1. Overview

With the release of Google's latest Android platform 4.1 (Jellybean), there have been significant changes for Flash support.
Flash players in mobile browsers are no longer supported. As of August 15, 2012, the Flash player browser plug-in is no longer available for download from the Google Play Store.
Please refer to the URL below for more details.

- Adobe AIR and Adobe Flash Player Team Blog / An Update on Flash Player and Android

First, let's take a look at the changes on the Web.
Currently, Flash is handled in browsers using a plug-in. However, many mobile browser developers, including Google, are no longer supporting plug-ins for the latest versions of their mobile browsers. As such, mobile browsers can no longer support Flash content.

Let's look at native apps. In the past, it was easy to use Flash content from native apps by embedding Flash SWF files in HTML files using the WebView provided by the Android platform. However, just like the browsers, WebView is also no longer supported in the latest Android platform. Consequently, it is no longer available.

2. Converting Existing Flash Contents

Adobe announced through several channels that services that used Flash in the conventional mobile environment can be replaced by using HTML5 and AIR.

2-1. Flash to HTML5

Looking at HTML5, it will replace Flash on the web, and the mobile browser in Android 4.x delivers improved HTML5 support so that most HTML5 content works smoothly.
Also, a few companies, including Adobe and Google, are supporting you by providing solutions to convert existing Flash content to HTML5.

Below are two good conversion examples.

- Flash Professional CS6 / Toolkit for CreateJS

  A free plug-in from Adobe. It can be used for authoring Flash content.
  It helps you convert to HTML5 + JavaScript at the content writing stage before exporting to SWF.
- Google Swiffy

  A beta solution provided by Google. When an SWF file is uploaded to the website, it is converted to HTML5 + JavaScript.
  Google also provides a plug-in that can be used with Flash tools.

  By using the services above, you can convert your Flash content to HTML5 + JavaScript. However, the converted outcome still has many restrictions including limited Action Script support, HTML5 compatibility issues with old versions of mobile browsers, and performance issues after conversion. Therefore, the same level of outcome as the conventional Flash player is not guaranteed.

2-2. AIR for Android Native Applications

Secondly, in terms of applications, native apps that are written for AIR can be used normally on the latest Android platform.

- Adobe AIR

Adobe AIR is a runtime environment that supports easy creation of RIA (Rich Internet Applications) through languages such as HTML, AJAX, Flash and FLEX.
It supports multiple platforms including PC and mobile.
AIR-based applications use a separate extension called ANE (AIR Native Extension) and can access device platform APIs written through native code, such as Java, C++ and C#.

On the other hand, the conventional method of embedding Flash SWF files into HTML files using WebView is no longer supported. Therefore, portions created using SWF files must be converted into HTML5 or must be rewritten as a native application using AIR instead of using HTML through WebView.
When doing so, you can use the similar Flash production process and use AIR SDK features when using Adobe's latest Flash tool, Flash Professional CS6.

3. Developing Native Apps Using AIR

Developing native applications using AIR poses many advantages.
Therefore, the use of AIR in app development will increase.
Below, this document provides basic examples of how to write native apps in AIR instead of in Flash.

The following components are required to develop content using AIR.

- JDK: Java SE Development Kit
  http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Adobe Flash Professional CS 6.0
  http://www.adobe.com/products/flash.html
- A device running Android 2.2 or later

For information about downloading and installing these components, please refer to the related links.

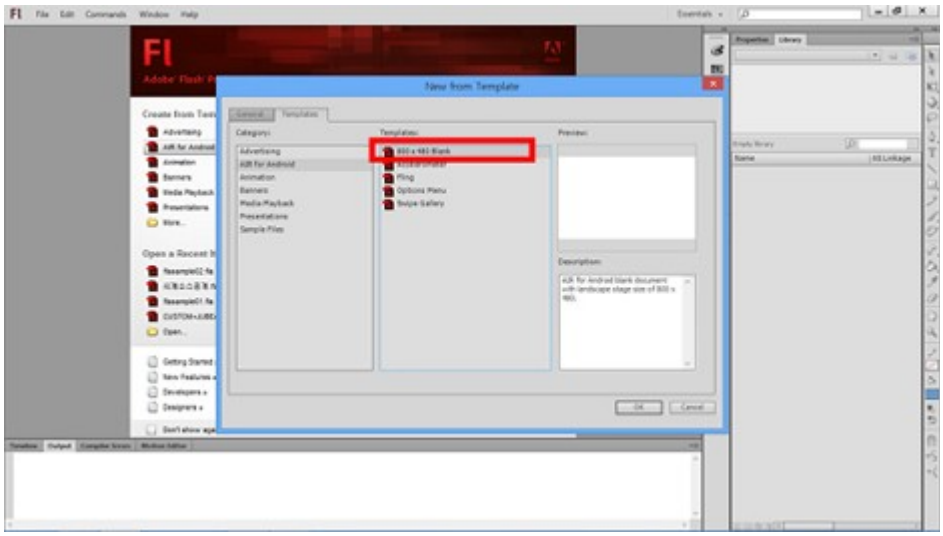
Although AIR supports various languages and development environments, this document focuses on using Flash tools in a similar way to conventional methods of creating Flash content.
Further, the examples are categorized into three categories below, depending on whether the device's native features and action Script are used or not.

