The 8th International Scientific Conference
eLearning and software for Education
Bucharest, April 26-27, 2012
10.5682/2066-026X-12-115

TRADE-OFFS IN DEVELOPING HIGHLY INTERACTIVE MLEARNING CONTENT USING THE CURRENTLY AVAILABLE DEVELOPMENT PLATFORMS

Mihai HAZI, Adrian BARBU, Adriana MĂLUREANU
ASCENDIA Design, Bucharest
E-mail: mihai.hazi@ascendia.ro, adrian.barbu@ascendia.ro, adriana.malureanu@ascendia.ro

***Abstract**: The current day smart-phone is packed with a great assortment of sensors (ambient light, proximity, GPS, accelerometer, compass, gyro) and input modalities (touch, camera, microphone). This set of capabilities has spurred tens of thousands of developers to publish hundreds of thousands of mobile applications. The apple app store and the android market sum a total app download count of over 20 billion. How many of these apps have an mLearning focus and what sort of capabilities do they currently employ? What are some successful use cases extracted from this short market analysis? Considering the rapidly expanding mobile software and hardware space e-learning also has to adapt and move to this new medium. Most of the highly interactive desktop e-learning content has been traditionally developed using Adobe Flash. Does Flash still hold up in the mobile context or do the developers have take up HTML5 and/or native development for this type of content? This paper strives to give an answer to these questions by performing a short app markets analysis and developing an example app for a highly interactive use-case using three combinations of platforms and tools: flash/air, native and a mixed solution. The stumbling blocks encountered, and trade-offs made are expanded upon. Factors like the relative ease of implementation, the speed of deployment, multi-platform reach and app performance are also analyzed and trade-off points identified. As a final step conclusions are drawn as to the best path to choose for present day versus future development of highly interactive mLearning experiences.*

***Keywords:** smart-phone capabilities, mobile development trade-offs*

I. INTRODUCTION

The paper presents the point of view of an e-learning software developer specialized in using the flash platform tools on the challenges and opportunities of crossing over from developing content for the desktop to the mobile platforms (smart-phones, tablets). This transition also implies a rethinking of the way one packages content: content for the flash player running in the browser for the case of a PC to content packaged as a mobile application (app) targeted for a specific platform.

The focus is placed on the market analysis and performance comparison of highly interactive applications - applications that use many of the current day smart-phone's capabilities and sensors. Typical uses for such categories of applications:

- User interface control
- Gaming
- Simulation
- Image stabilization
- Navigation
- Augmented reality [4]

II. CAPABILITIES USAGE ANALYSIS

In the first part of the paper we look at the strengths of mobile devices and to what extent they are currently used in the applications developed for them.

2.1. SWOT analysis of mobile devices

Strengths	Weakness
- Portability, pocket-ability; - Help to contextualize the learning experience [2]: -Communications -Spontaneous -Geo-sensitive -Short periods of use -Focused activity	- In the case of mobile phones screen real-estate limits the possibility of directly porting PC based course-ware. - More limited hardware resources/ flash player unavailable in iOS.
Opportunities	**Threats**
- Exponential growth of the smart-phone market. - By early 2013 smart-phones running Android will reach the 70$ price point.	- General trend is to drop FlashPlayer support across all mobile OS's - The acceptance of AIR compiled application depends of ecosystem owner policies (and, moreover, good will)

2.2. APPS - used capabilities analysis

The following table shows the result of a short analysis of the top apps in the android market for the Games, respectively the Education categories (both paid and free) from the perspective of the capabilities and sensors of the smart-phone that were used.

Table 1 - apps capabilities chart

Application subgroup	Application name	Sensors used	Required permissions
Games - paid			
	Draw something	Acceleration	Location Internet Control vibrator
	Where's my water	Acceleration	Internet
	Doodle jump	Acceleration	Internet Control vibrator
	Grand Theft Auto 3		Internet Bluetooth connections Control vibrator
	Minecraft		Internet Control vibrator
Games - free			
	Ninja chicken		Internet
	Ceramic destroyer		Internet
	Parking frenzy		Location Internet
	Spanzuratoarea		Internet
	Drag Racing		Internet Control vibrator

Education - paid			
	Star chart	Acceleration Magnetic field	Location Internet
	Color & draw for kids		Internet
	Monkey preschool lunchbox		
	Kids learn to read		Internet Control vibrator
	Toddler tapping zoo		
Education - free			
	Noul Cod Rutier		Internet
	MyPlay Chef HD		Location Internet
	Chestionare Auto		Internet
	50 languages		Record audio Internet Control vibrator
	How to draw - easy art lessons		Internet

A conclusion of this short capabilities analysis is that the most popular applications in the games and in the education categories under-utilize the capabilities of current day smart-phones.

