Extensible multi-platform educational game framework

Eugenio J. Marchiori [1] , Ángel Serrano [1], Javier Torrente [1], Iván Martínez-Ortiz [1], Baltasar Fernández-Manjón [1,2]

[1] Facultad de Informática, Universidad Complutense de Madrid, 28040 Madrid, Spain
[2] Laboratory of Computer Science, Massachusetts General Hospital, Harvard University, Boston, USA
emarchiori@fdi.ucm.es, aserrano@e-ucm.es, {jtorrente, imartinez, balta}@fdi.ucm.es

Abstract. This paper presents an extensible multi-platform educational game framework. This new framework enhances the *point-and-click* adventure game model used in a preexisting educational game framework (eAdventure) by extending it with new game metaphors and interactions. These include mini-games (e.g. puzzles, stories, word-games) suited for a greater variety of subjects (e.g. math, history, science) that are configured through a plug-in architecture. Targeting multiple platforms allows for the transparent deployment and use of the developed games in new mobile devices (e.g. tablets) and other systems of growing interest in the educational community. Unifying these aspects into one platform is a challenging task because it must implement the appropriate balance between expressivity, game production costs and possibilities to re-use successful educational game models. The basic adventure metaphor is used as a backbone to provide a strong narrative to drive games and engage the students.

Keywords: Educational game framework, plug-in architecture, adventure games, multi-platform, serious games

1 Introduction

Educational video games are available in very different flavours, going from simple (and inexpensive) puzzle games to complex (but costly) 3D immersive environments. Different genres are also available, but authors like Amory *et al*. [1] or Dickey [2] point out that the adventure game genre is especially well suited for educational use, because these games promote reflection and improve intrinsic motivation, and thus facilitate the acquisition of knowledge.

Several advantages can be achieved by narrowing the gaming field to specific formulas of proven educational value. First, successful gaming models such as *point-and-click* adventures can be reused, helping to reduce the risk that the lack of rigorous research proving games' educational gain imposes [3]. Second, specific and easy-to-use authoring tools and frameworks can be created and used, which reduces development costs and complexity [4]. These advantages can be of particular

importance in settings where the high development cost is used as an argument against the use of educational games [5].

However, constraining the game genre too much creates its own set of problems. Technical solutions applied in education must be flexible and adaptable in order to fit local needs, teachers' pedagogical approaches, and particularities of each subject or domain (e.g. [6]). Extending the gaming model with new functionality is usually required to meet the requirements of each use case (e.g. puzzles, word games or math challenges). These specific extensions can increase the perceived complexity the tools, increasing the difficulty of creating and maintaining the games, so it should be done only as needed and not for the general case.

We are applying these ideas in the development of the next generation of the eAdventure educational gaming platform[1] (i.e. eAdventure 2.0, formerly <e-Adventure>). eAdventure is a research project that started in 2005 with the aim of providing educators with a game authoring tool specially devised for education. Since its origins, eAdventure has been focused in the *point-and-click* adventure game genre (games like *Monkey Island™* or *Myst™* are classic examples of adventure games). Now the platform is an established product and we are using the input gathered from educators, users and our experience developing and deploying games to improve it.

In this paper we present the approach behind eAdventure 2.0, which allows for the inclusion of other successful game approaches while keeping the focus on adventure and story driven games. eAdventure 2.0 is still a single unified (though hybrid) platform that allows for simple reuse of existing games, and promotes new game development in a cost effective way. A plug-in architecture allows for the incorporation of the new features, allowing not only the customization of the games, but also the customization of the platform for specific use cases, while avoiding the platform overload for general use. These extra features can be new game genres or metaphors, such as puzzles, *fill-in-the-blanks* exercises, simple simulations or other interactive mini-games. More approaches and metaphors can be developed through the use of an API (Application Programming Interface) provided by the eAdventure 2.0 platform. To prepare for changing behaviors in today students and gamers, this platform allows for the automatic deployment of games in different target platforms, ranging from traditional PCs, to web browsers and new mobile devices (i.e. Android *smartphones* and tablets).

