A Novel Concept of a Cartoon-Generator Application on a Mobile Phone

doi:10.3991/ijim.v4i2.1266

J. Stergar[1], A. Šulić[1] and B. E. Rodrigues Silva[2]
[1] University of Maribor, Maribor, Slovenia
[2] Instituto Politécnico do Porto, Portugal

***Abstract*—In this work the development and deployment of a new concept of cartoon-generation on a mobile phone is presented. An interactive multimedia application was designed for a push-approach of sending short e-cartoons from a mobile phone to a central WEB server. The concept of a unique cartoon-generator will be discussed. The multimedia Authoring Environment Flash was used for the interactive application design and implementation. The architecture and the design process of the interactive cartoon-generator application will be presented including storyboarding with final testing of the application on a Nokia mobile phone.**

***Index Terms*—content generation, Flash Lite, multimedia authoring, mobile phones, interactive multimedia application.**

I. INTRODUCTION

With the advent of powerful new mobile phones running on more advanced networks, wireless carriers, content developers, and handset manufacturers are bringing to market a range of new content and devices. Mobility companies see great opportunities but also face significant challenges in deploying these products [1].

Key among these challenges is the ability to quickly and easily create, deploy, and manage content and applications ranging from the mobile phone user interface to vibrant and user-demanded rich applications. Worldwide mobile phone shipments reached over 1.19 billion units in 2009. It is critical that wireless carriers, content developers, and handset manufacturers have the tools they need to quickly respond to challenges of this complex and growing mobility market [2].

The inclusion of a proven tool such as Adobe Flash CS3 with improved Flash Lite capabilities will assist in lowering the technological barriers to creating dynamic mobile applications and real-time content. Flash Lite also benefits from leveraging an existing Flash development community of over 1.3 million Flash developers worldwide [3].

With only minor adjustments to accommodate the unique specifications and needs of handsets, developers already skilled in Flash can easily bring their skills and expertise to the mobile space. By allowing for an easy transition from Flash development on another platform to Flash development in the mobile environment, Flash Lite further shrinks the barriers to adoption. The key barrier among these is the substantial difficulty involved in efficiently creating time-sensitive content that fully leverages the mobile phone's unique and powerful always-on, always-connected capabilities. While WAP and Web browsing functionality is increasingly prevalent on mobile phones, those platforms severely limit the form and type of data capable of being displayed and therefore slow adoption by consumers and enterprises alike. Instead of a rich platform complete with interactive multimedia, background downloading, and applications that integrate and respond to user-configured data, the mobile phone is still, by and large, a statically driven, one-dimensional platform with limited navigation and display options as well as exasperating data usage experiences [2].

Flash Lite allows for the efficient and rapid creation and deployment of content and interfaces to mobile phones, enabling mobility companies to customize their devices with powerful user interfaces, content, and applications. Flash Lite also enables over-the-air management of content, creating the possibility for the dynamic creation and modification of new content and campaigns even after the device has been purchased – further increasing the ability of the carrier to differentiate itself from its competitors [4], [5].

II. THE MULTIMEDIA AUTHORING ENVIRONMENT

Adobe Flash Lite is a lightweight version of Adobe Flash Player, a software application published by Adobe Systems. This version is intended for mobile phones and other non-phone, portable electronic devices. It allows users of these devices to view multimedia content and applications developed using Adobe's Flash tools, which had previously been available only on personal computers.

Flash Lite is a development technology implemented at the client-side, or user interface layer. In this realm Flash Lite competes with other technologies like Qualcomm's uiOne markup language and Sun's JavaFX Script. Recent changes to ActionScript allow Flash Lite to better integrate with and even compete with device-layer technologies like Java ME and BREW. Flash Lite should not be considered a mobile operating system like Symbian OS, Windows Mobile, Mac OS X for mobile: it is a technology for developing applications that run on a mobile operating system [6], [8], [10].

A. *Arguments for Flash Lite*

Using Flash technology on a mobile phone provides the same advantages as using Flash on the desktop. Rich, interactive, compelling user experiences can be created that have a consistent display across a range of platforms. Developing applications with Flash Lite can also result in

a quicker time to market and lower developer costs than using J2EE or C++ [1].

