CMER Centre for Mobile Education and Research

Developing Apps for the BlackBerry PlayBook with Adobe AIR
Lab # 1
Getting Started with Flash Builder and ActionScript 3.0

Introduction
The objective of this lab is to give you the basic information required to begin developing applications for the BlackBerry Playbook using ActionScript 3.0 and the Flash Builder 4.6 IDE. In this lab, we'll be dealing with creating, testing, and building your application.

A small self-test is included at the end of this lab to test your understanding of its contents

Software Requirements:
- Flash Builder 4.6 (http://www.adobe.com/products/flash-builder.html)
- Adobe Flex SDK (Included with Flash Builder)
- PlayBook SDK for Adobe Air 2.0 (https://bdsc.webapps.blackberry.com/air/download/sdk/)
    -PlayBook SDK must be integrated into Flash Builder. Choose to do so when asked during its installation
-PlayBook Simulation Environment
-PlayBook Deployment Tools for Mac (included in Lab 2 file)
-Updated Java JRE Enviroment (http://java.com/en/download/index.jsp)

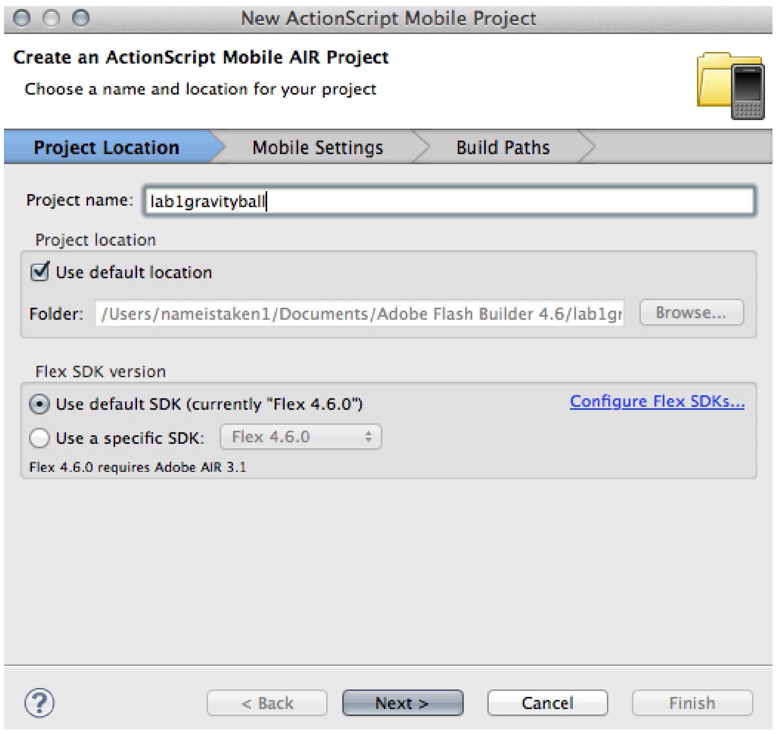
Note: This guide can also be used with Flash Develop (http://www.flashdevelop.org/), which is an open source alternative to Flash Builder. The code presented here will work with both. However, interactions with the IDE will differ, and because it is more user friendly – Flash Builder will be used. If you choose to use Flash Develop, the Flex SDK can be found here: http://sourceforge.net/adobe/wiki/Projects/

This guide assumes you have set up your development platform correctly, and are familiar with the basic functions of your IDE. It also assumes you have the necessary tools to build and sign PlayBook applications. For more information on this, refer to: https://bdsc.webapps.blackberry.com/air/

Start a new project
Create a new *ActionScript Mobile Project* navigating to:
File > New > ActionScript Mobile Project

