Development of a Game Engine for Accessible Web-Based Games

Javier Torrente(✉), Ángel Serrano-Laguna,
Ángel del Blanco Aguado, Pablo Moreno-Ger,
and Baltasar Fernández-Manjón

UCM, Madrid, Spain
jtorrente@e-ucm.es

Abstract. The Web is rapidly shifting towards more dynamic and interactive content. One clear example is the increasing use of web-based digital games. However, the more interactive a piece of content is, the more difficult it is to make it universally accessible. Besides, users are increasingly demanding ubiquitous access to the content and applications they use (including games), resulting in a need for ensuring that Web content is also multiplatform. These trends are adding an extra technology challenge for ensuring content accessibility. In this paper we describe our technical approach to create an accessible multiplatform game engine for the new version of the eAdventure educational game authoring platform (eAdventure 2.0). This approach integrates accessibility as a core design principle instead of adding accessibility features *a posteriori*. We expect this to facilitate the creation of web-based digital games that are accessible regardless of the context (device, assistive tools available, situation, etc.) in which they are being used. In this work we describe the general architecture, as well as some specific examples of accessibility adaptation plug-ins already available.

1 Introduction

Rich Internet Applications (RIAs) and interactive content are gaining importance in modern web to enrich the navigation experience. In particular, digital games are increasingly being used in the web, not only for leisure but also for 'serious applications' like education [8], health [4], advertising [11] and even as an alternative to Captchas [1]. The drawback is that RIAs create new issues from an accessibility perspective. The problem grows for digital games where interaction cycles are extremely short and feedback is usually provided on multiple channels. Although the problem has been identified, and research is being conducted on how to address it [17], the fact is that the current level of accessibility of digital games is still rudimentary.

The limited accessibility of digital games is not motivated by a single reason. An apparent lack of awareness of game developers and the cost overhead that accessibility adds to any game development project are surely among the most relevant. A proposed approach to address these issues, at least partially, is to integrate accessibility into game development software [14], instead of focusing on ad hoc solutions for each particular title. On the one hand, this increases the visibility of accessibility

A. De Gloria (Ed.): GALA 2013, LNCS 8605, pp. 107–115, 2014.
DOI: 10.1007/978-3-319-12157-4_9

among developers, as it translates the problem to a language they are familiar with. On the other hand, it allows reusing previous efforts across different game development projects, resulting in significant savings and cost reductions.

When games are deployed in the web there are also further technical challenges that are difficult to address. Web content can be used (by definition) in different contexts and deployed on multiple platforms, which adds uncertainty to what technologies or assistive tools would be available. Besides, web content must be conformant to standards to ensure interoperability.

In this paper we present our ongoing development efforts in the eAdventure 2.0 game engine and how it is being designed and implemented to accommodate accessibility from its very inception. Once development is complete, game authors will be able to make accessible games more easily and automatically deploy them on the Web using HTML5 and WebGL. Most of the solutions proposed could also be applied to other types of RIAs and interactive contents.

This paper is structured as follows: Sect. 2 provides a short overview of the state-of-the-art in digital game and RIA accessibility. Section 3 introduces the eAdventure platform: what is it, what prototypes have been already developed to explore accessibility in games, and why a new version is being developed based on Web technologies. Section 4 describes the technical design rationale to introduce accessibility in the eAdventure 2.0 game engine, with Sect. 5 providing examples on how the architecture presented allows adapting the games for two specific user profiles. Finally Sect. 6 wraps up our contribution and outlines future lines of research.

2 Background

Web accessibility has traditionally focused on granting equal opportunities of access to the vast majority of the content and applications that populate the Internet, which used to be rather static and not highly interactive. Interest on making RIAs accessible is more recent. This unbalanced distribution of efforts is reflected on current status of web accessibility standards. While the Web Content Accessibility Guidelines (WCAG), which deal with static content, are a mature and stable technical standard, its counterpart for RIAs, the Accessible Rich Internet Applications (WAI-ARIA) specification, is still a draft.

Concurrently the gaming field is gradually starting to explore how to increase accessibility of digital games, not necessarily focusing on the Web [17]. The first accessibility guidelines specifically targeted to digital games were proposed by the Special Interest Group on accessibility of the International Game Developers association [5] on 2005. These guidelines provided a compendium of good practices grouped by types of disability and exemplified through case studies of games that included features to support accessibility that were available at the time. Since then, the state-of-the-art on game accessibility recommendations has been pushed forward not only by IGDA but also by other advocators and dedicated institutions [2]. However, the field is not mature enough to produce an official standard or technical recommendation similar to W3C specifications, lacking of reference tools and appropriate conformance levels.