- Not using the device's native API
- Using Action Scripts
- Using the device's specialized features, such as gyro sensors using ANE

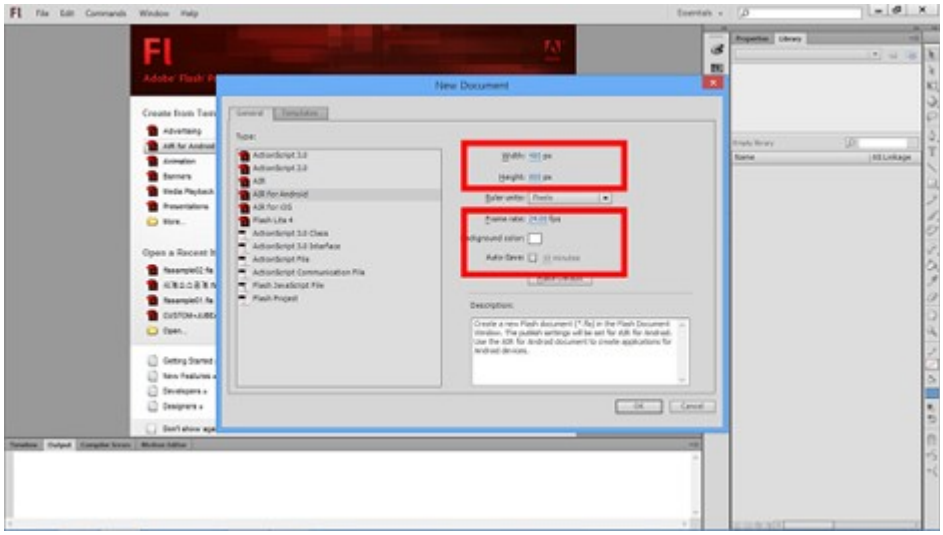
3-1. Not Using the Device's Native API

The following explores an Android native application that consists of simple text and buttons using Flash tools.
In the first screen after running the program, select "AIR for Android" under "Create from Template".

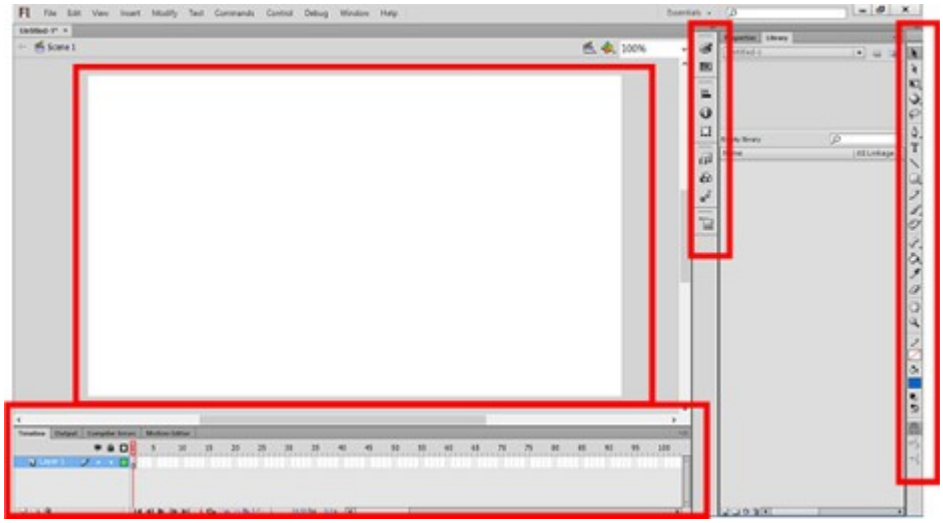
On the next screen, select the “Templates" tab and select “800 x 480 Blank."