1. Specify a project name
2. Select an SDK (recommended to choose Flex 4.6.0)

3. Select Next to manage your mobile settings

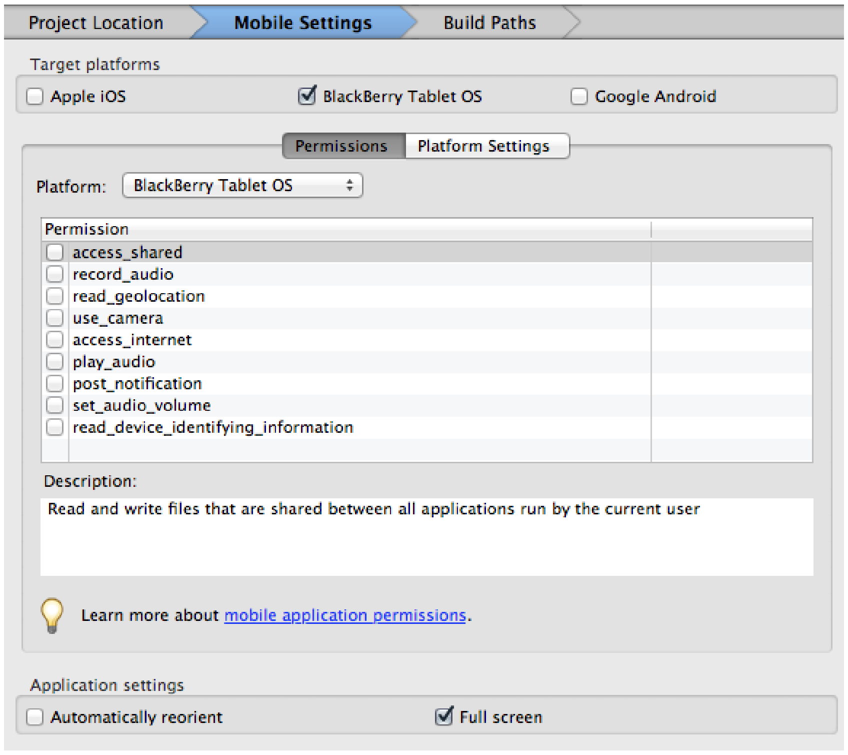

New ActionScript Mobile Project
Create an ActionScript Mobile AIR Project
Choose a name and location for your project

Project name: lab1gravityball
Project location
☑ Use default location
Folder: /Users/nameistaken1/Documents/Adobe Flash Builder 4.6/lab1gr

Flex SDK version
● Use default SDK (currently "Flex 4.6.0")

○ Use a specific SDK: Flex 4.6.0
Flex 4.6.0 requires Adobe AIR 3.1

Figure 1: Starting a new project



Project Location | Mobile Settings | Build Paths

Target platforms

Apple iOS [✓ BlackBerry Tablet OS] Google Android

Permissions | Platform Settings

Platform: BlackBerry Tablet OS

Permission
access_shared
record_audio
read_geolocation
use_camera
access_internet
play_audio
post_notification
set_audio_volume
read_device_identifying_information

Description:

Read and write files that are shared between all applications run by the current user

Learn more about mobile application permissions.

Application settings

Automatically reorient [✓ Full screen]

4. Select the *BlackBerry Tablet OS* Platform. (If this option is unavailable to you, it is because your PlayBook SDK was not installed correctly. Reinstall it, and remember to select “Implement into Flash Builder”)

5. Deselect *Automatically reorient*. This lab will not implement that feature. In future labs, however, we will create dynamic graphical assets that will work with the reorient property.

