ON BRINGING ADVENTURE GAMES INTO THE MOBILE GAMING SCENARIO

Marco Furini
Computer Science Department - University of Piemonte Orientale
Via Bellini 25/G - 15100 Alessandria, Italy
Email: furini@mfn.unipmn.it

KEYWORDS
Mobile Gaming, Game on low-resource devices, embedded games, 3GP.

ABSTRACT

Videogames developed in 80s are very popular in the current mobile scenario. These games don’t require several system resources and hence they are well suited for the limited system resources of current cellphones. In this paper we focus on a genre that was very popular in the past, but is not receiving much attention today: the adventure games. These games are proper for the current mobile scenario as they require very little system resources. We propose a software architecture to potentially transform an audio mobile device into an adventure games console. The adventure game is described through MPEG7 and is embedded into a 3GP file. Our approach ensures interoperability, transparency, lightness and security. The simplicity of our approach along with the large availability of audio devices may bring new life to adventure games, which can become an exciting game genre also for today’s users.

INTRODUCTION

A recent research indicates that the mobile-gaming market has grown, since its first appearance in 1997, to a total of $590 millions in revenue in 2005, with a forecast of $1.5 billion by 2008 (see, ABI (2006)). The interest around this market is so high that giants of the game industry, like Electronic Arts and Ubisof , are entering the mobile-gaming market.

Looking at the best sellers video games of the Verizon wireless provider, it is interesting to note that several titles come from mid 1980s: Tetris, Pac-Man and Pong are some examples. This popularity is due to the multimedia features of current cellphones, which are comparable to the one of the gaming console of the mid 80s. This is why, the gaming industry is using old video games for the current mobile scenario.

In this paper we focus on a particular genre of videogames: the adventure games. These games were very popular in the 80s and 90s and although the adventure game market suffered from the introduction of high performing game consoles, today, some adventure games are still released, and many fans develop their own adventure games with tools like Adventure Game Studio and Visionaire.

The main characteristic of these games is that the player has to deal with problem-solving rather than to shoot against something; Being focused on a narrative story, instead of being based on reflex-based challenges, adventure games are well suited for the mobile scenario, as they don’t require several system resources (some adventure games are text-based and don’t even require a graphical screen). Famous adventure games included Zork, King’s Quest and The Secret of Monkey Island.

The contribution of this paper is to propose a simple, transparent, inter-operable and secure mechanism to potentially transform any portable music player into an adventure game console. The idea is to describe the script of an adventure game through a text-based description and then to store this description inside a media stream without modifying its structure. For interoperability reasons, our approach involves standard mechanisms like the Advanced Audio Coding, the MPEG7-DDL for the multimedia content description and the 3GP media container format as the file format.

Our proposal is composed of a script manager, a scene manager and an interaction manager, which cooperate to produce the adventure in a transparent and secure way. Transparency ensures that every audio player can potentially become an adventure game console. In fact, the only requirement is the presence of an enhanced player able to read and understand the stored adventure game description. Hence, only a simple software update is necessary. To achieve transparency, we consider the 3GP file format, which is a container of different multimedia resources. Security is essential to avoid any malicious alteration of the adventure game description and is achieved through the development of a security mechanism that ensures that only a legal 3GP file owner can enjoy the full features of an adventure game and also ensures that alterations cannot be done to the 3GP file. This mechanism is developed with classic security tools (encryption, watermarking and hash functions).

The characteristics of our approach, along with the never-ended interest around adventure games and considering the large availability of audio devices, may bring
new life to adventure games, which can become an exciting game genre of current cellphones.

The remainder of this paper is organized as follows. First, we review the 3GP, AAC, MPEG7 and the used security tools and then we present our proposal.

PRELIMINARIES

In the following we present basics of the standards used in our mechanism: i) the 3GP (the media file container of the adventure game); ii) the AAC (the audio stream format used to encode the narrator voice) and iii) the MPEG7-DDL (the markup language that describes the adventure game). We also review the tools used by our security mechanism.