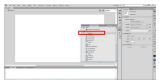
Note that you can set the screen resolution, frame rate, and background color in the “General” Tab.
The default resolution is set to WVGA (800 x 480).



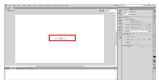
The content editing screen that follows consists of a work area, timeline, output and error indicator panels, scene component, and library properties window, as well as Flash authoring tools.



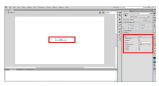
Save the file as “tutorial.fla”, activate the Components menu, and select Button components from the list of displayed items.



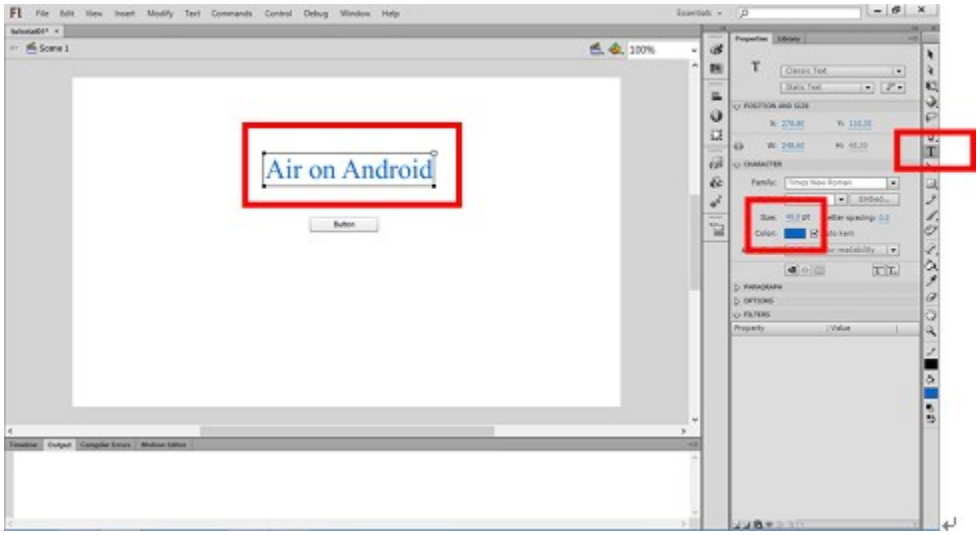
Drag & drop the selected button component to the center of the scene composition screen.



Component setting values can be changed in the Properties window.

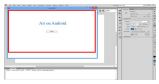


Use the Text Tool to create a label at the top of the screen and change the size and color of the font as you wish.

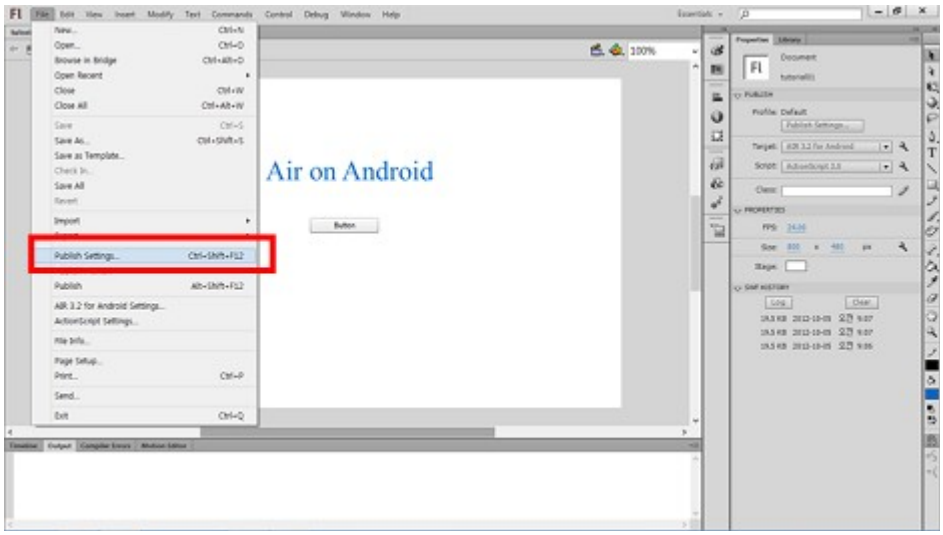


Your current progress can be previewed using the Publish Preview option under the File menu.

