Building Accessible Flash Applications: An XML-Based Toolkit

Paloma Cantón , Ángel L. González , Gonzalo Mariscal , and Carlos Ruiz

Consejería de Educación de la Comunidad de Madrid

School of Computing, Technical University of Madrid

CETTICO, Facultad de Informática.
UPM, Campus de Montegancedo, 28660 Boadilla del Monte, Madrid, Spain

Abstract. The use of Flash as a web-based multimedia development tool has spread lately. Although a big effort has gone into improving its accessibility, there are still accessibility gaps requiring programming or purchase from another supplier. This makes building an accessible Flash application an ad hoc, complex and time-consuming task. With the aim of lightening the accessibility-related workload, we have implemented a toolkit that helps to create accessible multimedia Flash resources. This toolkit specifies the accessibility features as XML configuration files. It includes a library that works like a wrapper abstracting the logic layer of the different events and iterations from the physical layer. This way, new functionalities can easily be added. Additionally, it has been successfully used to build teaching and teaching support systems.

Keywords: E-learning, web accessibility, flash accessibility, education.

1 Introduction

Multimedia tools for developing applications and contents on the Internet are in widespread use. Very often their use is debatable and merely ornamental. In other cases, though, they are indispensable, for example, to build teaching or teaching support systems. Multimedia resources are a key component in this kind of systems. In this scenario, there are several solutions, although the market is now moving towards the use of Flash to develop multimedia applications embedded in web pages[1].

Until 2002, using Flash to develop accessible multimedia contents was a one-off, tough and complex task. Additionally, the developed contents were not compatible with standard assistive technologies. However, the release of Flash MX marked a shift towards the improved construction of accessible Flash-based resources.

Macromedia Flash MX and Flash Player 6 (Adobe bought up Macromedia at the end of 2005 and is now the owner of Flash) are the first accessible versions of Flash platforms. Developers can use Flash MX to create US guideline-compliant accessible multimedia applications To assure accessibility on Microsoft Windows systems, Flash uses the Microsoft Active Accessibility (MSAA) API. Even so, these features are limited and do not guarantee adequate accessibility and usability for some types of applications and disabilities. For example, it overlooks key aspects like the application context, scanning input methods, captioning, languages... These are elementary features for developing activities or support material for teaching children with special educational needs (SEN). Developers, then, are left with two alternatives: develop the application ad hoc to solve a specific problem or buy special-purpose libraries to meet those accessibility needs. The first alternative calls for specialized developers, increasing the cost of production and reducing reuse. The second entails the purchase of external libraries, supposing that they exist and meet the general and special needs. Generally, more than one library will be necessary to meet the application's needs.

This raises several questions: Wouldn't it be better to have a wrapper integrating and abstracting accessibility needs? And to be able to easily integrate and use any new accessibility functionality you want to add as just another library? And even be able to have an external configuration specification for any Flash application defining the necessary accessibility requirements depending on the user profile?

In response to these problems, we designed the Accessibility Toolkit (ATK). This toolkit facilitates the creation of accessible Flash applications, allowing the definition, configuration and use of accessibility elements in a simple and reusable manner. The creation of ATK is part of the framework described in . It is being applied in the *Proyecto Aprender* (Learn Project) and *Internet en la Escuela* (Internet at School) [4] projects, and works like a Flash accessibility wrapper.

This article is organized as follows. The next section summarizes the work related to multimedia accessibility and Flash in order to analyse Flash's accessibility properties. Then the philosophy underlying this paper and the solution implemented through the development of the Flash accessibility libraries is presented. Finally, we present the results, conclusions and future lines of research.

2 Related Work

SVG (Scalable Vector Graphics) is a language describing graphics and graphical applications that has become an alternative to Flash for rendering multimedia content on the web. It is a W3C recommendation based on XML and CSS that has a number of strengths: it is an open standard, allowing different compatible implementations; it is easy to edit; the graphics code can be easily searched; and one CSS can be used for several graphics. Although more and more browsers are including a SVG viewer and plug-ins have been developed, it is less commonly used than Flash.

As regards add-on libraries containing accessibility features and operating as extensions of Flash, worthy of note are Hi-Caption and Caption Component (CC) for Flash They manage captions and synchronize files with the respective text and videos in different formats. Zoomyfier can create and integrate zoom and panoramic applications into Flash to improve viewing for visually impaired people.

Another notable option is AccRepair which is useful for rapidly and effectively verifying compliance standards set out by US Section 508. Specifically, it defines a test to verify accessibility properties use in objects that are included in the movie and have alternative names and descriptions, tabulation order, etc.

3 Flash Accessibility Properties

Flash offers a series of basic accessibility features through its API, allowing communication through MSAA. These are: (a) define whether an object is accessible, (b) define an alternative text for any graphical element, (c) define keyboard shortcuts to controls, and (d) set a tabulation order of elements.

