display updates only Refresh Install Selected

マネージャー左のメニューから「Available packages」を選択し、必要なパッケージにチェ ックを入れ、ダウンロードします。どれが必要かは、きちんと検証はしていないのですが、 AFA で開発を行うのみであれば、最低限、

- Android Repository の Android SDK Platform-tools, revision 1 (adb という実行ファイルが必要。また ddms という便利なツールも含まれる)
- Third party add-ons の Google Usb Driver package, revision 4 (win の場合、デバイスを USB 経由でつないだ際のドライバが必要)

の2つはダウンロードしておいた方がいいですね(他にも必須項目あるかもしれません)。

さて、必要なパッケージのダウンロードが済んだら、またもや環境変数「Path」の設定が 待っています。おなじみの手順で[システム変数の編集]ダイアログを表示し、今度は Android SDK 内の「tools」フォルダと、「platform-tools」フォルダへのパスを追加します。 例えば、C ドライブ直下の「Android」フォルダ内に「android-sdk-windows」を展開した 場合には、既存の環境変数 Path に加えて、

 $; C: \cite{Android} and roid \cite{sdk} windows \cite{tools}; C: \cite{Android} and roid \cite{sdk} windows \cite{point} platform \cite{tools}; c: \cite{Android} \cite{Android} \cite{sdk} windows \cite{point} platform \cite{tools}; c: \cite{Android} \cite{Android} \cite{sdk} windows \cite{platform} platform \cite{tools}; c: \cite{Android} \cite{sdk} \cite{sdk} windows \cite{platform} \cite{sdk} \cite{s$

というような形で追加するわけですね。

必要なパッケージをインストールできたか、パスがきちんと通ったか、を確認するには、 コマンドプロンプトで「adb version」と入力してみましょう。「Android Debug Bridge version 1.0.26」のようにバージョン番号が表示されたら成功です。

さらにコマンドプロンプトで「ddms」と入力してみましょう。「Dalvik Debug Monitor」 が立ち上がればインストールが成功しています。

Dalvik Debug Mor e Edit Actions	nitor De	vice I	Help					[- • ×	
					Info	Threads	VM Heap	Allocation Tracker	Sysinf 1 >	
Name					DDM-aware	?	-			
HT019P8052	224			Online	App descrip	tion:	-			
-					VM version:		-			
					Process ID:		-			
					Supports Pr	ofiling Contro	l: -			
					Supports H	PROF Control:	-			
(,		開発時は	こなにか	と役に立つ	モニタツー	- <i>)</i> l
+ 🖌 – 🛞 🤅	0) 🛞 (ē 🖳 🖡			[Dalv	ik Debu	g Monitor]		
Log										
Time		pid	tag	Message					*	
01-24 02:00:	D	193	dalvik	GC_EXPLICIT freed	d 825 objects / 7	468				
01-24 02:18:	Ι	193	EventL	Aggregate from 12	95801298120 (k	og),				
01-24 02:18:	D	193	dalvik	GC_EXPLICIT free	d 888 objects / 1	392				
01-24 02:26:	D	86	dalvik	GC_FOR_MALLOC	freed 29407 obj	ects			_	
01-24 02:30:	D	193	dalvik	GC_EXPLICIT free	d 348 objects / 1	479				
Filter:										

■ 「ddms」で「Dalvick Debug Monitor」が立ち上がる

実機を USB 経由で繋いで Android デバイスとして認識されるかをチェック

さて、2つの環境がインストールできました。ここで実機がある場合には、USB ケーブル で実機と PC をつないでみましょう(逆に言うと、この時点まで実機はつながない方が無難 です)。

そして、コマンドプロンプトから「adb devices」と入力します。この時点で、端末の番号 が表示されれば成功です。USB 経由で実機へと作成したアプリをインストールする事がで きます。

■ デバイス接続の確認



また、この時、コマンド自体は受け付けられるものの、端末番号が出てこない場合には、 対応するドライバがまだ用意できていないか、端末が Android デバイスとして認識されて おらず、単なる大容量ハードディスクとして認識されている可能性があります。

比較的新しい端末で開発を行う場合には、まだ Google 側でドライバの配布が追い付いてい ないケースがありますので、自前でドライバの設定が必要です。このケースでは、端末内 に用意されているドライバを使用できるケースもありますが、ケースバイケースですので、 web 等でその端末のドライバ設定の情報を検索してみるのが良いでしょう。

また、ハードディスクとして認識されてしまっている場合には、USB ケーブルを抜き差し すると認識しなおしてくれるケースもありますが、それでも認識されない場合には、[コン トロールパネル]・[デバイスマネージャー]を使ってチェックしてみましょう。