This paper is structured as follows: section 2 presents the motivation behind the extensible hybrid platform. Section 3 shows the basic architecture to support this platform. Later, section 4 presents the implementation of the new architecture using available technologies. Finally, section 5 presents some use cases for the new platform advantages and section 6 presents the final remarks.

2 Motivation: hybrid educational game model

During the last 5 years the eAdventure platform (formerly <e-Adventure>) has been under intense development and use. Currently in its 1.3 version, the platform has

[1] http://e-adventure.e-ucm.es

grown to meet the demands and requests of users. This platform was designed for the development of "adventure games" [7] and was used in several experiments with teachers and students [8], as well as in other academic settings.

Experience using the platform has shown that there are many limiting factors that cannot be addressed within the current eAdventure architecture, stimulating a new approach. We propose a new model (Fig. 1) where plug-ins were introduced in the game editor to add flexibility in the creation or modification of adventure games complemented by mini-games. The new model also extends the system to easily export these games into different platforms, including new mobile devices (e.g. Android tablets)

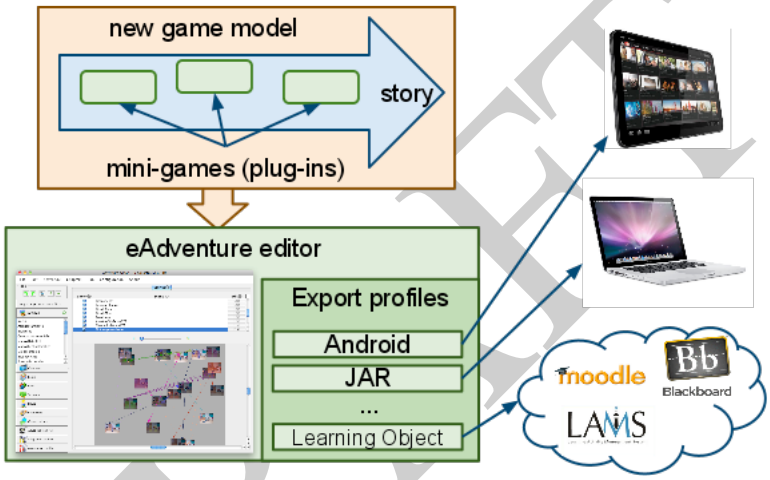

Fig. 1. Abstract model of the eAdventure 2.0 platform for the creation of story-driven adventure games complemented with mini-games.

Teachers who wish to replicate simple games to test specific skills or increase the expressiveness of the platform in other ways usually expect and demand mini-games, plug-ins and pre-made components. The use of mini-games complements the current "adventure" gaming model, by allowing the story (the core of adventure games) to become the driving force that takes the player through the different puzzles and challenges. This changes the game model, from a purely adventure game metaphor, to a hybrid game genre. Several plug-ins are included with the eAdventure 2.0 platform. These range from simple puzzle games to more complex plug-ins to support accessibility aspects within the games. Besides, expert users or those with programming knowledge can create new plug-ins targeting the platform by the use of the API.

Moreover, the e-learning community shows a growing interest in mobile devices (e.g. tablets), as well as emerging web technologies. Targeting these platforms within the old approach required the creation of specific versions of eAdventure that increase the cost, create compatibility problems and are hard to maintain by the eAdventure development team. A new platform-agnostic engine core helps to deal with all these problems.

Other platforms are taking similar approaches, like the *Adventure Maker*[2] and *Raptivity*[3] systems. The *Adventure Maker* platform allows for the creation of 2D adventure games and provides a plug-in architecture. However, it does not provide educational tools (e.g. in-game evaluation features), not all games are cross-platform (just the simplest of them being playable on *iPhone* devices), and only provides a single game metaphor. The *Raptivity* system is more focused on educational content, it is very easy to use and provides many different puzzles or game metaphors. However, the creation of rich adventure games is not currently possible.

3 eAdventure 2.0 Architecture

The multi-platform plug-in based approach used in eAdventure results from a single core engine implementation (Fig. 2, a) and a unified game editor (Fig. 2, d). Different platforms require different customizations of the engine, given that images and other assets must use platform-dependent representations. Besides, this approach also allows for optimizations for each platform (Fig. 2, c), taking into account its constraints and advantages (e.g. more RAM memory in desktops vs. local installations in mobile devices).

