

Design and Production of Audio Technologies for Video Games Development

A Thesis
Presented to
The Academic Faculty

by

Claudio Baccigalupo

In Fulfillment
of the Requirements for the Degree
Master of Science

Advisor: Goffredo Haus
Assistant supervisor: Giacinto Attanasio

Corso di Laurea in Informatica
Università degli Studi di Milano
December 2003

TABLE OF CONTENTS

THE AUDIO COMPONENT IN CONSOLE VIDEO GAMES

INTRODUCTION	1
Professional Figures	1
Evolution of Audio Programming	2
A Taxonomy of Video Games	3
Game Design and Audio Design	4
1 GAME CONSOLES AND AUDIO FORMATS	6
Shared Resources in Game Consoles	7
History of Game Consoles	7
The Current Scenario of Game Consoles	10
Digital Audio	11
Audio Compression	12
Audio Outputs	13
Multi-channel Encoding	13
2 SOUND EFFECTS	15
Reproducing Sound Effects	16
Production of Sound Effects	17
Sound Effects and Sound Design	18
Movie-like Sound Design	19
Realistic Sound Design: Issues	21
Realistic Sound Design: Random Noise Synthesis	22
Realistic Sound Design: Synthesis From Physical Data	23
3 3D AUDIO	26
Elements of a 3D Audio System	26
Perceiving the Position of a Sound	27
Binaural Synthesis	28

	Environmental Acoustic Models	28
4	SOUNDTRACK	33
	History of Game Music	33
	Sampled Music and Synthesised Music	35
	Purpose of an Interactive Soundtrack	36
	The History of Interactive Soundtracks	37
	Composing an Interactive Soundtrack	38
	Examples of Interactive Soundtrack	40
5	INTERFACE	43
	Musical Games	43
	Rhythmic Games	44
	Benami	45
	Innovative Musical Games	45
	Audio-games	46
6	DIALOGUES	50
	In Sync with the Animations	51
	Speech Recognition	54
7	NETWORK	55
	Vocal Input	56
	Network Transmission of Vocal Data	56
	Alteration of Vocal Data	58
	Multilingual Support	58
 IMPLEMENTING THE AUDIO ENGINE FOR A MICROSOFT XBOX VIDEO GAME		
	REFERENCES	60

**Design and Production of Audio Technologies
for Video Games Development**

SECTION I

**The Audio Component
in Console Video Games**

INTRODUCTION

How do you develop audio for a video game? The last thirty years have witnessed quite an evolution in this domain. In 1973 Atari's first worldwide success video game Pong produced a single sound: the homonymous "Pong" provoked by the collision between the ball and the racket. A single sound constituted the whole audio engine. Nowadays, a video game production requires an average of 18–24 months to complete, where the audio team, including an average of 3 professionals, weighs on almost 15% of the final budget [39] (1 to 2 million dollars [54]). The whole video games market reaches annual profits of about 17 billion dollars in Europe and the United States only [47].

PROFESSIONAL FIGURES

In the recent past, the music composer of a video game used to be a programmer as well. With time, the two roles have progressively detached and become more specialised, resulting in the dawn of three distinct professional roles:

- the **sound designer**: supervises the audio department, creates the sound effects, sets the master volumes and the filters, manages the final mix;
- the **composer**: creates the environmental music played during the game (sound-track);
- the **audio programmer**: integrates sounds and music inside the code of the video game.

Both composer and sound designer are professional roles inherited, almost unchanged, from the movie industry. On the other side, the audio programmer is a new breed of professional, peculiar to the video games sector, where music and sound effects cannot be just “wedged in” and edited during post-production as with movies, but have to be carefully programmed to actively respond to the game events and to the player control.

EVOLUTION OF AUDIO PROGRAMMING

Audio in video games contrasts with audio in movies in that it is reactive: the audio experience changes every time the game is played and is determined by several events. Recent years have seen a key evolution in how audio reacts to game events; sounds are not anymore the sole result of a “racket/ball” impact, but of a wide range of events driven by the player or by the system itself. Besides sound effects, modern games include ambient reverberations and peculiar acoustic models to improve the sonic quality of each game level. Present-day consoles can apply audio filters in real time and manage “positional” audio, which changes along the position of the player in the level. Some titles also include an interactive soundtrack, whose shape evolves according to the actual game played.

Space Invaders (1978) embodied one of the first reactive audio systems: the less the remaining time, the faster the music was playing. But such “music” was strictly limited by the two audio channels of the console. The latest consoles in the market are cheaper than 200 euros and support hundreds of simultaneously playing voices, filters, reverberations, 3D audio, and much more. The evolution of audio technologies has followed step by step the improvement of the underlying hardware.

The audio programmer has evolved as a professional figure more and more oriented towards a high-level development, who has the adequate tools to abstract the coding from the technical details, to focus on how to make a “globally good use” of audio. In some recent titles, audio is not a simple “embellishment”, but the same core of the game; without audio the game would not be funny, not to say playable. Immersion and interactivity are

key elements in a video game, and a good audio system achieves these goals more cleverly and subtly than a valid graphical interface.

In this thesis, I will describe many of the specifications required to the audio component of a modern console video game. I will retrace several stages of the history of audio in video games to highlight how these requirements have changed over time. I will pinpoint some programming techniques that meet these requirements, abstracting from the underlying hardware. Finally, I will report on the work I have carried out as an audio programmer to develop a video game for the Microsoft Xbox console, emphasising which requirements I have fulfilled and how.

A TAXONOMY OF VIDEO GAMES

The musical features of a video game mostly depend on the genre the video game belongs to. On the whole, console video games can be categorised into six groups:¹

- **Puzzle Games:** the main goal is to solve problems or brain-teasers;
- **Sport Games:** simulations of either real-world or imaginary sports;
- **Action/Kill'em All Games:** the largest time is spent fighting against other players or against the computer. Fights can involve humans, monsters, tanks, aeroplanes, etc.; they can be hand-to-hand, include weapons or other tools. Either the only goal is to fight and win, or the game combines more adventures where the “enjoyment” relies more in fighting than in researching;
- **Platform Games:**

the main goal implies moving through a series of levels, running, climbing, jumping, and using other means of transporting. Players and settings are shown laterally or in the front, rather than from above, thus creating a sense of ‘up’ and ‘down’ as indicated by the word Platform [15];

¹These six categories are suggested in [12], while the Interactive Digital Software Association (IDSA) divides market into 13 genres, and Mark J.P. Wolf distinguishes between 42 different genres [60].

- **Role Games:** the player selects a character and moves it through a series of rooms or connected places, often solving enigmas or quests in order to proceed;
- **Simulation Games:** simulations of real-life experiences, such as piloting flights, building cities, driving vehicles.

Each of these genre has different audio specifications. A simulation game requires realistic sound effects to respond to the game events. In a sport game, on the other hand, spoken comment is fundamental, and should not be repetitive. The soundtrack has normally a secondary role in puzzle games, while it can give hints to the players about particular situations or dangers in an action or a role game. A good video game does not need to implement all the possible audio specifications, but should meet the requirements described in its Audio Design document.

GAME DESIGN AND AUDIO DESIGN

The Game Design is the document developers prepare, when designing the game, to describe all the desired features of the game, either at a detailed level (name of the players, description of the scenery, etc.) or at an abstract level (moods invoked, malignity of enemies, etc.). Similarly to the Game Design, a document is compiled indicating the features required by the audio: the Audio Design. The Audio Design comprehends detailed elements (which objects emit sounds, dialogue lines, etc.), and abstract elements (music timbre, realism of the sound effects, etc.), and involves both musical features and indications on how to implement them, which should be taken care of by the audio programmer. These indications concern:

1. **Game Consoles and Audio Formats:** Which hardware is the game developed for? Which audio resources are available? Which formats should be used for reproduction and for compression?

2. **Sound Effects:** How are sounds reproduced? How do they respond to game events?
Which features are required?
3. **3D Audio:** How are sounds modified based on the position of the player in the game?
Which level of realism does the game aspire to?
4. **Soundtrack:** How and when is background music played? How does it respond to
game events?
5. **Interface:** What is the overall role of audio in the game? What information should
it bear to the player? How can the player interact with audio?
6. **Dialogues:** Does the game contain dialogues? In which and how many languages?
How are dialogues bound to animations?
7. **Network:** Does the game include online matches? Which audio data is intended be
sent on the network? In which format?

The joint work of game producer, audio programmer and sound designer leads to the definition of these points. The game producer sets the duration of the project, the available budget and the main objectives and, similarly to the movie industry, is in charge of the economic development of the project. The audio programmer indicates the time needed to implement each point and fixes the limitations given by the hardware and the development tools. The sound designer, according to the game category and to these guidelines, defines the Audio Design document. Once approved, the programmer will implement its code, while the composer and the sound designer will create the music and sound effects.

In the next chapters, I will illustrate in details each of the aforementioned specifications.

CHAPTER 1

GAME CONSOLES AND AUDIO FORMATS

The first question addressed by the Audio Design document is: “Does the game contain audio at all?”. As the same word evokes, a “video game” is a ludic system based on the presence of graphics, not necessarily sound. *Tennis for Two*¹ (1958) and *Spacewar* (1963) are considered the precursors of the modern video games, and did not contain any acoustic element. As a confirmation, *Magnavox Odyssey* (1972), the first game console in history, did not emit any sound, yet it sold an outstanding 100,000 units in one year. Nowadays audio is a peculiar characteristic of every game and every console.

The term audio implies a combination of soundtrack and sound effects, although a game does not necessarily contain both. The use of audio in video games often reflects a cinematic setting: sound effects underline narrative or descriptive events in a scene, while the soundtrack forges the atmosphere and highlights the key frames. Similarly to movies, dialogues can be present. Music, sound effects and dialogues are inserted in movies during the post-production, thus they do not interfere with any other element in the development. In a video game, on the other hand, they have to be developed together with the rest of the game, since they share common resources. The format in which sound events are stored primarily depends on the technical characteristics of the game console the title is developed for.

¹Created by engineer William Higinbotham at the Brookhaven National Laboratory, it is considered by some the first video game in the history [48], and described by others as nothing but “a more-or-less standard physics demonstration” that “revisionists have turned into a ping-pong game.” [60].

SHARED RESOURCES IN GAME CONSOLES

Within the game code, the part dedicated to audio is commonly known as the Audio Engine. The Audio Engine makes use of both reserved and shared drivers and resources such as processor (CPU), memory (RAM), bus, and network.

Video game programmers tend to think that audio counts for something slightly more than nothing, and that it should use the smallest amount of shared resources, to leave them available for graphics in first place. This prejudice has often led to develop Audio Engines with very small resources, as explained by Pong designer, Al Alcorn [34]:

Nolan [Bushnell] wanted Pong to feature the sound of roaring crowds when a player scored a shot and Ted Dabney (Bushnell's partner) asked for a 'boo and a hiss' when players lost. I said, "Screw it, I don't know how to make any one of those sounds. I don't have enough parts anyhow." Since I had the wire wrapped on the scope, I poked around the sync generator to find an appropriate frequency or a tone. So those sounds were done in a half a day. There were the sounds that were already in the machine.

Current Audio Engines are able to access more common resources; a typical scenario of shared resources used for audio include: *CPU*, to calculate 3D audio positioning, to apply filters and effects in real time, to manage data streams; *memory*, to load samples to be played, from the hard drive or from other peripherals; *bus*, to transport audio data from an external drive (DVD, cartridge, hard disk) to the central memory; *network*, to transmit audio data in an online game.

HISTORY OF GAME CONSOLES

Audio Engines make use of shared resources as well as of hardware components that have been specifically designed for audio. Since the Eighties, game consoles have been equipped with oscillators, sound and noise generators, integrated synthesiser chips; modern consoles also offer dedicated audio processors, private RAM memory and access to mass storage (hard disks, DVDs). A brief chronology of such technological evolution is presented hereafter, focusing on those game consoles which have transformed vanguard into reality (more

video games chronologies are presented in [31, 12, 46, 41, 22]):

Magnavox Odyssey (1972)

The world's first video game console, included six cartridges, two controllers and translucent plastic overlays to be applied on the TV screen to choose the graphics of the game. It lacked of any audio hardware and sold more than 100,000 units.