■ デバイスマネージャーでチェック

大容量ハードディスクドライブとして認識されてしまっている場合には、[操作]・[ハードウ ェア変更のスキャン]を選択して、デバイスを再認識させましょう。すると、「Android Phone」という項目が表示されます(表示されない場合には、大容量ドライブとして認識さ れているデバイス項目を削除して再実行してみて下さい)。この時点で正しくドライバが認 識されないで、「!」マークが表示されている場合には、該当項目を選択し、ポップアップ メニューより、「ドライバソフトウェアの更新」メニューを選択します。そして、手動での ドライバインストールを選択し、Android SDK ディレクトリ内の「google-usb_driver」フ オルダを選択すると、対応するドライバが組み込まれます。

3. Android アプリを作成する前に考えておいた方が良い事

いろいろな環境が準備でき、「AS でアプリ作れるなら今までのノウハウ活かしていっちょ やるかー」と、なるわけですが、その前に少し考えておいた方がいい項目が何点かありま す。

解像度の変化に対する方針

Android の端末は、さまざまな解像度を持っています。そこで、解像度が変化した場合に、 コンテンツの表示をどうするのか、という方針を決めておいた方が安全です。 同じ端末でも端末上部に表示される通知バー部分を表示したままアプリを起動するか、フ ルスクリーンで起動するかによって、アプリの使用する画面サイズは変化してきます。



■ 通知バー部分の表示・非表示

また、気を付けないといけない点は、端末によってタテヨコ比(アスペクト比)も異なる点で す。web のコンテンツでいうところの、ブラウザのサイズ変更に対応してきっちりと各パ ーツを再配置する、リキッドレイアウトのような考え方が必要になってくるわけですね。 そういう意味では、今まで Flash で web コンテンツを作成されていた方にとっては、「ブラ ウザ上にムービーをどう表示するか」というノウハウが活かせますね。

■ 画面サイズに関する選択の例

方針	説明
おまかせ作戦	サイズ変更に特に対応しない。
	AIR の自動サイズ調整に任せる。
アスペクト比合わ	アスペクト比に応じて画面内のパーツの位置を調整し、「はみだし」部
せ作戦	分が無いようにする。
	拡大/縮小に関しては、AIR の自動サイズ調整に任せる。
NO_SCALE 作戦	stage.scaleMode=StageScaleMode.NO_SCALE を適用し、自動サイ
	ズ調整による拡大/縮小を適用せず、自前で各パーツのサイズを算出・
	設定する。
mxml 作戦	(Flash Builder & mxml のレイアウトに任せる)

このあたりの処理には以下のようなプロパティやメソッドを使って端末の情報を取得し、 対応することになるかと思います。

プロパティ/メソッド	用途
stage.scaleMode	コンテンツ全体の拡大/縮小表示モードを決める
stage.stageWidth	表示領域の幅・高さを取得
stage.stageHeight	
Capabilities.screenDPI	表示端末の解像度情報を取得
Capabilities.screenResolutionX	
Capabilities.screenResolutionY	
stage.align = StageAlign.LEFT;	コンテンツの表示の基準点を左上に設定

■ 異なる解像度に対応する際に役立つプロパティ、メソッド

また、通知バーの表示/非表示の設定は、[AIR Android 設定]ダイアログの「一般」タブ内

の「フルスクリーンモードにする」のチェックで切り替えることができます。

■ 通知バーの表示/非表示の設定

アプリケーションとインストーラーの設定 (ベータ版)							
一般 デプロイ アイコン	2 権限						
出力ファイル:	testApp.apk						
アプリケーション名:	testApp						
アプリケーション ID :	air. testApp 例:com.yourdomain.appname						
バージョン:	1.0.0 バージョンラベル:						
起動時の縦横比:	縦長モード・						
	フルスクリーンモードにする						
	■ 自動回転を有効にする						

もし、コンテンツの伸縮やパーツの位置調整を行う方針で行く場合には、「resetControls メソッド」のような適当な再配置専用のメソッドをあらかじめ作成しておき、配置関係は 集中管理して行うような仕組みを用意しておくのがお勧めです。

デバイスの回転に対する方針

デバイスが回転した際、コンテンツの表示をどうするのか、という方針もあらかじめ決め ておいた方が良い物のひとつです。

デバイスが回転した際、その向きに合わせてコンテンツも自動的に回転して表示するには、 [AIR Android 設定]ダイアログの「一般」タブ内の「自動回転を有効にする」のチェックで 切り替えることができます。

■ 通知バーの表示/非表示の設定

アプリケーションとイン	ストーラーの設定 (ベータ版)
一般 デプロイ アイコン	/ 権限
出力ファイル:	testApp apk
アプリケーション名:	testApp
アプリケーション ID :	air. testApp 例:com.yourdomain.appname
バージョン:	1.0.0 バージョンラベル:
起動時の縦横比:	〔縦長モード ▼
ſ	フルスクリーンモードにする自動回転を有効にする