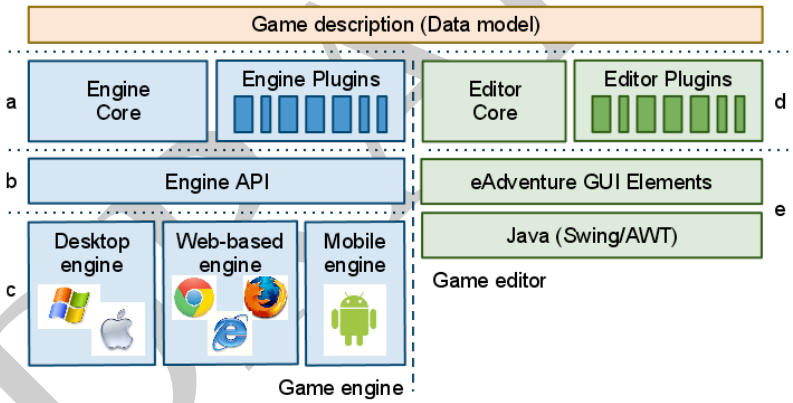

Fig. 2. Overview of the basic eAdventure engine and editor architectures. This includes: (a), engine core and plug-ins: (b), engine API; (c), specific platform implementations; (d), editor core and plug-ins; and (e), editor GUI (Graphic User Interface).

The engine exposes an API to the plug-ins (Fig. 2, b) that allows each particular plug-in implementation to target multiple platforms simultaneously, ensuring that the plug-in, in most cases, will not affect the multi-platform compatibility. At the same time, this API allows for new elements to be registered for the specific platform in which they are needed, in case of specific asset representations (e.g. specific video codec support).

[2] http://www.adventuremaker.com/
[3] http://www.raptivity.com/

This way, the game description (an instance of the abstract game data model) is represented by elements in the eAdventure engine core (e.g. frame animations) or those available through engine plug-ins (e.g. video player). In the engine, these elements use a common API, some parts of which are implemented for the specific underlying platform. The editor uses a set of core components, that allows for the direct use of the basic game elements (e.g. game character) and editor plug-ins that allow for the edition of complex elements (e.g. conversations). To provide a common *look&feel* to all editor plug-ins, a set of GUI elements is provided which includes the common elements required in game edition (i.e. graphic asset management, image and animation preview, etc).

Plug-ins can be added at both the engine and editor levels, adding flexibility to the development process and opening new opportunities to include new value-added features. From the end-user perspective, engine plug-ins introduce elements that need specific representations during the game (e.g. support for specific accessibility features), while editor plug-ins aim to improve usability, reduce the time needed to create a game, or facilitate the reuse and adaptation of complex gaming structures (e.g. conversations are made up of several independent lines, questions, etc.). From the technical perspective, engine plug-ins can help to extend the functionality of the system in different ways, such as adding new accessibility mechanisms or helping to increase compatibility (e.g. a plug-in can allow specific assets to be used in all platforms). Editor plug-ins are mostly used to represent the same information with different degrees of flexibility/complexity in the editor.

4 eAdventure 2.0 Implementation

This section presents the technical solutions used in the implementation of the eAdventure architecture. The most relevant of these include the definition of the API, how dependency injection has been used to achieve modular code, the definition and implementation of an extensible data model and the representation of such a data model at run-time through the use of a dynamic model based on game objects.

4.1 API Definition

To ease the platform extensibility and the support for plug-in development, a high-level API was defined including all the general interfaces describing eAdventure games and game elements. Every part of an eAdventure project is defined using elements available in the API, thus exposing the full functionality of the system for plug-in development. It must be noted that the use of the API in itself is limited to expert users (i.e. programmers) who may want to increase the functionality of the platform, while regular users of the platform (i.e. game creators) need not be aware of its complexity.