One of the major strengths of the Flash platform on the desktop is the one million-plus developer community that has years of experience designing and developing user interfaces, games, animations and e-learning applications.

B. The used version of Flash Lite

Flash Lite 1.1 supports Flash 4 ActionScript. Flash Lite 2.0 and 3.0 support Flash's ActionScript 2.0. All three versions also support the World Wide Web Consortium's Standard SVG Tiny, a mobile profile of the consortium's Scalable Vector Graphics (SVG) recommendation. Unlike SVG, Flash Lite can add audio and interactive elements without the use of other technologies such as JavaScript. As with Flash, Flash Lite is able to read and redraw external XML content.

C. Features and Functionality

Flash Lite offers the same timeline based features that are implemented in the regular PC based Flash platform: MovieClips, Buttons, Vectors, Gradients, Bitmaps, User Input, digital audio and Scripting language.

As this version has been designed for use on mobile devices that have much less processing power and memory than desktop computers, it has a reduced feature set when compared with Flash 7 and upper. In fact, Flash Lite is based on the Flash 4 scripting engine. Macromedia has gone for distribution rather than functionality with the release of version 1.1.

One of the features of version 1.1 is the ability to access native device properties. From within the designed application it is possible to: get battery level status, send SMS messages, get network connectivity status, dial phone numbers and launch other applications.

These new capabilities enable content creation that can interact with the host device giving the users a much better "mobile" experience.

One of the most important features of Flash Lite 1.1 is the ability to send and receive data over HTTP. This enables to load data (and SWF's) into applications from a web server, giving the benefit of dynamic content in an installed application.

D. Flash Lite 1.x ActionScript

ActionScript is used to add programming logic and interactivity to Flash Lite applications. The version of ActionScript in Flash Lite 1.0 and Flash Lite 1.1 software from Adobe - referred to collectively as Flash Lite 1.x ActionScript - is a hybrid of Adobe's Adobe Flash 4 ActionScript, plus additional commands and properties specific the Flash Lite player, such as the ability to initiate phone calls or text messages, or get time and date information from the device.

Flash Lite 1.x ActionScript consists of the following parts:

Flash Player 4 ActionScript: This includes operators (for example, comparison and assignment operators), movie clip properties (for example, _height, _x, and _y), Timeline control functions (for example, gotoAndPlay() or stop()), and network functions, such as the loadVariables() and loadMovie() functions (Flash Lite 1.1 only).

Phone integration commands and properties: Flash Lite provides commands that let you, for example, query the

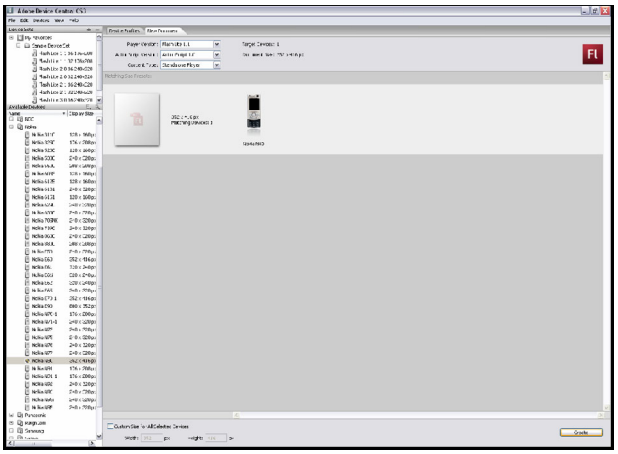

Figure 1. Choosing the device parameters in Adobe Device Central.

date and time information from the device, initiate a phone call or short message service (SMS) text message, or start external applications installed on the device.

Platform capability variables (Flash Lite 1.1 only): These properties provide information about the capabilities of the device or Flash Lite runtime environment. For example, the _capLoadData variable indicates the applications capability to load data over the network.

Host environment communication function: The *fscommand2()* function provides enhancements to *fscommand()*, including the ability to pass an arbitrary number of arguments and to retrieve immediate return values (rather than having to wait until the next frame, as with *fscommand()*. Like the *fscommand()* function, the *fscommand2()* is used to communicate with the host environment or system – in this case, the mobile phone or device.