A good example of an application with high educational potential and that makes good use of a smart-phone's capabilities is “Smart Tools”: A set of tools that enable the user to measure length, angle, distance, height, direction, sound and detect nearby metals by making use of the following phone capabilities:

- Camera
- Microphone
- Location (fine) – GPS
- Accelerometer
- Magnetic field
- Orientation sensor

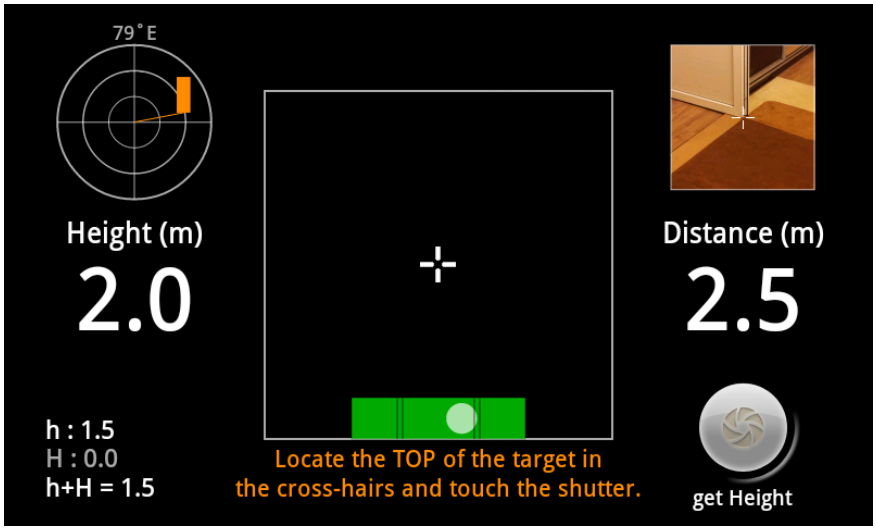

Figure 1. SmartTools in action

Figure 2. SmartTools – Sound Meter and Metal Detector

III.TRADE-OFFS BETWEEN ADOBE AIR AND NATIVE APPLICATION DEVELOPMENT

In the following paragraphs we look at what set of capabilities are available to both flash/AIR and native developers and then at the ease of implementation, ease of distribution and app performance for the two main development options identified.

3.1. Availability of mobile device capabilities to developers

The device capabilities directly available at the AIR API (Application Programming Interface) are currently only the accelerometer and the geolocation.

Beginning with AIR 2.7 for mobile Adobe has introduced the ability to extend the AIR runtime with code libraries that contain native code wrapped with an ActionScript API. This has opened access to native platform features not supported by AIR.

As a result, developers in the flash community have made publicly available Native Extensions for AIR [1] for accessing the following device capabilities (both Android and iOS):

- Proximity sensor
- Gyroscope
- Control device vibration

However, at present, there remains a set of capabilities and sensors that only native *android* developers can access:

- Ambient temperature
- Gravity
- Light

- Magnetic field
- Pressure
- Relative humidity

3.2. Ease of implementation

Adobe AIR cross-platform development

From the perspective of a developer with experience in using the flash platform for desktop web development, transitioning to mobile development requires the following tools: Flash Builder 4.5 with AIR SDK 2.7 or Flash Professional CS 5 with the latest player updates from Adobe. For accessing the speed improvement brought by the direct GPU rendering model (Stage3D) on mobile devices, AIR SDK 3.2 is needed, together with a framework that makes accessing the Stage3D API's simple - the Starling Framework has been proposed by Adobe for this purpose.

