CHAPTER 2

Push Start Button: The Rise of Video Games

If video games had parents, one would be the bespectacled academic world of computer science and the other would be the flamboyant and fun penny arcade, with a close cousin in Las Vegas. Many of the thematic concepts of the earliest video games (such as racecar driving, hunting, baseball, and gunfights) had first been seen in the mechanical novelty game machines that lined the Victorian arcades.[1] These novelty game machines date back to at least the nineteenth-century *Bagatelle* table, a kind of bumper-billiards. The *Bagatelle* developed into the pinball machine, first made famous by the *Ballyhoo* in 1931, created by the founder of Bally Manufacturing Company, Raymond Maloney. Within two years of the *Ballyhoo*, pinball machines were incorporating various bells and buzzers, which served to attract players and generate excitement. One early example of pinball sound was found in the Pacific Amusement Company's *Contact* (1934), which had an electric bell, designed by Harry Williams of Williams Manufacturing. Various electric bell and chime sounds were incorporated into the machines in the following decades, before electronic pinball machines became the fashion in the 1970s.

Related to the pinball and novelty arcades were gambling machines, notably the one-armed-bandit-style slot machine. The earliest slot machines, such as the *Mills Liberty Bell* of 1907, included a ringing bell with a winning combination, a concept that is still present in most slots today.[2] Playwright Noël Coward noted that sound was a key part of the experience in Las Vegas: ''The sound is fascinating . . . the noise of the fruit machines, the clink of silver dollars, quarters, nickels'' (cited in Ferrari and Ives 2005). As in the contemporary nickelodeons, sound's most important early role was its hailing function, attracting attention to

the machines (Lastra 2000, p. 98). More important is that sound was a key factor in generating the feeling of success, as sound effects were often used for wins or *near wins*, to create the illusion of winning.[3] Indeed, the importance of sound in attracting players and keeping them interested was not lost on these companies when they later ventured into the video arcade games market. Many of the same companies that were influential in the development of pinball machines also made slots, or became associated with slots through the creation of *pay out* machines, a combination of slots and pinball, which was developed in the 1930s during the Prohibition (Kent 2001, p. 5). It was these companies—Williams, Gottlieb, and Bally, for instance—that would become among the first to market electronic video arcade games.

The very earliest electronic video games, including William Higinbotham's never published tennis game of 1958, *Tennis for Two*, and *Spacewar!* (1962, developed at the Massachusetts Institute of Technology), had no sound. However, the first mass-produced video arcade game, pinball company Nutting Associates' *Computer Space* (1971), included a series of different ''space battle'' sounds, including ''rocket and thrusters engines, missiles firing, and explosions.''[4] A flyer advertising the machine highlights its sound-based interactions with the user: ''The thrust motors from your rocket ship, the rocket turning signals, the firing of your missiles and explosions fill the air with the sights and sounds of combat as you battle against the saucers for the highest score.''[5] The first real arcade hit, however, would be Atari's *Pong* (1972), which led to countless companies entering the games industry. By the end of the year following its original release, Williams had introduced a version of *Pong* called *Paddle Ball*, Chicago Coin had launched a very similar game called *TV Hockey*, Sega of Japan had introduced *Hockey TV*, and Brunswick offered *Astro Hockey*. Midway had cloned *Pong* with *Winner*, and created a follow-up, *Leader*. As *Pong's* designer Al Alcorn explains, ''There were probably 10,000 *Pong* games made, Atari made maybe 3,000. Our defense was . . . 'OK. Let's make another video game. Something we can do that they can't do''' (cited in Demaria and Wilson 2002, p. 22). The answer was *Space Race*, which would be cloned by Midway as *Asteroids* (1973). The video game industry had been born.

Pong was to some extent responsible for making the sound of video games famous, with the beeping sound it made when the ball hit the paddle. The *Pong* sound—as with many early games successes—was a bit of an accident, Alcorn recalls:

> The truth is, I was running out of parts on the board. Nolan [Bushnell, Atari's founder] wanted the roar of a crowd of thousands—the approving roar of cheering people when you made a point. Ted Dabney told me to make a boo and a hiss when you lost a point, because for every winner there's a loser. 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 half a day. They were the sounds that were already in the machine. (Cited in Kent 2001, pp. 41–42)

It is interesting to note, then, that the sounds were not an aesthetic decision, but were a direct result of the limited capabilities of the technology of the time.

Despite these humble beginnings, most coin-operated (coin-op) machine flyers of the era advertised the sound effects as a selling feature, an attribute that would attract customers to the machines, much as had been witnessed with pinball and slot machines. Drawing on their heritage, these early arcade games commonly had what was known as an *attract function*, which would call players to the machines when nobody was using them, and so games like *Barrel Pong* (Atari, 1972) or *Gotcha* (Atari, 1973) had ''Electronic sounds ... [which were] always beckoning.''[6] Also interesting was the proliferation of advertisements boasting ''realistic'' sounds (including that of *Pong*). It is not mentioned how players are to judge the realism of ''flying rocket'' sounds in Nutting's 1973 *Missile Radar*, or those of Project Support Engineering's 1975 *Jaws* tie-in *Man Eater*, which advertised a ''realistic chomp and scream.''[7] Of course, most players today would laugh at the attempts to describe these low-fidelity blips and bleeps as realistic. This drive toward realism, however, is a trend we shall see throughout the history of game sound.

In the arcades, sound varied considerably from machine to machine, with the sound requirements often driving the hardware technology for the game. A 1976 game machine programming guide described how the technical specificity drove the audio on the machines, and vice versa: ''Sound circuits are one of several areas which show little specific similarity from game to game. This is a natural result of designers needing very different noises for play functions of games where the theme of the machines varies greatly. For example, a shooting game requires a much different sound circuit design than a driving game.''[8] Indeed, genre sound codifications (discussed in chapter 7) began quite early, although the coin-op arcade games also developed in a particular way owing to the sonic environment of the arcade. Sound had to be loud, and sound effects and percussion more prominent, in order to rise above the background noise of the arcade, attract players, and then keep them interested.

Sound was difficult to program on the early machines, and there was a constant battle to reduce the size of the sound files owing to technological constraints, as Garry Kitchen, developer for many early games systems described: ''You put sound in and take it out as you design your game.... You have to consider that the sound must fit into the memory that's available. It's a delicate balance between making things good and making them fit'' (cited in Martin 1983). Typically, the early arcade games had only a short introductory and ''game over'' music theme, and were limited to sound effects during gameplay. Typically the

Box 2.1
Sound Synthesis in Video Games

(Note: There are ample excellent discussions of synthesis on the Internet, in journals, and in books on acoustics, computer music, synthesis, and so on. I will, therefore, only quickly summarize the main types relevant to video game audio here, with a note to their relevance.)

Programmable sound generators (PSGs) are sound chips designed for audio applications that generate sound based on the user's input. These specifications are usually coded in assembly language to engage the *oscillators*. An oscillator is an electric signal that generates a repeating shape, or wave form. *Sine waves* are the most common form of oscillator. An oscillator is capable of either making an independent tone by itself, or of being paired up cooperatively with its neighbor in a pairing known as a *generator*. Instrument sounds are typically created with both a waveform (tone generator) and an envelope generator. Many video game PSGs were created by Texas Instruments or General Instruments, but some companies, such as Atari and Commodore, designed their own sound chips in an effort to improve sound quality.