III. THE FRAMEWORK FOR APPLICATION DEVELOPMENT AND TESTING

It's crucial that the tools to create content and experiences on mobile devices are being delivered to both designers and programmers, alike as suggested by [12]. The Adobe Flash CS3 was used as the multimedia authoring environment. It is the first version of Flash since Adobe acquired Macromedia. This is significant because it is the first version of Flash that is integrated tightly and intuitively with the other tools that designers of digital experiences have been using all along – specifically Photoshop and Illustrator.

The authoring environment still shows these roots by making any designer feel immediately at home. The tools that are most immediately apparent obviously enable the creation of visual elements. Flash is also a comfortable environment for developers. The Actions panel where developers write their ActionScript code is newly improved.

Creating and testing mobile content has one key difficulty that desktop software doesn't – the large quantity of devices that can be targeted and therefore must be tested. Adobe has provided a Flash emulator the so called "Adobe Device Central" to greatly help creators of mobile content all the way through the life cycle of a mobile application (Figure 1.). Designers and developers can now test their mobile content of these types: Flash Lite,

raster images (bitmaps of different types), mobile web, as well as video content targeted at mobile devices.

Choosing the target device, previewing layout, monitoring memory usage, and previewing the performance that can be expected on the target device are all features of Adobe Device Central (Figure 4.). Since the mobile market moves so quickly, Adobe publishes device profile updates on a regular basis.

IV. THE DESIGN OF THE INTERACTIVE CARTOON-GENERATOR APPLICATION

A. *The Environment Setup*

In the beginning of the application design process for the Flash Lite application one has to decide which handsets will be targeted. Flash technology has been shipped on over 400 millions of Nokia phone across the globe. Flash technology has become broadly available, because it is supported across all Nokia platforms and is widely integrated with Nokia's mobile WebKit browser. That was one of the key elements in choosing our mobile device [4].

Mobile phones vary quite extensively in features and input methods. Therefore we should emphasize the following:

- screen size,
- input devices (five-way navigation, touch screen, etc.),
- processor speed (important for animation and complex graphics) and
- sound support.

Mobile phones usually lack a keyboard or mouse therefore the interface had to be designed in a way that is intuitive and easy to use with the input methods supported by the respective device. Obviously navigational keypads are the proposed solution.

Flash Lite can take input from joysticks, touch screens and navigational keypads. Also functions can be assigned to key press events (0-9,* and # keys). There is also the possibility to map "Soft Keys" present on some mobile phone types, to key press events.

Thus, after this few assumptions the setup (main target device from Adobe Device Central) and development of the Application can be started. This is the first step on the development of any application.

B. *The Storyboard*

As stated in [7] storyboards are a natural representation and they can be used to simulate functionality without worrying about how to implement it. When designing a visual interface, rough pictures of the screen layouts are sketched to design the concepts of functionality and interactions. The screens are then usually tied together by storyboarding techniques: the designer annotates the

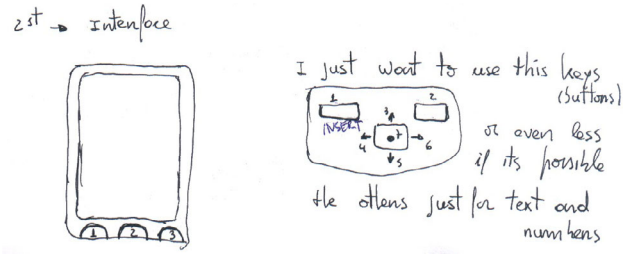

Figure 2. A cut out of the storyboard sketching process.