The 3GP File Format

Our approach introduces the adventure games inside 3GP files. This file format is based on the ISO Base Media File Format (readers can refer to MPEG (2005) for a detailed description) and can contain time-based audio-visual information with an object-oriented structure (each object represents a media object). In particular, the file is made of data structures called boxes, which may contain actual media data or metadata (information to define the media properties). By combining these boxes, different multimedia objects can become a single presentation (called movie in 3GP files). Figure 1 shows a simple example. The file has three main boxes: Movie (moov), Media Data (mdat) and User Data (udta). The moov box includes all the data related to the presentation; it may contain several trak boxes, each of them represents the container of an individual object. The mdat box is the media data container, while the udta box is used to store descriptive metadata (in MPEG-7 format) related to the whole presentation. As we show in the following, our proposal uses a trak box for the audio and a udta box for the game description.

The AAC File Format

Our approach uses the Advanced Audio Coding (AAC) format to encode the vocal narration of the adventure game. AAC is an MPEG standard audio encoding algorithm and can be used inside 3GP files. This format provides high audio quality at low bit-rates and is gaining wide adoption in the marketplace. Readers can refer to Brandenburg (1998), Herre and Purnhagen (2002) for further details about this format.

Here, we simply highlight that an AAC stream is composed of a set of self-encoded audio blocks. This is the main difference with other encoding algorithms, where to decode an audio block, it is necessary to have several (adjacent) audio blocks. Conversely, an AAC player can jump from one block to another without any problem, and, as we better show in the following, this is a fundamental characteristic for our proposal. In fact, in an adventure game, the player might jump from one audio portion to another, depending on the user’s choices.

Security Tools

The goal of a content protection mechanism is to disclose the material only to authorized users. Today, this protection is achieved through a digital right management system that wraps the media file with a control mechanism that is usually based on classic security techniques like encryption and information hiding.

With encryption, the player has to know the cryptographic key (which should be kept secret to the user to avoid unauthorized usage or distribution) to play out the media file. Information Hiding is a technique that hides information inside a media file. It is achieved...
through watermarking techniques as explained by Cox et al. (1997), where a watermarking key is used to generate a random sequence during the embedding process. The random sequence is used to spread out the hidden information (the watermark) into the media file. The knowledge of the watermarking key is essential to extract the watermark. The security of the watermark relies on the fact that it has to be: hidden, imperceptible, directly connected to the media content, statistically invisible, robust and tamper resistant. If all these properties are met, extracting and altering a watermark is hard with no knowledge of the watermarking key.

**OUR PROPOSAL**

In this section we present details of our proposal that aims at describing and storing an adventure game inside a 3GP file in a transparent, light, inter-operable and secure way, so that adventure games may be potentially played over any mobile audio device. Our focus is on adventure games for two main reasons: i) old games are getting new life thanks to their use over modern cellphones and adventure games were the most popular games in the 80s and ii) adventure games are well suited for the mobile scenario where system resources are limited, and where several devices are still released without any graphical screen (think of portable MP3 players).

The architecture we propose aims at transforming an audio mobile device into an adventure game console. As we mentioned, we seek for: **lightness**, **interoperability**, **transparency** and **security**. Lightness is essential for a scenario with limited resources and is achieved through MPEG7-DDL, which allows describing and synchronizing different media streams through a simple and light text-based description; Interoperability ensures that the produced file can be used over several different devices and is achieved through the usage of standard tools like MPEG7-DDL and the AAC format (for the encoding of the vocal narration); Transparency is ensured by the usage of the 3GP files. Security ensures that only a legal owner of the media file can play out the adventure game and also that alterations of the adventure data (or a part of it) are not allowed. It is guaranteed by a security mechanism designed using classic security tools such as watermarking and hash function.

In the following we show how a text-based adventure game is produced, protected and played out.

**Production of the Adventure Games Script**

Let us consider a simple *Knight Adventure: A knight enters into the castle. He is looking for a treasure.* Three doors are before him: one leads to a death-trap; another to lion cage and only one is the door that has to be opened in order to reach for the treasure. Although very simple, this story present a character (the knight) that has to take decisions (which is the right door?) in order to accomplish a task (to find the treasure).

Users can affect the story development by interacting with the story. Since user’s choices are unknown in advance, the adventure game plot has to contain all the possible story developments that can happen. Hence, multiple storyline and multiple user’s choices have to be described in advance. Figure 2 shows the proposed software architecture where the script, scene and interaction manager cooperate to produce an adventure game.

