Processes and tools for sound design in computer games

V. Gal*, C. Le Prado*, J.B. Merland+, S. Natkin*, L. Vega*

* CEDRIC/CNAM, + Cryo interactive

1 Abstract

This paper is a survey of the process and technology used in sound design for video games. The first part of the paper addresses the general state of the art in the design of video games: market and technology constraints, production process, game design and level design practices, game engines. The second part is devoted to the sound aspects of games: sound components, interactive composition, sound engines. We analyze the probable evolution of this technology from mixed recorded sound to generative composition. As a conclusion we discuss how sound designers who are not interested in video games can use the game technology and, reciprocally, what game designers can gain from the development in other fields of interactive music.

2 Rationales

In 1999 the CNAM (Conservatoire National des Arts et Métiers), the Universities of La Rochelle and Poitiers in collaboration with IRCAM (Institut de Recherche et Coordination Acoustique/Musique) and the CNBDI (Centre National de la Bande Dessinée et de l'Image) decided to create a new postgraduate training in video game design and development. To define the contents of this training, the authors of this paper interviewed representatives of all the main activities involved in the game industry. This work was completed by a bibliography analysis. This paper is a survey of the result of this work in the field of sound design. To understand the specific features of sound design in video games, it is necessary to presents general aspects of the game industry design process and technology. The first section of this paper is devoted to this subject. In the second part we focuss the presentation on sound design and the possible evolution of the sound technology is the subject of the last section. The conclusion addresses the relationships between sound design in games and other fields of interactive music.

3 Game industry state of the art

3.1 A few words about the game industry

The computer game industry is one of the most important fields of the interactive multimedia industry. A 21.1 Millions $ revenue is forecasted in 2003 [Gal 02]. The market is split between PC and console games (PS, PS2, Xbox, GameCube…) with two special cases: game for small console (Gameboy, Palm.) and Online persistent games (like Everquest), which are sold on a subscription basis.

In the last twenty years game editors and distributors were generally also producing most of their products and developing the software tools used in games. There is an increasing trend to separate, like in more classical audio visual fields, these three domains into editing company, studio and specialized software editors.

Console manufacturers are the great winners of the game industry: each time a console game is sold, the manufacturer gets royalties. Moreover the console manufacturer controls the game design and development.

3.2 Writing for games

3.2.1 Introduction

Writing for games is a rather difficult task. Of course it is an interactive composition and, as in other fields of open work, the author must leave a controlled freedom to the player. But, in the opposite of the art installation field or interactive music composition, the game industry is driven by marketing goals. Game is mainly entertainment, hence the player must solve non trivial but not too complex problems, leading to a succession of goals in a reasonable amount of time. The player must feel an open interactive work, but should be driven to the game solution. To solve this paradox the game industry has invented several techniques derived from game theory and object oriented specification, which are summarized in this section.

3.3 Game design and level design

A game is first and foremost an imaginary universe. Then the first step of the game specification is to define the main aspects of this universe: Era and style, context of the game, goal to be reached, main types of objects involved, user perception of the game… This part of the game definition is called the game design.
We will consider two opposite examples. In the game Black and White, the player is a God of a small world. His main goal is to be recognized and honored by his people. To reach this goal he can be a good God, helping people in their life or a kind of devil, known for his cruelty. The objects of the game are, for example, the representation of the God (a hand), the people classified by sex and profession, the animals, the houses and monuments, the materials to built house and monuments. There is also a good and a bad angel which are an on line help … Other objects are used as elements of the staging, for example virtual camera allowing several point of views of the same scene (first person, third person)…
On the other hand consider a car racing game such as Gran Turismo 3, in this case, the objects are the set of car, which can be used, the elements of possible racing circuit…

In both cases objects have numerous attributes: geometry and appearance, variable parameters (strength, speed, robustness), type of actions allowed on the object (move right, jump, ring, explode…)…In term of sound design the nature and characteristics of all sound sources are defined at this level. It can be associated with a given object (sound produced by the heroin when she is swimming), or an object by itself (Environmental sounds and music). For example each possible car is associated with a motor sound. The ability to alter this sound as a function of the car speed is also specified at this time.