Mattel Intellivision (1980)

Mattel replied to the successful Atari's VCS/2600 system released on the previous year selling the first console equipped with a dedicated sound chip, the four-channels Programmable Sound Generator (PSG) General Instruments AY-3-8914. Three of the four channels worked with a tone generator, while the last worked with a white noise generator. Each channel allowed for individual frequency and volume controls. This architecture (known as PSG 3+1) was reproduced in many successive consoles, such as Colecovision (1982), Nintendo NES (1983), Atari 520 ST (1985), Sega Master System (1986), Nintendo GameBoy (1989), and Sega Game Gear (1991). Because of the analogue nature of the signal, shared computation resources were almost untouched by audio.

Nintendo NES (1983)

The most successful console of the Eighties, selling more than 62 millions units and 500 millions games, introduced digital audio signals in video games. An audio channel combined a Pulse Code Modulation (PCM) with a digital/analogue (D/A) converter. Audio samples were stored in memory, and loaded by the central CPU before being transmitted to the converter. Audio weighed for almost 10% of the size of the game code.

Sega MegaDrive (1989)

A Frequency Modulation (FM) synthesis chip was integrated for the first time in a console, allowing for six channels of synthesised sound, additive synthesis and support for the

MIDI standard. MegaDrive was also the first system with a portion of RAM memory (8 kB) dedicated to audio. The acoustic result was impressive: while the PSG chip (Texas Instruments SN76489NAN) managed sound effects and occasional music events, the FM chip (Yamaha YM2612) offered six synthesised stereo channels, and the PCM chip played 8-bit samples at a frequency rate of 22 kHz. The Sega MegaCD, a MegaDrive accessory released in 1991, was also innovative, because it introduced the Compact Disc (CD) as a game support, allowing audio samples to be stored in the Red Book format,² while an additional processor (Motorola 68000) managed the stream of data from the CD.

Nintendo SuperNES (1991)

The first Digital Signal Processor (DSP) was introduced, a wavetable synthesiser supporting eight stereo channels with programmable frequencies and volumes, and effects such as reverberations, filters and balance. Nintendo SuperNES included one dedicated audio processor (Sony SPC-700), one DSP (Sony 16-bit), 64 kb of memory and a 16-bit stereo D/A converter; 24 Mb of memory on the cartridge (the game support) were also available for audio. More than 40 millions units were sold.

Sony PlayStation (1994)

The Compact Disc was imposed as a standard game support. A single chipset (SPU) controlled the hard drive and the audio, with 1 MB of dedicated memory, and managed 24 audio channels in Red Book quality, real-time effects such as ADSR³ changes, loops, reverberations and pitch modulations.⁴ Sony PlayStation resists as the best-selling game console

²Red Book is the name of the document containing the specification of the CD-DA (Compact Disc - Digital Audio) format, defined as a stream of 16 bit, 44.1 kHz digital audio. It also details all the technical and physical characteristics of a CD: material, location of the data, tracks, sampling, coding, arrangement of sectors and blocks on the disc.

³ADSR stands for Attack, Delay, Sustain, Release: volume dynamics typical of musical instruments, whose volume increases up to a maximum value before decreasing. Delay is the time between the beginning of the sound wave and the Attack; Attack is the time between the end of the Delay and the moment of maximum volume; Sustain is the level of volume while the sound is playing; Release is the time for the volume to decrease from the maximum level to zero.

⁴The pitch value indicates the frequency change to apply to single notes; a pitch of 100 (respectively, -100) increases (resp., decreases) a note of a semitone.

ever, with the astonishing number of 70 millions units sold.

Sony PlayStation2 (2000)

The DVD was defined a new support for video games. The hardware architecture remained the same as that of its predecessor, with the sole addition of the first hardware codification for multi-channel audio (Dolby Pro Logic II).

Microsoft Xbox (2001)

An internal hard disk and 64MB of RAM memory were included, which allowed for smaller latency during the reproduction of audio samples. Every audio functionality was integrated in one specific unit (MCP) composed of four dedicated processors; five DSP units and a hardware encoder for the multi-channel standard Dolby Digital 5.1 were included. 3D audio could be computed directly via hardware; a microphone could be used as an input interface and the console allowed the transmission of audio signals on the network.

THE CURRENT SCENARIO OF GAME CONSOLES

In the current scenario, three video game systems battle for the market lead: Japanese Sony PlayStation2 and Nintendo GameCube and American Microsoft Xbox. In terms of sold units, Sony console leads with 60 millions produced units,⁵ followed from afar by Xbox (10 millions) and GameCube (9 millions).⁶ Microsoft is at the cutting edge in terms of technology, as shown in Table 1.

⁵In September 2003, Sony officially announced having sold 60 millions units, with a current monthly production of 3 millions units.

⁶Microsoft and Nintendo are still battling for the second position. As of July 2003, Microsoft had declared having sold 9.4 millions Xbox, against 9.6 millions GameCube sold by Nintendo, but with a superior production growth rate. More data can be found in [5, 61].

Table 1: A comparison of last generation consoles.

	Sony PlayStation2	Nintendo GameCube	Microsoft Xbox
<i>general</i>			
Internal architecture	128 bit	128 bit	128 bit
CPU Processor	EE 295MHz	IBM Gekko 485MHz	Intel Pentium III 733MHz
Video Processor	GD 147MHz	ATi Flipper 202.5MHz	nVidia NV2A 233MHz
Rendered polygons	66M/s	6–12M/s	125M/s
Memory	16MB	40MB	64MB
Game support format	DVD	MiniDVD	DVD
Hard disk	n.a.	n.a.	8 or 10 GB
<i>audio</i>			
Hardware audio voices	48 at 48kHz	64 at 48kHz	256 at 48kHz
Multi-channel support	Dolby Pro Logic II	Dolby Pro Logic II	Dolby Digital
DSP Processors	1	1	5
Synthesised audio format	MIDI	MIDI	DLS2

DIGITAL AUDIO

Game hardware has progressively evolved from analogue to digital audio. Digital audio is based on the conversion of sound waves into discrete values, obtained by sampling the wave multiple times a second, and storing the amplitude of the wave at each sampling moment. According to the Nyquist theorem, the sampling rate should at least double the highest recorded frequency to avoid reproduction errors. Digital audio with a sample rate of 44.1 kHz is referred to as of “highest quality” because it surpasses twice the highest frequency humans can hear (about 20 kHz); audio files contained in a common CD have such a quality.

Each sample can be represented with a 16-bit binary number that indicates the amplitude of the wave at the time of sampling, and whose value ranges within $-32,768$ and $32,767$. A wave oscillating back and forth from $-32,768$ to $32,767$ represents the most powerful sound the format can encode, a wave oscillating between -1 and 1 the quietest, and a sequence of zeros denotes absolute silence. The range of values for the amplitude is

large enough to accurately represent even the smallest differences in volume. This sampling method is called Pulse Code Modulation (PCM) and is the most common.

An encoding audio system similar to the PCM format is called Adaptive Pulse Code Modulation (ADPCM). Rather than storing the amplitude of the wave for each sample, the difference in amplitude with the precedent sample is stored. This difference generally requires less bits than the absolute value, thus the same sound can be encoded in a smaller file without losing quality. In average, an ADPCM file is smaller than an equivalent PCM file in a relation of 3.5:1.

The three last generation game consoles support PCM and ADPCM digital audio at 16 bit, 44.1 kHz; in other words digital audio of the “highest quality”. Indeed, both Xbox and PlayStation2 offer 48-kHz hardware channels, with 20-bit sampling, which is even better than the quality of a CD. Nonetheless, typical games use 44.1-kHz samples, firstly because they require less memory, and secondly because any player will hardly make use of a music system where the difference between the two formats can be perceived. 48-kHz audio samples are normally used only by professionals, for instance in Digital Audio Tape (DAT).

AUDIO COMPRESSION

One minute of uncompressed music in PCM format (16 bit, 44.1 kHz) takes up more than 5MB.⁷ Compressed audio has a smaller size, takes less space for storage and less time for data transfer. None of the three main consoles contain a hardware implementation of an efficient compression method; it is up to the audio programmer to include one and integrate it with the audio system. Choosing the right format is a question of balance between the size of the compressed files and the fact that decompression requires computation time and increases latency time to access data.

⁷16 bit * 44,100 Hz * 60 s = 42,336,000 bit = 5.04 MB

AUDIO OUTPUTS

Apart from the Fairchild VES system,⁸ every console in the last thirty years has used the television speakers to reproduced audio signal. With the arrival of digital audio, the unknown and generally monophonic acoustic quality of TV sets has proven inapt to faithfully reproduce game music and sound effects. Personal computer games, on the other hand, have been usually played with the support of two or more speakers, yielding a better acoustic result.

Modern consoles present three audio outputs: a RF output (for the television set), a stereo output (for headphones), and a digital output. The last one, combined with multi-channel encoding systems, is able to offer an excellent acoustic fidelity, and to exploit common Home Theater systems, such as Dolby Surround, Dolby Digital and Digital Theater System (DTS). Research studies quantify in 30% the number of players who owns any of these systems.

MULTI-CHANNEL ENCODING

Multi-channel systems are based on a simple principle: they read one signal in input and decompose it in many output signals, one for each available speaker.⁹ Dolby Surround, the first widespread multi-channel system, allows to “thread” up to four distinct channels (central, left, right, and monophonic rear) in one stereophonic track. The encoding algorithm is analogue, based on the difference in phase and level between the four channels to encode. The result has some limitations: the surround channel is unique and monophonic, its frequency response is constrained in the 100–7000 Hz range (thus excluding the highest frequencies), and the separation between channels is limited to about 30 dB (that is, a sound addressed to one speaker is partly reproduced on the others as well).

Dolby Pro Logic II is a more advanced system, which digitally implements the decoding

⁸Fairchild VES (1976) reproduced sound through integrated speakers, not through television speakers.

⁹More information on multi-channel encoding can be found in [21].

phase, breaking down the limitations given by the analogue nature of Dolby Surround. Two rear channels are stereophonic, and have no limitation in the frequency response, while the separation of the five channels is net. Sony PlayStation2 and Nintendo GameCube include the Dolby Pro Logic II decoding system.

The most recent Dolby system, known as Dolby Digital 5.1, is completely digital, from the encoding (using 16-bit, 48-kHz samples) to the decoding in the speakers. Six channels are present: three on the front (central, left, right), two on the back (right, left), and one (the “.1”) reserved for low frequencies (sub-woofer). Surround channels are stereophonic and without frequency limitation. Digital samples are compressed with a factor of about 10:1. Microsoft Xbox includes Dolby Digital 5.1 decoding.

An alternative to Dolby Digital 5.1 is the Digital Theater System (DTS) decoding system. The number of available channels is the same (5.1), and a compression system is used as well. However the sampling is done at 20 bit, 48 kHz; thus the resolution of the information is slightly superior. The compression algorithm is also less destructive, and this hones the sound quality. In optimum conditions, DTS contains three times the information of an equivalent Dolby Digital signal. None of the three consoles supports the decoding of a DTS signal via hardware, but third-party libraries are available for decoding DTS via software on Sony PlayStation2¹⁰ and Nintendo GameCube, with a computational cost of about 5–10%.

¹⁰Implementing a DTS decoder for Sony PlayStation2 is described in [17].

CHAPTER 2

SOUND EFFECTS

Pong (1972), the first successful video game, included nothing but a single homonymous sound. Still, without that unique sound effect, not as many players would have been attracted. The game was first tested in a Californian pub in 1975, where people got immediately curious at the machine, and hypnotised by the sound [11]:

One of the regulars approached the Pong game inquisitively and studied the ball bouncing silently around the screen as if in a vacuum. A friend joined him. The instructions said: 'Avoid missing ball for high score.' One of [them] inserted a quarter. There was a beep. The game had begun. They watched dumbfoundedly as the ball appeared alternately on one side of the screen and then disappeared on the other. Each time it did the score changed. The score was tied at 3-3 when one player tried the knob controlling the paddle at his end of the screen. The score was 5-4, his favor, when his paddle made contact with the ball. There was a beautifully resonant "pong" sound, and the ball bounced back to the other side of the screen. 6-4. At 8-4 the second player figured out how to use his paddle. They had their first brief volley just before the score was 11-5 and the game was over. Seven quarters later they were having extended volleys, and the constant pong noise was attracting the curiosity of others at the bar. Before closing, everybody in the bar had played the game. The next day people were lined up outside Andy Capp's at 10 A.M. to play Pong.