*Script manager*

Since the story development depends on the user’s choices, the story, along with all the possible story developments, is divided into basic audio chapters.

An audio chapter contains the vocal narration of a portion of the adventure and can be of four possible types: initial, interactive, sequential and ending. The initial chapter is the first audio part that is played out; only one initial chapter per adventure game is allowed. The interactive chapter allows users to interact with the story. The sequential chapter does not allow interaction (it just presents information to the user) and the ending chapter is the one that ends a storyline (note that multiple ending chapters may be present). For instance, our *Knight Adventure* may be composed of six different chapters:

- **Ch 1. (Initial)** A knight enters the castle. He is looking for a treasure.
- **Ch 2. (Interactive)** He has three doors before him. Left, right or center door?
- **Ch 3.1. (Ending)** He opens the door and fifty swords hit him.
- **Ch 3.2. (Ending)** He opens the door. A lion appears and roars at him. He tries to escape, but the Lion is already upon him.
- **Ch 3.3. (Sequential)** He opens the door and he takes the stairs up to the roof.
- **Ch 4. (Ending)** A box is partially hidden, but he spots it and he finally finds the treasure.

![Image](image.png)

Figure 2: The proposed software architecture to produce and protect adventure games.
He opens the door and fifty swords hit him.

Table 1: MPEG7 description of an audio chapter.

<table>
<thead>
<tr>
<th>AudioSegment</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;label&gt;“41”&lt;/label&gt;</td>
</tr>
<tr>
<td>&lt;MediaTime&gt;</td>
</tr>
<tr>
<td>&lt;MediaTimePoint&gt;00:22:30&lt;/MediaTimePoint&gt;</td>
</tr>
<tr>
<td>&lt;MediaDuration&gt;00:00:10&lt;/MediaDuration&gt;</td>
</tr>
<tr>
<td>&lt;/MediaTime&gt;</td>
</tr>
<tr>
<td>&lt;TextAnnotation&gt;&lt;FreeTextAnnotation&gt;</td>
</tr>
<tr>
<td>Re opens the door and fifty swords hit him.</td>
</tr>
<tr>
<td>&lt;/FreeTextAnnotation&gt;&lt;/TextAnnotation&gt;</td>
</tr>
<tr>
<td>&lt;/AudioSegment&gt;</td>
</tr>
</tbody>
</table>

Table 2: A text-based description of the possible choices a user can take.

<table>
<thead>
<tr>
<th>ROWSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;ROW num=&quot;2&quot;&gt;</td>
</tr>
<tr>
<td>&lt;QUESTION&gt;What door would you like to open? Left, center or right?&lt;/QUESTION&gt;</td>
</tr>
<tr>
<td>&lt;DEST1&gt;3.1&lt;/DEST1&gt;</td>
</tr>
<tr>
<td>&lt;DEST2&gt;3.2&lt;/DEST2&gt;</td>
</tr>
<tr>
<td>&lt;DEST3&gt;3.3&lt;/DEST3&gt;</td>
</tr>
<tr>
<td>&lt;/ROW&gt;</td>
</tr>
<tr>
<td>&lt;ROW num=&quot;3&quot;&gt;</td>
</tr>
<tr>
<td>&lt;QUESTION&gt;&lt;/QUESTION&gt;</td>
</tr>
<tr>
<td>&lt;DEST1&gt;4.&lt;/DEST1&gt;</td>
</tr>
<tr>
<td>&lt;DEST2&gt;&lt;/DEST2&gt;</td>
</tr>
<tr>
<td>&lt;DEST3&gt;&lt;/DEST3&gt;</td>
</tr>
<tr>
<td>&lt;/ROW&gt;</td>
</tr>
</tbody>
</table>

Adventure Game Protection

The security mechanism is in charge of protecting the content with the following goals: i) every user can listen to a pre-defined story and ii) only a legal owner can enjoy the adventure game. To this aim, the vocal narration (AAC stream) is produced with the following rules:

- A single pre-defined storyline is linearly encoded so that the audio chapters that compose the story can be sequentially found from the first to the last (no interactive chapters are present). This story is in clear and any player can play it out.
- All the other audio chapters are encrypted and stored in the second part of the file. Without the decryption key, the play out produces audio noise.

- The first and the second part are separated by 120 seconds of silence, so that, if played with an ordinary player, the pre-defined storyline would not be immediately followed by audio noise.