In the academia, research initiatives on digital games have also emerged [17, 18]. Some of these initiatives have focused on the production of games that could be

enjoyed by players with and without disabilities alike, while others have focused on the special needs of players with disabilities only [12]. Other experiences have focused on making popular games accessible, instead of developing an accessible game from scratch [3].

Comparatively, very few cases have explored how game technologies and development software can support accessibility. For example, in [10] a Game Accessibility Framework is introduced from a conceptual perspective. Moreover, the additional requirements of Web games remain as an open issue.

3 eAdventure

eAdventure (formerly <e-Adventure>) is an open source, high-level game authoring tool [6]. Unlike more complex tools (e.g. Unity [16]) it targets low-profile and user-generated games that could be used in different contexts, especially 'serious applications' and education. The types of games that can be produced with eAdventure are limited to 2D point-and-click games and conversational adventures. This genre is typically considered more appropriate for educational settings (and more accessibility-friendly) due to the focus on exploration and reflection as opposed to time pressure or fast-paced action [7].

3.1 Versions 1.X and 2.0

eAdventure has been in development since 2005, being v1.5 the latest version available. On 2011 it was reaching its end of life. It is built on Java, which is rapidly becoming an obsolete technology for Web clients due to the need of installing browser plug-ins and recent security holes found in Java Applets. This presents a problem in online education (a.k.a. e-learning) environments, where everything lives on the web. For that reason, we started the development of a new eAdventure game engine from scratch (v2.0). The main aim in the development of eAdventure 2.0 is to provide an extensible and multiplatform engine to supports game deployment as HTML 5 (using WebGL) Web Applications. As HTML 5 cannot be fully deployed in some devices yet (e.g. computers with old browsers or some smartphones and tablets), the eAdventure 2.0 also has native support for specific platforms (e.g. Android devices).

Both branches of the eAdventure engine currently coexist. The internal architecture is completely different in both cases. The former one is referred to as version 1.X (stable but rapidly becoming obsolete) while the new one (unstable) is referred to as version 2.0.

3.2 Previous Work on Accessibility

Previous work has already explored the introduction of accessibility in the eAdventure platform using version 1.X. In [13] the development and integration of accessibility modules for adapting the game interface dynamically is described. Three user profiles were considered: (a) screen reader users (blind); (b) speech recognition users (limited

mobility in hands) and (c) users that need high contrast settings (low or limited vision). Different alternatives for users requiring screen readers were further explored in a subsequent experiment [15]. The experience gathered on these previous research activities has been used to design the core set accessibility features that will be supported by eAdventure 2.0 out-of-the-box.

4 Implementation Proposed

The basic architecture of the new version of the eAdventure engine (2.0) was described in a previous publication, which can be consulted for further details [9]. In this paper we focus on how accessibility fits within this architecture.

4.1 Engine Architecture

The 2.0 engine is modular, multiplatform and extensible. It is built upon an API that supplies functionality (e.g. access to the data model and art resources, etc.) for all basic processes of the application (e.g. rendering, collision detection, etc.) to all internal components (see Fig. 1), and enables cross-component communication. All the platform-independent functionality of the Engine API is implemented by the Engine Core, the main controller of the application. This way most of the code of the engine is implemented only once. Platform-dependent components provide implementations for the rest of the Engine API (e.g. image rendering, video reproduction, input/output, etc.).

The eAdventure data model (the description of the game) is constituted by EAdElements. An EAdElement holds no computation logic, just a piece of the description of the game or one of its components. This includes characters, items or game scenarios, but also effects triggered in the game in response to user's interactions. These effects can produce feedback for the user. For example, eAdventure 2.0 supplies effects to display formatted text on the screen, or to play a sound track. At runtime, the game engine reads the EAdElements from a XML file and translates them to GameObjects, which are the minimal game functional units, that can be manipulated.

eAdventure 2.0 uses the concept of plug-ins to support functionality and platform extension. An eAdventure plug-in is a set of classes and interfaces extending and using the Engine API. Plug-ins are programmed as independent units that are loaded at start-up. For example, plug-ins can contain extensions of existing EAdElements or GameObjects, new implementations of parts of the API, etc. A configuration file defines the plug-ins that the game engine must load at start-up.