For example, the API includes a platform agnostic definition of colours, positions and other parameters within games. At the same time different assets such as images or animations have specific interfaces defined in the API that allow for different

implementations of these elements. In addition, the basic game scene model, which includes different elements, is also accessible through the API, allowing for specific extensions (i.e. the implementation of plug-ins).

4.2 Dependency injection

To deal with the complexity of platforms and devices in eAdventure, links between interfaces from the API and concrete implementations are established through dependency injection. This creates highly decupled code that allows for better testing, simple configuration of different platform-specific code (though configuration files) and favours code modularity. In our approach, *Google Guice*[4] is used, which works through configuration modules. Dependency injection is transparent for the user.

One configuration module is used for each of the platforms where eAdventure is supported. For example, the system provides a *RuntimeImageAssetRenderer* interface, which provides methods to render image assets in the games. Rendering is not accomplished in the same way in a Desktop platform as in a mobile platform like *Android*. However, we can use *RuntimeImageAssetRenderer* interface and its methods without caring about the concrete implementation, which is injected at run-time.

4.3 Data model

The data model is represented by a tree structure that is stored using a serialized xml representation. This xml, although complex, is human-readable which can assist expert users during the creation of games.

Every data element extends the *EAdElement* basic unit (Fig. 3, a). There is no computation logic within these units as we apply the MVC (Model-View-Controller) pattern. Besides, to perform the serialization in an automatic way, Java *annotations* are used to mark the persistent parameters that are to be stored in the XML.

4.4 Dynamic game model: game objects

The data model is interpreted within the engine core to create the game. Every *EAdElement* is translated into a dynamic game object (*GameObject*, Fig. 3, c) that can be drawn and manipulated by the game engine. The specific *GameObject* for each *EAdElement* is obtained though a *Game Object Factory* (Fig. 3, b). Implementing the Factory pattern, combined with the dependency injection technique, allows for the easy addition of new *EAdElement* types with specific *GameObject* representations.

[4] http://code.google.com/p/google-guice/

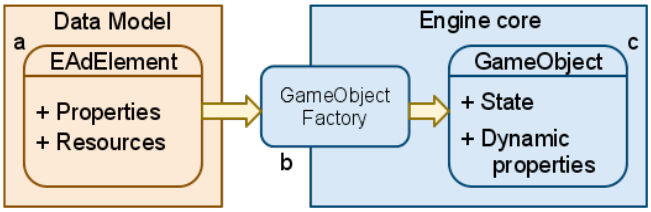

Fig. 3. *EAdElements* (a), which define the game model, are used to create *GameObjects* (c), through a *Game Object Factory* (b). These new elements hold the dynamic state, which changes during the game. The data contained by the *EAdElement* will not change during the game.

As *EAdElement* is the minimal unit in the data model, the game object is the basic unit at the engine core. It has a graphical representation, and holds all the game logic, including user interaction. But still it is an abstract element, because rendering mechanisms need to be defined in every supported platform.

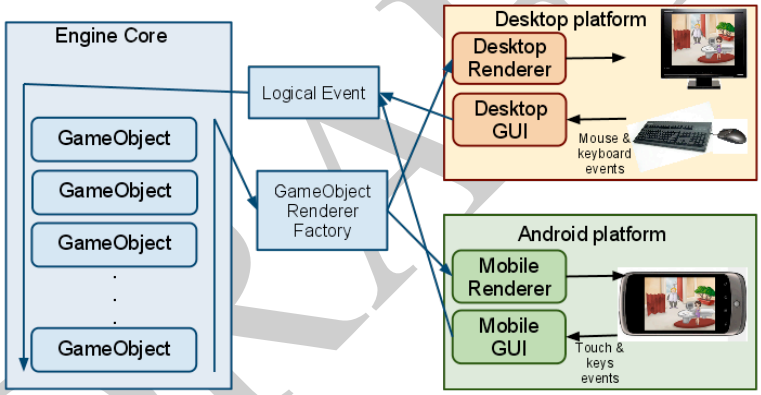

Fig. 4. *GameObject* is an abstract element in the engine core. Must be interpreted for the different platforms to make it available to the user, through rendering, for GUI interactions.