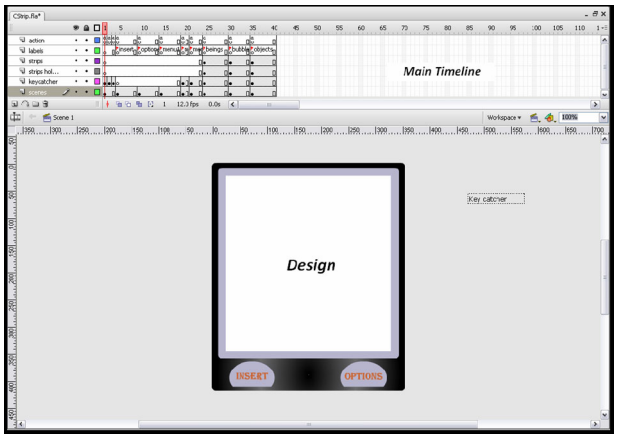

Figure 3. Main timeline with emphasized "Keycatcher"layer.

sketches to illustrate sequences of system responses to end-user actions. Despite the process can be performed by suitable tools we used a standard approach when designing the user interface (Figure 2.).

C. *Verical Design and Structure – The Layers*

The main timeline has six different layers (Figure 3.):

- Action Layer
- Labels Layer
- Cartoon and CartoonHolder Layers
- KeyCatcher Layer and
- Scenes Layer

1) *The "Action" Layer*

In this layer all of the actions (scripting language ActionScript) that occur on the main timeline and that make the work interactive are integrated. Thus here the variables are initialized counting the numbers of cartoon characters. The "fscommand2" function is executed (systems calls) to put the application on full screen. Also Soft Keys are configured in the Action Layer.

2) *The "Labels" Layer*

Here the names for groups of frames intended to be managed are identified. Movement in the timeline can be controlled by other means for example with frame numbers. Nevertheless with group frames the work stayed more intuitive and a better overview is achieved.

3) *The "Strip Cartoon" Layer*

With strips we'll be referring different cartoon characters used in the carton-generator application. In the Strip Cartoon layer the strips are implemented for selection of different characters: the beings, the bubbles and the objects. One can see these strips animated here almost as they will appear live in the "main scene" of the running application after selection.

4) *The "Strips Holder" Layer*

In this layer all the strips exactly as in the "Strips layer" are included. The major differences are the size and the position that differ, because this layer has a different function. When the user selects any cartoon characters from the Strip Cartoon layer any of the MovieClip instances is duplicated and forced to appear on the main scene. With other words, the Strip Cartoon layers function is to show the strips to the user in an optimal way (screen size) whereas the function of the Strip holder layer is to

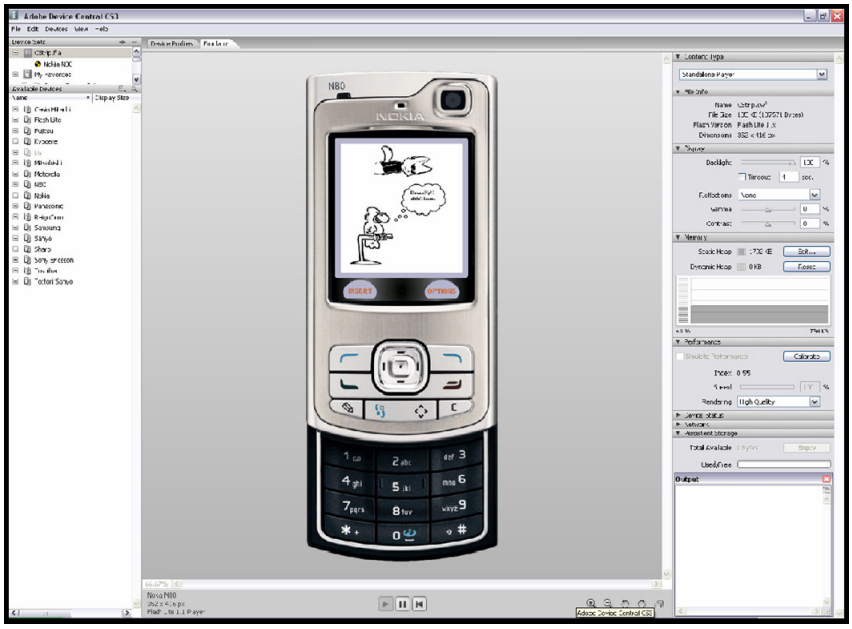

Figure 4. Testing the application on the Adobe Device Central

duplicate the selected strip cartoon in the sense of real appearance.