自動回転を有効にした場合には、画面が回転した際にコンテンツの表示を調整する必要が 出てきます。何もしないと、本当にそのまんま表示方向が変わるだけです。

とりあえず自動回転を有効にした所



通知バーを表示している場合には、通知バーの分だけその向きでのタテ方向の領域が占有 される点にも注意です。タテ・ヨコの比が縦向き時と横向き時で変わってくるわけですね。 画面の向きが変更された場合に任意の処理を実行するためには、

stage.addEventListener(StageOrientationEvent.ORIENTATION_CHANGE,func);
function func(e:Event):void{

switch(stage.deviceOrientation){
	case StageOrientation.DEFAULT:
	break;
	case StageOrientation.ROTATED_LEFT:
	break;
	case StageOrientation.ROTATED_RIGHT:
	break;
	case StageOrientation.UPSIDE_DOWN:
}	

}

と、orientationChange イベントで向きの変更を拾い、stage.deviceOrientation で変更後 の向きを確認し、対応した処理を実行します。ここからコンテンツ上のパーツを再配置す る関数を呼んであげてもいいですね。

また、この設定のオン/オフにかかわらず、デバイスは回転します。当たり前ですが、横む ければ横向きますし、逆さにすれば逆さになります。そこで、回転に対応する別の手法と して、enterFrame イベント等で定期的に stage.deviceOrientation の値をチェックし、変 更があったら任意の処理を実行してもいいですね。

たとえば、メモ用アプリを作成し、写真を表示できる機能を作成したとします。このよう なケースでは、デバイスの向きを変更した際に、コンテンツ全体の向きを変更する必要は ありませんが、写真を横向きにした方が見やすいですよね。そこで、 stage.deviceOrientationの値によって、コンテンツ的な向きはそのままで、表示している 写真や一部のコントロールの向きだけを変更してしまう、といった対応方法を取る、とい った事もできます。



■ コンテンツの向きはそのままで、デバイスの向きの変化に対応した処理を作成

また、コンテンツの向きは、自動回転させるだけではなく、任意のタイミングでコードから変更することもできます。

stage.setAspectRatio (StageAspectRatio.LANDSCAPE);	//横向きに
<pre>// stage.setAspectRatio(StageAspectRatio.PORTRAIT);</pre>	//縦向きに

のように「stage.setAspectRatio メソッド」の引数に、「StageAspectRatio クラス」の2つ の定数のどちらかを指定して実行します。つまり、コードから任意の向きに変化させる場 合には、「タテかヨコか」のどちらかにしか変更できません。

これまたピンポイントで端末の向きを変更したい時に便利ですね。特にスマートフォンで は、端末の向きが重要な意味を持つ場合があります。例えば、ソフトウェアキーボードで す。

	端末の向きによ	こってソフ	トウェアキー	-ボードの)方式が変化す	ることも
--	---------	-------	--------	-------	---------	------

■ 1417	이미러		776.			クエル変化	699-6	τ	
*		Ē		— 1:42					
フリック	入力はタテ								
タテ	たて	C 4	14	44					
					ノフリ	ック入力	したい場合	合には、	
ထို	あ	か	さ	X DEL	縦向]きが便利			
←	<i>t</i> -	tì	It						I
	,	<i>.</i>	104						
SYM	ま	せ	5	space					
あA	% ° 大↔小	わ	。?!	لے					
3 0						a 1	1:43		
フリック	入力はタテ	゠	やすい						
	////	1// /(/)0	1-301		/	ローマ	字入力した	こい場合に	には
1 ***	-11 20	ヤフノ	247/	civaci		_ 横向き;	が便利		
0199	いシ	1771	21/1	SiyaSt	siyasi		-		
q	W	e	r t	У	u	I C) p		
а	S	d	f g	g h	j	k I			
カナ	Z	X	С	V	b n	m	\otimes		
あA	SY	м ,		لے space	•	T	لې		

画像は「Simeji」という Android 界隈では超メジャーな日本語 IME を使用している場合の テキスト入力画面です。縦向きと横向きの場合のソフトウェアキーボードの状態に注目し てください。縦向き時はフリック、横向き時はローマ字入力となっていますね。 テキスト入力の方式は、ユーザーさんによって得手不得手がありますので、長いメモを取 るようなアプリを作成する場合、ピンポイントで setAspectRatio メソッドを使用すれば、 コードから端末の向きを変更し、ユーザーさんの得意な方式のキーボードを表示できるよ うにする、等の処理が作成できますね。

ちなみに、実はこのあたりの「コードからの向きの変更」処理、少し前までは

「stage.setOrientation メソッド」で実行できました。が、なんやかんやあって、今は setAspectRatio メソッドで向きを変更するようになっています。ヘルプ等も含め、少し前 の資料には stage.setOrientation メソッド方式の手順を紹介しているものも存在していま すのでご注意を。

多言語対応に関する方針

Java で Android アプリを作成する場合、多言語対応に対する対策として、あらかじめ言語 ごとに res/values フォルダ内にサブディレクトリを作成し、strings.xml を用意する、とい った方法が用意されています。AFA では、この方式は(現状)使用できません。そのため、多 言語対応を考えている場合にはあらかじめなんらかの対策を立ててからアプリ作成を始め た方が安全です。