Next steps of the game specification are called level design. A level of the game is a combination of a virtual space, a set of puzzles to be solved in this space, the main actions to be done by the player to reach a given goal. Generally the level is first defined by the geometry of the space: a given maze, a race circuit. It is possible, if the game uses a 3D sound engine, to specify the acoustic characteristics of each room in the space.

Then the level designer chooses the positions and actions associated with the objects in this level. The goal is either implicit (win the game on this circuit) or explicit (find three elements of a totem to open a gate). In both cases the player is conducted by an implicit scenario, which limits the number of possible effective actions. In Black and White, he can move everywhere in the universe, but the pieces of the puzzle to be solved are well positioned and easy to find (at least for the first levels) to determine a quasi-linear sequence of actions. In the car game, you must follow the circuit until the end of the race!
To keep the sensation of freedom, several solutions are used: first, a set of independent actions can be performed in any order, in more complex games the player can pursue, in the same space, several games in parallel. This is the case in Black and White, where the player (God itself) can have at any time a look to the level of achievement for each task (and rest on the seventh day!).

3.4 Games engines and game platforms

Several companies (Infograme, Darkworks, Virtools…) have developed a programming environment, which allows to develop the game according to the previous steps. Objects specified in the game design are implemented by the programmers and the team of graphic artists as classes in an object oriented library using all the facilities of object oriented programming (heritage, polymorphism…). The level designer defines the geometry using a standard 3D tool such as 3ds max or Maya. A scripting language is then used to specify the level in term of objects in the space. In an ideal world tools like Direct Music producer should be used to define the interactive composition and EAX for sound spatialization. In practice, like for other audiovisual media, sound design is the last piece of the chain.

The implementation of a game relies generally on software called a game engine. A game engine is first a set of software libraries. Each library performs a set of functions needed to code the dynamic and interactive behavior of the game. General purpose game engine includes a 3D graphic engine used to code interactive animation, a sound engine which includes sound synthesis, effect and spatialization functions and more specific libraries like Artificial Intelligence engines used to code the gameplay and physic engines used to simulate physical systems (like cars for example).

<table>
<tr><th colspan="4">Architecture Levels</th><th>Examples</th></tr>
<tr><td colspan="4">Level design scripts editors</td><td>God.move(right, 2);Wait_Event;
On button.click God_Anger:=new(thunder)
God_Anger.lightning, God_Anger.sound</td></tr>
<tr><td colspan="4">Game classes</td><td>class thunder
methods: lightning , sound</td></tr>
<tr><td colspan="4">Game engine library: general purposes game oriented libraries</td><td rowspan="2">

Create_new_object(God, god_geometry.vrml, god_texture.gif,god_voice.wav)</td></tr>
<tr><td>Graphic engine</td><td>Sound engine</td><td>Physic Engine</td><td>IA engine</td></tr>
<tr><td colspan="4">General purpose multimedia libraries (Direct X, Open GL, Open AL…)</td><td>GlMatrixMode(); alsourceplay(source1)</td></tr>
<tr><td colspan="4">Operating system</td><td>Windows, PS2 Monitor…</td></tr>
<tr><td colspan="4">Hardware</td><td></td></tr>
<tr><td>Central Processor, memory…</td><td>Graphic accelerator</td><td colspan="2">Sound card</td><td>PC, PS, GameCube, XBox</td></tr>
</table>

When a collection of game relies on the same universe (Ubi Soft Ray Man games, for example) the studio can define a specific game engine at the game classes level.

In several game engines there is also a software monitor which schedules events defined by the level design script either on a synchronous mode or asynchronous mode. On console platforms the synchronous monitor approach is almost always used. On PC (as Windows 95,98… are highly asynchronous operating systems) the asynchronous approach is possible.

Synchronous Monitor	Asynchronous monitor
Cycle reset timer read the game pad inputs, compute the next state sound and image synthesis wait for time out end cycle	wait for game pad input or time out compute the next state sound and image synthesis reset timer

The use of portable game engines[1] allow to minimize the work to be done to create multiplatform games. The Criterion engine[2] is a typical example of game engine. It contains a 3D Graphic engine, a sound engine originated from Sensora, a Physic engine originated from Karma and an IA engine. A game developed with

[1] http:// 3dgraphics.about.com/cs/gameengines/
[2] http://www.renderware.com/

Criterion game engine can be, in principle, ported on PC, Xbox and PS2 consoles. In practice the differences between the hardware of platforms induce to adapt, even in certain cases change, the design of games.