5) The "Keycatcher" Layer

In the "KeyCatcher" layer all the key catchers are included. The necessity of using the key catchers is for the reason that different situations can occur in the running application. Therefore the same key could have different meanings in different scene scenarios of the application.

6) The "Scenes" Layer

Here the design part is more visible. Different kind of scenes for the different cartoon scenarios along the main timeline appear in this layer. All included strips are intended for the main scene with menus and selection scenes.

D. Horizontal Design and Structure – The Frames

In the foregoing section the main timeline in the vertical way was explained, thus layers were the main topic. In this subsection the frames and group of frames on the timeline will be discussed (Figure 3.).

The only frames in the application not having label names are the first three. They are essential for the different application states which will be explained next.

First the similarities – the design is the same for all three frames. It is constituted from the border that does not change in all scenes of the application and two buttons – one for the insertion of strips and the other one for the belonging options. One button is applied to the left and the other one to the right soft key.

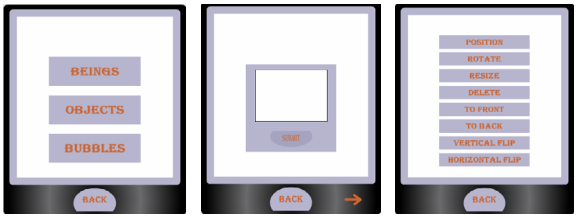

Figure 5. The "Insert", "Text Insert" and "Editing Options" graphical user interfaces.

All the three frames have different actions. Therefore they are split into three frames and also three different key catchers. The first frame function is dedicated for making the system calls, variables initialization and it is also programmed for distinguishing between buttons. The second one is to make the characters appear if selected (in the menus part they have to disappear). The functionality is important for strip editing.

The third frame differs from the second merely in the key catcher. Here the user can edit all elements, for example changing their position, resizing them, etc. This action is possible because of the different types of key catchers. For editing we used the 4-way keypad press events with the "Enter" press event.

The second frame functionality is to select elements (characters), and the third ones functionality is to edit some properties. Both are used with the 4-way keypad and Enter press events (Figure 5.).

1) The "Insert" frame label

Here in this part, the Insert menu is implemented. It certainly supports all three types of strips implemented: the beings, the objects and the bubbles. One can choose among them and then chooses the respective selection part. The Design part has like we already mentioned the border (appearing unchanged through the whole application) and four buttons. Three of them are designed for choosing among the strips and one to return into the main scene. Basically this part only has the objective of the strips selection one can choose for insertion into the main scene.

2) The "Options" frame label

This label is very similar to the one mentioned before. It represents another menu. Here we choose among the options of the application, for example the "clear all" option that deletes the entire main scene instead of deleting characters one by one. The "Help" and the

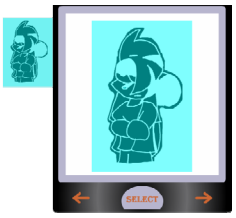

Figure 6. The two step altering process of cartoon characters.

"Send" options don't make anything yet. Their functions are reserved for future use for publishing the content to a WEB server. Finally the "exit" option exits the application.

3) The "Menu1" frame label

This label belongs to the same category as the last two frame labels mentioned before. Like the other two this label has the same design. It represents a menu where one can choose the options related with the properties of the strips, like "position", "rotate", "resize", "delete", "to front", "to back", "vertical flip" and "horizontal flip".

4) The "Menu2" frame label

In this menu we solved the differences that the select arrow has in accessing the same strip properties already mentioned in the foregoing Menu1 label. This menu is fundamentally different from the other menus because it is intended solely for the bubbles strips. Here the text is inserted by the user adding dialogs, comments and other information into selected bubbles. After the insertion procedure the user has to confirm the text input using the "Submit" button. Then the text will be shown inside the bubble. Another difference is the implemented key catcher. It enables the user to change from the current menu (Menu2) to a different one (Menu11). That is because of the same properties the bubbles have compared to other non-text strips after the text input. After text input competition the cartoon characters must support all the editing functions mentioned in the Menu1 subsection.

5) Menu11" frame label

This menu is a copy of "Menu1". The differences here are with a different key catcher which additionally enables the user to change from this menu to "Menu2". The option was added with the goal of simplifying the user interaction with more intuitive changes between menus.