いろいろな方法があるかとは思いますが、筆者の場合には、英語と日本語のテキスト表記 を管理する xml ファイルを作成し、アプリに同梱する手法を取ってみました。さらに表示 文字列を集中管理するクラスを一つ作成し、そこにユーザーが選択した設定の文字列情報 をまとめて読み込んで管理するようにしてみました。そして、アプリに表示する文字列は、 必ずこのクラス経由で取得する、というルールを作成し、運用するようにしました。

■ xml ファイルで言語単位で表示文字列を管理



ついでに表示文字列の管理と xml ファイルの作成を一括でできるように、Excel のワーク シートで表示文字列を管理し、マクロで各 xml ファイルを作成できるようにしておきまし た。このあたりは、自分の手慣れたアプリをサブツール的に使用がお勧めです(いっそ AIR で専用サブアプリを作成してしまってもいいですね)。

🗶 🛃 🧐 77111	 ・ ・ ・	多言語対応xml 数式 データ 校閲 表示	作成用 - Microsoft Excel	
	MS P ゴシック 10 B I U - 10 B J U - - 10 D J D - 10 - 10	·A·A· · Ă·· · Ă·· · Ă·· · Ă·· · · Ă·· · · · ·	標準 ・ 職条件付き書式・ 調マ・%・, 第テーブルとして書式設定: 第・%・% ・ 受セルのスタイル・ 5 数値 5 スタイル	□□ 挿入、 Σ、 ▲ ●●● 挿入、 ● ▲ ●●● 前次、 ● ▲ ②●● 並べ替えと 検索と セル 編集
	A1 - 🤄 🎜		Excel のワークシー	トで表示すろ文字列を一括管
A	В	C		
2	app.sm.data.activity.(@name=='Msg').quit	理&書き出しする仕	組みを作ってみました。
4 5 6	日本語版のバス 英語版のバス	C:¥Users¥ume¥Desktop¥Android検 C:¥Users¥ume¥Desk y op¥Android検	この手のサブツール	~的なものをあらかじめ作っ
7 8 9	アクティビティ名 TextCardWindow	コントロール名(分類名) title_default tag default	ておくと楽にな、管	理・変更が楽になりますね。
10		okButton	夏叔	Add
11		cancelButton	戻る	Return
12		upDate	更新	Update
13		picNatFound	登録された画像が見つかりませんでした。 移動、もしくは、削除された可能性があります。	Registered images could not be found. Movement, or may have been deleted.
14		cantMoreAttach	このメモには、これ以上添付ファイルを追加できません。	This card can't add any more attachments
15		deleteAttach	この添付ファイルを削除しますか?	Delete this attachment?
16		deleteHelp	ヘルブ用の添付ファイルは削除できません	The attach for Help notes can't be remove
17	CardLineWindow	deleteButton	削除	Delete
18		deleteMsg	このメモを削除しますか?	Delete this card?
19		cantAdd	このページはこれ以上メモを追加してきません	Can't add more notes on this page.
20		update	史新	Opdate
20		caricei nawTitla	キャノビル ページカイトルを1 カレアイださい	Variuei
23		deleteHeln	ヘルブ用のマモは削除で考ません	The Help notes can't be removed
24	Msg	quit	アプリケーションを終了しますか?	Quit Application?
25	MicWindow	noVoice	ボイスデータが破損しているため、再生できませ ん。	Can't find the voice data.
	Stringリソース一覧/ダミーのカードデ		j 4	
コマンド				

■ 表示文字列を Excel で一括管理してみました

多言語対応を考える場合には、アプリ制作前にどのような方針で行くのかをちょいと考え てから進めてみましょう。後でステージ上やライブラリ内のテキストを必死になって変え たりするのは、超めんどくさいです(やりました)。

どんなファイルを同梱するかに関する方針とアクセス方法

AFA で作成するアプリでは、設定用の xml ファイルや、表示したいムービーや画像の swf ファイル、png ファイルなどの各種ファイルを同梱してパッケージングする事ができます。 画像などは、開発中に差し替えることを考慮して、png ファイルを適宜読み込んで使用する、 なんて運用をする事が多いかと思いますが、そのまんまパッケージングできるわけですね。 同梱するファイルを決めるには、Flash の[AIR Android 設定]ダイアログボックス内の「一 般」タブ内の「含めるファイル」欄で設定を行います。この際、ファイル単位だけでなく、 フォルダ単位でも同梱するファイルを指定できます。