An other main point is the percentage of the CPU time allocated to the different functions.. Image synthesis is the most expensive function. This is the consequence of a marketing constraint (The quality of the game is often considered from the graphic point of view, in particular the level of detail related to the number of polygons generated for each frame) and a technical goal: the frame refreshment rate must be at least 30 frames/s. Several console manufacturers require that any game edited on the console have frame rate up to 60 frames/s.

This problem must also be considered from a hardware point of view. Gamers PC may rely on a wide number of processors, graphic accelerators, sound boards… A game designed for a PC must adapt its behavior, and in particular the graphic quality to each configuration. Such a property of the game code is called scalability. On the opposite in a console the hardware is fixed and less powerful than the one of expensive game oriented PC configuration.

Platform	Graphics	Sound	IA and other
PC (game configuration)	60	30	10
Console	75	15	10

Typical percentage of relative CPU consumption by functions in a game

Hence a game scheduler is generally defined to allow the time according to various level of importance: Graphic is a periodic task with high bandwidth, sound is a periodic task of low bandwidth, other functions are considered as asynchronous tasks with various priorities.

4 Sounds in Games

4.1 Game sound design

Sound Design for games follows generally the process described in the previous chapter. Principles of the sound design are defined during the game design phase. Unfortunately the sound is generally integrated in the game at the end of the realization, when the whole interactive graphic part is finished. Generally the sound designer has to prepare his elements without being able to test them before this step of the process.

The main goal of sound in most games is to increase the feeling of immersion. Hence ambiance sound and music are generally a main focus of this work. The diversification of foley effects is also important to avoid boring feeling of repetition due to short loops of music. More and more often, spoken dialog between games characters are used. There is a usual recording work, then voices have to be transformed according to ambiance and acoustic considerations. There is a trend to use real time transformations in the two last cases, at least on PC games. The most difficult work in the composition of music for games is the design of transitions. Music and ambiance sound rely on small recorded or midi files loops. When the player moves from a scene to an other or on interactive events, the sound must move from one set of loops to an other. There are no general rules to design transition, but basic music modulation principles and the use of continuous parameter transformations seems to be a good basis of thought [Harland00]. Many games use non-interactive animations, called cinematics, to present the narrative aspects of the game. This part of the sound design is similar to the work done in the classical audio visual field.

4.2 A short history of sound technology for games.

From a simple point of view, one can consider three ages of sound technology in games.

The first generation is related to games which support is ROM cartridge, like Nitendo NES or Gameboy games. In this case the sound design is both constrained by the lack of storage and computing power. Sound dedicated hardware is a rather poor midi FM synthesizer. The sound designer may use a small set of recorded sounds with a low sampling rate and a poor quantification level. So most of the sound design is a composition playing with,

in the best case funny and generally irritating, sound and music loops stored in the cartridge as midi files and played by the synthesizer.

The second generation corresponds to the status of games on consoles like PS1 and PS2, Sony Playstations. In this case there is place available on the CD or DVD to store good quality recorded sounds, the console can offer a wave table synthesizer and a rough 3D sound generator. The sound design is still limited by two main factors: As it was pointed out in the previous section, there is a little percentage of the processing power which can be used by sound real time effects, moreover the RAM allocated to samples is still limited. For example a PS2 can deal with 48 channels of 16 bits audio at a sample rate of 48Khz and has access to 2MB of dedicated RAM. Hence the composition is a mix of recorded music treated as a continuous stream in non interactive animations or dialogues, recorded short loops used in the ambiance music, recorded or generated foley effects with a very simple real time treatment. Games like Woody Woodpecker, Gran Turismo 3, Jack and Dexter, Silent Hill 2 give interesting examples of several kinds of sound design in this context.