Subtractive synthesis, common in PSGs, starts with a waveform created by an oscillator, and uses a filter to attenuate (subtract) specific frequencies. It then passes this new frequency through an amplifier to control the envelope and amplitude of the final resulting sound. Subtractive synthesis was common in analog synthesizers, and is often referred to as *analog synthesis* for this reason. Most PSGs were subtractive synthesis chips, and many arcades and home consoles used subtractive synthesis chips, such as the General Instruments AY-8910 series. The AY-8910 (and derivatives) found its way into a variety of home computers and games consoles including the Sinclair ZX Spectrum, Amstrad CPC, Mattel Intellivision, Atari ST, and Sega Master System.

Frequency modulation (FM) synthesis was one of the major sound advances of the 16-bit era. FM synthesis was developed by John Chowning at Stanford University in the late 1960s, and licensed and improved upon by Yamaha, who would use the method for their computer sound chips, as well as their DX series of music keyboards. FM uses a modulating (usually sine) wave signal to change the pitch of another wave (known as the *carrier*). Each FM sound needs at least two signal generators (oscillators), one of which is the carrier wave and one of which is the

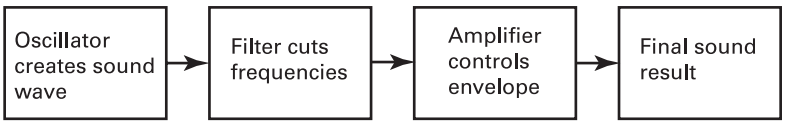

Figure B2.1
Subtractive synthesis method of sound generation.

Box 2.1
(continued)

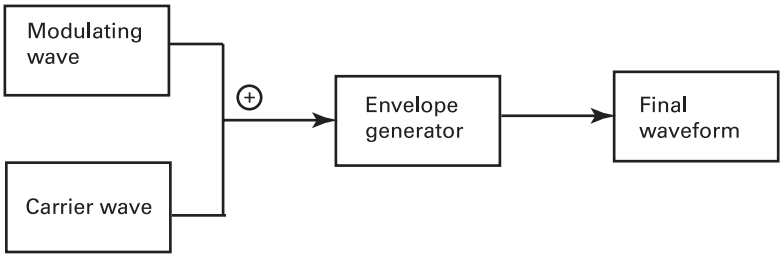

FIGURE B2.2
FM synthesis method of sound generation.

modulating wave. Many FM chips used four or six oscillators for each sound, or instrument. An oscillator could also be fed back on itself, modulating its original sound.

FM sound chips found their way into many of the early arcade games of the late 1970s and early 1980s, and into most mid-1980s computer soundcards. Compared with other PSG methods of the era, FM chips were far more flexible, offering a much wider range of timbres and sounds. Arcades of the 16-bit era typically used one or more FM synthesis chips (the Yamaha YM2151, 2203, and 2612 being the most popular).

WAVETABLE SYNTHESIS, also introduced in the 16-bit era, uses preset digital samples of instruments (usually combined with basic waveforms of subtractive synthesis). It is therefore much more ''realistic'' sounding than FM synthesis, but is much more expensive as it requires the soundcard to contain its own RAM or ROM. The Roland MT-32 used a form of wavetable synthesis known as *linear arithmetic*, or LA synthesis. Essentially, what the human ear recognizes most about any particular sound is the attack transient. LA-based synthesisers used this idea to reduce the amount of space required by the sound by combining the attack transients of a sample with simple subtractive synthesis waveforms.

GRANULAR SYNTHESIS is a relatively new form of synthesis (having begun with the stochastic method composers, such as Iannis Xenakis, in the 1970s), which is based on the principle of *microsound*. Hundreds—perhaps thousands—of small (10–50 millisecond) granules or ''grains'' of sound are mixed together to create an amorphous soundscape, which can be filtered through effects or treated with envelope generators to create sound effects and musical tones. Leonard Paul at the Vancouver Film School is currently working on ways to incorporate granular synthesis techniques into next-generation consoles (see Paul 2008 for an introduction to granular synthesis techniques in games).

music only played when there was no game action, since any action required all of the system's available memory.

Continuous music was, if not fully introduced, then arguably foreshadowed as one of the prominent features of future video games as early as 1978, when sound was used to keep a regular beat in a few popular games. In terms of *non-diegetic* sound,[9] *Space Invaders* (Midway, 1978) set an important precedent for continuous music, with a descending four-tone loop of marching alien feet that sped up as the game progressed. Arguably, *Space Invaders* and *Asteroids* (Atari, 1979, with a two-note ''melody'') represent the first examples of continuous music in games, depending on how one defines music. Music was slow to develop because it was difficult and time-consuming to program on the early machines, as Nintendo composer Hirokazu ''Hip'' Tanaka explains: ''Most music and sound in the arcade era (*Donkey Kong* and *Mario Brothers*) was designed little by little, by combining transistors, condensers, and resistance. And sometimes, music and sound were even created directly into the CPU port by writing 1s and 0s, and outputting the wave that becomes sound at the end. In the era when ROM capacities were only 1K or 2K, you had to create all the tools by yourself. The switches that manifest addresses and data were placed side by side, so you have to write something like '1, 0, 0, 0, 1' literally by hand'' (cited in Brandon 2002). A combination of the arcade's environment and the difficulty in producing sound led to the primacy of sound effects over the music in this early stage of game audio's history.

By 1980, arcade manufacturers included dedicated sound chips known as *programmable sound generators*, or PSGs (see box 2.1, ''Sound Synthesis'') into their circuit boards, and more tonal background music and elaborate sound effects developed. Some of the earliest examples of repeating musical loops in games were found in *Rally X* (Namco/Midway, 1980), which had a six-bar loop (one bar repeated four times, followed by the same melody transposed to a lower pitch), and *Carnival* (Sega, 1980, which used Juventino Rosas's ''Over the Waves'' waltz of ca. 1889). Although *Rally X* relied on sampled sound using a *digital-to-analog converter* (a DAC: see box 2.2, ''Sampling''), *Carnival* used the most popular of early PSG sound chips, the General Instruments AY-3-8910. As with most PSG sound chips, the AY series was capable of playing three simultaneous square-wave tones, as well as white noise (what I will call a *3+1 generator*, as it has three tone channels and one noise channel; see box 2.3, ''Sound Waves''). Although many early sound chips had this four-channel functionality, the range of notes available varied considerably from chip to chip, set by what was known as a *tone register* or *frequency divider*. In this case the register was 12-bit, meaning it would allow for 4,096 notes (see box 2.2). The instrument sound was set by an envelope generator, manipulating the attack, decay, sustain, and release (*ADSR*) of a sound wave. By adjusting the ADSR, a sound's amplitude and filter cut-off could be set.

Box 2.2
Sampling

A *bit*, derived from binary digit, is the smallest unit of information in computer language, a one (1) or zero (0) (also sometimes referred to as ''on or off,'' or ''white or black''). In referring to processors, the number of bits indicates how much data a computer's main processor can manipulate simultaneously. For instance, an 8-bit computer can process 8 bits of data at the same time.