■ ファイルを同梱する際の設定

アプリケーションとイン	マストーラーの設定 (ベータ版)
一般 デプロイ アイコ	2 権限
出力ファイル:	assetsTest.apk
アプリケーション名:	assetsTest
アプリケーション ID:	air. assetsTest 例:com.yourdomain.appname
バージョン:	1.0.0 バージョンラベル:
起動時の縦横比:	縦長モード▼
	フルスクリーンモードにする
	■ 自動回転を有効にする
レンダリングモード:	CPU -
含めるファイル:	assetsTest.swf assetsTest-app.xml assets data
	同梱するファイルは「一般」タブ内の 「含めるファイル」欄で指定できます
	OK キャンセル パブリッシュ ヘルプ

つまりは、すべての素材を Flash ドキュメント内で管理しなくても OK なわけですね。

さて、このようにして同梱したファイルにコードからアクセスする場合、ファイルへのパ スの指定はどうすればよいのでしょうか。おなじみの Loader&URLRequest で同梱した画 像ファイル等を読み込む場合には、そのまま相対パスでアクセス可能です。

//「piyo.png」という画像ファイルを同梱した場合

var req:URLRequest = new URLRequest("piyo.png");

var loader:Loader = new Loader();

loader.load(req);

addChild(loader);

フォルダごと同梱した場合には、フォルダ名から指定すれば OK です。

//「data」フォルダ内に「piyo.png」という画像ファイルを同梱した場合

var req:URLRequest = new URLRequest("data/piyo.png"); var loader:Loader = new Loader(); loader.load(req); addChild(loader);

AIR アプリでおなじみの File を使ってアクセスしたい場合には、いろいろなアプローチが あるのですが、とりあえず「app:/ファイルパス」のような形式で同梱したファイルへとア クセス可能です。

// piyo.png」という画像ファイルを同梱した場合
var f:File = new File("app:/piyo.png");
//ファルを開き、バイト配列へと内容を取得
var s:FileStream = new FileStream();
s.open(f,FileMode.READ);
var data:ByteArray = new ByteArray();
s.readBytes(data);
//取得した値を利用。今回は画像を Loader に表示。
var loader:Loader = new Loader()
loader.loadBytes(data);
addChild(loader);

バイナリとして同梱ファイルを扱いたい場合には File を利用すれば OK ですね。

ちなみに、File を利用していろいろな場所からファイルの読み書きを行う場合、次の3種類のパスの指定方法を覚えておくと便利です。

指定方法	パス
File.applicationDirectory	同梱ファイルのルートです。
または、	読み取り専用ディレクトリです。
app:/	
File.applicationStorageDirectory	いわゆる「アプリケーション専用ディレクトリ」
または、	です。
app-storage:/	ここは読み書き可能です。アプリの設定などを保
	存する場合にはここで OK ですね。
File.desktopDirectory	SD カードのルートです。
File.documentsDirectory	SD カード上にフォルダを作成したり、ファイルの
	読み書きをしたい場合にはこちらを基本にパスを
	作成していきます。

■ File のパス指定あれこれ

この仕組みを活用する場合には、何を読み込むのか、また、どんなフォルダ構成で管理す るのかをあらかじめ決めておいた方が良いですね。

4. 開発・デバッグの際に知っておくと便利な仕組み

AFA でアプリを開発する場合に便利だった仕組みを 2 点ご紹介したいと思います。ひとつ は実機でテストする際に、おなじみの trace 関数やエラーメッセージ等を確認する方法、も うひとつは、条件付きコンパイルを利用したデバッグの方法です。

DDMS を使って trace 関数やエラーメッセージを確認する

AFA でアプリを作成し、書き出す際には3パターンの書き出し方法が用意されています。

	-
タイプ	説明
デバイスリリース	いわゆる本番用
	実際に実機にインストールしたり、配布する際に使用
エミュレーターリリース	PC 上で動作する Android エミュレータにインストールする
	際の方法。PC 上でテストしたいときに使用
デバッグリリース	trace 関数の出力や、エラーメッセージの確認等をしたい際
	に利用できる方法

■ 3パターンの書き出し方法

このうち「デバッグリリース」で書き出した場合には、tracce 関数やエラーメッセージ等、 Flash の「ムービープレビュー」で表示した際に[出力]ウィンドウに表示されるおなじみの 情報を、Android のログへと出力してくれます。

■ デバッグモードで書き出す

アプリケーションとイ	ンストーラーの設定 (ベータ版)	×
一般 デプロイ アイ:	コン 権限	
証明書:	mePodLabo¥bu¥tde_Android¥mycertp12 🖌 参照… 作成…	
パスワード:	•••••	
	☑ このセッション中はパスワードを保存する	
ー Android デプ	ロイタイプ	
◎ デバ	7200-2	
○ I ₹2		
◎ デバ	yグ	

そして、この Android に書き出されたログを確認する際に便利なツールが、「DDMS(Dalvik Debug Monitor)」です。DDMS は Android でアプリを作成する際に、様々な用途に使える モニタツールなのですが、そのうちのひとつに、端末から出力されるログを、リアルタイ ムで確認できるという機能があります。