These tools and frameworks provide a gentle learning curve to the flash platform developer for transitioning to mobile content development.

As part of the development process, application debugging and profiling on the device is very important. Because of the less strict Android policy for installing unknown/unsigned apps on devices, a compiled .apk package (from Flash Builder) can easily be tested on a variety of Android devices.

iOS native development

A developer transitioning to highly interactive content development for iOS has a steeper learning curve ahead of her. The development tools and frameworks needed are: XCode suite (available only on MAC OS X) and a 2D game development framework for Objective-C (Cocos2D has the largest community/learning resources available; while Sparrow is a game engine with an API much more familiar to flash developers [3]).

As an intermediary solution, for developers that do not work on a MAC, the native iOS toolchain can also be run on a VMware image on a PC, but with a serious speed drawback when running the app on the XCode simulator.

For the debugging process XCode native development currently has the advantage that it offers live debugging on the device, and the ability to monitor in real time the entire iOS operating system on the device (it is not currently possible to do live debugging of a cross-compiled AIR application for iOS).

3.3. Ease of distribution

Being a cross platform tool, a single flash builder project (a single codebase) can be compiled and packaged for multiple platforms: AIR for iOS, AIR for Android, AIR for desktop.

3.4. App performance

As a result of in-house testing, AIR 3.2 Stage3D direct mode rendering Starling performance is double that of AIR 2.7 GPU rendering and up to 75% of the performance of native iOS rendering.

IV. CONCLUSIONS

From the point of view of mobile device capability availability, the Adobe AIR solution poses no restrictions to developers. With the exception of the Bluetooth connectivity (requested by the GTA3 app) all other capabilities or sensor data are currently accessible to the flash developers either directly through the AIR API or through a native extension.

At the moment, considering the rendering performance boost brought by the latest AIR SDK (3.2), the relatively short mobile game development time, and the fact that Adobe refocused their runtime and tool development efforts on cross-platform high-performance games and on high performance video for the web, the flash platform continues to offer significant value for the development of highly interactive mLearning content.

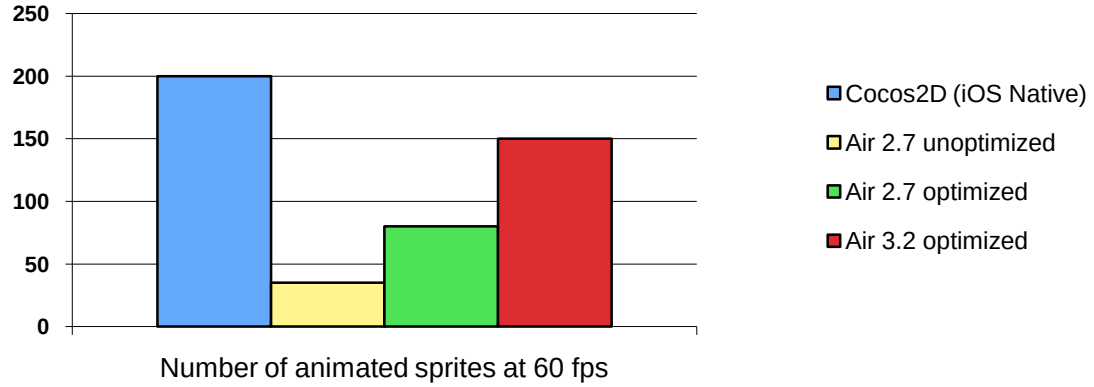

Figure 3. A comparison chart of rendering performance using native apps or several iterations of Adobe AIR framework

References

[1] Adobe Flash Platform Help, Introducing native extensions for Adobe AIR, http://help.adobe.com/en_US/air/extensions/WSb464b1207c184b14-70ccb6bd12937b4f2d6-8000.html

[2] thenextweb.com, Mobile Apps: a look at what makes a good app great, http://thenextweb.com/mobile/2011/07/16/mobile-apps-a-look-at-what-makes-a-good-app-great/

[3] Sparrow framework, Sparrow vs Starling, http://forum.sparrow-framework.org/topic/sparrow-vs-starling

[4] Augmenting the reality of learning, http://www.dashe.com/blog/augmented-reality/augmenting-the-reality-of-learning