The implementation(s) of all the parts of the Engine Core and API (e.g. EAdElements, Game Objects, Plug-ins, Core functionality, Platform-dependent components) are completely separated from the interfaces that define them. The interfaces are bound to the code components (i.e. classes) dynamically at start-up, using a technique called dependency injection (Google Guice is used for this purpose). This structure enhances the flexibility and adaptability of the engine, as the behavior of any component can be replaced dynamically (e.g. to better suit the needs of the user).

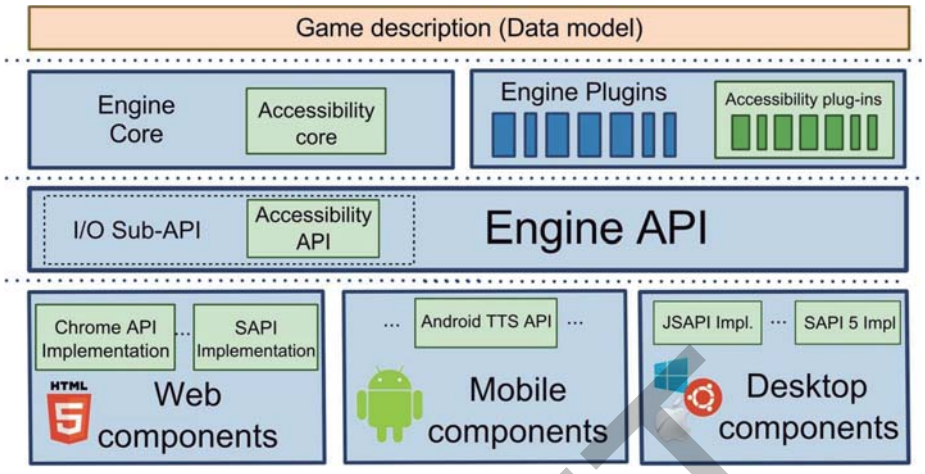

Fig. 1. Architecture of the eAdventure 2.0 engine

4.2 Accessibility Support

Accessibility is present in different components of the game architecture, including: an Accessibility API (part of the main Engine API); an Accessibility Core, responsible for implementing part of the Accessibility API and also for setting up accessibility features at start-up; Accessibility Plug-ins, a set of engine extensions to deal with particular functionalities; and Platform-dependent accessibility Components that implement parts of the Accessibility API that handle inputs and outputs, like speech recognition or text-to-speech (TTS).

Most of the code that deals with accessibility is implemented either as Accessibility Plug-ins or Platform-dependent Components. The Accessibility Plug-ins (a specific type of eAdventure plug-ins) allow changing the behavior of any component or element of the game engine. For example, a plug-in could adapt how the visual elements are rendered, or the complexity of the text or other pieces of game content. Platform-dependent Accessibility Components are designed to allow interoperability with external components by connecting the engine's I/O modules to different platform-dependent implementations of the Accessibility API (e.g. Android or HTML5). In this manner the game engine can take advantage of technologies or support tools that the user may already have installed (e.g. the JAWS screen reader, Mac Voice Over system, etc.). This favors using well-tested and implemented aiding technologies, and also allows the applications to be lighter and speed up loading times.

In the process of setting up a game for a particular user, some game content (e.g. images and text) may need to be adapted. Occasionally the content can be adapted dynamically (e.g. apply a filter to the image) but sometimes it is necessary that the engine is fed with alternative versions of these resources. For that reason, game content is highly decoupled and encapsulated. Using a namespace convention, different versions of the text scripts and images are organized in folders. When a resource is loaded in the game, the engine fetches the best version available for the characteristics of the

user. If none of the versions for that resource suit the user needs, then it will attempt dynamic adaptation.

5 Case Studies

To exemplify how the engine works, two case studies are presented, focusing respectively on color vision deficiency and screen reader users.

5.1 Adaptation for Color Vision Deficiency

Users with color vision deficiency (CVD) may have troubles playing a game if color schemes are used to convey information. The color schemes used may need to be adapted or replaced by other identification techniques (e.g. icons). Users with CVD may also have problems reading text if its color cannot be distinguished from the background.