Since then, the number and quality of sound effects in video games have grown considerably. Many action and simulation games include sound effects recorded from real world situations in order to simulate environments, means of transporting, animals, people. Dialogues, which were initially synthesised and later on recorded, are now completely integrated with sound effects and music. A game acoustic environment can be very complex, while a single sound effect can change the outcome of a game. Quoting Brian Schmidt [51]:

Imagine you are in a cave, and your torch has extinguished or has been stolen. During minutes, the screen is black, and you have to listen to get out; you have to fight your enemies with sound.

REPRODUCING SOUND EFFECTS

Sound effects are played according to the game events. In order for the response to be immediate, the system should ideally be able to reproduce any sound effect at any given time. This requirement is hard to fulfil even on the most recent consoles, and one of the tasks of the audio programmer is to design a system able to respond in the best possible way to every game event.

The first element to consider is the device where sound effects are stored. Hard disks, for instance, have an average access latency time of 30 ms and a throughput rate of 12 MB/s, while DVD players have ten times the latency and a quarter of the throughput rate. Whenever possible, the programmer should store audio on low-latency devices.

The compression format is another relevant factor. Compressing audio has the advantage of reducing the size of the samples, but the disadvantage that decoding is required prior to the execution, which can generate delays when responding to events. The best choice is usually a hardware-decodable compressed format, such as ADPCM, supported by every last generation console. The programmer is free to implement other proprietary decompression systems using standard formats (such as Ogg Vorbis¹ or MP3²), being aware that software decoding always worsens the response time.

The audio programmer can also exploit several techniques to improve physical access to audio files and to reduce latency time, for instance:

- samples stored on DVDs should be all positioned on the same disc layer and in the inner disc section, where the head can read faster;
- samples stored on hard disks should be packed in large sequential chunks, readable

¹Ogg Vorbis is an open-source audio compression format, developed to replace proprietary coding formats such as MP3, VQF and AAC. Ogg Vorbis compressed audio files have a smaller size and, according to different measures, a better acoustic quality than MP3 files.

²MP3 (MPEG-2 Audio Layer-3) is the most common standard of audio compression, with a size reduction of about 12 times, and an almost undetectable loss in fidelity. The MP3 standard has been developed under the control of the Motion Picture Experts Group (MPEG) and is formally an ISO standard.

by the head in few consecutive passes;

- when RAM memory is available to the audio system, average response time can be lowered by maintaining only the beginning of each sample in memory, and loading the remaining part from the mass storage while the first part is being reproduced.

PRODUCTION OF SOUND EFFECTS

Producing sound effects is a task entrusted to the professional figure known as the sound designer, who creates in average about 500 different sound effects for an action or platform game.³ The creation occurs in harness with the audio programmer: together they compile and maintain an update listing of the sound effects, with these descriptive fields:⁴

- name of the source audio file
- common name of the sound effect
- position
- category (weapon, climate, vehicle, ...)
- description
- associated animation file
- duration
- audio format
- file source (recording, sound library, ...)
- version

³Several techniques to produce sound effects are described in [55].

⁴An optimal organisation of the production flow in an audio department (music, sound effects, dialogues) is described in [4].

SOUND EFFECTS AND SOUND DESIGN

Apart from creating and testing sound effects, the sound designer manages the sound design of the game, that is, the high-level acoustic impact of the game.

Ico⁵ is an example of a recent title whose brilliance resides precisely in the choice of a sound design which perfectly matches the game environment. The permeated background music themes are rare and mostly remark the main episodes. As for the rest, the sound is limited to a few environmental effects: seagulls shouting in the mist overlooking the sea, whooshing trees caressed by a mild wind, the steps of the characters echoing in a vast empty hall and crackling torches enlightening a corridor, outstandingly contribute to drag the player along the adventure.

The sound designer commonly faces three typologies of problems when planning the sound design: how to manage audio priorities, global levels and spectral values.⁶

Audio priorities define which audio components hold a key role in the narration of the story. Some questions that need to be answered are: should environmental effects be predominant, or is the game guided by the soundtrack? What is the task of dialogues, and what is their priority? Can sound effects abruptly interrupt dialogues or not?

Global levels define the volume of the different components of the game (sound effects, music, dialogues) and the overall dynamic musical range, that is, the difference between the loudest and quietest levels. Levels can also be assigned to groups of sound effects; for instance the loudness of the player's steps in relation to the environmental sounds. Global levels are assigned in subsequent steps: firstly the sound designer assigns each sound effect a proper volume, then all the volumes are levelled off and normalised. Some titles let the players control these levels using a proper menu where music, sound effects, and dialogues volumes can be set.

Managing spectrum values means to control the frequency spectrum of sounds in order

⁵Winner of the "Game Developers Choice Awards 2002" for "Game Innovation Spotlight".

⁶These three typologies have been first highlighted in [57].

to eliminate problems related to non-linear mixing. For instance, the sound designer can decide to apply an equalisation curve in areas where constant music and heavy environmental sound effects are played, in order to reserve some space in the frequency spectrum for other immediate sound effects or dialogues. Another good technique is to attenuate the medium-low frequencies to enhance clarity when dialogues are present.

The sound designer can choose, according to the game typology, between two different sound effects approaches: a movie-like approach and a realistic approach. In the first case, “overacted” sound effects are allowed to aggravate or underline the peculiarity of a scene as in movies: a slap can resonate over measure, or a shout can break the calm in the middle of a silence. In the second case, the intent is to faithfully reproduce the sound of the real world, without this kind of tricks. Both movie-like and realistic sound designs imply some programming issues the audio programmer should deal with.

MOVIE-LIKE SOUND DESIGN

Unlike movies, video games do not have a linear plot: events can occur at any moment, and a countless number of times. To maintain a cinematic setting, sound effects should not be repetitive, and the right balance should be maintained in distinct occasions: for instance, an arrow cast in the middle of a battlefield should not have the same volume as one cast in the absolute silence.

Avoiding repetitions

The straightforward solution to avoid repetitive sounds is to create many different audio files for the same purpose (variations); however this method results in more cons than pros. First of all, this approach can indefinitely increase the work of the sound designer, who would only know the number of required sound effects but not the number of audio files to record. Moreover, the administration of the shared resources would be badly affected, since more audio files require more memory. Finally, effects which are very often repeated

throughout the game would not improve much: how many audio files should be recorded, for instance, for the player's steps not to sound repetitive?

The audio programmer is in a charge of providing a valid method to manage sound variations. In particular:

- the number of audio files to record for each sound effect (variations) should be fixed in advance, in relation to the available resources;
- variations should be intelligently played throughout the game, for instance playing some variations less often than others, to produce an effect of “surprise” in the players;
- variations should be appropriately loaded or unloaded from memory, with usually one variation constantly present in memory, ready to be played when necessary.

The audio programmer has also the faculty to add real-time variations of the loaded audio files in the game code. A simple method consists in slightly varying the pitch of a sound effect every time it is played: this allows human ears to clearly distinguish the effect, and to suffers less from repetition. Other filters can be applied to obtain similar results, depending on the tools offered by the underlying game console.

Modifying volume to remark scenes

In movies, the volume of a sound effect inserted in a scene is an instrument that can underline a particular narrative situation. In horror movies, for instance, unreal silences are suddenly broken by exaggeratedly powerful screams. In video games, a similar effect is not as easy to implement: the volume of a sound effect has to be set in real time, based on the position of its source, the narrative situation and the global volume. When designing the game, the sound designer has the faculty to compile a list of specific cases where this technique of modifying an effect's volume should be applied to increase the emotionalism of a scene.

Balancing the whole set of sounds

While in movies the sound of a single scene is balanced in post-production, a good volume balance is achieved in video games only after several trials, testing audio levels both in environments with few sound effects and in saturated settings. An excellent video game mixing sometimes requires more time than a movie mixing. Initially the sound designer sets the pad value for each effect, ranging from -100 dB (silence) to 0 dB (full volume). During the game, the effect is played at this volume; if the game includes a 3D audio system (described in the next chapter), the pad increases in relation to the distance of the effect and to the spatial characteristics of the game level. At each moment, the global pad is the sum of the pads of all the effects in play.

The global pad can reach a positive value (higher than 0 dB) when too many effects are played together: in this case, the sound “distorts”. To avoid such an unpleasant situation, a maximum limit of simultaneous effects can be fixed. When playing a sound effect would lead to exceed this threshold, one or more playing sounds are stopped to prevent pad from surpassing 0 dB. The priority of each sound is indicated in the sound design, and used to decide which sounds to stop when necessary.

REALISTIC SOUND DESIGN: ISSUES

The techniques illustrated so far endeavours to thrill the player with a non-realistic application of sound effects. Many titles do not long for this behaviour, at least not for every sound effect: games set in 3D worlds usually have the implicit goal of simulating real-world sounds as faithfully as possible. Again, the audio programmer has to consider the problem of repetition. In the real world, events never emit the “same” sound twice; examples are a barking dog or a bouncing ball. In a virtual world, on the other hand, digital audio samples sound the same every time they are played. Rather than just playing digital audio samples, the audio programmer can introduce automatically synthesised sounds to increase the realism of game scenes. Hereafter, two techniques are illustrated that generate non-repetitive

sounds: the first is based on mathematical grounds, the second on physical grounds.

REALISTIC SOUND DESIGN: RANDOM NOISE SYNTHESIS

How can a large and continuous sound such as the noise of the wind be reproduced throughout a game without causing a sense of repetition? Reiterating many times a few “wind” samples is not an acceptable method. The proposed solution consists of synthesising the sound of the wind using random noise generators, where changing of a few parameters along time avoids repetition.⁷

First, a pseudo-casual numbers algorithm is launched to generate a random noise: the random values are used to define the amplitude of a wave at each moment. The Linear Congruent Algorithm (LCA), for instance, is able to generate a sequence of 232 non-repeated values, and the result is a noise known as white noise, which sounds like a static buzz (shshshsh). The LCA algorithm is excellent in the sense that, working at a frequency of 44.1 kHz, presents the first repetition only after 27 hours.

A red noise wave is obtained instead taking a random value as the amplitude only every n samples, and interpolating between two random values to obtain the internal missing values. The frequency at which a random value is taken is called noise phase; the highest the noise phase, the more chaotic the noise.

With two noise generators (one white, one red) and a low-pass filter, a brilliant wind sound effect can be obtained, which turns to be more realistic and less repetitive than one created with digital samples. A white noise generator as a sound source and a low-pass filter with a cut-off value of 1,000 Hz and a Q value⁸ of about 162 create the effect of a wind with irregular whistles. To simulate the increase and decrease of flurries, the cut-off value can be obtained in real time from the red noise generator, rather than being fixed. It is also possible to manually set how rapidly the wind floats and the size of the fluctuations.

⁷A full description of this technique can be found in [10].

⁸The cut-off value is the frequency at which the signal level is reduced to 3 dB. The Q value is the resonance degree at the cut-off value frequency.

REALISTIC SOUND DESIGN: SYNTHESIS FROM PHYSICAL DATA

Neither digital samples nor noise generators exist in our world, where sound waves are the consequence of the vibration of solid or elastic bodies.

Many action, simulation, sport and platform games are set in tridimensional worlds, and follow the classical laws of Physics: each solid element (players, objects, floors, walls, etc.) behaves with respect to its physical properties (mass, volume, elasticity, shape). For instance, objects react differently to impacts: they either break, bend, move, bounce, or stay still, depending on their physical nature. Thus, graphics obeys to physics rules. Similarly, physics data can be used to command the audio of the game. If a game contained a description of all the physical properties a realistic world, then sounds could be generated from the physical data of the objects, rather than using samples or noise generators.

This approach is theoretically excellent, but inapplicable in practical terms, because of the size of the required data and of the complexity of the calculations to compute. Yet, it is worth being analysed because it offers the outstanding advantage of an automatic and faithful synthesis of real-world sounds, without “tricks” such as digital samples and noise generators. Hereafter a method introduced in [33] is illustrated which concretises this idea.