Each platform manages the rendering for every individual game object (Fig. 4). The GUI events sent from the different devices are translated into platform-independent logical event that can be directly interpreted by the *GameObject*.

4.5 Adding Plug-ins

The eAdventure core platform can be extended through plug-ins. An eAdventure plug-in is a set of classes and interfaces extending and using the API. Plug-ins are programmed as independent units that are loaded at start-up in both the editor and the engine (dynamic plug-in installation is not supported). Each plug-in must have a descriptor file that identifies the initialization class that must be called either in the editor or engine.

Two distinct types of plug-ins are considered: editor plug-ins and engine plug-ins. Editor plug-ins are used to manipulate and edit structures already defined by the API or even other plug-ins. These plug-ins increase the usability of the editor (e.g. by providing ready made components) and increase the productivity of the user (e.g. by simplifying the creation of some repeated elements). As an example, the eAdventure platform includes several editor plug-ins to allow the creation of in-game basic types of puzzles. Each of these puzzles are made of basic game objects, but the plug-in editor deals with the composition of these elements in more complex structures. Engine plug-ins can add new functions to the game engine (e.g. different styles of rendering for some game object) or new data elements with their corresponding interpretation during run-time.

5 Case studies

eAdventure 2.0 allows for the creation of complex and varied educational games that can run in multiple platforms. An example of how to include jigsaw puzzles is presented. Also we present plug-ins that can be used to extend the platform with new features such as accessibility enhancements.

5.1. Jigsaw puzzle

One of the plug-ins added to the main editor is a jigsaw puzzle editor. With this editor, a puzzle can be defined only giving a few parameters (i.e. the image formed by the complete puzzle and the number of pieces to be split into). These parameters are used to create the basic elements in the API to represent the puzzle. These elements can be interpreted natively by the engine core, thus requiring no modification to the engine. From the game perspective, the successful completion of the puzzle can be used in the game as a condition to be met in order to advance in the main adventure.

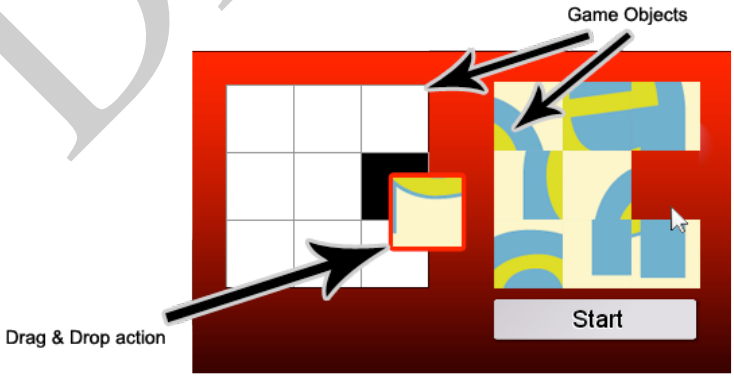

Fig. 5. *Jigsaw Puzzle* is interpreted as a group of standard *Game Objects*. Each of these elements has associated a series of actions (e.g. drag & drop), all defined by the API.

5.2. Adding accessibility features

Engine plug-ins were used to enhanced the accessibility of the platform. One of the stereotypes considered was people with low vision. One of the typical problems that people with low vision have is that they perceive an altered brightness of the colors. To deal with that it is necessary to render the game using a high contrast combination profile that allows them to distinguish game elements (objects and characters) from the background. While this problem is relatively simple to solve in text-based applications (e.g. the Web) it is much harder in highly interactive visual applications like games where images with heterogeneous distributions of colors are the most frequent type of content. In these applications how each element is rendered cannot be decided in isolation, but the whole context of the application must be considered (i.e. the interrelation between elements is important).

Other considerations must also be taken into account. For example, some elements of the game interface (e.g. buttons) need to be larger, along with the font size, which is especially challenging to deal with if the text is embedded directly in one of the game images.

To deal with these needs we added new accessibility features to the eAdventure platform using the plug-in model. When the system identifies that the user needs a low-vision configuration, scene backgrounds are rendered using a low-brightness filter and game elements are rendered using a high-brightness filter. Game elements that are not relevant in that moment are also rendered using the background filter (see Fig. 6).