つまりは、デバッグモードでアプリを書き出し、実機にインストールし、DDMS を立ち上 げておけば、操作に応じて配置した trace 関数の出力を確認したり、エラーの発生やその原 因を確認できるというわけですね。

			I	info Threads	VM Heap	Allocation Tracker	Sysinfo	Emulator Conti
				DDM-aware?	-		1	
	Online		222	App description:	-			
com.google.and	Iroid 440	装	8600	VM version:	-			
		**	F	Process ID:	-			
			9	Supports Profiling Control:	-			
			9	Supports HPROF Control:	-			
						DDMC	~14 1	manaid DIZ
	🖗 🗈 📓	H					CIL A	narola 0149
og						確認でき	きます。	
-		nid	tag	Mossage	/		N/1 ~ 1-1-	
ime		più	tay	message	/	traco 🖾	本年(1) 経	単わ エラーッ
01-24 07:51:59.091	V	161	RenderScript_jni	surfaceDestroyed		_ trace 阕	数の結	果や、エラーノ
01-24 07:51:59.091 01-24 07:52:01.061	V I	161 85	RenderScript_jni ActivityManager	surfaceDestroyed Displayed activity a	ir.traceTest,	trace 関	数の結 D内容↓	果や、エラー> ちここで確認で
01-24 07:51:59.091 01-24 07:52:01.061 01-24 07:52:01.121	V I D	161 85 822	RenderScript_jni ActivityManager dalvikvm	SurfaceDestroyed Displayed activity a GC_EXTERNAL_ALL	ir.traceTest, .OC freed 40	trace	数の結 D内容も	果や、エラー> もここで確認て
11me 01-24 07:51:59.091 01-24 07:52:01.061 01-24 07:52:01.121 01-24 07:52:01.121 01-24 07:52:01.141	V I D W	161 85 822 822	RenderScript_jni ActivityManager dalvikvm IInputConnectionW	surfaceDestroyed Displayed activity a GC_EXTERNAL_ALL finishCemposingTex	ir.traceTest, .OC freed 40 t on inactiv	trace 関 /.A: セージの B0 e II ます。	数の結 D内容♥	果や、エラー> らここで確認で
11me D1-24 07:51:59.091 D1-24 07:52:01.061 D1-24 07:52:01.121 D1-24 07:52:01.141 D1-24 07:52:03.991	V I D W I	161 85 822 822 822 822	RenderScript_jni ActivityManager dalvikvm IInputConnectionW air.traceTest	surfaceDestroyea Displayed activity a GC_EXTERNAL_ALL finishComposingTex trace simasita!	ir.traceTest, .OC freed 40 t on inactiv	trace 関 /.4 セージの eli ます。	数の結 り内容も	果や、エラース もここで確認で
Ime D1-24 07:51:59.091 D1-24 07:52:01.061 D1-24 07:52:01.121 D1-24 07:52:01.141 D1-24 07:52:03.991 D1-24 07:52:03.991	V I D W I I	161 85 822 822 822 822 822	RenderScript_jni ActivityManager dalvikvm IInputConnectionW air.traceTest air.traceTest	surfaceDestroyea Displayed activity a GC_EXTERNAL_ALL finishCamposingTex trace simasita! a~aā¼4ãàā¾4ãàā¼4	ir.traceTest, .OC freed 40 .t on inactiv	trace 関 /.A: セージの ⁸⁰ ます。	数の結 D内容も	果や、エラース
Ime D1-24 07:51:59.091 D1-24 07:52:01.061 D1-24 07:52:01.121 D1-24 07:52:01.121 D1-24 07:52:03.991 D1-24 07:52:03.991 D1-24 07:52:03.991 D1-24 07:52:04.011	V I D W I I I I	161 85 822 822 822 822 822 822 822	RenderScript_jni ActivityManager dalvikvm IInputConnectionW air.traceTest air.traceTest air.traceTest	surfaceDestroyea Displayed activity a GC_EXTERNAL_ALL fr finishQrmposingTex trace simasita! ä~ää¼äää¾äää¼ ArgumentError: Erro	ir.traceTest, OC freed 40 t on inactiv or #2004: 0	trace 関 /.A; セージの BBO e Ir ます。 One of the parameter	数の結 D内容 rs is invalid.	果や、エラース
Ime 01-24 07:51:59.091 01-24 07:52:01.061 01-24 07:52:01.121 01-24 07:52:03.141 01-24 07:52:03.991 01-24 07:52:03.991 01-24 07:52:03.991 01-24 07:52:03.991 01-24 07:52:03.991 01-24 07:52:04.011 01-24 07:52:04.011	V I D W I I I I I I	pi0 161 85 822 822 822 822 822 822 822 822 822 822 822 822 822 822	RenderScript_jni ActivityManager dalvikvm IInputConnectionW air.traceTest air.traceTest air.traceTest air.traceTest	surfaceDestroyea Displayed activity a GC_EXTERNAL_ALL fr finishCimposingTex trace simasita! ă`āā¼ăăā¾ăăā¾ ArgumentError: Erri at Error\$/throwE	ir.traceTest, OC freed 4(t on inactiv or #2004: (rror()	trace 翔 /Ap セージの 280 e Ir ます。 One of the parameter	数の結 D内容(rs is invalid.	果や、エラーフちここで確認で