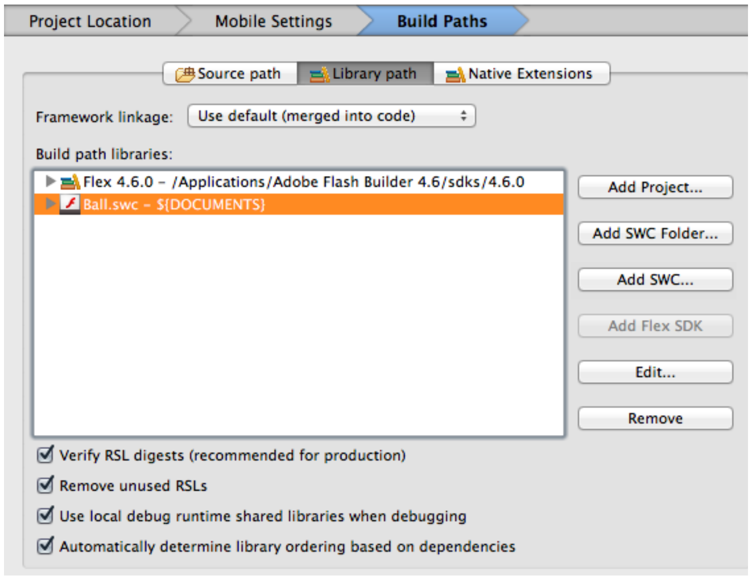
6. Select *Full Screen*. This will allow the app’s boundaries to exactly match the screen boundaries.

Figure 2: Mobile Settings

7. Take note of the permissions. The description below gives you a basic overview of what each permission allows. Although we will not be using any permissions for this lab, they are key to making use of the PlayBook’s hardware.

8. Select Next to select custom *Build Path*s. We will be using a referenced library for this lab.

Definition from Wikipedia.org:
*An **Adobe SWC** file is a package of precompiled Flash symbols and ActionScript code that allows a Flash or Flex developer to distribute classes and assets, or to avoid recompiling symbols and code that will not change. SWC files can be generated by the Flash authoring tool, and by Flex. They are sometimes referred to as class libraries.*



Project Location | Mobile Settings | Build Paths

Source path | Library path | Native Extensions

Framework linkage: Use default (merged into code)

Build path libraries:

Flex 4.6.0 – /Applications/Adobe Flash Builder 4.6/sdks/4.6.0
Ball.swc – ${DOCUMENTS}

[✓ Verify RSL digests (recommended for production)]
[✓ Remove unused RSLs]
[✓ Use local debug runtime shared libraries when debugging]
[✓ Automatically determine library ordering based on dependencies]

Figure 3: Build Paths

9. Add Ball.swc, included in lab1.zip.

This .swc includes a MovieClip, called Ball, which contains a symbol of a gradient shaded ball of 100px by 100px. It is exported using the linkage class Ball.

It is considered good practice to have all your Classes and Libraries start with an uppercase letter, to differentiate them between packages and variables.

In this window, you can also set the Application ID (or name). Click finished when your ready, and Flash Builder will automatically generate the necessary xml and descriptor files.

Creating Your First Application

Hello World is boring. So, instead, we'll be designing balls that move on stage, being affected by gravity and stage boundaries.

ActionScript 3.0 is an object oriented programing language (OOP). Using OOP languages at first can be overwhelming, and complicated, but they are a very efficient, modular, powerful, and clean way to code. We'll be taking advantage of ActionScript's OOP capabilities, and be creating packages and classes.

Look at your package explorer in Flash Builder (to the left of your code editor). You'll notice that Flash Builder automatically created a default package with a class that has the same name as your project. This is your document class (your main function, main object, index, or home – but in ActionScript 3 it is referred to as your document class). This class will automatically instantiate when your program starts, so we'll be using it to instantiate our other classes, as well as call all our functions.

Not yet though, let's get started by creating the hierarchy that will be our program. Create a new package (*File > New > Package* **or** *Right click on project > New > Package*) inside the */src/* folder in your project and name it **ballpack**.

Now create a new ActionScript Class inside your *ballpack* package, called **BallCode**.

Note that it is considered good practice to name your packages will all lowercase, and have the first letter of your classes a capital. This is to help you differentiate between packages and classes when working with your libraries.

When you're done, your project should look something like Figure 4.