The audio stream (AAC format) and the adventure description (MPEG7-DDL format) have to be stored inside the 3GP file. As previously described, a 3GP file is organized through boxes. Hence, the AAC stream is stored inside a trak box, while the static MPEG7 description is stored inside the udta box. In this way, transparency and lightness and interoperability are achieved: thanks to the use of the 3GP file format, every player that can handle 3GP files can handle the AAC stream and the MPEG7 description.

To ensure security, the 3GP file has the following data watermarked in it, as shown in Figure 3:

- $\alpha$, the key for the audio stream decryption: The key is watermarked in the first audio chapter and the player can retrieve it while playing it out. The key is watermarked as it has to be provided to the media player, but not the the user.
- $k$, the decryption key for the audio description $A_{desc}$: Again, the key is needed by the player to decrypt the adventure game description, but it has to remain unknown to the user. The key is watermarked inside the first audio chapter.
- $WID = H(E_K(A_{desc}))$ is watermarked to protect the audio from unauthorized alteration. It ensures integrity via a lightweight verification procedure that compares the hash of the whole adventure game description (encrypted) against the $WID$. Note that, for performance reasons, the use of stronger cryptographic tools is avoided.
\[ \text{CID} \text{, the audio chapter identifier (watermarked inside any audio chapter, with the exception of ending scenes); This ID uniquely identifies each chapter.} \]

Data are hidden into the file using classical spread-spectrum techniques as the ones proposed by Cheng et al. (2002). These techniques require a watermarking key to spread out the data in the media file. Since the software player must know the watermarking key to read the watermarked data, the key is hidden in the player’s code with suitable software engineering techniques (hence users don’t know the key). To avoid illegal copies, each 3GP file is released for a specific instance of a software player.

**Adventure Game Play Out**

To enjoy the adventure game an enhanced player is necessary. In fact, if played with an ordinary player, only the unencrypted audio stream is rendered. The enhanced player is in charge of: i) playing the audio stream, ii) decrypting the encrypted data, iii) interacting with the user and iv) jumping from one audio chapter to another depending on the user’s choices. As shown in Figure 2, to perform all these tasks, the player cooperates with the scene and with the interaction manager. The player retrieves the hidden data during the play out of the first audio chapter: the embedded watermark \( WID \) (for weak integrity verification), \( k \) (for decrypting the adventure game description), and \( \alpha \) (for decrypting the second part of the audio stream). Right after the extraction, the player checks the integrity by computing \( H(E_K(A_{desc})) \) and by comparing it with the retrieved \( WID \). If the integrity check fails, reproduction is interrupted, otherwise the audio description and the second part of the audio track are decrypted. Once the security check has passed, the rendering of the adventure game is done in cooperation with the interaction and the scene manager as described in the following.

**Interaction manager**

The interaction manager is in charge of handling the interactions between the user and the system. It is activated when an interactive audio chapter is played out; by using the scene transition table, it identifies the question to pose to the user and, using the answer and the scene transition table, it gives the next audio chapter CID to the scene manager. Note that the interaction interface depends on the available hardware: in cell-phones the interactions are done through the keyboard, but more complex systems may be available (e.g., a pad, a voice or visual recognition system).

**Scene manager**

The scene manager is in charge of identifying the audio chapter to play out. It controls both the player and the interaction manager. By cooperating with the interaction manager, it gets the CID of the chapter that has to be played out; using this CID, it accesses the MPEG7-DDL description and finds out the corresponding audio segment in the AAC stream; it analyzes the audio-segment description and gives the play out timing information to the enhanced player.

**CONCLUSIONS AND DIRECTIONS**

In this paper we presented a software architecture to transform an audio device into an adventure game console. The proposed mechanism is provided with features like: i) Transparency (no modification to the 3GP file), ii) interoperability (3GP files can be managed by mobile device), iii) lightness (the MPEG7 description requires limited system resources), iv) security (the digital content is protected from illegal usage and modifications). Our proposal can be expanded with multi-modal recognition system (e.g., a voice recognition system, a point-and-click interface) and with multiplayer capabilities.

**ACKNOWLEDGEMENTS**

This work has been partially supported by the Italian M.I.U.R. under the MOMA initiative.

**REFERENCES**