Ime 01-24 07:51:59.091 01-24 07:52:01.061 01-24 07:52:01.121 01-24 07:52:03.991 01-24 07:52:03.991 01-24 07:52:03.991 01-24 07:52:04.011 01-24 07:52:04.021 01-24 07:52:04.021	V I D W I I I I I I I	pi0 161 85 822 822 822 822 822 822 822 822 822 822 822 822 822 822 822 822	RenderScript_jni ActivityManager dalvikvm IInputConnectionW air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest	surfaceDestroyed Displayed activity a GC_EXTERNAL_ALL fr finishCamposingTex trace simasita! ă"ăă¼ăăă¾ăăă¼ ArgumentError: Errr at Error\$/throwE at flash.filesyster	ir.traceTest, .OC freed 40 t on inactiv or #2004: (rror() m::File/set	trace 翔 /Ap セージの e Ir ます。 One of the parameter nativePath()	数の結 D内容(果や、エラークもここで確認で
Ime 01-24 07:51:59.091 01-24 07:52:01.061 01-24 07:52:01.121 01-24 07:52:03.991 01-24 07:52:03.991 01-24 07:52:03.991 01-24 07:52:04.011 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021	V I D W I I I I I I I I I	pi0 161 85 822	RenderScript_jni ActivityManager dalvikvm IInputConnectionW air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest	surfaceDestroyer Displayed activity a GC_EXTERNAL_ALL fr finishCemposingTex trace simasita! ä~äät4äää34äää34 ArgumentError: Err at Error\$/throwE at flash.filesyster at flash.filesyster	ir.traceTest, .OC freed 40 t on inactiv or #2004: (rror() m::File/set m::File()	trace 翔 /Ap セージの 280 e In ます。 One of the parameter nativePath()	数の結 D内容€	果や、エラース
Time 01-24 07:51:59.091 01-24 07:52:01.061 01-24 07:52:01.121 01-24 07:52:03.991 01-24 07:52:03.991 01-24 07:52:04.011 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021	V I D W I I I I I I I I I I	pic 161 85 822	RenderScript_jni ActivityManager dalvikvm IInputConnectionW air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest	surfaceDestroyea Displayed activity a GC_EXTERNAL_ALL fmishCamposingTex trace simasita! ă~äăt4ăăă4ăăă4 ArgumentError: Err at Error\$/throwE at flash.filesyster at flash.filesyster at traceTest_fla:	ir.traceTest; OC freed 4(t on inactiv or #2004: (rror() m::File/set m::File() :MainTimeli	trace 翔 /Ag セージの e If ます。 Dne of the parameter nativePath() ne/func()[traceTest_	数の結 D内容(rs is invalid. fla.MainTime	県や、エラー> らここで確認て eline::frame1:9]
Ime 01-24 07:51:59.091 01-24 07:52:01.061 01-24 07:52:01.121 01-24 07:52:03.091 01-24 07:52:03.091 01-24 07:52:04.011 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:04.021 01-24 07:52:06.141	V I D W I I I I I I I I D	pi0 161 85 822 823 824 825 826 827 828 <	RenderScript_jni ActivityManager dalvikvm IInputConnectionW air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest air.traceTest dalvikvm	surfaceDestroyer Displayed activity a GC_EXTERNAL_ALL fmishCemposingTex trace simasita! ă"ăă¼ăăă¾ăăă¼ ArgumentError: Err at Error\$/throwE at flash.filesyster at flash.filesyster at traceTest_fla: GC_EXPLICIT freed	v ir.traceTest, .OC freed 4(t on inactiv or #2004: (rror() m::File/set m::File() :MainTimeli 2845 objec	trace 翔 /Ap セージの 980 e If ます。 One of the parameter nativePath() ne/func()[traceTest_ ts / 199400 bytes in	数の結 D内容↓	県や、エラーク らここで確認て eline::frame1:9]