Bits can also be used to describe sound fidelity or resolution. *Bit depth* is used to describe the number of bits available in a byte. Higher bit depths result in better quality or *fidelity*, but larger file sizes. 8 bits can represent 2^8 (binary being base 2), or 256 variations in a byte. Adding one bit doubles the accuracy, or number of levels. At 16 bits, there are 65,536 possible states ($2^{16} = 65{,}536$). When recording sound, 256 divisions are not very accurate, since the amplitude of a wave is rounded up or down to fit the nearest available point of resolution. This process, known as *quantization*, distorts the sound or adds noise. CD quality sound is considered 16-bit, although often the CDs are recorded in 24-bit and converted to 16-bit before release. Figure B2.3 simplifies the process, by showing a 4-bit sample (16 sample points along the positive and negative amplitudes), with amplitudes sampled at 16 times per second. The black wave line shows the original sound wave, and the gray line shows the sample points that would occur. As you can see from the gray line, the original sound is considerably changed by the *sampling* of the sound at a low rate.

A *sample* contains the information of the amplitude value of a waveform measured over a period of time. The sample rate is the number of times the original sound is sampled per second, or the number of measurements per second. Sample rate is also known as sample frequency: A CD quality sample rate of 44.1 KHz means that 44,100 samples per second were recorded. If the sample rate is too low, a distortion known as aliasing will occur, and will be audible when the sample is converted back to analog by a *digital-to-analog converter* (DAC). Analog-to-digital

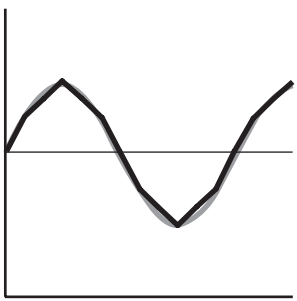

Figure B2.3
Bit depth, showing a 4-bit sample.

Box 2.2
(continued)

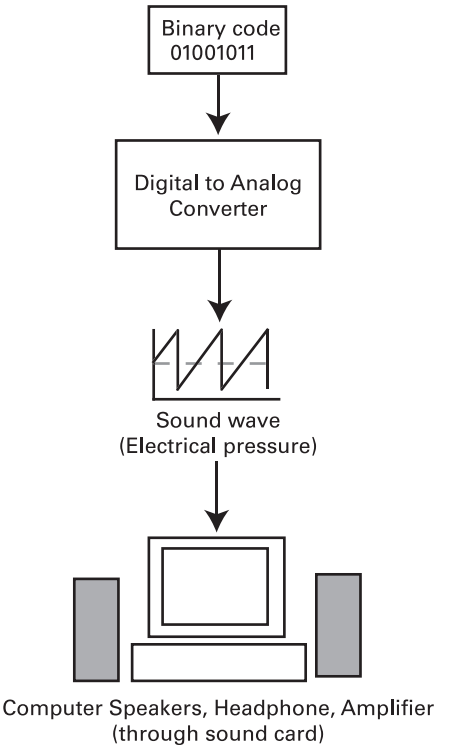

Figure B2.4
Digital-to-analog converter (DAC).

converters (ADCs) typically have an anti-aliasing filter that removes harmonics above the highest frequency that the sample rate can accommodate.

The recreation of a sound wave from sample data (binary code) to an analog current (an electrical pressure soundwave) is performed by a DAC (figure B2.4). DACs have bit depths and sample rates. The higher the bit rate and sample rate, the better the resulting sound. DACs most often work through *pulse code modulation* (PCM, otherwise known as raw, or AI2 synthesis), which refers to an analog sound converted into digital sound by sampling an analog waveform. The data is stored in binary, which is then decoded and played back as it was originally recorded. The downside of this method is the amount of space required to store the samples: as a result, most PCM samples in early games were limited to those sounds with a short envelope, such as percussion. 8-bit PCM samples commonly had an audible hiss because of resolution problems.

Box 2.2
(continued)

Adaptive differential PCM (also known as adaptive delta PCM, or ADPCM), is essentially a method of compressing a PCM sample. The difference between two adjacent sample values is quantified, reducing or raising the pitch slightly, to reduce the amount of data required. ADPCM uses only 4 bits per sample, therefore requiring only one quarter of the space of a 16-bit PCM sample. This works well for lower frequencies, but at higher frequencies can lead to distortion. ADPCM speech chips made their way into late 1980s coin-op machines, such as in the OKI Electric Industry Co.'s OKI 6295 chip, used in *Hit the Ice* (Williams, 1990, which used a YM 2203 and two speech chips, since it had a lot of voice parts, including announcers and crowds), and *Pit Fighter* (Atari, 1990, using a YM2151 and a speech chip).

The AY-3-8910 (and derivatives) found its way into a variety of home computers and game machines including the Sinclair ZX Spectrum, Mattel Intellivision, and the Sega Master System. Similarly, another popular arcade chip, the Texas Instruments SN76489, was shared with a few computers of the time, such as the BBC Micro, as well as consoles like the ColecoVision and the Sega Genesis. The SN76489 was also a 3+1 sound chip, although the frequency divider was limited to 10-bit, meaning only 1,024 possible pitches, and was, therefore, slightly inferior to the AY series.[10] Most of these chips were capable of playing short, low-fidelity samples, typically used for sound effects, or percussion, using *pulse width* or *pulse code modulation* (see box 2.2).

By 1980, most game systems had co-processors specifically to deal with sound, although the majority of games had yet to develop any continuous music. Roughly half of coin-ops were using DACs (such as Nintendo's original *Donkey Kong* of 1981) and half PSGs, usually the AY series (such as Atari's *Centipede* of 1980), the SN chip (such as Nintendo's *Sheriff* of 1980) or Atari's own custom chip, the Pokey (such as in *Battle Zone* or *Missile Command* [both Atari, 1980]). Soon it became increasingly common to use more than one sound chip in a coin-op game, as in *Front Line* (Taito, 1982), which used four AY chips and a DAC. The additional sound chips were typically used for more advanced sound effects, rather than increased polyphony for music. The likely reason for this was a combination of the arcade's atmosphere and the difficulty in programming music, as discussed above. Competing machines had to be loud, with short, simple, but exciting sounds that would attract players. The advantage of separate chips for music, however, meant that any music included could play without being interrupted by the sound effects having to access the same chip. As this idea became more common, an increasing number of games incorporated music, such as *Alpine Ski* (four AY chips and a DAC, Taito, 1983) and *Jungle Hunt* (four AYs and

Box 2.3
Sound Waves

Sound waves are described using three properties: wavelength, frequency, and amplitude (see figure B2.5). (The fourth, velocity [velocity = wavelength × frequency] is typically the same for all sound waveforms and so is not discussed here.)

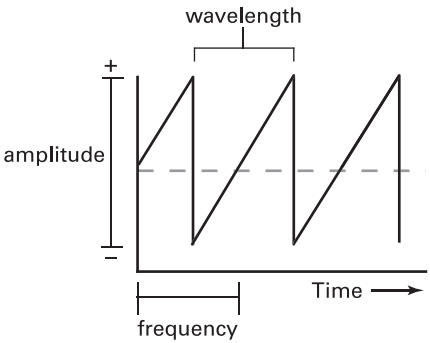

Figure B2.5
Anatomy of a sound wave.

Wavelength is the distance from one peak of a wave to the next, or the distance between maximum compressions. *Frequency*, the technical name for pitch, is a measure of the number of pulses (waves) in a given space of time. It is measured in Hertz, or CPS (cycles per second). For example, a note with a frequency of 440 Hz (A), means that in one second, 440 pulses occur. Shorter wavelengths result in higher frequencies. *Amplitude* is the measure of the amount of energy in a wave (technically, the amount of compression the wave is under), typically described as intensity, or loudness. The more energy a sound has, the more intense, or loud, the sound that results. Loudness is measured in decibels (dB).