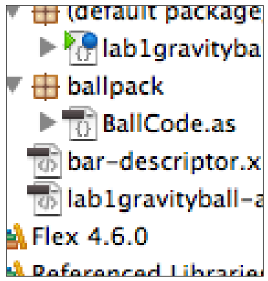


Figure 4: File structure

Place the code from BallCode.txt into your BallCode class.
Refer to the comments (in the code) and appendix for explanations on how it works.

BallCode.as Appendix:

Package – tells the compiler what package the class is a part of.
Import – tells the compiler what libraries this particular class requires and will use to run
Public class – this class can be publicly accessed/used/ and instantiated by other classes

Private var – variables that can only be accessed/modified by this class, because they were declared outside of a function and inside the class, there scope is the whole class.
Public function ball code – this is the constructor function. This function is automatically run when this class is instantiated.
public function updateballVelocities(b:Ball):void – This function accepts a variable of type Ball() as a parameter, and returns no value
private function – A function that can only be called privately. I.E. Only by this class.
degree to radians – Most ActionScript properties accept radians as values. To make this code more readable, I've decided to use degrees, and simply convert it to radians before I gave the value to the property.
move() and addBall() are functions that will be used by our document class.

Place the code from documentclass.txt into your document class, or main file.
Refer to the comments (in the code) and appendix for explanations on how it works.
lab1gravityball.as Appendix:
swf header – Very important line. This declares all the major properties of your application.

Done. But does it work? Let's see.

Test Your Application using the PlayBook Simulator
To run your app in the simulator, right click your project (the main folder with the phone on it) and select *Run As > Configurations*

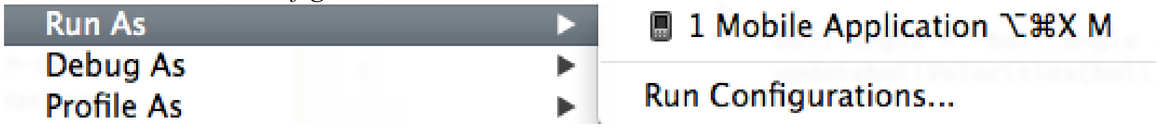
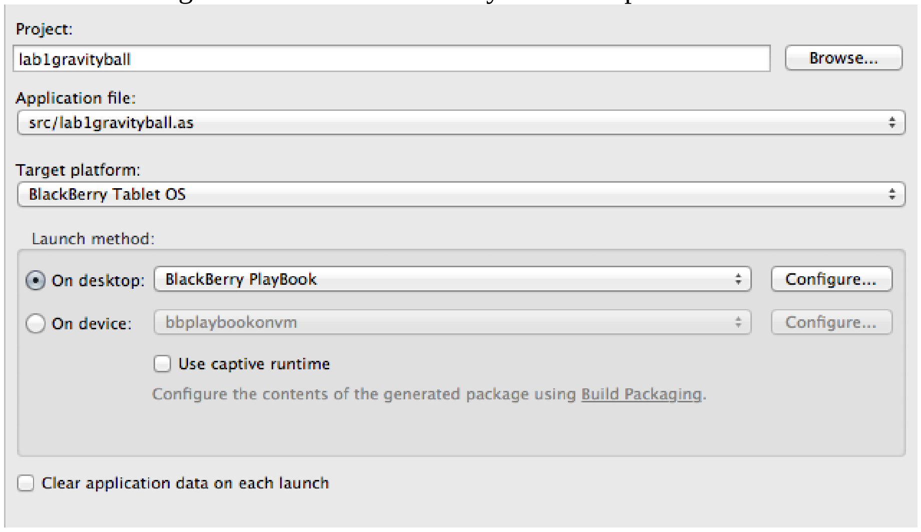


Figure 5: Run As...

Figure 6 shows the settings to run the simulator on your desktop.