When exporting to an SWF file, the program emulates and displays the results.

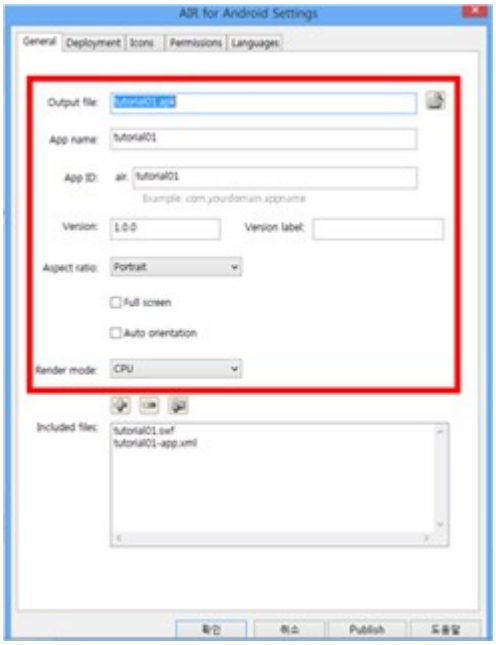


To acquire your intended results in the binary format of an actual Android application, you must select “Publish Setting” in the File menu and adjust several options.



In the Publish Settings window, click the Tool icon next to “Target: AIR 3.2 for Android” combobox then enter the detailed settings window.

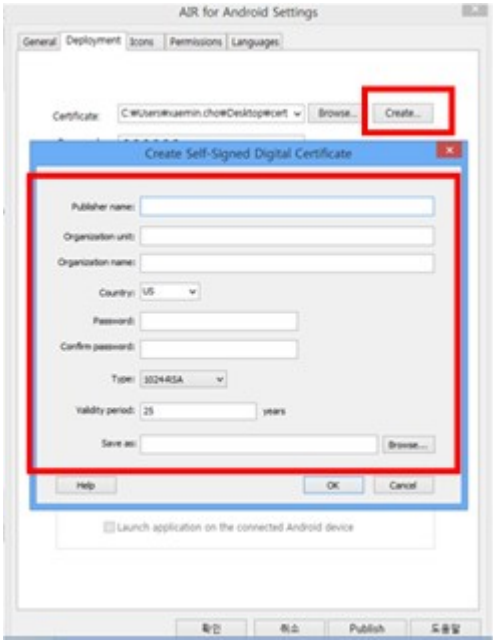
When the “AIR for Android Settings” window appears, select the “General” tab, then set the properties such as output file/app, and name/App ID/version.
To remove the status bar at the top of the screen when the app is running, check “Full Screen”.
To allow the app to automatically orient, check “Auto Orientation."
The Render Mode properties have two options, CPU and GPU.
Selecting GPU improves the performance of applications written using AIR.
However, it also restricts the use of a few features and properties.



In the “Deployment" tab, select a certificate for app certification, and choose release type from Actual Device, Emulator, or Remote for Debugging.
Then, you must decide whether to embed the AIR runtime for running the app within the app

binaries, or to have it downloaded and installed through an external store when it is not present.
If it is embedded within the binary file when building, it increases the size of the binary file by approximately 8 MB.

You can also easily generate a certificate.
To generate a certificate, just fill out the necessary information such as publisher name, organization unit, organization name, country, and password.

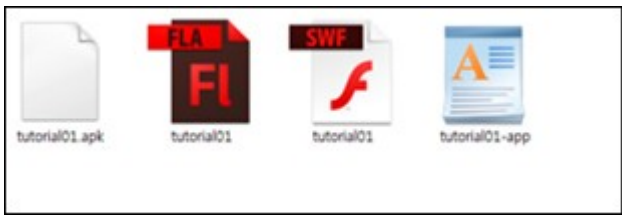


On the “Icons” tab, you can register icons from among 36x36, 48x48, or 72x72.
When not specified, the default Android icon is used.