Regular, or *periodic*, waveforms are considered pleasing to the ear, and can take several forms, including:

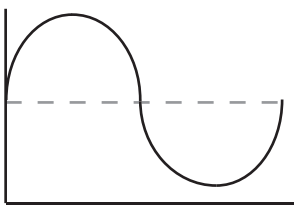

Figure B2.6
Sine wave.

Box 2.3
(continued)

Sine waves have only one frequency associated with them—they are ''pure'' in that they have no harmonics. They are also referred to as ''pure tones.'' In games, sine waves are often used for certain sound effects (laser, alarm), or for flute-like melodic parts.

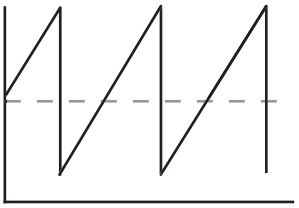

Figure B2.7
Ramp wave.

Sawtooth waves are so named because they resemble the teeth on a saw. They are also sometimes referred to as *ramp waves*. Sawtooth waves typically ramp upward and then drop sharply, although the opposite are also found (inverse/reverse sawtooth waves). Sawtooth waves contain both odd and even harmonics. Sawtooth waveforms in games are used to create bass parts, as it resembles a warm, round sound.

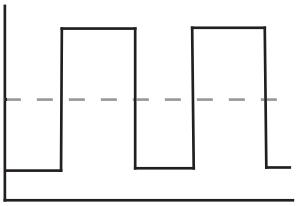

Figure B2.8
Pulse wave.

Pulse waves contain only odd harmonics, and are rectangular waveforms with ''on'' and ''off'' slopes, known as the *duty cycle*. When the duty cycle is of equal length in its ''on'' and ''off'' period, it is known as a square wave. Changing the duty cycle options (changing the ratio of the ''on'' to ''off'' of the waveform), alters the harmonics. At 50 percent (square wave), the waveform is quite smooth, but

Box 2.3
(continued)

with adjustments can be ''fat,'' or thin and ''raspy''). Square waves are often referred to as ''hollow'' sounding.

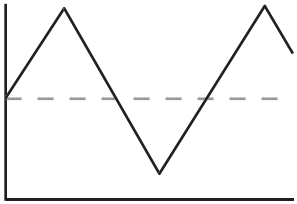

Figure B2.9
Triangle wave.

Triangle waves contain only odd harmonics, like pulse waves; however, in triangle waves, harmonics finish much faster, and so the resultant sound is much smoother, sounding similar to a sine wave.

Figure B2.10
Noise.

White noise is a sound that contains every frequency within the range of human hearing in equal amounts. In games, it is commonly used for laser sounds, wind, surf, or percussion sounds. Pink noise is a variant of white noise. Pink noise is white noise that has been filtered to reduce the volume at each octave. It is also commonly used for rain or percussion sounds in games, sounding like white noise with more bass.

a DAC, Taito 1983). Sometimes, as many as five synthesis chips and a DAC were used (such as *Gyruss*, Konami, 1983, which appears to use at least two chips for sound effects, one for percussion, and at least one chip to create a rendition of Bach's *Tocatta and Fugue in D minor*).[11]

Speech chips, which could be used for short vocal samples or for sound effects, also began to see more prominence in the early 1980s.[12] Atari included a Texas Instruments TMS5220 chip (which had been used in *Speak 'n' Spell*, the popular family electronic game) in several games, such as *Star Wars* (1983) and *Indiana Jones and the Temple of Doom* (1985). With a separate chip to handle sound effects and voice, the primary sound chip's noise channel could be freed up, allowing for more complex music and advanced sounding effects, such as in *Discs of Tron* (Atari, 1983), which was also one of the first games to use stereo sound.[13]

In a market driven by fierce competition, innovation was a key ingredient in the success of many early arcade games, as a manual for programming games describes as early as 1976: ''Today, jaded players have become bored by the myriad of variations of these first games and increasingly more dramatic game action is required to stimulate the average player who might still play a fifteen year old pin ball machine, but is not at all interested in last year's video game'' (Kush N' Stuff Amusement Electronics, Inc., 1976, p. 6). In addition to the technological hardware advances that distinguished arcade machines from their competitors, there were also some novel beginnings in the software programming of game audio in some of the very earliest arcade games. Although it was to some extent a response to the technological constraints of the time, looping was an aesthetic that developed in the early years of game music. There were a few early examples of games with loops (such as those discussed above), but it was not until 1984 that music looping in video games began to gain real prominence. This change to a looping aesthetic is most obvious when examining the ColecoVision games, where there is a clear division between the nonlooping games of 1982 and 1983 (e.g., *Tutankhamun*, *Miner 2049er*, *Jungle Hunt*, *Dig Dug*, *Congo Bongo*, and so on) and the games of 1984, most of which have loops (e.g., *Gyruss*, *Sewer Sam*, *Tarzan*, *Burger Time*, *Antarctic Adventure*, and *Up N Down*), despite the fact that the hardware remained the same. Such a change in aesthetic is also evident in Nintendo's home console games, where the very first games released in 1983 and 1984 (*Donkey Kong*, *Donkey Kong Jr.*, *Popeye*, and *Devil World*) had only very short one- or two-bar loops (*Popeye's* loop was eight bars, but it was the same two bars transposed into different pitches), but later games increased the number and length of looping parts.

There were also some nonlooping programming practices during this era that would go on to influence future developments in game music. *Frogger* (Konami, 1981) was one of the first games to incorporate dynamic music. The

game, in which the player guides a frog past cars and over moving logs into a series of four safe-houses, used at least eleven different gameplay songs, in addition to ''game over'' and the level's start themes. The player began in the main gameplay theme, and when he or she successfully guided a frog into a safe-house, the song would switch to another quite abruptly, continuing until a new frog either was successfully guided into another safe-house (moving onto a new song), or died (returning to the gameplay song). Since the maximum time a gameplay could last before arriving at a safe-house or dying was about thirty seconds (much less as the levels increased), the songs did not need to loop. A similar approach was found in Jetsoft's *Cavelon* (1983). The player moved about the screen capturing various items and pieces of a door, and when the player captured a piece, the loop changed into a new sequence after a brief ''win item'' cue. Each time the player stopped moving, the music also stopped, an approach that was also seen in *Dig Dug* (Namco/Atari, 1983). These techniques are discussed further in chapter 8.

INVADERS IN OUR HOMES: THE BIRTH OF HOME CONSOLES

Although home game consoles had existed before their coin-operated counterparts, it was not until the success of video games in the arcades along with the decrease in the cost of microprocessors that home consumer versions were mass-produced. The Magnavox Odyssey, released in 1972 (in black and white, with no sound) had some success, but it was Atari, piggybacking on their arcade hit and releasing *Pong* on the Sears Tele-Games system in 1975, which really brought gaming home to the masses. By the following year, some seventy-five companies had launched a home version of *Pong*, nearly all using a General Instruments chip that had been made available to any manufacturer, which became known as the ''*Pong* Chip'' (i.e., the AY-3-8500: see Kent 2001, p. 94). Not only would the graphics of *Pong* be reproducible, but the *Pong* sound was carried into hundreds of versions of the game.