The next version of technology start to be used in games on the latest (2002) PC generation. In this cases, even if the sound is still a long way from images, there is much more power left to sound synthesis and sound cards, like Creative Soundblaster, includes efficient DSP processors and are able to perform in real time complex sound effect as 3D localization and reverberation. Hence there is a trend to replace a part of recorded by real time generated sounds and music. This will allow, for example, altering voices or foley effects according to the game context. This evolution is still limited by compatibility goals (a game is designed for several platforms and not only the most powerful PC), the lack of investment for sounds in game production, and the lack of interactive composition tools and good game sound engines. This last point is discussed in the next section.
It is also interesting to point out that this evolution from recorded to generative built objects can also be observed in the image synthesis field. The reader is invited to hear, for example, the sound of Rez, Black and White, Alien vs. Predator games.

4.3 Example of tools

In this section we present briefly three tools mainly designed for the sound design of games, which represents the evolution of this technology.

OpenAL (Open Audio Library) [OpenAL 01] is a standard API for sound programmers. It was mainly defined to offer a portable alternative to the sound part of Microsoft DirectX API (DirectSound). OpenAL principles where derived from existing API and the widely used OpenGL graphic API. OpenAL has been implemented on all major home computer operating systems, and several studio has an OpenAl implementation for consoles. OpenAL allows to describe the main sound aspects of a virtual 3D scene (sound sources characteristics, room acoustic, listener positions) and to program the evolutions of these elements in the coding of a game. It includes calls to sound effects to alter in real time audio streams. It contains also the functions to handle audio sample and play them. Up to now it does not include synthesis function (like DirectMusic). OpenAL is a significant example of the need for portable libraries in the game audio field.

EAX is a technology developed by Creative Labs [Jot 01]. It is first a set of sound 3D and spatialization functions (low level audio rendering API), which includes OpenAL. It is also an authoring tool (EAGLE) for the design of a game acoustic. The sound designer starts to work with the geometric description of a game level. It can be a 3D scene scheme generated by a software modeler (Discreet 3ds max for example). The sound designer uses a graphic interface to describe the acoustic of each room and occlusion between rooms. This description generates a binary description, which can be included in the game program. The game programmer can then access to this data structure through a library. It passes, as parameters, the current positions of sound sources and the listener. It gets, as output parameter, the EAX functions and parameters that have to be called to get the best 3D sound positioning and acoustic.

DirectMusic Producer [Hays 98] is a tool for interactive composition designed by Microsoft. It relies on the DirectMusic API, which is the musical component of Direct X. DirectMusic includes general MIDI and more general functions oriented on generative music composition. It supports the DLS (DownLowdable Sound standard) for wavetable synthesis. DirectMusic producer is a graphical interface, which allows to define complex composition styles, which evolve according to real time events. A style is composed by three kinds of elements: Bands, Motifs and Patterns. The pattern editor is the key of the interface and is also a limited MIDI sequencer. A pattern includes all the elements of a music style (mainly harmony principles) according to score and MIDI parameters. The composer is able to derive, from a given pattern, a set of variations, which can be used randomly

or according to events handling. A pattern is applied to a counterpoint element called the Motif. Many Motifs can also be derived for a given one by simple transformations. The combining of a Pattern and a Motif leads to a score. The Band specification is the interface to specify a set of possible orchestrations. Then Direct Music Producer allows you to bind combinatorial choices between these three elements to game events. Although Producer is based on a classical compositions scheme, it can be used in many other ways to define almost any kind of interactive sound generation. The main criticism made to DirecMusic Producer (apart from being a Microsoft standard), is its complexity.

5 Conclusion

In the last section we have presented some advanced tools for the sound design in the computer game field. In most of the game studios, in the best case, these tools are under investigation. Most of the sound for games, even for PC, are still developed using very limited sound API. Even in DirectMusic Producer, there is no facility to edit graphical patches like in Max MSP or JMax for example. Game sound engines are far from the real time sound tools used in other field of interactive composition (such as IRCAM FTS for example) [Dechelle 98]. Moreover, sound effects used in games are still rather limited to the standard MIDI. There are some technical reasons for such lacks, in particular the hardware limitations of sound boards and processing power available. But it will probably quickly evolve in the next years, and we think that it is important for the game community to look at what has been done in other fields. In the opposite, the game industry has already realized powerful and very cheap tools and sound restitution environment. We think also that composers, sound artist and even audiovisual sound designers can take advantage of all these developments. For example, we are working on the implementation of the sound installation presented in [Natkin01] (a virtual reality sound system based on mobile computers), using the EAX technology on laptop computers.