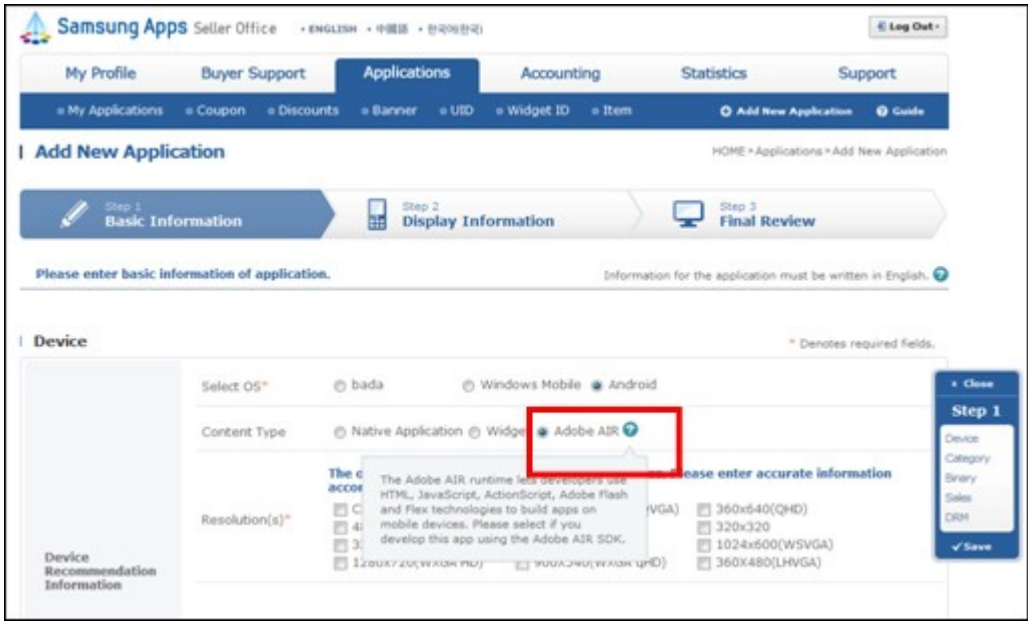
Using major features on Android devices such as Wi-Fi connections and cameras requires permission to use them. You can select the permissions you wish to have on the “Permissions” tab.
For example, the WAKE_LOCK permission prevents the device from returning to the lock screen when the app is running.

On the “Languages” tab you can select the supported languages for your app.

When you are done adjusting the settings, click Publish and your app is published.
In the directory where you saved the project, there is an FLA file, a SWF file, an XML file containing application settings information, and the APK file, which is the application binary file.

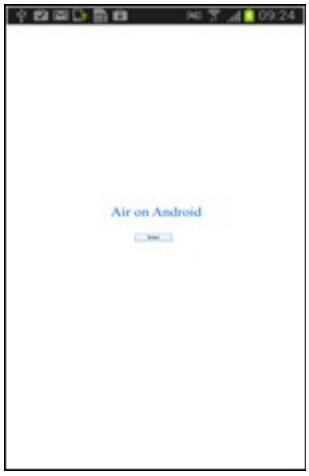


Once created, the APK file can be registered and sold at Samsung Apps.
Also, when registering your app at the Samsung Apps Seller Site (http://seller.samsungapps.com/), you must choose “Adobe AIR” as the Content Type.



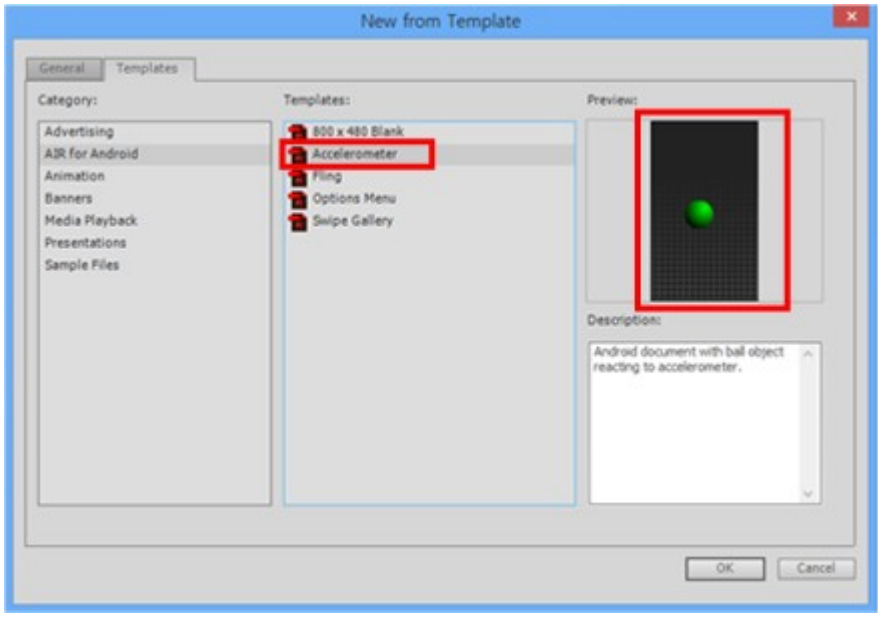
When the app is installed and run on the device for the first time, but the AIR runtime is not embedded in the app, a one-time runtime installation process is required to run your AIR app.