Table 1 shows the accessibility needs by disability and whether they are directly enabled by Flash, whether they will have to be programmed or whether an external application will have to be used. Some of the needs are related to one or more disabilities, but the table has been simplified by placing them under a single disability. Looking at Table 1, it is clear that not all the accessibility needs can be met, not even by combining different functionalities to make a user need-dependent accessible Flash application. The answer would be a tool that can standardize the use of the available features and abstract the content of the Flash movie from user needs.

Table 1. Accessibility needs by disability and availability in Flash

Disability Type	*Needs*	*Native Flash*	*Through external applications*
General	Language management	No	
	Configuration management	No	
	Tabulation order	Partial	
	Element clustering	No	
Visual	Document description	Yes	
	Object description	Yes	
	Video description	No	
	Control description	No	
	Font size management	No	
	Image size management	No	Zoomifyer
Hearing	Captioning synchronized with audio and video	No	HiCaption, CC
Motor	Device-independent controls	Yes	
	Other types of pointing devices (scanners)	No	
Cognitive	Context management	No	

4 Solution: An XML-Based Toolkit for Accessibility in Flash

The way to solve the above problems is to provide an abstraction of the different ways of interacting with users, irrespective of their disability and the use of assistive technologies. This abstraction should produce:

- Conceptual definition of the user actions This is an inherent definition of direct handling interfaces. For example, we must define the select, move, request help actions, etc. They must all be defined with a view in the first place to their meaning. The library automatically translates the user actions to the equivalent concept within the application. The programmer only has to deal with the defined concepts and not with how the user interacts.

- Mechanisms to translate the input from assistive technologies to one of the conceptually defined actions. In the case of concept keyboards, for example, the selection of a keyboard concept may be associated with the selection of an element from the multimedia scene or a specific action like go forward or go back. This translation also has to be applied when the system has to communicate with the user. This is done thanks to the conceptual information associated with each interactive object and with the scene in which it appears. This information is used to send the right output to each assistive technology (such as, descriptive text, visual or audio rendering, etc.) and to the selected language. A programmer-transparent wrapper should take charge of this translation.

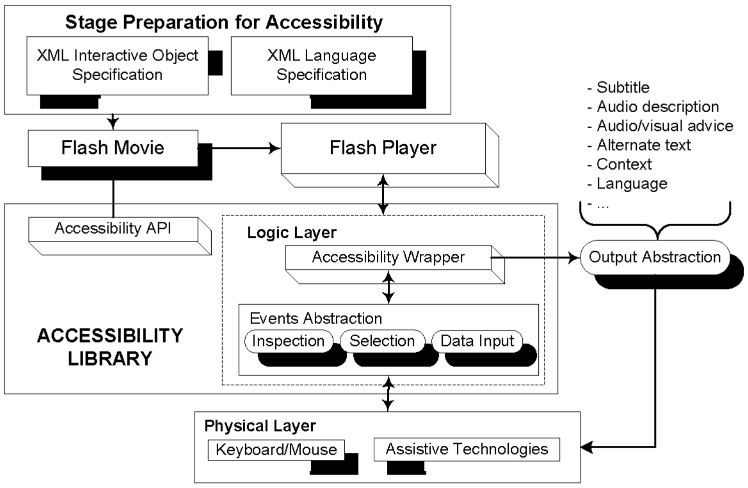

Fig. 1. Overview of the architecture

All the conceptual information, like the best representations and the correlation between concepts, should reside outside the Flash application and will be used by the wrapper to do its job. To do this, we decided to define a set of XML files that could be used to gather this information externally.

For this purpose, we defined the Accessibility Library (AL), built into the ATK. This library can externally define and manage the accessibility needs of the Flash application. This improves reuse and provides effective profile-based tailoring of the application. Figure 1 shows an overview of the AL's operation and composition. As illustrated in Figure 1, the configuration parameters and the conceptual description of the scene are established at an earlier stage of the design process. On the other hand, the AL interprets the files defined earlier on and defines the logic of the multimedia scene. In the following we will describe each component in more detail.

4.1 Preparation for Accessibility

Before starting to program a multimedia scene, a scene preparation process is enacted during the design stage. At this point, the accessibility parameters can be configured depending on the type of user that is going to use the application. At this stage, we will also add the conceptual information associated with each interactive object and the relations between objects, and we will allocate the different alternative representations of the concept (image, text, speech, etc.) that can be used during scene execution. Finally, concepts have to be defined for the actions that the user or the system can take. These actions can be triggered by user interaction with the system or by system-generated events. These are the conceptual actions that programmers have to deal with when they implement the scene. All this information is defined through a web-form-based application that is part of the ATK. The web forms are used to generate a set of XML files. The configuration parameters can be preset or modified at run time. This way, it is possible to tailor the behaviour of the library. This will determine how to interact with the application, that is, everything related to scene accessibility and usability. Library behaviour is programmer transparent.