Synthesising Sounds From Physically Based Motion

This technique is able to reproduce sounds which are generated by the motion of solid objects. It can be applied in virtual environments where objects are animated on the basis of physical data. This technique is comprised of three subsequent steps.

In first place, the motion of animated objects which are to generate the animations (visual component) and the sounds (acoustic component) are calculated. This computation can be done with any non linear method with finite elements that satisfies these properties:

- temporal resolution: the simulation should use an integration time-step smaller than approximately 10^{-5} s, otherwise it would not be able to adequately model the highest frequencies humans can hear (about 20 kHz);

- dynamic deformation modelling: the method should model elastic deformations, whose vibrations are the main responsible for physical generated sounds; rigid body simulators are not permitted;
- surface representation: the simulation technique must contain some explicit representation of the object surfaces, since there vibrations transition from the objects to the surrounding medium;
- physical realism animation must produce a visible motion; since sounds are generated based on the animation, motion should be visually acceptable for the intended application to have an acceptable acoustic component.

Once the solid objects motions have been calculated, the motion of the surfaces is analysed to determine how the pressure of the air will be affected; fluctuations in pressure that correspond to audible frequencies (20–20,000 Hz) are isolated using pass-band filters.

Finally, the propagation of the generated acoustic pressure to the listener is calculated. The intensity of the sound is attenuated based on the distance, the gaudiness of the air and the position of the listener in space.

The described technique is very interesting since it can generate both visual and acoustic components of a game in parallel, based on physical data. The computational cost required to synthesise sounds in addition to graphics is mostly irrelevant, and never surpasses 5%. Different tests have verified that synthesised waves have a very similar spectrum to the equivalent real-world sounds.

Unfortunately, the technique requires elastic deformations to be modelled in the game, which makes the method unsuitable for any actual video game. Even titles with the most advanced physical systems represent elements with rigid bodies, since the visual outcome of using elastic bodies is not yet considered good enough to justify its computational cost. However this limit is slowly coming off: the first physical engine for video games supporting elastic bodies has recently been distributed, and next years will see the publication of

titles which will implement physical elements using elastic bodies.

Synthesising sounds from physically based motion also presents some limitations, for it is not able to simulate reflected or diffracted sounds, which play a significant role in our perception of an acoustic environment. Tracing an analogy with the synthesis of images, the proposed method is equivalent to a “local” illumination model; adding reflex and diffraction would make it a “global” illumination model. Somehow, though, calculating a “global sound” may result harder than calculating a global illumination, because sound waves cannot be assumed to propagate instantaneously. Moreover the actual listener model is designed to receive every acoustic pressure wave with the same strength from each direction; in the future this model will be substituted with a more complex and realistic one which, similarly to human ears, will be able to distinguish acoustic information based on their origin.⁹

⁹Sound synthesis based on physical data is also detailed in [13].

CHAPTER 3

3D AUDIO

A 3D audio system reproduces sounds through simple headphones or speakers, offering the perception they come from arbitrary points in space. In a traditional stereophonic system, pan variations “shift” the point where the sound comes from within the space included between the two speakers, creating the illusion of a “phantom” source that moves around the listener. However, a stereophonic system is not able to position the sound sideways or behind the listener, not even beyond or above. This is the goal of a 3D audio system.¹

ELEMENTS OF A 3D AUDIO SYSTEM

Many modern games are set in tridimensional worlds, even in genres that would have used 2D representations in the past.² The fundamental idea of 3D audio is to offer a way to use the same 3D representation of the world for every aspect of the game, sharing the calculated data for both the graphic and the acoustic components.

These are some key elements to understand a 3D audio system [29]:

- single listener: there is a unique “listening” object that represents where the sound is received in the 3D environment. The listener has a position and an orientation in the game space, which are used to calculate the correct output for the speakers;
- multiple sources: each object produces a sound in the space and is represented by a source;
- output determination: whatever the speakers’ configuration of the player may be, 3D

¹An introduction to different 3D audio implementations (A3D, EAX, DirectSound) can be found in [53].

²More information on 3D audio in video games can be found in [42].

audio has to be faithfully reproduced. To do so, the system first calculates the panning and panning values of each audio source with respect to the listener and then distributes the sound adequately on the different speakers.

PERCEIVING THE POSITION OF A SOUND

To proceed with 3D audio, it is necessary to understand how humans localise sound in space using nothing but two ears [26]. A sound generated in the space creates a wave which propagates to the listener's ears. When the sound comes from the left, the wave reaches the left ear beforehand, thus the signal on the right ear is delayed with respect to the left ear. Moreover, the right signal is attenuated because of the head obstructing the wave. Both signals are then subject to complicated processes of interaction with the human body, the head and particularly the pinna (outer ear). Various folds in the pinna modify the signal content, strengthening some frequencies and softening others, according to the incidence of the wave on the ear. Each ear acts as a complicated tone controller; humans unconsciously make use of the delays, amplitude differences and tonal information of each ear to determine the position of a sound.

The transformation of a sound from a point in the space to the auditory canal can be accurately measured: these measures are called Head Related Transfer Functions (HRTF), and are estimated inserting miniaturised microphones in the auditory canals of a man or a human dummy [27]. A specific signal is emitted from the speakers, and the signal recorded by the microphones is stored. Each HRTF consists of hundreds of numbers describing the time delay, the amplitude and the tonal transformation of a sound with respect to the specific position of the source. Repeating the experiment for different positions leads to a HRTF database which describes the characteristic transformation of a particular head.

BINAURAL SYNTHESIS

A 3D audio system works by simulating the process of human hearing, but reversing the calculated transformations. A sound processed by a 3D audio system is filtered using the inverse HRTF function of the point in space where it comes from, in order to mislead the brain to think the signal comes from that point. This process is also called binaural synthesis.

HRTF functions are one of the numerous existing models to cheat the human brain to perceive a sound from different points in space that have been implemented in recent years by different production houses.³ For instance the MCP chip-set included in the Microsoft Xbox console allows to directly use positional 3D audio thanks to the Sensaura hardware implementation of HRTF functions [52].

ENVIRONMENTAL ACOUSTIC MODELS

A 3D audio system does not only deal with sounds in space, but also with modelling the acoustic environment where the game takes place. It has to regard of the viscosity of the mean (air, fog, water), of the present solid geometry (corridors, open spaces, caves), of the presence of objects in the scene (blocking, smoothing, letting sound through). The most relevant acoustic elements to describe an environment are: distance factors, Doppler effect, reverberation, obstruction and occlusion.

Distance factors

A feature of 3D audio engines is the distance effect. The farther the source of a sound, the higher the attenuation. The sound designer establishes a “minimum distance” value: when the source is within this threshold, the sound is played at the original volume (it is too close to be attenuated). When the source is beyond this imaginary frontier, the related sound

³A more precise but computationally expensive method based on beam-tracing techniques is described in [23].

loses half of its strength for each unit of measure crossed (in the real world, 6 dB for each meter). The volume keeps lowering up to a “maximum distance”, when it is too far away to be heard. The higher the maximum distance is, the longer a sound can be perceived. Different sound sources have different minimum and maximum distances; for instance, a fly becomes silent after 50 cm of distance, while the noise of an aeroplane disappears after several kilometres.

Doppler Effect

A 3D audio engine can also simulate the Doppler effect, which occurs when the length of a sound wave changes while the source comes closer or moves farther from the listener. When the source comes closer, the wavelength decreases, and vice versa. Race or simulation games can increment their realism simulating this effect.

Reverberation

When an object in a room produces a sound, the acoustic wave expands from the source reaching walls and other objects, where its energy is absorbed or reflected. All the reflected energy is technically called reverberation. Assuming a direct path exists from the source to the listener, the listener first hears the direct sound, followed by the reflections of the nearest surfaces, called first-order reflections. After some tenths of second, the number of reflected waves become very large, and the resulting reverberation is characterised by a dense ensemble of waves travelling in every direction, called diffuse reverberation. The time needed by the reverberation to fall 60 dB below the initial level is called reverberation time. Generally, reverberation in a small room disappears much faster than in a large room.

Reverberation is an important acoustic phenomenon. There is at most one direct path from the source to the listener, while millions of undefined indirect paths can be present, particularly in a room where a sound wave bounces hundreds of times before being absorbed. The perception of the reverberation depends on the type of sound and of the type of environment. Reverberations containing many high frequencies are associated with rooms

with solid, reflecting walls, which do not easily absorb higher frequencies. Similarly, a thud reverberation is typical of rooms with padded materials, such as carpets or drapes. Hence, reverberation conveys useful information about the composition of the surrounding space. Reverberation also helps establish measures of distance. When the distance between the source and the listener increases in a reverberating space, the level of the direct sound decreases very fast, while the level of reverberation does not. This can be used to infer the distance of the sources: the “driest” sounds (without reverberation) come from the closer sources, while the most reverberating sounds come from the farthest sources.

In a virtual 3D world, reverberation is often created considering a simple geometrical model of a simulated space. From the position of the sound sources, of the listener, and of the reflecting surfaces (walls, floors, roofs), different procedures can be applied to calculate the time and the direction of every first-order reflection.⁴ These reflections can be reproduced combining three components:

- an initial delay related to the time needed to cross the distance of reflection;
- an attenuation or filter to approximate the losses in transmission and reflection;
- a binaural synthesiser to correctly “spatialise” the reflection.

This method, theoretically justified by acoustics, is computationally expensive, and it is not clear whether all the first-order reflections should be modelled with this accuracy. Subsequent reflections contain millions of waves, travelling in every direction. A geometrical approach is not optimal to obtain an effect similar to the real world. Less precise mathematical methods are required, able to create a significant sonic impact in relation to the level.⁵

In modern consoles, the computation of reverberations is simulated on the basis of a description of the environment. A set of possible environments is defined (open spaces,

⁴For instance, beam-tracing algorithms ([24]) and ray-tracing algorithms ([36]).

⁵Algorithms for calculating reverberations are described in [25].

close, large, etc.), and each one is associated with a particular filter. During the game, the audio engine checks the zone where the player is located and applies the related filter.

In the I3DL2 (Interactive 3D Audio Level 2) standard [1], 24 significant environments have been defined with different reverberation: Alley, Arena, Auditorium, Bathroom, Carpeted Hallway, Cave, City, Concert Hall, Default, Forest, Generic, Hallway, Hangar, Living Room, Mountains, Padded Cell, Parking Lot, Plains, Quarry, Room, Sewer Pipe, Stone Corridor, Stone Room, Under Water. Filters related to I3DL2 reverberations can be implemented in hardware, as it happens on the Microsoft Xbox, for a more immediate application.

Obstruction and Occlusion

The path of a sound wave can be blocked by physical elements between the source and the listener. If an obstacle blocks both the direct wave and the subsequent reflections, occlusion occurs; if on the other hand the obstacle blocks only the direct wave but not the reflections, obstruction occurs. For instance: in a closed room, a sound coming from the outside is occluded. Depending on the thickness and on the material of the walls, a more or less large portion of the wave reaches the listener. If the source is found behind a small obstacle (a pole, a poster, another person), the incoming sound is obstructed. Depending on the trajectory of the reflected wave, the sound reaches differently the listener.

On modern consoles, the obstruction effect can never be faithfully reproduced. This would imply calculating all the reflected trajectories by which a sound bypasses an obstacle; not just the computation is onerous, but physical data describing the world is often unavailable.

Other techniques can rather be used, which emulate the effect of obstruction based on the fact that long waves (with a low frequency) diffract around solid objects better than short waves (with higher frequencies). After having defined a value of “quantity of obstruction”, this is used as the gain of a low-pass filter to emulate the attenuation of higher

frequencies caused by obstruction. The “quantity of obstruction” is a number included between 0 (silence, the sound is completely obstructed) and 1 (maximum, the sound is not obstructed at all). Some methods⁶ can approximate rapidly and precisely this value.

⁶A technique with a low computation cost is described in [58].