When the AIR runtime is successfully installed, your app is automatically loaded.

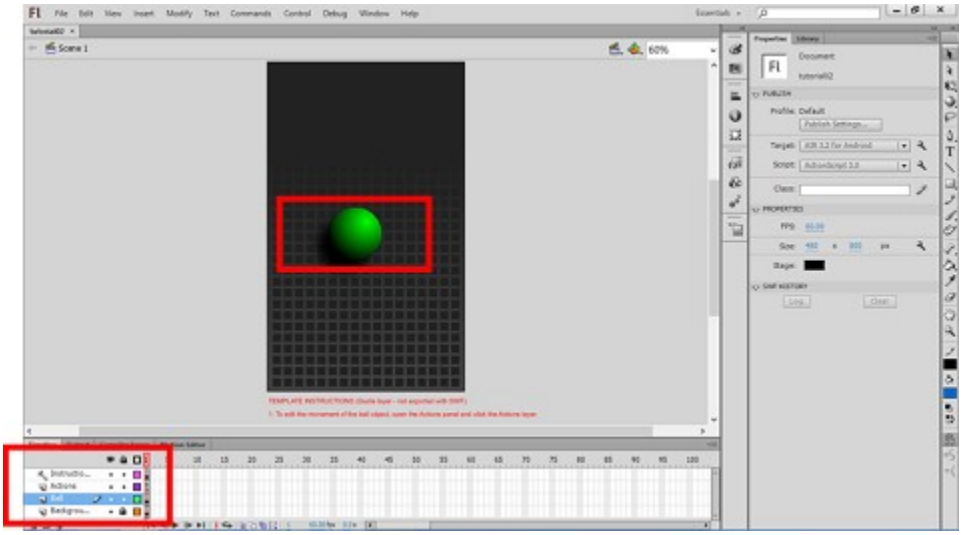


3-2. Using Action Scripts

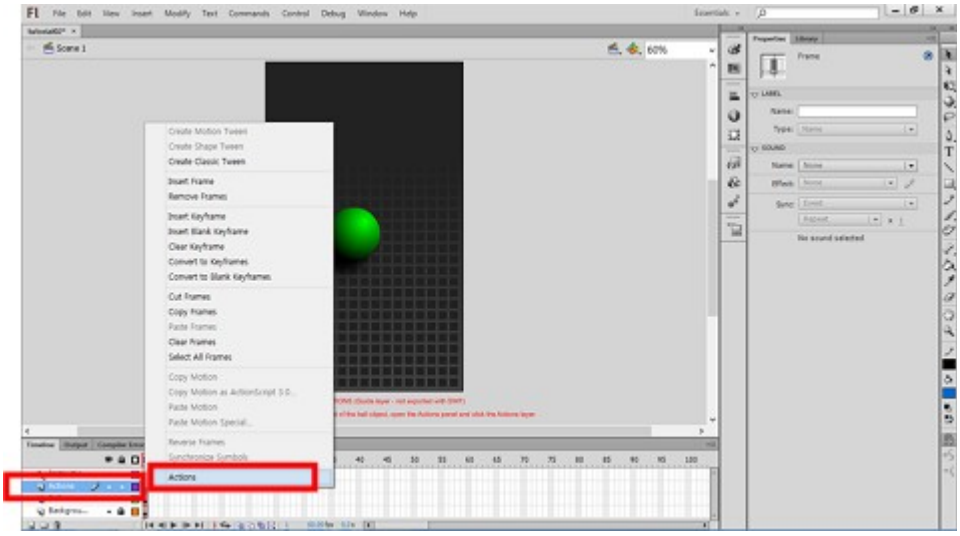
Let’s try writing an Android App that uses the device's accelerometer by using the templates provided in the Flash tool.
Run the program, select “AIR for Android” in the “Create from Template” screen, and select Accelerometer under Templates.