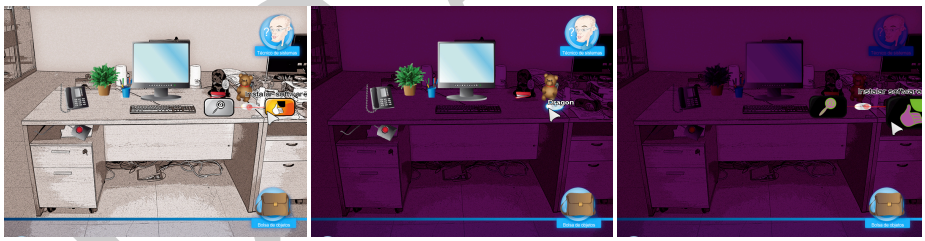

Fig. 6. From left to right, first image shows a scene rendered in normal mode. Second image shows the same scene with low-vision rendering activated. The last image shows the scene when an object is selected (all others are rendered as part of the background to improve usability).

6 Final remarks

The new eAdventure model allows for a hybrid game model creation that combines new game genres with the adventure game metaphor as a backbone. This increases the game expressivity but at the same time maintains the simplicity of game creation and maintenance, achieved though the use of plug-ins.

The plug-in architecture facilitates the development of the whole platform and allows the evaluation of new features in educational games from both technical and pedagogical perspectives.

Next steps in the project are to increase the number of plug-ins available to simplify the development of more complete educational games. For example, we are considering the development of new plug-ins to introduce social features to the platform such as game leader-boards, achievement badges and others to ease the inclusion of new sorts of media into educational video games.

7 Acknowledgements

The Ministry of Education (grants Movilidad I-D+i PR2010-0070 and TIN2010-21735-C02-02) and the Ministry of Industry (grants TSI-020110-2009-170, TSI-020312-2009-27) have partially supported this work, as well as the Complutense University of Madrid and the Regional Government of Madrid (research group GR35/10-A and project e-Madrid S2009/TIC-1650), and the PROACTIVE EU project (505469-2009-LLP-ES-KA3-KA3MP) and the GALA EU Network of Excellence in serious games.

References

1. Amory, A., Naicker, K., Vincent, J. and Adams, C.: The Use of Computer Games as an Educational Tool: Identification of Appropriate Game Types and Game Elements. British Journal of Educational Technology, vol. 30, issue 4, pp. 311-321 (1999)
2. Dickey, M. D.: Engaging by design: How engagement strategies in popular computer and video games can inform instructional design. Educational Technology Research and Development, vol. 53, issue 2, pp. 67-83 (2005)
3. Hays, R. T.: The effectiveness of instructional games: a literature review and discussion. Technical Report 2005–2004 for the Naval Air Center Training Systems Division: Orlando, FL. (2005)
4. Torrente, J., Moreno-Ger, P., Fernández-Manjón, B., and Sierra, J. L.: Instructor-oriented Authoring Tools for Educational Videogames. 8th International Conference on Advanced Learning Technologies (ICALT 2008), Santander, Spain, IEEE Computer Society (2008)
5. Federation of American Scientists. Summit on Educational Games: Harnessing the power of video games for learning. http://www.fas.org/gamesummit/Resources/Summit%20on%20Educational%20Games.pdf (2006)
6. Baltra, A.: Language Learning through Computer Adventure Games. Simulation Gaming, vol. 21, issue 4, pp. 445-452 (1990)
7. Moreno-Ger, P., Martinez-Ortiz, I., and Fernández-Manjón, B.: The <e-Game> project: Facilitating the Development of Educational Adventure Games. Cognition and Exploratory Learning in the Digital age (CELDA 2005), Porto, Portugal, IADIS (2005)
8. Moreno-Ger, P., Torrente, J., Bustamante, J., Fernández-Galaz, C., Fernández-Manjón, B. and Comas-Rengifo, M. D.: Application of a low-cost web-based simulation to improve students' practical skills in medical education. Int. J. Med. Inform. vol. 79, pp. 459-467. (2010)