6) The "Beings", "Bubbles", "Objects" frame labels

Because of the similarities in structure of the three strip cartoon types we will focus just on the core and the differences in their structure. Each strip cartoon is characterized by its graphical environment – the frame and the soft keys. Its core consists of two MovieClips with the respective timeline and layers for the ActionScript. An additional invisible button is added for interactivity in a separate layer for each cartoon character (the blue transparent surrounding of the character, Figure 6.).

The characters can be altered by activating the key catcher therefore pressing the "Enter" button. The two MovieClips are necessary because of the implemented editing concept. The usually minimized MovieClip the so called "beings holder" is an altered copy of the original "beings" MovieClip. We used this concept to maximize the capabilities of visual representation of the cartoon elements on the mobile phone screen. We could also explain the "beings holder" frame label as the pointer to the preview option and the "beings" frame label as the pointer to the final view of the cartoon character. The cartoon character properties are therefore not edited on the final screen but prior to their selection in a separately editing screen.

V. APPLICATION TESTING

Despite that it's often easier to interact with an emulator than with an actual device, the tester doesn't get a true sense of application usability. Emulators will normally allow for the use of mouse and keyboard, while actual devices obviously do not. While this doesn't directly affect functional testing, it's always a good idea to do some amount of testing on the actual device to evaluate its usability [9], [11].

A. Workflow for authoring Flash Lite applications

The process for creating Flash Lite content is an iterative one that involves the following steps (Figure 7.):

1) Identify target device(s) and Flash Lite content type

Different devices have different screen sizes, support different audio formats, and have different screen color depths, among other factors. These factors may influence the application's design or implementation. In addition, different devices support different Flash Lite content types, such as screen savers, stand-alone applications, or animated ring tones. The content type for which we are developing also determines the features that are available to our application.

2) Creating and testing applications in Flash

Adobe Flash CS3 includes a Flash Lite emulator (Adobe Device Central) that lets the developer test the application without having to transfer it to a device. The developer uses the Adobe Device Central to refine the application design and fix any problems before testing it on the mobile device (Figure 4.) [13].

3) Testing the application on a target device or devices

This step is important because the emulator doesn't emulate all aspects of the target device, such as its processor speed, color depth, or network latency. For instance, an animation that runs smoothly on the emulator might not run as quickly on the device, due to its slower

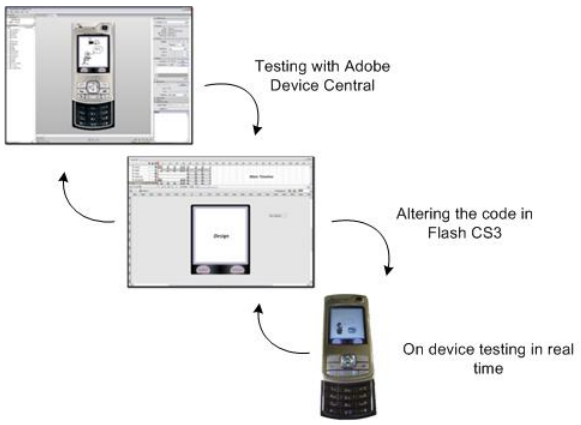

Figure 7. The interactive development and testing process using Adobe Device Central

Figure 8. The final implementation of the Cartoon-Generator on the Nokia N80.

processor speed. Or a color gradient that appears smooth in the emulator may appear banded when viewed on the actual device.

After testing the application on a device, the designer may find the need to refine the application's design in the Flash authoring tool. This it what we followed since the beginning of development of the application – when we changed or redesigned something in the work, we tested it in the Device Central. In our opinion this is a very good program for application testing to verify and accordingly modify the information about the file information, memory usage, performance, output messages, etc.

The figure above a scenario of our final application on the mobile phone is shown. Once we performed the necessary tests on the Adobe Device Central with guidelines from [13], the application was additionally tested on the Nokia N80 Mobile Phone (Figure 8.). Thus the true sense of the application usability was tested because it is often easier to interact with an emulator than with an actual device [11]. Therefore we really got a better approximation how the application works in practice.