These problems are solved using the dependency injection technique. GameObjects that are responsible for visual elements of the scenes are created using an alternative version that alters the rendering code. For example, at runtime, the GameObject used to control and render a game scenario (interface GOScene) is bound to an alternative implementation (e.g. class GOSceneCVDImpl) that overrides the draw() method making interactive elements more distinguishable. Similarly, the GameObject that represents effects for showing text in the game (interface GOTextEffect) is bound to a different implementation (class GOTextEffectCVDImpl) that draws the text on a clear background using a high-contrast color scheme.

5.2 Adaptation for Screen Reader Users

The most important needs of screen reader users are (1) avoiding the mouse as input device (they can use a keyboard) and (2) providing non-visual feedback (i.e. audio-based). While the first issue poses no significant challenge from a technological perspective, the second is a more complex issue. Dealing with non-visual feedback will typically require using text-to-speech technologies (a full voiced game may be too expensive). Web-based TTS are cumbersome as no reference API or implementation is has been adopted and implemented for the HTML5 specification.

In the case of eAdventure, a TTS API was defined (as part of the Accessibility API) to abstract all this complexity. At start-up the Accessibility Core inspects the context where the game has been launched, gathers information about the guest operative system and platform, and starts a discovery process to investigate potential TTS engines and other assistive tools installed. Considering this information, the available implementations of the TTS API that were packaged with the game are analyzed, discarding those that are not applicable in the current context. Available options are prioritized and iterated through, trying to set-up the best alternative for the user (Fig. 2).

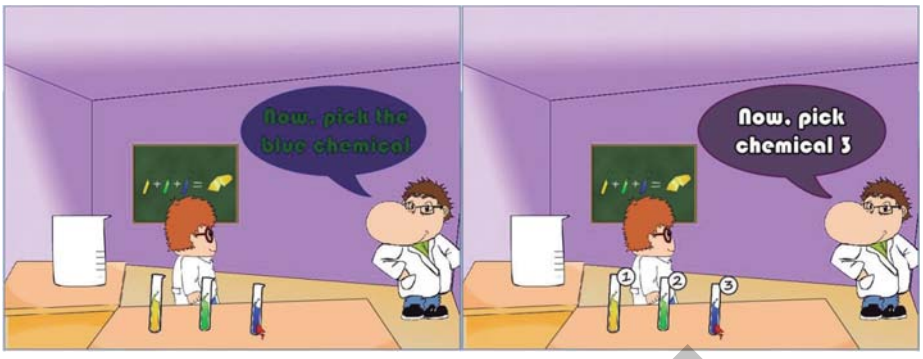

Fig. 2. On the left, the original game screenshot. On the right, adapted version for CVD. Two adaptations are performed: (1) chemicals are identified with numbers instead of colors; (2) green-blue text color scheme, which may be hard to read, is replaced by a high-contrast alternative (Color figure online).

For example, suppose the game is launched in the Chrome Web browser on a Windows Machine (XP or above) and the Accessibility Core discovers, through a browser plug-in, that a screen reader (e.g. JAWS) is installed. It will first try to load an implementation of the TTS that connects to the screen reader, so that the voice used will be familiar to the player. If the process fails, it would try to take advantage of the TTS API Google Chrome browser provides. Next it would try to connect to Microsoft's Speech API (SAPI) provided by the OS, through another browser plug-in. Should all these alternatives fail, then the TTS feature would be disabled, as the game engine does not include a built-in TTS engine.

At start-up the Accessibility Core will also load alternative implementations of the GameObjects that show text on the screen, in a similar way as described in Sect. 5.1. When these accessible objects are rendered, they also invoke the TTS API. In this manner, each piece of text that is displayed on the screen is also played back using the TTS API (if available). Other GameObject effects are also adapted to enhance the audio feedback that is conveyed to the user. For example, when the player enters in a new scene, the game engine will generate a textual description of the scene and reproduce it using the TTS API.

6 Conclusions and Future Work

With the increasing presence of digital games on the web on one hand, and the need for ubiquitous access to content in the other, making accessible games is becoming even harder, as new technological problems are added (e.g. how to deal with text-to-speech technologies and screen readers on different platforms). The introduction of accessibility features could be facilitate by providing game development software that supports the production of games that can be delivered through the web on computers and also on other platforms, like mobile devices.

It can be assumed that HTML5 will eventually make Web games run on every single Internet-enabled device. However, the standard has not fully been debugged and adopted in several platforms, like smartphones and tablets. Thus it is still necessary to provide a native version of the games for some platforms.