4.2 Accessibility Library (AL)

Strictly speaking the Accessibility Library will automatically interpret the information generated during the preparation stage to tailor navigation and interaction with the objects that are part of the application. Additionally, it will provide the necessary support for programmers to specify the logic of the scene. With a view to developers, AL contains the Accessibility API. This API has been built as a Flash MX accessibility functionality wrapper and add-on designed to meet all the described needs. It is implemented through OOP in the Flash programming language: ActionScript v2.0. The AL is divided into four components:

1. Configuration Manager. Interprets the configuration file defined in the preparation stage and automatically establishes some parameters for the application.
2. Input/Output Manager. Provides mechanisms to translate: the input from an assistive technology (or standard input device) to one of the conceptually defined actions, and a conceptual representation to the respective output of an assistive technology (or standard device output). The application will be independent of the input/output device in use (keyboard, mouse, screen, assistive technologies).
3. Context Manager. Provides mechanisms to inform users whereabouts they are in the Flash movie. For example, it informs users of the scene they are viewing and what actions they should take.
4. Language Manager. Allows different languages to be used or the language in use to be tailored.

4.3 Accessibility Wrapper

As already mentioned, all the finer points that make programming tough are left out of the Flash application and placed in a series of XML files. All developers have to do is follow the guidelines established in the ATK manual focusing on the logic layer of abstraction without having to bother about all the little accessibility details involved in making an accessible Flash application. The logic layer of abstraction uses functionalities provided by the configuration manager, input/output manager and language manager. Among other things, its job is to manage the logic of navigation within the application, to tailor language and set the language used.

4.4 Abstraction Model

The abstraction that we propose for simplifying the actions in response to user events is based on the model described in Figure 2 shows a simplification of this model that will be used as a basis for building the AL. The above model puts all the devices into two major input or output categories. A device can belong to both categories and offer the respective input and output services. As shown, the input devices are wrapped by an interface divided into three main categories. It is the interface that allows the user actions to be given standard treatment. The user actions are transformed into events that will be properly processed by the events manager.

In our case, the events manager will be the listener system defined by Flash. The events will be gathered and translated to their conceptual representation, if any. The programmer will only have to bother about the notices of appearance for actions, like, for example, capture information, move (either around the screen or through selectable elements), select, request help, irrespective of what device the user uses to take the action, be it a qwerty keyboard, concept keyboard, pointing device, etc. This process will be programmer transparent and carried out by the library.

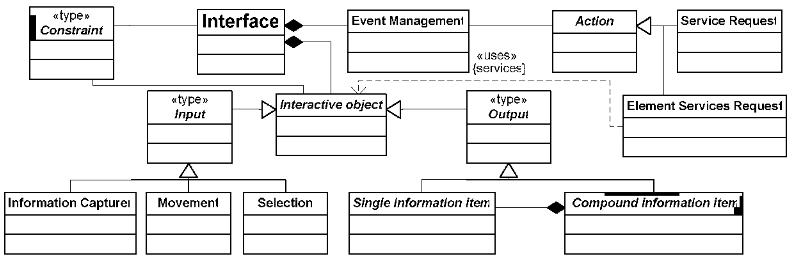

Fig. 2. Device Abstraction

As regards the output, programmers have to work with concepts that represent the information that they want to transmit to the user. The information to be displayed will be gathered externally in XML files created in the previous stage. The media to be used to display the information will be specified depending on the selected configuration. In this way, an output telling users how to take an action could produce captioned output, an audio description, an animation, etc. All this will be synchronized, and, at the end of the output, the programmer will be notified by an event. Additionally, these outputs could be displayed in a developer-transparent manner.

5 Results

The results presented in this paper are the fruit of a line of research that kicked off by defining a methodology for making accessible Flash movies The methodology was applied in *Proyecto Aprender*, and the results were moderately successful. From this experience, we learned that the production process needed to be improved and simplified. This is what the Accessibility Toolkit (ATK) has managed to do.

Table 2 shows how the development effort got smaller as we developed our approach further. The first two columns (ad hoc and I1) correspond to the first phase of the *Proyecto Aprender*. Several development teams divided into two groups worked on this phase. The first group developed the application ad hoc, whereas a second group prepared a smaller version of the AL, which it then applied to develop the activities. Although the second group fell behind the first group at the start, as the number of scenes to be developed grew, it gradually caught up again, and it managed to finish ahead (development time was 25% less). The ATK described in this paper was developed in the second phase. In this phase, the AL was integrated and extended with a toolkit to facilitate the tasks to be performed in the earlier stage. These new resources were used to develop the second phase of the '*Proyecto Aprender*: Attention deficit, autism and behavioural patterns'. The use of the toolkit and the proposed methodology led to a 50% increase over I1 in the time required for the Preparation Stage, whereas there was a sizeable drop in the development time for the next stages. In actual fact, there was an overall improvement of 50% over the ad hoc implementation, and development time was more than 30% shorter than using I1.