■ Dalvik に表示された trace 関数のメッセージやエラーメッセージ

日本語のメッセージは文字化けしてしまうのですが、アルファベットなら OK です。お手 軽に[出力]ウィンドウ代わりに使える便利なツールとして活用してみましょう。

条件付きコンパイルを利用してマルチプラットフォーム対応やデバッグ対応を切り替える

Flash には「条件付きコンパイル」機能が搭載されています。詳しくはアドビの上条さんの blog をご覧ください。

http://cuaoar.jp/2010/10/actionscript-30.html

http://cuaoar.jp/2010/10/actionscript-30-1.html

この仕組みを利用して、iOS用のコード、Android用のコード、さらにはデバッグ用のコー ドなどを切り分けて記述しておき、用途によってコンパイルする箇所を切り替えながら管 理すると、同じ Flash ドキュメントやクラスファイル内での、マルチプラットフォーム対 応や、デバッグ時のみに走らせたいコードなどを整理整頓することができます。

いちいちコメントアウトしたり元に戻したりしながら確かめるよりも楽ですね!

5. コンポーネントどうするか問題

AFA を使ってアプリを作成する場合の大きな問題点として、「コンポーネントをどうしよう」という問題点があります。

現状では、AFA では Android アプリの標準的なボタンやメニュー、スクロールするリスト といったコンポーネントは提供されていません。Flash Builder の次期プレビュー版である burrito には用意されているのですが、Flash の場合には自前で用意する必要があります。 このコンポーネントに関してですが、ヒム・カンパニーの永井さんが、ご自身のブログで ひとつの解決方法をご提示して下さっています。

■AIR Android アプリ作成: Flash, Flex Hero

http://himco.jp/air-for-android/ ■2-6-2:Menu コンポーネント http://himco.jp/air-for-android/?p=286 ■翻訳記事 No:106 「ケビン ホイトのコンポーネント」 http://www.himco.jp/articles/pdf/kevinHoytComponents.pdf

上記の翻訳記事の内容を拝見しますと、

本ドキュメントは、Adobe のプラットフォームエバンジェリストであるケビン ホイト (Kevin Hoyt) のブログで公開されている記事「Some Flash Android Components」のコ ンポーネントの使い方をヒム・カンパニー 永井勝則が自主的に記述したものです。

との記述がされています。日本語で読める AFA 対応のコンポーネントの使い方の記事があるというのは本当にありがたいですね。多謝です。

また、永井さんの blog には、AFA 開発のノウハウやサンプルコードが満載です。びっくり するくらい充実しています。AFA で開発をしようと考えている方は必見です。

6. 現状、AFA ではできないこと

「そういえば AFA であれってできないのかな?」という項目を調べてみて、現状ではどうやらできなさそうな項目をまとめてみました。

項目	説明
インテントの発行	AFA ではインテントの発行はできません。
	URI スキーマは、StageWebView 上では利用できます。
	ただし、navigateURL 等では、http:、tel:、sms:で、ブラウ
	ザ、電話、ショートメッセージ(メールのような物?日本では
	あまり馴染み無い)の各アプリが呼び出せるようです。
インカメラへのアクセス	いわゆる「自撮り」用のカメラへのアクセスはできません。
	自撮り用のカメラへのアクセスというか、「2台目のカメラ」
	へのアクセスができない状態なのかもしれません。
バッテリー残量等へのア	ガラケーの時のコンテンツではおなじみのバッテリー容量情
クセス	報ですが、AFA ではアクセスできません。
ウィジェットの開発	AFA ではいわゆるウィジェットは加発できません。
	フルスクリーンを利用するアプリのみが開発できます。

■ 現状、AFA ではできないこと

この他にもまだいろいろとあるかもしれません。

ただ、AFA はスタートしたばかりの技術ですので、今後のアップデートで可能になる物も 出てくるかと思います。今後の動向に注目ですね。

7. 描画速度に関して

AFA での開発を進めるにあたって、おそらくみなさんが一番悩むのは描画速度に関する問題になるかと思います。やはりまだまだ PC に比べると、そんなに速度は出ません。 そこで、様々な手法で描画速度を上げるチューニングを行うわけですが、正直言って、この項目に関してはまだ手探り状態です。そこで、有効だと思われる方法を列記するに留め置きたいと思います。

- 表示リストの改装を深くしない。理想は 1~2 階層
- GPU レンダリングを効果的に利用する
- cacheAsBitmap、cacheAsBitmapMatrix を効果的に利用する
- 画面に表示するインスタンスを事前に生成できる場合には、事前に用意しておく(生成 と表示のタイミングを切り分ける)
- いわゆる画像プールの仕組みなどを使用し、なるべく BitmapData を使いまわす
- カクついてしまう画面遷移の演出などは避けたり代替のものを利用する
- 画面全体を一気に動かすのではなく、細かくパーツごとの位置を移動する

などなどです。

このあたりはいろいろと実験してみて、有効な方法を模索し続けたいと考えております (azoth 系のサードパーティ製のコンパイラでビットマップの描画部分だけ最適化して手動 パッケージングしてみたらどうかなあ…とか。今度試してみます)。

さてさて、本流とはちょっと外れた話題をとりとめもなくいろいろと書き連ねてきました が、AIR for Android、単純に楽しいですよ!新しいデバイスでのコンテンツ作りというの はやはり楽しいですね。また、身近に持ち歩く端末上で、自分自身が作成したアプリを動 作させることができるというのは、本当に便利です。かゆい所に手が届くものも作れそう ですね。

今後もいろいろと研究し、コミュニティの皆様方にとって面白い情報を提供できるように 頑張りたいと思います。

ではでは。