Although there would be other popular consoles, it was another Atari release, the Video Computer System, or VCS (later known as the 2600), relying on a cartridge system, that was to revolutionize home gaming and become the longest-running console ever, sold from 1977 until 1992. The Atari VCS saw limited success when it was first released, and the machine struggled during its first few years of production. In 1980, however, Atari licensed the popular arcade game *Space Invaders*, which became a best seller and helped to spur on the sales of the VCS. Eventually, over 25 million homes owned a VCS, and over 120 million cartridges had been sold.[14]

The sound chip in the VCS was manufactured specifically by Atari for sound and graphics, and was known as the Television Interface Adapter, or TIA chip. The audio portion had just two channels, meaning whatever music and sound effects were to be produced could only be heard on two simultaneous voices, mixed into a mono output. Each channel had a 4-bit waveform selector, meaning there were sixteen possible waveform settings, though several were the same or similar to others. Typically, the usable waveform options were two square waves (one treble, one bass), one sine wave, one saw wave, or several noise-based sounds useful for effects or percussion.[15] Sound effects were often reduced to simple sine wave tones of one volume, or noise. The trouble with the tonal sounds, however, was that each channel had a different tuning, so that in music, the pitch value would often be different between the bass and the lead voice.

The awkward tuning on the VCS was due to the TIA's 5-bit pseudo-random frequency divider, capable of dividing a base frequency of 30 KHz by 32 values. Starting with one base tone, that frequency was then divided between 1 and 32 to obtain the other notes in the *tuning set*, or note options available to the composer. To compound the problem, there were slight variations between the frequencies on the NTSC (the North American television broadcast standard) and PAL (the European format) versions of the machine. At times, pitches were off by as much as fifty cents (half a semitone) (Stolberg 2003). Depending on the random division, tuning sets could be quite variable, as some sets would allow for more bass notes, while others would allow for more treble, and since many sets would have conflicting tunings between bass and treble, they were useless for most tonal compositional purposes. Paul Slocum, creator of an Atari VCS sequencing kit for *chip-tunes* composers who incorporate the old sound chips into contemporary compositions, advises, ''Although each set contains notes that are close to being in-tune, you can still end up with songs that sound pretty bad if you aren't careful'' (Slocum 2003).

The tuning set example shown in table 2.1 gives us five tonal voices from which to choose our melody or bass. Pitches are given as closest to equal tuning temperament, but depending on whether or not the system is NTSC or PAL, the actual pitch can vary. For instance, A4 (440Hz) on the lead square-wave voice would be off by thirteen cents on an NTSC machine, and by twenty-seven cents on a PAL machine. Examining the tuning set, the most complete range in terms of a chromatic scale within any one octave is the square wave, which allows only six out of the twelve notes (A, B, C, D, E, and G in the fifth octave), though on a PAL machine these were nearly all very out of tune (tuning calculations are from Stolberg 2003).

The fact that the tuning was different between different voices (there may have been a G available in the bass, but only a G-sharp in the treble channel, for

Table 2.1
An example tuning set for the Atari VCS, showing a typical frequency selection, and distance in cents from equal tuning on NTSC and PAL formats.

Bass, Pitfall

NOTE	NTSC	PAL
F#2	−6	−19
F1	+16	+4
D#1	+4	−9
C1	0	−11

Low Bass

NOTE	NTSC	PAL
B0	−11	−22
G#0	0	−13

Lead (square wave)

NOTE	NTSC	PAL
E8	−11	−25
E7	−11	−25
A6	−14	−27
E6	−11	−25
C6	+2	−11
A5	−14	−27
E5	−12	−25
D5	−16	−29
C5	+2	−11
A4	−13	−27
F4	0	−13
E4	−11	−25
D4	−16	−29
C4	+3	−11
A3	−14	−27
G3	−17	−31
F3	+1	−13
E3	−11	−25

Saw

NOTE	NTSC	PAL
C7	+2	−1
C6	+2	−1
F5	0	−1
C5	+2	−1
F4	0	−13
C4	+3	−11
A#3	−2	−15
F3	+1	−13
C3	+3	−11
B2	−3	−16
A#2	0	−14
A2	+5	−8
F2	0	−12
D#2	−5	−18
C2	+3	−11

Square

NOTE	NTSC	PAL
B8	−9	−23
E8	−11	−25
B7	−11	−25
G7	+4	−9
E7	−11	−25
B6	−9	−23
A6	−13	−27
G6	+4	−9
E6	−11	−25
C6	+2	−11
B5	−10	−23
A5	−14	−27
G5	+4	−9
E5	−12	−25
D5	−16	−29
C5	+2	−11
B4	−9	−23

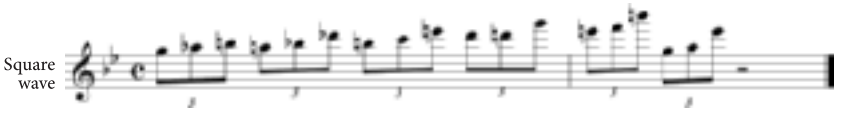

Figure 2.1
Tapeworm (Spectravision, 1982).

instance) complicated programming in harmony, and it is little wonder that very few VCS games included songs with both bass and treble voices. These complications in programming songs for the Atari VCS (in addition to the awkward assembly language) meant that there was very little music, and what there was tended to use rather uncommon keys or notes. For instance, if a composer chose the saw wave sound from the figure 2.1 chart, the bass (say, the lower two octaves of the chart) is limited to C, D-sharp/E-flat, F, A, A-sharp/B-flat, and B, and he or she would be left without either a full diatonic or chromatic scale to play with. Such limitations meant that a composer may end up with something as peculiar as the theme song for *Tapeworm* (Spectravision, 1982) (figure 2.1).

Comparing Atari games with their arcade counterparts shows a clear distinction in the capabilities of the chips. The bass lines of the arcade versions were often abandoned when ported to the VCS, since there was little chance of finding compatible lead and bass voices on the TIA chip (such as *Burger Time*, Data East, 1982). More notably, the songs often have a distinctly different flavor. *Up N Down* (Sega, 1983) in particular suggests that the Atari's tunings may have played a significant role in the sound of the machine, as the tune changed from a bluesy F-sharp minor groove (figure 2.2) to a very unsettling version based in C minor with a flattened melodic second (figure 2.3; see Collins 2006 for a more detailed discussion of this phenomenon).

Home console sound would be improved by Atari's chief competitors, Mattel and Coleco. Mattel's answer to the Atari VCS was the Intellivision (Intelligent Television), which was significantly more advanced in sound and graphics.[16] Also important was its modular design, allowing for extensions such as the Entertainment Computer System, consisting of a music keyboard and second sound chip, leading to six simultaneous channels. The original Intellivision used a General Instruments PSG sound chip that had been popular in the arcades, an AY-3-8914. The chip meant that the Intellivision could create recognizable renditions of precomposed music, such as Bill Goodrich's use of ''Flight of the Bumblebee'' (Rimsky-Korsakov) in the game *Buzz Bombers* (Intellivision Productions, 1983). Indeed, since most programmers were not musicians or were under strict time constraints, precomposed songs were frequently used on the early machines (see chapter 6). By the late 1980s, music composition on the Intellivision became easier when a program was created by programmer–composer Dave

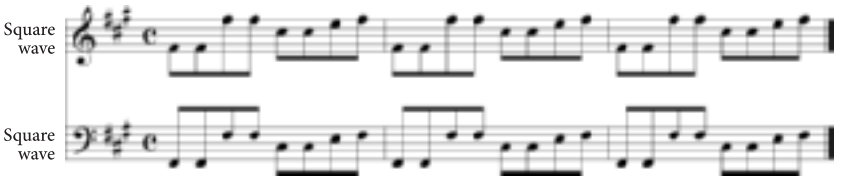

Figure 2.2
Up N Down — Arcade Version (Sega, 1983).