In this paper we have presented the eAdventure 2.0 architecture that will allow development of accessible 2D serious games meeting these criteria. The architecture was designed with extensibility and flexibility as key drivers. The main advantage of this approach is that eAdventure 2.0 would easily support extensions to accommodate more types of disabilities and/or new platforms.

This work has only addressed the technical problems related to game accessibility. However, making a game that is enjoyable for players with different types of profiles requires more than having a technology that supports it (e.g. game authors also need to integrate players' special needs into the game design).

The features here presented are currently on a prototype state. We are currently working to reach a more stable status. The next steps will be development and evaluation of accessible games using the eAdventure 2.0 game engine.

Acknowledgements. The Spanish Ministry of Science (TIN2010-21735-C02-02), the European Commission (519332-LLP-1-2011-1-PT-KA3-KA3NW, 519023-LLP-1-2011-1-UK-KA3-KA3MP, FP7-ICT-2009-5-258169), the Complutense University (GR35/10-A-921340) and the Regional Government of Madrid (eMadrid Network - S2009/TIC-1650) have partially supported this work.

References

1. A Gaming Replacement for Those Annoying CAPTCHAs (2013). http://readwrite.com/2012/05/03/a-gaming-replacement-for-those-annoying-captchas. Accessed 13 Feb 2013
2. A straightforward reference for inclusive game design (2012). http://www.gameaccessibilityguidelines.com/
3. Allman, T., et al.: Rock Vibe: Rock Band® computer games for people with no or limited vision. In: Proceedings of the 11th International ACM SIGACCESS Conference on Computers and Accessibility, pp. 51–58 (2009)
4. Arnab, S., et al.: Serious Games for Healthcare: Applications and Implications. IGI Global, Hershey (2012)
5. Bierre, K., et al.: Game not over: accessibility issues in video games. In: 11th International Conference on Human-Computer Interaction (HCII'05) (2005)
6. eAdventure website: http://e-adventure.e-ucm.es
7. Garris, R., et al.: Games, motivation and learning: a research and practice model. Simul. Gaming **33**(4), 441–467 (2002)
8. Johnson, L., et al.: NMC Horizon Report: 2013 Higher Education Edition (2013)
9. Marchiori, E.J., Serrano, Á., Torrente, J., Martínez-Ortiz, I., Fernández-Manjón, B.: Extensible multi-platform educational game framework. In: Leung, H., Popescu, E., Cao, Y., Lau, R.W., Nejdl, W. (eds.) ICWL 2011. LNCS, vol. 7048, pp. 21–30. Springer, Heidelberg (2011)
10. Ossmann, R., Archambault, D., Miesenberger, K.: Accessibility issues in game-like interfaces. In: Miesenberger, K., Klaus, J., Zagler, W.L., Karshmer, A.I. (eds.) ICCHP 2008. LNCS, vol. 5105, pp. 601–604. Springer, Heidelberg (2008)

11. Pempek, T.A., Calvert, S.L.: Tipping the balance: use of advergames to promote consumption of nutritious foods and beverages by low-income African American children. Arch. Pediatr. Adolesc. Med. **163**(7), 633–637 (2009)
12. Sánchez, J., Espinoza, M.: Audio haptic videogaming for navigation skills in learners who are blind. In: Proceedings of the 13th International SIGACCESS Conference on Accessibility (ASSETS), pp. 227–228 (2011)
13. Torrente, J., et al.: Implementing accessibility in educational videogames with <e-Adventure>. In: First ACM International Workshop on Multimedia Technologies for Distance Learning - MTDL '09, Beijing, China, pp. 55–67 (2009)
14. Torrente, J., et al.: Introducing accessibility features in an educational game authoring tool: the experience. In: 11th IEEE International Conference on Advanced Learning Technologies ICALT 2011, pp. 341–343 (2011)
15. Torrente, J., et al.: Preliminary evaluation of three eyes-free interfaces for point-and-click computer games. In: 14th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS), pp. 265–266 (2012)
16. Unity 3D website: http://unity3d.com/. Accessed 15 Feb 2013
17. Westin, T., Bierre, K., Gramenos, D., Hinn, M.: Advances in game accessibility from 2005 to 2010. In: Stephanidis, C. (ed.) Universal Access in HCI, Part II, HCII 2011. LNCS, vol. 6766, pp. 400–409. Springer, Heidelberg (2011)
18. Yuan, B., et al.: Game accessibility: a survey. Univ. Access Inf. Soc. **10**(1), 81–100 (2011)