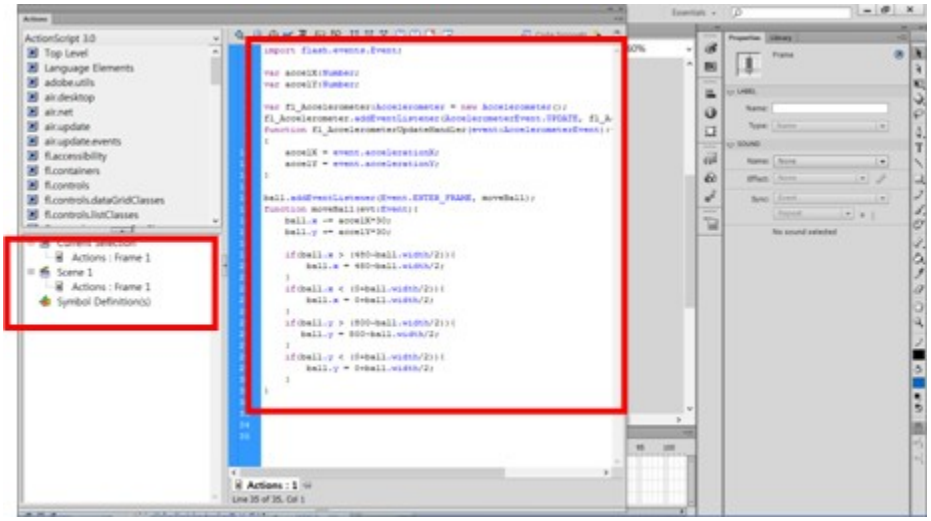
In the next screen, there is a green ball object, and the Background and Action layers are added to the timeline and scene workspace as shown below.



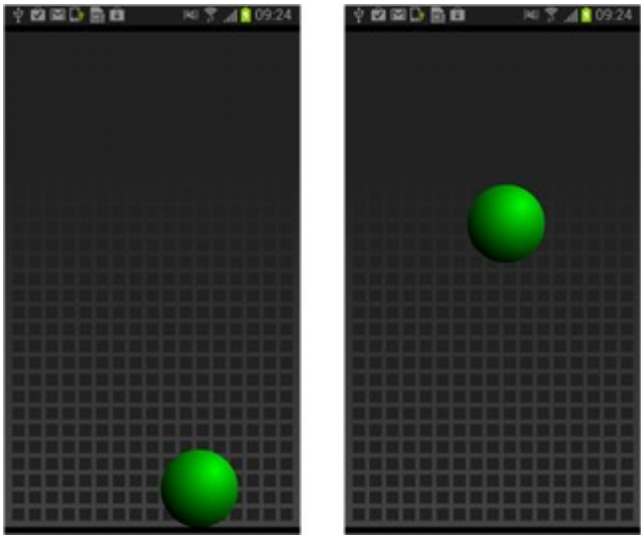
Select the Actions layer, and choose Actions in the submenu, then the Action Script Editor appears on the screen.



The script included in this sample consists of code to update the X and Y coordinate information by receiving the values of the device’s accelerometer by using the Accelerometer class object and EventListener API provided in the flash.sensors and flash.events packages.



When you publish your app and run it on an actual device, you can see that the position of the ball changes depending on the tilt and position of the device.



3-3. Using the Device's Specialized Features, Such as Gyro Sensors Using ANE

While most contents can be created in Flash and action Script, using the device's specialized features, such as gyro sensors and external libraries for things such as in-app billing, requires additional technologies and tools.
ANE, which stands for AIR Native Extensions, supports specialized device platform features and direct access to the API interface.
This means that by using ANE, you can access the specialized features and resources of the device directly from AIR.

You can download and test samples of various features (such as the gyroscope, notifications, and vibration) and tutorials from the Adobe Developers website.
http://www.adobe.com/devnet/AIR/native-extensions-for-AIR.html.
Also, you can create custom ANEs that you need and add them to a project. More information regarding creating ANEs is available from the Adobe Developers website.