Table 2. Table of times for applying accessibility in a Flash application

time (hours)	*ad-hoc*	*I1*	*ATK*
Stage 1 Accessibility: Stage Preparation	0	2	4
Stage 2 Accessibility: Accessibility Development	14	8	2
Scene Development	20	15	10
TOTAL:	34	25	16
Improvement over ad hoc development	-	**26.47%**	**52.94%**

Accessibility was not evaluated according to a standard evaluation methodology, as, according to Nielsen [11], the four basic ways of evaluating user interfaces are automatically, empirically, formally, and informally. Surveying the current state of the art, there are no fully automatic methods. Additionally, formal methods are very difficult to apply and do not scale up well to handle larger user interfaces. Also none of them is directly applicable for assessing software accessibility In this case, experts from CNICE (National Centre of Educational Information and Communication) and educators observing the behaviour of learners with SEN working with the system evaluated accessibility for compliance with ISO DIS 9241-171

6 Conclusions

Although first Macromedia and then Adobe have tried hard to make it possible to create accessible web-embedded Flash applications, the truth of the matter is that making accessible Flash movies is still hard work today.

ATK provides a tool that helps developers to create accessible web-embedded Flash applications in an easy way. Another noteworthy point is that ATK has evolved into a flexible framework that can be easily extended by adding new modules and developer utilities.

The potential future improvements that we are weighing up are the possibility of using templates to make it easier to change standard activities, that is, templates in which teachers specify what concept families or groups they want to teach pupils, and the toolkit selects the best elements from each family to build the activity. In this case, rules would have to be entered to define special cases, like "at least one from this group" or "two elements from group B for every one element from group A", etc. Finally, functions could be added to ease integration into e-learning or blended learning systems by providing calls out to SCORM-compliant systems in the API.

References

Macromedia Flash MX Accessibility FAQ. Is the Macromedia Flash authoring tool 508 compliant?, `http://www.adobe.com/macromedia/accessibility/features/flash/faq.html#itemA-5` (last visit 01-11-2007)

Cantón, P., González, A.L., Mariscal, G., Ruiz, C.: Towards a methodology for educating students with special needs. In: 5th Conference and Workshop on Assistive Technology for People with Vision and Hearing Impairments, CVHI 2007, Granada, Spain (2007)

Cantón, P., González, A.L., Mariscal, G., Ruiz, C.: Developing pedagogical multimedia resources targeting children with special educational needs. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A.I. (eds.) ICCHP 2006. LNCS, vol. 4061, pp. 536–543. Springer, Heidelberg (2006)

Gértrudix, M., Gálvez, M.C., Álvarez, S., Galisteo, A.: Design and Development of Digital Educational Content Institutional proposals and actions. In: Computers and Education, pp. 67–76. Springer, Netherlands (2007)

W3C. Scalable Vector Graphics (SVG). XML Graphics for the Web, `http://www.w3.org/Graphics/SVG/` (last visit 15-10-2007)

HiSoftware. HiCaption Studio, http://www.hisoftware.com/hmcc/index.html

National Center for Accessible Media (NCAM). Caption Component (CC) for Flash, `http://ncam.wgbh.org/webaccess/ccforflash` (last visit 05-11-2007)

Zoomify for Flash. Zoomify Inc., `http://www.zoomify.com/flash.htm` (last visit 12-12-2007)

HiSoftware. AccRepair, `http://www.hisoftware.com/access/repair.html` (last visit 1-11-2007)

González, Á.L.: Modelo para la Generación y Gestión en Tiempo de Ejecución de Procesos de Interacción Hombre-Máquina a Partir de un Lenguaje de Especificación de Relaciones con el Usuario. PhD Thesis dissertation, Technical University of Madrid (2003), `http://oa.upm.es/87/`

Nielsen, J.: Usability inspection methods. In: Conference companion on Human factors in computing systems, Boston, Massachusetts, United States, April 24-28, pp. 413–414 (1994)

Hideki, E., Bim, S., Vieira, H.: Comparing accessibility evaluation and usability evaluation in HagáQuê. In: CLIHC 2005. ACM Int. Conference Proceeding Series, vol. 124, pp. 139–147 (2005)

ISO DIS 9241-171. Ergonomics of human-system interaction - Part 171: Guidance on software accessibility. Draft International Standard. ISO (2006)

Sharable Content Object Reference Model (SCORM), `http://www.adlnet.gov/scorm` (last visit 22-11-2007)