CHAPTER 4

SOUNDTRACK

In the United States, kids between 2 and 18 years spend an average of 20 minutes a day playing video games; one out of three has a console in the bedroom, and each one spends approximately 2,000 hours playing games in her entire childhood [19]. Each video game has its own soundtrack. Not surprisingly, two out of three college students know the main theme of Super Mario Bros. However, almost no one remembers the name of the composer [6], Koji Kondo, whose productions are famous worldwide and whose name is almost unknown outside of Japan.¹

HISTORY OF GAME MUSIC

The history of music in console video games begins with Nintendo NES (1984). Predecessor consoles cannot be reasonably associated with the word “music”, since background themes were either very short² or executed only when the computation power of the machine was not exploited by any other game component³. The NES console could generate a triangle wave (in the range of 27.3 Hz to 55.9 Hz), a “noisy” wave (29.3 Hz to 447 KHz), and two square wave voices (54.6 Hz to 12.4 KHz) [14]; in total four sounds could be emitted simultaneously⁴. The very first NES games reserved one voice for sound effects, therefore the composer had to write the soundtrack struggling with only three voices. The first

¹Many of Kondo’s soundtracks have become successful records in sales both in Japan and abroad, and have been interpreted by several orchestras. His most representative album is “Super Mario World” (1991), edited by Warner Music Japan.

²Neither *Space Invaders* (1978) nor *Asteroids* (1979) had a proper background “music”, but only a non-diegetic signal of 4 and 2 notes, that is, a sound that remarked the game atmosphere and was not related to any game event.

³*PacMan* (1980), for instance, was not reproducing music throughout the game, but only at the beginning of each level, when more resources were available.

⁴More technical details about Nintendo NES audio system can be found in [56].

composers had to know every detail of the console they were working for, because their creativity was enormously limited by the technical specifications of the system. Played notes were often nothing more than pure sinusoidal waves. Programmers were frequently creating the music of the game in the spare time they were not dedicating to game coding. Nonetheless, some themes took form twenty years ago for Nintendo NES which still endure in our ears: from the songs of *Tetris* (1988)⁵, to *Final Fantasy* (1989)⁶ or the most celebrated *Super Mario Bros.*

For a long time, the professional figure of the composer could not be distinguished from the audio programmer, as Mike Pummel, composer for the label Acclaim, remembers [14]:

Early on, you were just thankful to get any sound out of the thing. There were numerous challenges to overcome at every turn. For instance, how can one produce a four-voice chord if the hardware only allows three voices at once? Composers figured out a way to assign one voice to play arpeggios so quickly that listeners believed they were hearing sustained chords. This not only produced the chord, but it freed up the other two voices to play other things. Tinny synthesized notes could be made to sound more impressive through real-time waveform shaping, such as using pulse-width modulation on a square wave. This boils down to constructing mathematical equations which superimpose the effect of articulations and decays upon pure waves. Spare voices were used to double melody lines, playing a millisecond off to create a primitive echo or chorused effect. Percussion sounds were created with carefully timed bursts of static and distortion on the noise track, but often had to be omitted entirely in favor of sound effects.

At the beginning of the Nineties, the distinction between programmer and composer took form with the success of consoles such as Sega Genesis/Mega Drive (1989) and Nintendo SuperNES (1990), which supported sampled audio. Composers could finally create music with proper musical instruments, importing it on the console only in the final phase. Yet the strongest limitation was given by the number of usable voices: only eight. With time, the audio hardware of consoles has improved and composers have become less and less involved in the technical details of the machine, dedicating more time to pure composition. The separation between composer and audio programmer is today complete, to the point

⁵Composed by Alex Pajitnov.

⁶Composed by Nobuo Uematsu.

where the composition of soundtracks is sometimes outsourced to external musicians, while the audio programmer is just in charge of importing them inside the game code.⁷

SAMPLED MUSIC AND SYNTHESISED MUSIC

Until the middle of the Eighties, the video game music game was programmed directly on the console hardware, as a series of commands sent to the tone generators to emit a specific sound wave for a defined duration of time. The MIDI (Musical Instrument Digital Interface) protocol was defined in 1982 as a standard format of commands to generate sounds, which could be used in synthesisers, keyboards, sequencers, mixing decks, computers. The MIDI format was first supported by the Atari 520 ST (1985) console, and is still supported by modern consoles, together with the WAVE format, which identifies sampled digital audio. Both formats have advantages and disadvantages.

The MIDI format

Composing in MIDI format equals sending the synthesiser a sequence of commands to reproduce music. The commands can indicate which notes to play, the duration of each note, the instrument to use, the time, volume, etc. MIDI music only includes “instructions” and not sampled audio, therefore a small space is required for storage, reason why MIDI has been the main format in video game music for a long time. MIDI music has the merit not to overload RAM memory, bus, mass storage and DMA channels, leaving more resources available, which is desirable even in the last generation consoles. Sequences of MIDI commands make it easy to dynamically change music properties (tempo, key, intensity) in real time, and this makes MIDI an optimal format to implement some musical transitions typical of an interactive game soundtrack (described in the next chapter). Musical quality is the big disadvantage of this method, since the MIDI format is limited by the synthesis capability of the underlying machine. The audio programmer can have some trouble in integrating

⁷More chronologies of game music can be found in [37, 38].

MIDI music in the game, since every musical transition for every game interaction has to be specified as a sequence of MIDI commands.⁸

The WAVE format

Composing in WAVE format means to record the music in a sampled digital format to later reproduce on the console the same exact recorded sound. This format requires much more space than the MIDI format, since the entire binary audio data is stored. Sony PlayStation (1994) was the first console to firmly point towards digital audio. Sampled music complied with the Red Book standard (16 bit, 44.1 kHz) and was reproduced directly from the CD, implying a substantial access to memory and to shared resources, the laser in first place. Given the linear nature of the laser reading from disc, variations were not possible, and an interactive soundtrack was difficult to obtain. Still, the sound maintained the quality of a CD, and could be produced recording even a complete orchestra. As Jack Wall claimed [59]:

It is the composer's job to add as much realism to game music as possible to bring the player into the experience. I can't think of a better way to not achieve this goal than to produce orchestral music solely with samples. The result of live performance is so much richer and more satisfying to the player.

PURPOSE OF AN INTERACTIVE SOUNDTRACK

The very first approaches to include background music in a video game reminded of the world of cinema: music underlined the characteristics of the players, the details of the environments, and key passages of the story could be narrated in a non visual form. As Royal S. Brown claims, music in movies can *narrativise* a scene [9]; in other words can relate things that neither actors, nor dialogues or images can describe. If all the video games had linear plots, the music could always start at the beginning of the game and be reproduced until the end. This solution is satisfying for game genres (puzzle, sport games),

⁸A detailed planning of a MIDI game soundtrack can be found in [45].

which for many years haven't had a soundtrack at all, while it is unacceptable for other genres (platform, action and role games). Platform games, for instance, have an average duration of 40 hours, and no game has such a long soundtrack. Novice players could even spend a much longer time to solve all the game levels. In short, the soundtrack of a game is necessarily shorter than the overall game duration, so the music has to be somehow "repeated" throughout the game.

An indefinite repetition of the same exact theme during the game fails to achieve the goal of "narrativise" the game: an ideal soundtrack should be able to interpret every game situation and to adapt accordingly. While a movie plot is linear and well-defined (and so is music), a video game plot is undefined, with an imprecise duration, and influenced by the player's actions. A music that is able to adapt in real time to such a plot is called adaptive soundtrack or, more commonly, interactive soundtrack.

THE HISTORY OF INTERACTIVE SOUNDTRACKS

The history of interactive music is surprisingly short, and only a few games have supported it. The first musical interactivity goes back to the Seventies, when games such as *Space Invaders* and *Asteroids* included an effective trick to acoustically communicate a sense of urgency to the players: the music tempo was increased. Yet, it is excessive to talk about "interactive music", given that the themes were made respectively of four and two notes.

The evolution of musical instruments during the Eighties had an influence on the video game market. Digital samples were used for the first time, while the MIDI format turned into a standard for synthesised reproductions. Successful video game usually included valid music, either original or reproduced from well known themes. And developers started to realise that even magnificent musical themes could sound boring if they had to be repeated during several hours.

A solution found was that to record many different songs, moving from one to the other in response to the game state. The audio programmer had to face several difficulties

(conform with the available memory, manage the constant disc reading), with a musical result that was not so exciting: the themes were not too many and the harmonic transition from one to the other resulted unappealing.⁹

Some titles though introduced small and effective stratagems to include interactive music. Snowboard game *SSX Tricky* (2001) abruptly turned down the volume of the main theme during the most vertiginous jumps, as if the music remained “glued” to the track. The absence of sound while hovering in the air created a sense of emptiness and “suspension” which strengthened the sense of vertigo effectively created by the graphics.

The middle of the Nineties saw the first real experiments of interactive soundtrack. A reference title is *Banjo Kazooie* (1998): different climatic situations are present in the game and the musical instruments playing the main theme gradually change as the player wanders between game levels. While getting closer to the beach, the music assumes a reggae-style arrangement; while moving towards the mountains, Christmas-style bells are introduced. A marine zone is distinguishable by a piratey-sound. The melody never changes, but the style continuously adapts to the situation. This was possible thanks a perfect coordination between programmer and composer, and the conviction that an interactive and original music could be much more valuable than a famous but repetitive music.¹⁰

COMPOSING AN INTERACTIVE SOUNDTRACK

The choice of including an interactive soundtrack has a strong impact of the creative process of the composer. In a classic scenario, the composer sticks to the musical dictates of the Audio Design and produces music that is later inserted by the programmer in the game code. This process is close to composing a movie soundtrack, where every detail, passage or dynamic can be taken care of, since every theme will be exactly reproduced as it was imagined.

⁹A reasonable critic to non-interactive game soundtracks composed by successful musicians can be found in [49].

¹⁰A more detailed chronology of interactive soundtrack in games can be found in [28].

This schema changes dramatically for an interactive soundtrack. Many more tracks have to be prepared, and have to be such that one can harmonically shift into another, depending on the state of the game. As Brian Schmidt explains [50]:

Any game can be thought of as consisting of a number of states and arcs connecting those states. A typical video game or pinball machine can have dozens or these states and connecting arcs. As a player plays a game, scoring points, hitting targets, etc., arcs are taken depending on the actions taken by the player, and the current game state changes accordingly. These changes in game states are accompanied by corresponding changes in the background music. This is done by composing a musical score for each state node and connecting arc. Scores for game states are usually full and complete musical compositions, while score for connecting arcs are typically short transition pieces that lead directly to the “next game state” music. The arcs can be as simple as a drum fill or as complex as a harmonic cadence setting up a new key and theme. When the game state changes, the music for the arc corresponding to the reason for the state change is played, usually a transition leading into the music for the next game state.

The composer’s melodic choices can be strongly limited because of the interactive soundtrack. Since themes can unexpectedly change from one to the other, their melodies are usually focused on the keynotes¹¹, with a few repetitions, so that a theme can harmonically wedge in with the keynotes of the subsequent theme [6]. The same transitions from song to song raise novel problems¹², as Thomas Dolby explains [2]:

At the point where a game player can either stand and fight or turn and run, there are various things that you can and cannot do. You want it to make sense. If he happens to turn around conveniently right before the downbeat, it might be as simple as kicking the thing up by a minor third, adding ten percent to the tempo, enhancing the two percussion tracks with sixteenth-note tom-toms that had previously been muted, and so on. If he does it just after the first beat of the bar, do we wait until the first beat of the next bar? That’s going to feel very slow; maybe it’s just one second, but it’ll feel like a lifetime. So is it acceptable to make this change on the second beat of the bar? That’s no strict formula; you can’t write a string of code that would deal with any of this. But you can create a set of higher-level tools that will allow a composer to make those kinds of decisions.

¹¹The keynote is the first note in the scale and the tonal centre “around” which every other note and chord revolves.

¹²More details about a soundtrack “granularity” (how often transitions can be applied for each theme) can be found in [30].

An interactive music composer, hence, needs to collaborate much more with the audio programmer than a “classical” musical composer. A strict bond is somehow re-established between composers and programmers, two professional figures which are now distinct, and whose teamwork can sensibly improve the acoustic quality of a game.

EXAMPLES OF INTERACTIVE SOUNDTRACK

Modern consoles support both MIDI and WAVE music: developers are to decide which format to adopt. This choice is decisive when the game requires an interactive soundtrack, for the MIDI format contains commands which can easily model interactions, while the WAVE format does not contain any concept of tempo, tonality or intensity.