VI. CONCLUSION

Although Flash Lite 1.1 was used with its limitations we managed to design a user friendly interface for cartoon creation and push-publishing on mobile phones. We designed the user interface and created three types of cartoon elements for the cartoon-generator. Every element can be edited regarding time and spatial composition. Text can be additionally added and altered. Also options for push-publishing on a WEB server are supported. The application could be further improved in the design aspect or at least could have much more characters included to make the application wealthier with histories or more critics or even more jokes to reach almost all age bands. Nevertheless our objective was to test the concept of the strip-generator based on the proposed framework on a real mobile phone without targeting a specific group or individuals.

REFERENCES

[1] J. Stergar, D. Miletić, "Rapid development and deployment of content for mobile devices", A. Hudobivnik, I. Humar, B. Vlaovič, Eds. Content and networking: proceedings of the International Symposium on Telecommunications, 2006, Ljubljana, Slovenia.

[2] D. Linsalata, A. Slawsby: Addressing Growing Handset Complexity with software Solutions. IDC, http://www.adobe.com/mobile/news_reviews/articles/2005/idc_whitepaper.pdf, 2005.

[3] R. Brady, S. Tonzi, New Flash Technology From Adobe Enables High-Impact User Experiences For Consumer Devices. Adobe Systems Incorporated. http://www.adobe.com/aboutadobe/pressroom/pressreleases/ 2005.

[4] Flash Lite Development for Nokia Devices, Adobe Developer Connection, Mobile and Devices Developer Center, http://www.adobe.com/devnet/devices/nokia.html.

[5] J. Ulm, "Designing Engaging Mobile Experiences", http://www.adobe.com/devnet/devices/articles/designing_engaging_mobile_experiences/designing_engaging_mobile_experiences.pdf, Adobe Systems Incorporated, 2007.

[6] E. Elrom, S. Janousek, T. Joos, AdvancED Flash on Devices: Mobile Development with Flash Lite and Flash 10, Friends of ED, 2009.

[7] J. A. Landay, B.A. Myers, "Sketching storyboards to illustrate interface behaviors", Conference on Human Factors in Computing Systems, pp. 193 – 194, 1996, Vancouver, British Columbia, Canada.

[8] J. Rasmusson, F. Dahlgren, H. Gustafsson and T. Nilsson, "Multimedia in mobile phones – The ongoing revolution" Ericsson Review no. 01, 2004.

[9] J. Harty, A Practical Guide to Testing Mobile Smartphone Applications, Morgan & Claypool Publishers, M. Satyanarayanan Eds., 2010.

[10] T. Jokela: Authoring Tools for Mobile Multimedia Content. Nokia Research Center. IEEE International Conference on Multimedia & Expo. pp. II-637 – 640. 2003.

[11] R. S. Barber, "Automated Testing for Embedded Devices", Software Testing Innovations Series, PerfTestPlus, Inc., http://www.perftestplus.com/resources/EA.pdf, 2006.

[12] D. Carroll, "Packing Lite: A mobile media interface design primer", http://www.adobe.com/devnet/devices/articles/packing_lite.html, Adobe Developer Connection Mobile and Devices Developer Center, 2007.

[13] W. Wang, "Overview of Adobe Device Central", http://www.adobe.com/devnet/devices/articles/introducing_device_central.html, Developer Connection Mobile and Devices Developer Center, 2008.

AUTHORS

J. Stergar is with the Digital and Information Systems Laboratory, Institute of Electronics and Telecommunications, University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova ulica 17, 2000 Maribor, Slovenia, (e-mail: janez.stergar@uni-mb.si).

A. Šulić is a co-worker of the Digital and Information Systems Laboratory, Institute of Electronics and Telecommunications, University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova ulica 17, 2000 Maribor, Slovenia (e-mail: andrija.sulic@rockpixel.si).

B. E. Rodrigues Silva is a Socrates Erasmus exchange student from Portugal, Instituto Politecnico do Porto. (e-mail: 1010587@isep.ipp.pt).

Submitted March 12th, 2010. Published as resubmitted by the authors March 18th, 2010.