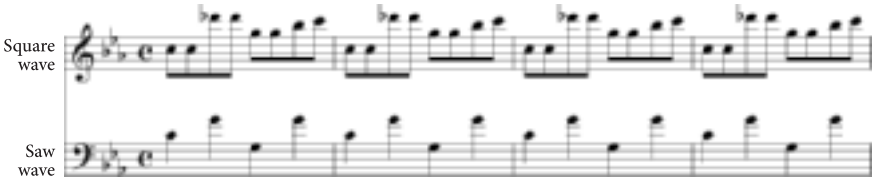

Figure 2.3
Up N Down — Atari VCS Version (Sega, 1983).

Warhol that could convert musical data (MIDI) files directly into Intellivision code; but by that time Intellivision had seen the peak of its success. (See chapter 3 for an explanation and discussion of the role of MIDI.)

Competing with Mattel and Atari was Coleco, who had experienced only moderate success with their earlier Telstar console. ColecoVision consoles, beginning in 1982, were shipped with the Nintendo arcade hit *Donkey Kong*, which helped spur on sales for the machine. The ColecoVision used the Texas Instrument SN76489 sound chip that had been common in arcade games.

Despite the moderate success of the ColecoVision and Intellivision, as well as the success of other companies entering the market during the early 1980s (e.g., General Consumer Electric's Vectrex, Emerson Radio Corp's Arcadia 2001), for a number of reasons the games industry saw a significant drop in sales by the mid-1980s.[17] It was the release of the Nintendo Entertainment System or NES (known in Japan as the Famicom) that would help to revive the games industry and secure its future. The Japanese company had previously barely been able to break into the North American market, at a time when ''the American companies showed little interest. Game publishers such as Sierra On-Line, Brøderbund, and Electronic Arts were more interested in making games for computers than for consoles, and toy companies like Milton Bradley and Mattel had left the industry entirely'' (Kent 2001, p. 307). Nevertheless, with hits like *Super Mario Bros.* (Nintendo, 1985) and *The Legend of Zelda* (Nintendo, 1986), as well as cunning business practices (see chapter 3), Nintendo was to capture the American market and prove to the public and to retailers that video games were here to stay.

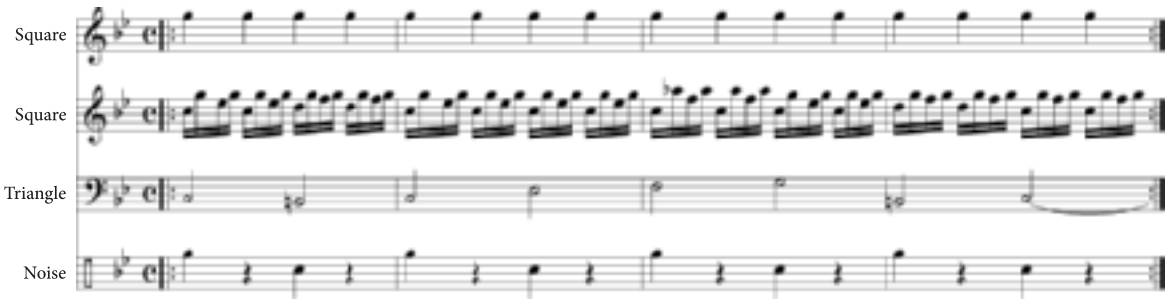

Figure 2.4
Castlevania, "Boss Music: Poison Mind" (*Akumajō Dracula*, Kinuyo Yamashita, Konami, 1987), showing the use of the tone channels in arpeggiating one channel.

The NES's sound chip, invented by composer Yukio Kaneoka, used a custom-made five-channel PSG chip. There were two pulse-wave channels capable of about eight octaves,[18] with four duty cycle options to set the timbre (see box 2.3). As well, one of the pulse-wave channels had a frequency sweep function that could create portamento-like effects, useful for UFOs or laser-gun sound effects. A triangle wave channel was one octave lower than that of the pulse waves,[19] and was more limited in pitch options, having only a 4-bit frequency control. The fourth, the noise channel, could generate white noise, which was useful for effects or percussion.[20] The fifth channel was a sampler, also known as the delta modulation channel (DMC), which had two methods of sampling. The first method was pulse code modulation, which was often used for speech, such as in *Mike Tyson's Punch-Out!* (Nintendo, 1987) or Tengen's *Gauntlet 2* (1990), and the second was known as *direct memory access*. This form of sampling was only 1-bit, and was more frequently used for sounds of short duration, such as sound effects (see box 2.2).

The NES's three tone channels were typically used in a fairly conventional way, with one channel for lead, one for accompaniment, and one for bass (and noise or DMC for percussion). The two pulse channels commonly worked as a chord or solo lead, with the triangle channel as a bass accompaniment. The most obvious reason for using the triangle as bass was the limitations of the channel, which included lower pitch, reduced frequencies, and no volume control. These limitations meant that many of the effects that could be simulated with the pulse waves, such as vibrato (pitch modulation), tremolo (volume modulation), slides, portamento, echo effects, and so on were unavailable for the triangle wave. At times, all three channels were used as chords (as in *Ultima's* battle music, in which two pulse waves create a chordlike lead in the first two channels, and the triangle creates the bass of the chord), or with one channel arpeggiated (as in *Castlevania*'s "Poison Mind," figure 2.4). The pulse channels also occasionally worked as counterpoint to each other, as in *Ultima's* "Overworld" music.

By altering the volume and adjusting the timing of the two pulse channels, phasing, echo effects, and vibrato could be simulated, as in *Metroid*'s ''Mother Brain'' and ''Kraid'' (Nintendo, 1987). *Metroid* also made other uncommon applications of the channels, such as the use of pulse wave for bass with triangle lead, in the ''Hideout'' music for the game. Indeed, *Metroid* represented a turning point in game music, as its composer Hirokazu ''Hip'' Tanaka explains:

> The sound for games used to be regarded just as an effect, but I think it was around the time *Metroid* was in development when the sound started gaining more respect and began to be properly called game music.... Then, sound designers in many studios started to compete with each other by creating upbeat melodies for game music. The pop-like, lilting tunes were everywhere. The industry was delighted, but on the contrary, I wasn't happy with the trend, because those melodies weren't necessarily matched with the tastes and atmospheres that the games originally had. The sound design for *Metroid* was, therefore, intended to be the antithesis for that trend. I had a concept that the music for *Metroid* should be created not as game music, but as music the players feel as if they were encountering a living creature. I wanted to create the sound without any distinctions between music and sound effects.... As you know, the melody in *Metroid* is only used at the ending after you killed the Mother Brain. That's because I wanted only a winner to have a catharsis at the maximum level. For [this] reason, I decided that melodies would be eliminated during the gameplay. By melody here I mean something that someone can sing or hum. (Cited in Brandon 2002)

The noise channel was nearly always employed as percussion in songs, although there were some interesting uses of it as sound effects in the music, such as radio static in *Maniac Mansion* (LucasArts, 1990), and a skipping record sound in the same game. The fifth channel (the DMC) was rarely used for music, but was instead used for sound effects in games, although there are a few examples of samples taking on the role of bass, such as in *Journey to Silius* (in which the triangle channel is used like Linn drum toms, Sunsoft, 1990), and more commonly as percussion, such as in *Contra* (Konami, 1988) and *Crystalis* (SNK, 1990). With the possibility of sampled sound, sound effects for the Nintendo system were far in advance of other 8-bit machines, and even included the occasional fuzzy vocal sample, as in *Mike Tyson's Punch Out!* Despite these advances in sound design, mixing was rarely if ever a consideration, and sound effects and music would often clash with each other aurally.