A MIDI format example: Mark of Kri

Mark of Kri (2003) is an action game presenting an interactive MIDI soundtrack: the whole audio setting has been developed by two people in a period of one year [16]. The choice of the MIDI format was driven by two factors: the need to musically underline many game events, and the narrowness of the console audio resources (only 2 MB of memory). The set of events the music had to respond to was defined in the Audio Design document: the number of enemies on the screen, the closeness of the enemies, the fact that the player was under attack, the health status of the player, the status of the weapon, the environment, the presence of special characters. The music was made by two overlapping tracks: one describing the environment and the other the combat status. The environment track was composed of several themes, each with a duration a 10–20 minutes, selected depending to the environment the player was in. For the combat track, the composer distinguished between four different game situations:

- no enemy close to the player
- 1–2 enemies in proximity;

- 3–5 enemies in proximity;
- more than 6 enemies in proximity.

Each situation was associated with a theme of growing intensity.

Every time the player moved from a setting to another, the environment theme would change; every time the number of enemies on the screen increased or decreased, the combat status track would change. These changes would always occur at the end of a musical measure (signalled by an appropriate MIDI event), to avoid undesirable harmonic variations.

The tracks were associated with different MIDI channels: while the environment track would control some percussion instruments (drums, timpani) and affect their timber and intensity, the combat status track would control the brasses (trumpets, horns) and enable or disable their execution. The changes in the two tracks were not “lined up”, as shown in Table 2, so to avoid a sense of repetition.

Table 2: Alignment of MIDI tracks in Mark Of Kri’s soundtrack.

Environment Track	1	2	3	4	5
Combat Track	1	2	3	4	5

The entire MIDI game soundtrack had a duration of about an hour; thanks to the interactivity, it would not sound repetitive even after several hours of game. The overall size was 800 kB; hence it perfectly fit in the 2 MB of memory fixed by the hardware.¹³

An example in WAVE format: Halo

Halo: Combat Evolved (2001) is an action game implementing an interactive soundtrack with sampled audio [40]. Composer Marty O’Donnell explains the audio production in this way [44]:

I believe that music is best used in a game is to quicken the emotional state of the player and it works best when used least. If music is constantly playing it

¹³Another example of interactive soundtrack in MIDI format can be found in [20].

tends to become sonic wallpaper and loses its impact when it is needed to truly enhance some dramatic component of game play. In Halo, there are more than 80 minutes of original music, but these minutes are spread over the course of a single player's experience, which could extend from 20 to 70 hours of play. Therefore, much of the time no music is present.

The whole soundtrack was made of 16-bit stereo ADPCM samples. To create the interactivity, composer and programmer worked tightly: the composer created several themes for different game events, and the programmer integrated them in the game. O'Donnell explains [44]:

The music engine is relatively simple in construction. It consists of three basic file types within a single soundtag. The `in`, which starts the piece, the `loop` which is the middle section and plays for an indeterminate amount time, and the `out`, which is how the piece ends. In addition, there is the `alt-loop`, which plays instead of the loop if called by the game, and the `alt-out`, which plays if the end is called during the `alt-loop`. The looping sections are made up of as many looping soundfiles as desired, and each looping soundfile can be of any length and weighted to determine the likelihood of being played. The level designer only needs to insert the commands: `start`, `start-alt`, and `stop`, in order to call any given music soundtag. This made it relatively easy for me to sit with each level designer and “spot” a level in the same way I might sit with a film director and “spot” a movie. Within each music soundtag, I could also set the preferred type of transition (immediate, cross-fade, or wait until the end of the current loop) between the `alt` and `out` soundfiles. We can give real-time volume commands to the currently playing soundtag and also give it an overall duration limit. This kind of soundtag was also used for ambient background sounds.

The sound design tools to easily manage an interactive soundtrack were partly supplied by the Microsoft Xbox libraries, and partly implemented by the same programmer. In the second part of this thesis, we will take a deeper look at some Xbox programming libraries which help implement an interactive soundtrack without the limitations of the MIDI format.

CHAPTER 5

INTERFACE

The acoustic nature of video games has often been subject to experimentation. This chapter focuses on musical games and audio-games, for which the developers had the courage to take the role of music from a simple ornament, to a support of the whole game interface. Audio can provide players with clearer and more direct information than graphics, and give unsighted players the instruments to get oriented and move through game levels.

MUSICAL GAMES

In the previous chapter, the word “interactive” has been employed to describe movie-like soundtracks, able to follow game events and to acoustically underline narrative and environmental aspects of the levels. The most proper term would have been “adaptive”, since music “adapts” to the game situations, but does not “interact” with the players. A soundtrack can be correctly defined as “interactive” when it is directly influenced by the explicit choices taken by the players. For many types of game, such an effect is to avoid. In a platform game, for instance, if the player realises that by going up a staircase the volume of percussive instruments systematically increases, time could be wasted “playing with music” on the stairs, rather than trying to achieve the main goal of the game. For platform, action and role games, a soundtrack is valid when it can unconsciously communicate with the player, can generate the right atmospheres to get oriented, and not divert the player’s attention from the main goal.

A new typology of games, though, is on the rise: musical games, which are not characterised by a specific goal, but by the game dynamic: to finish each level, it is necessary to interact in the best possible way with music. The soundtrack assumes a new meaning:

rather than being a simple decorative element, it becomes the main element.

RHYTHMIC GAMES

PaRappa The Rapper (1997) is the archetype of a subcategory of musical games: rhythmic games. The main character, the puppet PaRappa, has to learn different rap and hip-hop music styles to catch on the girl he is in love with. Players listen to the songs of the instructors on each level, and endeavour to imitate them pressing the right combinations of keys on the joypad. If the combination is right, the song continues with the following verse, otherwise it is interrupted and players have to take over the rap at time of music.

A similar concept was reprised by *Um Jammer Lammy* (1999), winner of the AIAS¹ award in 2000 as “Best original composition” [8]. The soundtrack is divided in two parts: the game music, with which the player controls the musical abilities of the character, and the cinematic non-interactive music, which underlines the weird and hallucinogenic transitions between levels. In *Um Jammer Lammy*, the player controls the guitar ability of the character by pressing the correct keys on the joypad on time with the reproduced music. The progress of the game is indicated by the music itself: if a song starts sounding weird, the player realises a mistaken note was played, and that she needs to concentrate more; if the player is not precise, not only the guitar starts playing out of time, but also the song becomes progressively more distorted and filtered as more beats are lost. The game response is not only auditory, because graphical distortions appear on the screen to signal errors, but the most immediate and subconscious reflexes come from music. This tight integration of audio in the game is probably the motivation for the assigned award.

Mad Maestro! (2002) and *Space Channel 5* (2000) do not differ very much from the previous titles: in the first, a classical music orchestra has to be directed, following the speed and tempo of the score; in the second, the player impersonates Ulala, a journalist of the future in charge of chasing away the Morolian aliens from Earth by imitating their

¹Academy of Interactive Arts and Sciences

dance steps.

BENAMI

Dance Dance Revolution (2001) presented the players with a new game interface: a foldable platform to dance on, on time with music. The screen shows upward-scrolling arrows; when an arrow enters the “action zone”, the player has to jump on the corresponding arrow on the platform. The soundtrack contains famous themes; the speed and rhythm of the scrolling arrows change according to the genre of the song (from pop to techno). This kind of game is called Benami. Variations have been proposed with different interactive instruments: Samba do Amigo (2000) includes two maracas, while in Gitaroo-Man (2002) the external device is a plastic guitar, to be played on time with music.

More recent titles on the same concept are Britney’s Dance Beat (2002), based on the songs and video-clips of the American pop-star Britney Spears, and Aerobics Revolution (2003), both compatibles with the foldable platform of Dance Dance Revolution. In particular, the last one draws on the “Diet” mode to develop an actual fitness programme: the dynamic of the game is to follow the trainer on the screen, dancing to the notes of eight different songs. The “Shape Up” mode is indicated to tone up precise zones of the body; the “Stretch mode” serves for the back, legs, and pelvic muscles.

INNOVATIVE MUSICAL GAMES

The role of the soundtrack is enhanced in musical video games: it does not adapt to the goals of the game, but marks them and influences the commands of the player. Some recent titles have amplified this idea, with audio engines that control other engines, rather than being controlled. For the first time, the music is given a prevalent role over the graphics in the interface of the game.

Vib-Ribbon (1999) is a vanguard game in this sense, that takes the relation between music and game strategy in a totally new direction. Vibri, the small rabbit character, travels

among levels which are generated in real time based on the musical track being played. Pop ballads build slow and flat levels; techno songs create rapid and complex worlds. It is also possible to insert one's own audio CDs to generate completely new levels.

Rez (2001) epitomises the described concepts and expands them brilliantly in the most valid musical game published on Sony PlayStation2, winner of the "Game Developers Choice Award 2002" in the "Game Innovation Spotlight" category. If played at volume zero, Rez results a boring *shoot'em all* whose only goal is to shoot for the enemies and destroy them with the only button working on the joypad. But the soundtrack, composed by Tetsuya Mizuguchi, is so hypnotic and immersive that in a few seconds the player is transported inside the level and forced to remain glued to the chair and to eliminate the enemies at the time of music, to the point where it is not clear whether the soundtrack is guiding the player, or is adapting in real time to the player's behaviour.

AUDIO-GAMES

Many persons cannot imagine a game without graphics, as the same word "video game" suggests. Still, 7 millions people in the United States have severe visual problems, and cannot enjoy any game based on a purely graphical interface. The goal of the American project Zform [3] is to make games funny for people who lack the primary sense of vision, by replacing graphics with audio to describe the game interface and the 3D environments.

A classical game interface, made of menus and descriptive texts, is normally "shown" to the players; Zform proposes to have it "read" out loudly by a synthesised voice. The description of a whole 3D world by means of acoustic clues, though, and has to face several issues, for instance:

- What is in the position of the player?
- Is the player moving or still?
- Is the player walking in a corridor or crawling along a wall?

- In which direction is the player moving?
- Which objects surround the player and which does she possess?

Movements in a 3D world are the most difficult information to present to unsighted player. A graphical interface is able to simulate real-world scenes in a familiar way: exits are signalled by a “WAY OUT” sign, corridors are paths to follow, etc. To replace graphics with audio means to empathise with the needs of the unsighted in order to supply the highest number of acoustic clues to describe the surrounding world. Some of the clues proposed by Zsoft, and implemented over the game engine of Quake (1996), are detailed hereafter.

Exits

Most games located in closed environments requires the player to find a way out. When the way out is designed as a sliding door, an electrical noise could be assigned to the door for the unsighted to spot. This noise should be audible from all over the place, should increase in volume as the player gets closer, and should switch to a mechanical noise when the door opens up. This approach seems fair, but it does not actually help unsighted people, for several reasons. Firstly, it is to locate an exit when more doors are present; then, it is hard to go through a door once it has opened. Assigning a different noise to each door would not help either. The solution is to overcome the (graphical) idea of door offering the (acoustic) information the player needs: how to get through a gate to move to the next room. Zform proposes to insert a ventilator (an acoustic source of a smoke noise) in the middle of each corridor, perfectly audible when the player is in the corridor, and attenuated when the player is in any adjacent room. Finding a way out becomes much easier, since the fan noise signals not-occluded paths through a corridor. A player would only have to rotate on her axis up to the point where the fan noise sounds as coming from above (same volume in both ears), and then could walk in that direction to get through to the following room.

Steps

Un sighted players can infer the movement and speed of the player from the sound of her steps. However, they have no way to discern whether they are walking towards, or they are shifting along an obstacle (such as a wall), and in which direction they are shifting (left or right). The solution is to emit a crash effect sound (or a voice: “You have bumped into a wall”), balanced in stereo according to the incidence angle. If the player keeps crawling along the wall, a friction noise is emitted instead, balanced in the same way.

Orientation

Without vision, it is hard to keep track of the direction followed. This holds even when the movements are constrained to two dimensions (north/south, east/west). The solution consists in inserting some “orientation keys” in the 3D world. For sighted players, a key can be the moss growing on the north side of the trees, or clouds shifting constantly from west to east. For unsighted players, an audible clue can be given by the ventilators, producing two distinct sounds when the corridor is oriented on the north/south axis or on the east/west.