Project: lab1gravityball
Application file: src/lab1gravityball.as
Target platform: BlackBerry Tablet OS
Launch method:
[✓ On desktop]: BlackBerry PlayBook
On device: bbplaybookonvm
Use captive runtime
Configure the contents of the generated package using Build Packaging.
Clear application data on each launch

Figure 6: Running the desktop simulator

Or if you are using VMPlayer/VMFusion to simulate (recommended), then start up VM and boot the PlayBook OS. Use the settings in Figure 7.

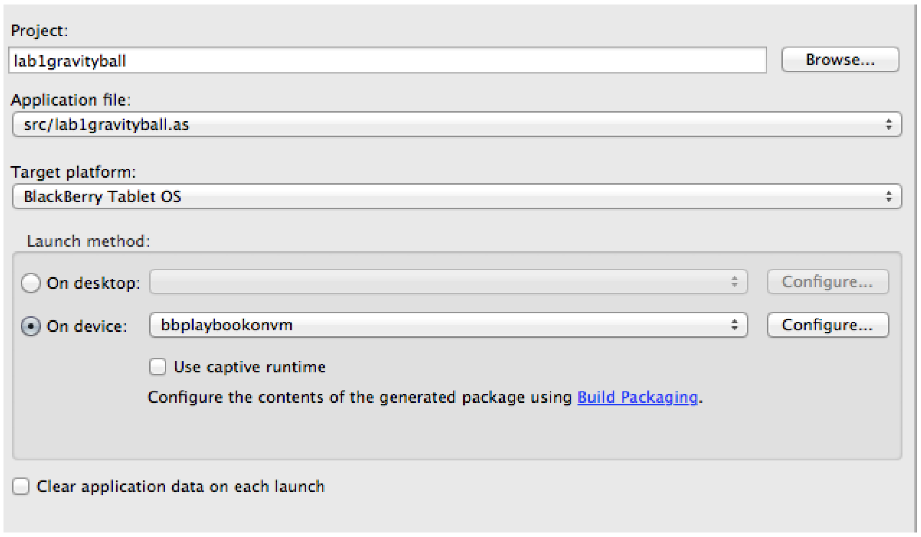

Project:
lab1gravityball
Application file:
src/lab1gravityball.as
Target platform:
BlackBerry Tablet OS
Launch method:
On desktop:
On device: [✓] bbplaybookonvm
Use captive runtime
Configure the contents of the generated package using Build Packaging.
Clear application data on each launch

Figure 7: Running the VM Simulator

Congrats. Your application should run. If not, check the compiler's error console for more information.

If you're happy with the way your application runs, then let's move on to the final part of this tutorial.

Build Your Application

To export your file into a .bar…

1. *File > Export*
2. In the *Flash Builder* folder, select *Release Build*.
3. Select *Next*.
4. Choose "BlackBerry Tablet OS" as your target platform.
5. Select a destination folder
6. Choose to Export as:
   Signed packages for each platform
7. Select *Next*
8. Select *Enable Digital Signing*
9. If you have yet to configure Flash Builder to automatically sign your apps, please click the *Signing* link to configure your author tool, other wise select Finish, and watch as your code is transformed into a PlayBook executable.

Now that your application has been signed, it can be be deployed to your Playbook and tested on a real device.

Deploying You Application

Export and sign your application. Refer to the instructions in Lab 1, if you need help.

Power on your PlayBook Device, and turn *Development Mode* on. To do so:
Touch *Settings > Security > Development Mode*
Remember your password, and IP Address.

Connect your PlayBook Device to your computer, and enter your password to grant access.

Put your exported *.bar* file, inside the Playbook_Tools folder included in the Lab 2 folder.

I highly recommend you read the included READ ME_INSTRUCTIONS file. It includes some important information.

Open *MacDeployer.app*, and follow the on screen instructions.

Test Yourself

Complete the following to test your understanding of the code:

1. Change the swf's frame rate
2. Create 5 balls
3. Slow your balls down
4. Create a function to remove a ball
5. Change the stage boundary to half the width