Nintendo games that were *ports* (copies) from early arcade games tended to use the same music and sound effects style, rather than to create their own songs.[21] This meant that these early Nintendo games, as in the arcades, had little in the way of song loops. *Donkey Kong* (1981 Nintendo for the arcade, 1983 for the Famicom), for instance, had short loops, only one or two bars long. By late 1984 to 1985, as arcade games and their music became more advanced, Ninten-

do's ports of these games followed suit, with longer looping gradually being incorporated into the games. Loop lengths were genre-specific, with the genres that had the longest gameplay (role-playing games and platform adventures) having the longest loops. These loops were made longer because players would spend more time on these levels than the levels of other games, as the games were designed to last for many hours. Shorter or more action-orientated genres (such as sports games or flight simulators) typically had very short loops or no music at all.

Unlike most popular music, the looping of the early game music did not typically follow a variation of a verse–chorus format. Rather, sections ranging from one to eight bars were typically found in the song-loop only once, one after the other, rarely returning to the original unless the entire loop was beginning anew. Loops were, however, often reused in other parts of a game, since system memory was always a concern. As Nintendo composer Koji Kondo states: ''I should admit that for each sound, music was composed in a manner so that a short segment of music was repeatedly used in the same gameplay. I'm afraid that the current gamers can more easily get tired to listening to the repetition of such a short piece of music. Of course, back in those days that was all we could do within the limited capacity'' (cited in MacDonald 2005).

A few looping examples can be broken down to see the looping structure in more detail. The sixteen-bar first level music of the action-adventure platformer *Castlevania* (table 2.2) had a one-bar intro (A) that repeated before moving on to the B section, which had a two-bar pattern (labeled here as B and B′) which also repeated once. The C and D sections also had two-bar patterns that repeated, followed by the repeating one-bar E section. This entire A–E song then repeated in a loop. The music for role-playing game *Ultima's* ''Ambrosia'' (table 2.3), however, had a much longer song. It began with an eight-bar A section that repeated once, followed by a four-bar B section that repeated, and a four-bar C section that was heard only once before the entire song loops.

Table 2.2
Castlevania ''Level One'' (*Akumajō Dracula*, Kinuyo Yamashita, Konami, 1987). Each block represents one bar.

Bar 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	A	B	B′	B	B′	C	C′	C	C′	D	D′	D	D′	E	E

Table 2.3
Ultima III: Exodus ''Ambrosia'' (Kenneth W. Arnold, Origin Systems, 1987).

Bars 1–8	9–16	17–20	21–24	25–28
A	A	B	B	C

These types of looping structures were fairly common in Nintendo games, though there were also games that used less conventional looping. The first-level overworld music to *Super Mario Bros.* (Nintendo, 1985), for instance, had four-bar sections with minor variations, and it alternated sections before looping the entire song. Looping was also dependent on the game state—whether a player was in battle, on an easy level, or entering a new area, for instance. Longer sections such as the *overworld* levels in adventure platform games tended to have more variation.[22] Battle music (or *boss music*) usually had much shorter loops than other songs in a game, with only one or two sections. This shorter looping in boss levels added to the tension, and since this part of the game usually lasted only a few seconds (or minutes at most), there was no need for longer loops.

At this stage in the development of game audio, the immersive power of audio was not yet understood, and music and sound effects were less responsive to action than they are today. Transitions between segments of music, for instance, would often cut abruptly without any regard for smoothness. Besides these *hard cuts*, transitions between sequences or loops were dealt with in several ways: either the loop was designed so that the last section would fit with the beginning of the loop, or a small transitional sequence was used in between looping. In *Castlevania's* Boss music (see figure 2.4), for instance, the fourth bar loops nicely back into the first. Transition segments typically used an effective glissando or rising scale of one or two bars. Another form of transition was akin to a break in a typical popular music song, in which the instrumentation would slow or pause briefly before returning to the start.

"WELL IT NEEDS SOUND": THE BIRTH OF PERSONAL COMPUTERS

Game consoles like the Nintendo were designed for gaming, although a few companies had tried to venture into business computer territory without much success (the Intellivision, for instance, "was seen as the heart of a computer system that would do family planning, stock analysis . . . ," according to designer Keith Robinson [cited in Demaria and Wilson 2002, p. 70]). Home computers like the IBM Personal Computer (PC), on the other hand, had been first designed as business machines, although games were often viewed as a way to make the machines more user-friendly. IBM introduced their PC in April of 1981. Intended largely for business use, the original PC had several advantages over its competitors in the personal computer market. IBM's upgradeability and open architecture—meaning the computer used standardized protocols and components—meant that third-party manufacturers could develop peripheral products (such as games,

soundcards, or joysticks) for the PC system, and even build their own PC systems, called clones (such as COMPAQ, who was the first to introduce such a system in 1982). Microprocessor sound was slow to develop, however, since the technology for the earliest computers had been developed primarily for business applications, which had little use for audio. The first IBM PCs and clones contained only a tiny speaker that could produce simple tones of varying pitch but of a fixed volume, designed to indicate errors or other messages, sometimes referred to as a *bipper* or a *beeper*.[23] Later, add-on soundcards would be designed as upgrades for early PCs, such as Mindscape's Music Board of 1986, a six-voice soundcard (see chapter 3).

By 1983, seeing the success of its major competitors such as Commodore and Apple (see below), IBM realized that their business could be expanded into a wider market by making the PC more accessible to the needs of the home user. With this in mind, the IBM PCjr was launched in 1984. To better compete with the other home computers being marketed at the time, several changes were made to the original PC. In addition to the usual floppy disks that had to be booted by command lines in DOS, the PCjr used cartridge ports that could be loaded by sticking the cartridge into the port and turning the computer on, which was much easier for children. The PCjr's sound enhancement over the original PC was created by the Texas Instruments SN76496, a 3+1 chip used in arcade games. The enhanced graphics and sound illustrated the importance of video games to the home computer market.

In an effort to market the PCjr, IBM hired US games industry leader Sierra On-Line to produce a PCjr game that would show off these newly enhanced colors and sound capabilities. Sierra had previously created graphic and text adventures for the PC, such as *Ulysses and the Golden Fleece* (1982), *Mystery House* (1980), and *The Dark Crystal* (1982). Sierra's answer to IBM's assignment was *King's Quest* (1984), the first ''3D'' graphic adventure game. But Sierra went one step further, creating an Adventure Game Interpreter, or AGI game engine, which would become the standard for programming Sierra's popular graphic adventure series such as *Space Quest*, *Police Quest*, and the *Leisure Suit Larry* series. The games still had a text interface in which users would type commands, but now had the additional feature of a character to be moved about on-screen in a three-dimensional space, using the keyboard, joystick, or mouse. The AGI format was designed for and modeled around the PCjr's sound chip, using all available sound channels. The PCjr versions of the songs were made up of four parts: the melody, two accompaniment parts, and one noise (generally for sound effects). When the same games were played on the original PC, only the first channel (the melody) was heard.