The design process in games leads to an interesting point of view for interactive composition. DirectMusic Producer relies on very classical musical schemes. In the opposite, one may think to start from the object oriented approach of game design to define a paradigm for interactive composition. Instead of harmony and counterpoint schemes, sound objects defined as audio sources or reflectors with interactive behaviors and environmental characteristics can be the starting point of a composition. The next step of composition is the equivalent of the level design. Sound objects are located in a 3D universe (real or virtual). The freedom left to the listener in his navigation through the space determines the openness of the compositions. This point of view is not quite new: electro acoustic music have taken, a long time ago, the sounds object point of view and the composition for sounds installations [LePrado 02] is often related to this process. But the game design paradigm includes in a structured way these schemes and allows controlling the interactivity effect. It may lead to new authoring tool for interactive composition.

Acknowledgments

We would like to thanks Jean Marc Jot for his comments on the last version of this paper

6 References

[Dechelle 98] Dechelle, Francois, Riccardo Borghesi, Maurizio De Cecco, Enzo Maggi, Butch Rovan et Norbert Schnell, "jMax: a new JAVA-based editing and control system for real-time musical applications", ICMC: International Computer Music Conference, Octobre 1998.

[Gal 02] V. Gal, C. Le Prado, S. Natkin, L. Vega, Processus et outils utilisés pour la conception et la réalisation des jeux vidéo, Rapport CEDRIC, A paraître, juin 2002

[Harland 00] "Composing for interactive Music", Gamasutra, Feb 2000, http://www.gamasutra.com/feature/20000217/hartland_01.htm

[Hays 98] T. Hays, "DirectMusic for The Masses", Gamasutra, Nov 1998, Vol2, N44, http://www.gamasutra.com/feature/sound_and_music/

[Jot 95]Jot, Jean-Marc et Olivier Warusfel, "Spat~ : A Spatial Processor for Musicians and Sound Engineers", CIARM: International Conference on Acoustics and Musical Research, Mai 1995.

[Jot 01]J.-M. Jot. "Perceptual and Statistical Models for Virtual Audio Environments". Proc ACM workshop on Acoustic Rendering for Virtual Environments, Salt Lake City, May 2001.

[Le Prado 02] "Sound Installation and spatialization" To appear in Women in Art and Technology, MIT Press 2002

[Merland 01] J.B. Merland. "Le son dans les jeux vidéo", document Cryo et cours CNAM, htpp://deptinfo.cnam.fr/Enseignment/DESSJEUX/

[Microsoft 02] "DirectMusic", Microsoft Documentation, http://msdn.microsoft.com/library/

[Natkin 01] S. Natkin, A. Topol, F. Schaeffer, Functional Specification of a Distributed and Mobile Architecture for Virtual Sound Space Systems, ICMC 2001, La Habana, Sept 2001

[OpenAL 01] OpenAL Specification and Reference, Loki Software, http://www.openal.org/snapshots/openal/docs/

[Rollins 00] A. Rollins, D. Morris, "Game Architecture and Design", Coriolis Ed, 2000

[Weske 00] J. Weske, "Digital Sound and Music in computer games", http://tu-chemnitz.de/phil/hypertext/gamesound/

Games

Rayman, Ubisoft, 1995 (PS1)

Gifts, Cryo 2000 (PS1)

Woody Woodpecker, Cryo, 2001 (PS2)

Silent Hill 2, Komany 2001 (PS2)

Gran Turismo 3, Sony, Polyphony Digitals, 2001(PS2)

Jack and Dexter, Sony, Naughty Dog, 2001 (PS2)

Metal Gear Solid 2, Komany, 2002 (PS2)

Rez, UGA, 2001 (PS2)

Black and White, Electronic Arts, Lionhead Studio, 2001 (PC)

Alien vs Predators, Fox Interactive, Sierra, 2002 (PC)