Objects

Un sighted players need to “see” the surrounding characters and objects without the sense of vision. To spot out other characters, a player can listen out for their steps and their impacts against walls. For other objects, Zform proposes that the game should only include elements which emit sounds even when they are still: animals, clocks, and so on. As a player gets closer to an object and picks it up, the sound changes to highlight this event: animals shout, clocks ring, etc. Sounds are influenced by the occlusion of the walls: inside a room, only a few objects can be heard, preventing unsighted player from bumping into walls to try and reach objects from contiguous rooms.

Environmental sounds

Acoustic artefacts can be used to embellish different game levels, and to describe each room: clashing cooking pans in the kitchen, ticking pendulum clocks in the studio, and so on. Similarly, environmental reverberations can help unsighted players: steps should be clumsy and hushed on a studio carpet, loud and echoed in a kitchen.

All the aforementioned techniques do not worsen the game to the eyes of sighted people, but rather make the interface more usable, easing the navigation and the progress of the game.

CHAPTER 6

DIALOGUES

In 1982, Mattel produced Intellivoice, a vocal synthesis module for the Intellivision console, introducing dialogues as a new component of video games. Five titles¹ made use of that accessory, advertised as follows [31]:

Video games that actually talk to you. Male and Female voices react to changing game situations immediately, are calm or excited, give you strategy tips, cheer you up or egg you on. [...] These are not fuzzy simulations, but voices complete with expression, produced by the ability of Intellivoice to duplicate realistic human speech electronically.

The goal was brave: real-time generation of human dialogues based on the game situation. Actually, the voices did not sound that realistic at all, probably because each game cartridge could only hold 4 to 8 kB of voice data, and this led Mattel to interrupt the production of Intellivoice. From those first experiments to today, much has changed regarding the integration of dialogues in video games. The approach still takes inspiration from the movies, where characters orally “narrate” the development of the story to the spectators (players), conducting them along the narrative thread of the movie (game). On the technical side, though, different problems arise when dialogues are present in movies or in video games.

Dialogues are normally added to movies in the post-production phase. Dubbers keep track of the duration of each scene and record voices which emotionally match what appears on the screen. Dialogues are the locking ring of the production chain, since no other element depends on them. This facilitates the localisation process, which is done by substituting the original audio track with a new one in a different language.

¹The five Intellivoice games were: Space Spartans (1982), B-17 Bomber (1982), Bomb Squad (1982), Major League Baseball (1983) and Tron Solar Sailer (1983).

A likewise approach has applied in video games for a long time. Dialogues were usually unaware of any other element of the game, and were only recorded in the final phase, with respect to the duration of the scenes. This technique still holds with some game genres (puzzle, sport), where spoken audio is meant to comment on the events, with no active interaction on the game.

The aforementioned technique is now considered obsolete for other game genres, specifically action and platform games.² Modern players expect the voice of the characters to be coherent with the animation of the mouth, rather than “pasted” in post-production over basic “open/close” mouth movements. Moreover, talking persons do not just move the mouth, but also the head, body and facial muscles, depending on the expressed emotions; video games should indeed include both a textual content and an emotive description of each dialogue. Finally, dialogue animations are subject to be dubbed in many different languages. If the animation of a dialogue line is limited by a predefined length, dubbers will have to pronounce that line with the same duration in each language, which may sound unnatural. All these disadvantages can be surpassed if dialogues are intelligently integrated within the production of the game, rather than at the end.

IN SYNC WITH THE ANIMATIONS

The inclusion of dialogues in the production of the game dramatically changes the activity of the animators, since basic movements of “open/close” mouth are replaced with more complex animations, reflecting the content of the dialogues. Formally, audio does not anymore rely on graphics, but the other way round. The advantages are clear: dialogues can look more natural, since dubbers are not limited by prefixed timings, and animations look more faithful, since the movement of the mouth is synchronised with the voice. Some new issues that rise from this approach are: how to rapidly model the animation associated to each dialogue? How to translate an audio file into a mouth animation? And how to

²A broader view on vocal applications inside virtual environment can be found in [35].

manage localisation?

The Talking Heads system

The *Getaway* (2003) includes facial and lips animations, synchronised with the dialogues, which were easily and automatically created thanks to an innovative system called Talking Heads.³ The system works by combining sequences of phonemes, the auditory elements which constitute the words pronounced by humans. Each phoneme corresponds to a viseme, which describes the position assumed by the mouth and tongue when that phoneme is pronounced. The English language, for instance, comprises 16 visemes. Talking Heads can realistically simulate the facial expression of talking characters by analysing the audio file to pronounce, extracting the phonemes, and obtaining the related visemes. The graphical engine includes a proper animation for each viseme (“O”: open mouth, “L”: tongue between the teeth, etc.), so that each sentence can be automatically visualised as a sequence of mouth and tongue animations.

The result is valid but not yet realistic, for humans also transmit emotions when pronouncing different words. Talking Heads include an emotive system to animate virtual character based on emotions. The system defines for each character a polygonal model of an “emotionally neutral” 3D face. When a sentence is pronounced, the pose of the face changes according to the expressed feelings. Since no algorithm can extract emotions from a speech line, it is up to the sound designer to indicate, next to each sentence, the associated feelings, for instance:

- (Anger 4) “Now listen to me, you’ll do what I say, ...”
- (Anger 2, Joy 3) “...or the kid will pay for it.”
- (Anger 0, Joy 2)

The numbers at the beginning of each line indicate the “quantity of primary emotions”

³More information on Talking Heads can be found in [43].

expressed. Talking Heads recognises six universal primary emotions (sadness, anger, joy, fear, disgust and astonishment); all the other nuances are obtained with combinations, as in the previous example (anger and joy mean treachery). 3D modellers specify the correct facial and body animations for each primary emotion, for instance joy is animated with an open smile, rage with frowned eyes. These animations are then automatically combined together according to the quantity indicated next to each sentence.

As a last expedient to increase realism, Talking Heads represents a wide range of subtle movements that humans do when talking. Eyelashes automatically flinch based on the expressed emotion: every six seconds if angry, every two seconds if lying or acting suspiciously: every four normally. Once in a while, the flinching is double. The head constantly swings at random; when a character uses keywords such as “yes”, “I agree”, “of course”, the movement becomes vertical; in the case of “no”, “I disagree”, it becomes horizontal. Even breathing is considered, with an animation executed every time a sentence ends. Finally, the eyes are animated to communicate that the character is alive and real.

Talking Heads is an advanced system to manage animations from dialogues, tipping over the axiom by which audio has to adapt to graphics, and setting the animator free from the inauspicious labour of creating an animation by hand for each dialogue line. Talking Heads helped the developers of *The Getaway* produce all the dialogue animations in a few hours rather than in many working days.

Talking Heads also offers interesting development ideas. While it is currently used in the design phase, to simplify the work of the animators, it could be soon applied in real time in the game. This means the characters will not come with any predefined facial animation, but will constantly move according to the pronounced sentences. Animators will only create the primary visemes, and the system will compose them depending on the phonemes. With such an engine, each sentence will be “visually” pronounced in the correct way. Even the localisation process will be simplified, since the graphical impact (animation) will remain coherent with the reproduction, independently from the “vocal”

translation of the sentences.

SPEECH RECOGNITION

The illustrated system offers the audio section (dialogues) an automatic control over a graphical component (animations) of the game. Still, the player has no way to interact with speech: the sentences pronounced by the characters are only those recorded by the dubbers. Players cannot vocally communicate with the characters and obtain appropriate answers. In short, players use the joypad, not their own voice, to communicate, and this hinders the realism of the dialogues.

Some recent titles overcame this drawback with the introduction of a new input device: the microphone. By means of this tool, the player does not have to choose among a pre-established list of sequences, but is free to pronounce unwritten thoughts which the system identifies and evaluates. A system able to translate each spoken sentence into a game event is called a speech recognition system. *Seaman* (2000) is probably the most advanced title so far to present an effective speech recognition system.

Speech recognition is still an experimental technology in video game. The microphone is not a common device for every console, and is currently sold only as a Microsoft Xbox accessory (The game pack of *Seaman* includes one as well). The situation is doomed to change, thanks to the outbreak of network video gaming, also known as net-gaming. With a theoretically infinite number of opponents, spread all around the globe, communication will become a primary factor. Speech recognition systems will soon let players orally command remote enemies and speak in real time with people of different languages, thanks to integrated engines of automatic translation.

CHAPTER 7

NETWORK

The future of video games is online. All of the three main game consoles support network gaming.¹ A match disputed among connected players is characterised by a continuous exchange of information by which remote consoles remain synchronised. The level of synchrony relies on the game genre: low priority for some puzzle games, high priority for many action, sport and platform games. That is, a delay in the transmission of a move to an opponent is tolerable in an online card game (puzzle), while it would be unbearable in a soccer game (sport).

Consoles guarantee uniformity among users, in the sense that every network player owns the same hardware and the same game support, and no opponent is advantaged since equivalent machines send and receive equivalent data streams. Since each opponent owns a copy of the game, all the graphical and audio data are stored locally, and the network communication includes only a small information which is needed to reproduce on one console what changed on the other. An online checkers game can serve as an example. When a player moves a draughtsman:

- the draughtsman moves on her screen;
- the associated sound is played;
- the information is sent to the opponent.

The transmitted information does not contain visual or acoustic component, but only the basic description of the move. On the other end, the listening console receives the data and consequently:

¹Different market strategies operated by the three consoles are described in [32].

- the opponent draughtsman is moved on the screen;
- the associated sound is played.

Summing up, neither graphical nor acoustic data are transmitted in a network game, but only the information needed to reproduce them.

VOCAL INPUT

In the aforementioned case, both opponents own a complete copy of the game. Acoustic data does not need to be transmitted since both players have the whole set of playable sounds in their console. Only the indication about which sound effect to play needs to be sent.

A device introduced in recent consoles works in contrast with this pattern and compels audio data to be sent: the microphone. A microphone in an online game allows opponents to verbally interact with each other. The role of the microphone depends on the game typology and affects the data stream sent to the opponents.²

NETWORK TRANSMISSION OF VOCAL DATA

The most immediate network application for a microphone is verbal communication, similarly to a telephone. In this case, the microphone is not exactly a gaming tool: the vocal input is not processed but sent and reproduced on the console of the opponent. Managing data audio stream on the network is crucial, and possibly requires an adequate compression format. A compression algorithm which well adapts to the context of video games is the Linear Predictive Coding (LPC).

Linear Predictive Coding

Linear Predictive Coding [18] offers optimal results on vocal audio data, requiring in average only 500 B for second of dialogue, almost 350 times less than the Red Book format.

²An introduction to network transmission of vocal data can be found in [7].

A device called vocoder is the founding element, which separates the different sound components (analysis), manipulates them, and packs them together (synthesis). Since analysis and synthesis are split and can occur on separate machines connected remotely, the vocoder is an optimal tool for transmitting vocal data on the network.

Real-world sounds are generated in nature thanks to two components: an oscillator and a resonator. The oscillator is the physical object that vibrates and produces sound waves (the strings in a violin, the vocal cords in humans); the resonator is the objects that amplifies the waves (the resonating chamber in a violin, the internal cavities of the head in humans). A vocoder simulates these two elements separately, representing the oscillator with a sampler, and the resonator with a digital filter which takes a sequence of samples in input and produces a sequence of filtered samples in output (emitted sounds). In particular, the filter of a LPC vocoder is modelled on the cavities of the human head, as a long tube composed of segments of different lengths where the emitted wave is partially absorbed and partially reflected. The “reflection coefficient” defines the quantity reflected at each point, and the sum of the coefficients describes the tube. The LPC filter maintains a state vector of the quantity reflected at each point, in order to simulate the behaviour of the resonator at each given moment.

Relevantly, the LPC filter is invertible: the original wave produced by the oscillator can be obtained from the generated one by applying the inverse filter function. The resulting wave has smaller energy, contains less information, thus has a more compact digital coding. When a player speaks through the microphone, her voice is compressed using the inverse filter function and sent to the opponent. The remote console receives the signal and restores the original wave applying the same filter function.

Since the LPC algorithm transmits the wave generated by the oscillator, rather than the filtered wave, the size of voice data is reduced. In the real world, this equals to communicate by sending the interlocutor the vibrations of our vocal cords, rather than the complete sound emitted by our voice. In order to faithfully reproduce the pronounced sentence, the

interlocutor would then need a human body identical to ours. This is the main limitation of the LPC algorithm: the reproduction is faithful only when the opponents have the same filter used by the first player. When this is not possible, the wave is reproduced using standard filters, and the effect is the reproduction of the original sentence using a reference voice rather than the original voice. This solution is tolerable, according to the game typology: the content of the dialogue is intact, while the form (voice) is simulated. The LPC algorithm can also be extended, to the detriment of transmission efficiency, to send the representation of the filter as well.

ALTERATION OF VOCAL DATA

In some games, the disparity of the resonator of the LPC algorithm among different players is not considered a defect, but a technique to add variance to dialogues. If a user plays the role of a robot in an action game, using a “robotic” filter to reproduce her dialogues will increase the realism of the level. In the same way, each character in a role game can have a voice tied to her role in the story; an “imposing” filter for the knight, a “distorted” filter for the wizard, and so on.

This process increases the immersion in role and action games, and avoids a loss in tension related to voices external to the scene. For instance, if the big enemy at the end of a level is remotely commanded by a little girl, the sentences she pronounces will result in contrast with the character she represents. The application of a specific filter helps overcome this inconsistency.

MULTILINGUAL SUPPORT

Opponents in online matches are located all around the world, but an invisible obstacle limits matches to national borders: the language. Two persons cannot communicate if they ignore each other language, and will hardly face one another online. The ideal solution is the design of a simultaneous multilingual translation system. Such a model is far from

being implemented and integrated in a video game console, but simpler models are already present on the market.

Microsoft Xbox is the most advanced console in this sector. Players are proposed with a kit for online matching, including an earphone-microphone to verbally communicate with the opponents. Developers are offered with a library for speech recognition for a finite set of English and Japanese words. The system can detect which words were pronounced by the player and communicate them textually to the opponents. The result is riveting since people of different languages can communicate, but is limited because only a few words are recognised, pronounced in English or Japanese, and transmitted textually. However it lends to future evolutions, such as an integration with a text-to-speech system to “read aloud” text to the opponents rather than showing it on the video.³ The actual system is indeed very valuable for players to communicate with virtual players (bot) spread on the network, rather than with human opponents. An American player can, for instance, vocally command her bot soldiers running on a remote server to fight a Japanese opponent who can, in turn, vocally activate his own AI characters for defence.

³Already in 1978, Magnavox produced without great success an accessory the Odyssey2 console The Voice, which phonetically pronounced the words written by the player on the keyboard.

**Design and Production of Audio Technologies
for Video Games Development**

SECTION II

**Implementing the Audio Engine
for a Microsoft Xbox Video Game**

REFERENCES

- [1] 3D WORKING GROUP OF THE INTERACTIVE AUDIO SPECIAL INTEREST GROUP, “Interactive 3D audio rendering guidelines - Level 2.0,” Sept. 1999. ([document](#))
- [2] AIKIN, J. and DOERSCHUK, R., “Thomas Dolby,” in *Keyboard*, p. 34, New Bay Media, LLC, Mar. 1995. ([document](#))
- [3] ANDERSEN, G., “Playing by ears,” in *Game Developer Magazine*, CMP Media LLC, Oct. 2001. ([document](#))
- [4] BAJAKIAN, C., “Game audio production: Process and strategy,” in *2002 Game Developers Conference*, GDC Lectures, Jan. 2002. [4](#)
- [5] BECKER, D., “Microsoft says Xbox sales on track,” Jan. 2003. [6](#)
- [6] BELINKIE, M., “Video game music: not just kid stuff,” Dec. 1999. [4](#), ([document](#))
- [7] BOLOT, J.-C. and FOSSE-PARISIS, S., “Adding voice to distributed games on the Internet,” *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings.*, vol. 2, pp. 480–487, Mar. 1998. [2](#)
- [8] BOYD, A., “Escape from bad audio,” in *Game Developer Magazine*, CMP Media LLC, Oct. 2001. ([document](#))
- [9] BROWN, R. S., *Overtones and Undertones: Reading Film Music*. University of California Press, Oct. 1994. ([document](#))
- [10] BURL, P., “The stochastic synthesis of complex sounds,” in *Game Programming Gems* (TREGLIA, D., ed.), no. 3 in *Game Programming Gems*, Charles River Media Inc., 2002. [7](#)
- [11] COHEN, S., *Zap! The Rise and Fall of Atari*. McGraw-Hill, 1997. [2](#)
- [12] COLLINS, K. E., “Video games audio.” [1](#), ([document](#))
- [13] COOK, P. R., “Physics-based synthesis of sound effects,” in *2003 Game Developers Conference Proceedings*, Mar. 2003. [9](#)
- [14] DEENEN, C., “Adventures in programming music & sound effects for video games,” in *Keyboard*, New Bay Media, LLC, Nov. 1992. ([document](#))
- [15] DEVSCAPE. ([document](#))
- [16] DOUD, C., “Composing, producing and implementing an interactive music soundtrack for games,” in *2003 Game Developers Conference Proceedings*, GDC Lectures, Mar. 2003. ([document](#))

- [17] EDE, R., “Techniques for implementing interactive surround sound,” in *2003 Game Developers Conference Proceedings*, GDC Lectures, Mar. 2003. 10
- [18] EDWARDS, E., “Linear Predictive Coding for voice compression and effects,” in *Game Programming Gems* (TREGLIA, D., ed.), no. 3 in *Game Programming Gems*, Charles River Media Inc., 2002. ([document](#))
- [19] EDWARDS, E., “Plugged-in generation; more than ever, kids are at home with media,” *The Washington Post*, July 1999. 4
- [20] FAY, T. M., “Puzzle scoring: An adaptive music case study for russian squares,” in *DirectX Audio Exposed: Interactive Audio Development* (FAY, T. M., SELFON, S., and FAY, T. J., eds.), Wordware Publishing, Inc., Oct. 2003. 13
- [21] FRATTAROLI, E., “L’audio multicanale e le console,” in *Evoution*, Apr. 2002. 9
- [22] FUCHS, M., “The history of computer games - from Spacewar to tournament.” ([document](#))
- [23] FUNKHOUSER, T., “Geometric modeling of sound propagation,” in *2003 Game Developers Conference Proceedings*, GDC Lectures, Mar. 2003. 3
- [24] FUNKHOUSER, T., CARLBOM, I., ELKO, G., PINGALI, G., SONDHI, M., and WEST, J., “A beam tracing approach to acoustic modeling for interactive virtual environments,” *Computer Graphics - Annual Conference Series*, vol. 32, pp. 21–32, July 1998. 4
- [25] GARDNER, W., “Reverberation algorithms,” in *Applications of Digital Signal Processing to Audio and Acoustics* (KAHRS, M. and BRANDENBURG, K., eds.), part 3, pp. 85–131, Kluwer Academic Publishers, 1998. 5
- [26] GARDNER, W. G., “3d audio and acoustic environment modeling,” Mar. 1999. ([document](#))
- [27] GARDNER, W. G. and MARTIN, K. D., “HRTF measurements of a KEMAR,” *The Journal of the Acoustical Society of America*, vol. 97, pp. 3907–3908, June 1995. ([document](#))
- [28] HARLAND, K., “Composing for interactive music,” *Gamasutra*, Feb. 2000. 10
- [29] HIEBERT, G., “Creating a compelling 3D audio environment,” in *Game Programming Gems 3* (TREGLIA, D., ed.), *Game Programming Gems*, ch. 6.2, pp. 595–599, Charles River Media, July 2002. ([document](#))
- [30] HIRST, A., “The art and craft of composing interactive music,” in *2003 Game Developers Conference Proceedings*, GDC Lectures, Mar. 2003. 12
- [31] INDUSTRIES, G., “Classicgaming.com’s museum,” Dec. 2000. ([document](#)), 6
- [32] ISENSEE, P. and GAMEN, S., “Developing online console games,” in *Game Developer Magazine*, CMP Media LLC, Mar. 2003. 1

- [33] JAMES F. O'BRIEN, P. R. C. and ESSL, G., "Synthesizing sounds from physically based motion," in *Computer Graphic Proceedings*, ACM SIGGRAPH Annual Conference Series, Aug. 2001. ([document](#))
- [34] KENT, S. L., *The Ultimate History of Video Games: From Pong to Pokemon—the Story behind the Craze That Touched Our Lives and Changed the World*. Rocklin, CA, USA: Prima Communications, Inc., 2001. ([document](#))
- [35] KREBS, E. M., *An Audio Architecture Integrating Sound and Live Voice for Virtual Environments*. Master's thesis, Monterey Naval Postgraduate School, 2000. [2](#)
- [36] L. SAVIOJA, J. HUOPANIEMI, T. L. and VAANANEN, R., "Virtual environment simulation - advances in the DIVA project," *International Conference on Auditory Display*, pp. 43–46, Nov. 1997. [4](#)
- [37] LEONARD, S., "Scores of glory, fantasy, and plumbing: The concise history and principal genres of video game music," Jan. 2001. [7](#)
- [38] MAGAZINE, X., "Sintetizzatori, campionatori ed effetti sonori," in *XBM*, Oct. 2001. [7](#)
- [39] MARKS, A., "Interview with Tommy Tallarico," *Gamasutra*, Mar. 1999. [I](#)
- [40] MARKS, A., "The use and effectiveness of audio in HALO: Game music evolved," *Music4Games*, 2001. ([document](#))
- [41] McDONALD, G., "A brief timeline of video game music," Apr. 2002. ([document](#))
- [42] MENSHIKOV, A., "Modern audio technologies in games," 2003. [2](#)
- [43] MOORE, G., "Talking Heads," in *Game Developer Magazine*, CMP Media LLC, Mar. 2001. [3](#)
- [44] O'DONNELL, M., "Producing audio for Halo," *Gamasutra*, Mar. 2002. ([document](#))
- [45] PATTERSON, S., "Interactive music sequencer design," *Gamasutra*, May 2001. [8](#)
- [46] PIDKAMENY, E., "Levels of sound," May 2002. ([document](#))
- [47] POOLE, S., *Trigger Happy: Video Games and the Entertainment Revolution*. Arcade Publishing, 2000. [I](#)
- [48] ROSENTHAL, M., "Dr. Higinbotham's experiment: The first video game [or: Fun with an oscilloscope]," 1996. [1](#)
- [49] ROSS, R., "Interactive music...er, audio," *Gamasutra*, May 2001. [9](#)
- [50] SCHMIDT, B., "Designing sound tracks for coin-op games," in *Proceedings of the 1989 International Computer Music Conference*, pp. 274–275, 1989. ([document](#))
- [51] SCHMIDT, B., "The art of noise - 3D audio," 2002. [2](#)

- [52] SENSAURA, “gameCODA™ concepts guide issue 1,” May 2002. ([document](#))
- [53] SIMPSON, J., “3d sound in games,” July 2000. [1](#)
- [54] SORGER, M., “Top down bottom up game audio: What i learned from Ken Griffey, Jr. Baseball.” [I](#)
- [55] STEVENSON, R., “The art of noise - game studio recording and foley,” in *Game Developer Magazine*, CMP Media LLC, Oct. 2000. [3](#)
- [56] TAYLOR, B., “2A03 sound channel hardware documentation,” Feb. 2003. [4](#)
- [57] VERRETTE, M., “Managing the nonlinear mix,” in *Game Developer Magazine*, CMP Media LLC, June 2003. [6](#)
- [58] VOGELSANG, C., “Obstruction using axis-aligned bounding boxes,” in *Game Programming Gems 3* (TREGLIA, D., ed.), Game Programming Gems, ch. 6.3, pp. 600–605, Charles River Media, July 2002. [6](#)
- [59] WALL, J., “Using living, breathing musicians in game music,” in *Game Developer Magazine*, CMP Media LLC, Mar. 2003. ([document](#))
- [60] WOLF, M. J., *Genre and the Video Game*. The Medium of the Video Game, University of Texas Press, 2002. [1](#), [1](#)
- [61] YAHOO! NEWS. [6](#)