Unlike the original IBM, the Apple II (first released in 1977) had been designed specifically for the casual home user and developed in part out of ideas borrowed from video games, as creator Steve Wozniak elaborates:

> A lot of features of the Apple II went in because I had designed *Breakout* for Atari. I had designed it in hardware. I wanted to write it in software now. So that was the reason that color was added in first—so that games could be programmed. I sat down one night and tried to put it into BASIC ... I got this ball bouncing around, and I said, ''Well it needs sound,'' and I had to add a speaker to the Apple II. It wasn't planned, it was just accidental ... So a lot of these features that really made the Apple II stand out in its day came from a game, and the fun features that were built in were only to do one pet project, which was to program a BASIC version of *Breakout* and show it off at the club. (Cited in Connick 1986, p. 24)

The original Apple II series had similar sound capabilities to those of the early PCs—a one channel bipper for warnings and errors. As with the PC, later versions of the Apple II improved sound capabilities, and third-party sound expansion boards were made available. Many, such as Sweet Micro Systems' Mockingboard, used the arcade chip AY-3-8912.[24] Some even used several of these chips, such as Applied Engineering's Phasor, which used four AY-3-8913 chips, creating a total of twelve tonal voices and four noise generators, in addition to a speech synthesis card. Such add-ons made programming music for games difficult, since several different versions of the code had to be included to accommodate each soundcard (see chapter 3).

In direct competition with Apple was Commodore, who had experienced great success with their VIC-20 computer, and who had topped all sales records with the Commodore 64 (C64). Like the Apple, the C64 was originally conceived as a game computer, and the advanced (for the time) graphics and sound were clear evidence of this. Appealing more to gamers, Commodore marketed the computers in department stores and toy stores, which were formerly the domain of home consoles, rather than PCs. In addition, the computer could be hooked up to a television set and played like a games machine. More directly targeting competing home game consoles, in 1983 Commodore offered a $100 rebate on the $400 machine to any customers who would send in their old game console (''Commodore Aims at Video Games,'' *New York Times*, April 12, 1983). It is interesting to note, then, that the Commodore 64 had been built around the concept of games, with games as a driving force in the technology, particularly in the area of sound.

The C64 sound chip (called the Sound Interface Device or SID) was a 3+1 chip, created by Robert Yannes in 1981, who had helped to engineer Commodore's earlier VIC-20, and who would later go on to create the sound chip for the Apple IIGS. Each tone on the chip could be selected from a range of waveforms—sawtooth, triangle, variable pulse, and noise. The noise channel could also operate as a simple pulse width modulation sampler. An independent ADSR envelope generator for each channel enabled the SID more accurately to imitate traditional instruments than previous chips, and each channel had a frequency range comparable to that of a piano. Each tone could also be subjected to a variety of effects

and programmable filters including ring modulation.[25] One of the biggest problems for game composers was the SID's filters (low pass, high pass, band pass, and notch), which would act differently on different versions of the C64 machine.[26] Attempts to overcome the problem varied. One game, *Beach-Head 2* (1985), allowed the user to select the filter settings for the sounds, to prevent the screeching that would occasionally be heard with the wrong filter setting. Composer Ben Dalglish explained, ''I tended to use 'static' filters as little as possible for exactly that reason—generally, I'd use filter sweeps, which were pretty much guaranteed to have the same effect irrespective of the start/end frequencies'' (cited in Pouladi 2004). In other words, the computer's chips often drove the resulting sound.

The limitations of memory were another major problem for game developers. The games were distributed on three kinds of media: 5.25-inch floppy, data cassette tape, and cartridge. The data cassettes—the most popular storage medium for games—had built-in audio converters to convert the computer's digital information into analog sound, though the loading of games on cassette was slower than it was on floppy. The floppy disks, on the other hand, offered faster loading but less storage, providing a total storage capacity of 170 kB, of which music was usually limited to between 5 and 10 kB. The most common approach to the limitations of memory was to reuse sequences of music, as described by composer Dave Warhol: ''I would talk with the game designer to figure out what type of music they were after. Then there were the architectural limitations, how much memory I had been allocated, and stuff like that. I often had to figure the structure of the composition (A:A:B:A) and only include copies of the portion that were unique, or have 'music subroutines' of a bar here and there'' (cited in Carr 2002a).

As on other early machines, looping was commonly used when space was limited, as composer Martin Galway elaborates: ''The [song conversion] was a nightmare since it's the tune right from the beginning of the movie [*Short Circuit*] with all the robotic short notes and arpeggios. The tune just built up so massive [*sic*] that the poor C64 was short of notes by about 30 seconds into it, so I had to fudge the end a bit and make it repeat, basically.''[27] One composer, Rob Hubbard, partially overcame this limitation by arranging the music into a series of subroutines or *modules*.[28] Each module could contain title music, in-game music, and game-over music using the same source code to share instrument tables (that is, each different timbre used in the game was set out in the code in advance, and just called upon when needed). Each song typically had three tracks (one for each channel), and each track consisted of a list of patterns (sequences) in the order in which they were to be played. The code would refer to specific sections of the module to be called when necessary, reducing the need to repeat any coding that would take up valuable space.[29] This modular approach to song construction

would go on to influence the MOD format on the Commodore Amiga (see chapter 3).

Some Commodore composers were quite adventurous with their music coding, and included random number generators into the code that could select from a group of sequence options. For instance, *Times of Lore* (Martin Galway, Microprose, 1988) used a selection of guitar solos that were repeated randomly for the eleven-minute duration of the song. In this way, the game's ten songs (over thirty minutes of music) could fit into just 923 bytes. A similar random generation was used in *Rock Star Ate My Hamster* (The Codemasters Software Company Limited, 1988), a rock management game. The music for the band's practice session was chosen from a random combination of sixteen sequences. Even more advanced randomization was developed at LucasFilm Games, in a soccer game called *Ballblazer* (1984), which incorporated fractal-based algorithmic music during gameplay. Programmer-musician Peter Langston designed a series of jazz sequences and a standard bass line. The lead was improvised on (algorithmically varied) by the computer. Another interesting approach was seen also in *Lazy Jones* (Terminal Software, 1984), which had twenty-one songs, one of which was selected when the character entered or left one of the ''room'' levels of gameplay. There were eighteen rooms in total, and each room had its own four-bar ''song,'' which actually played like a segment of one greater song (the title music). Even if the character left the room at, say, bar twenty-one, the rest of the loop would play before it would transition back into the theme song. Most of the loops worked well together, in part because of the ragtime-like bass lines, the same timbres used, and the fact that the game used only two channels. These ideas are discussed further in chapter 8.

Cover songs were very popular on the Commodore 64, partly because its advanced sound chip enabled more recognizable renditions than other systems of the time, although a few composers for the C64 favored original tunes, such as Ben Dalglish: ''Covers, on the whole [were more difficult], simply because I was a perfectionist when it came to things like getting fast guitar solos right note-for-note ... and also because of the arrangement challenge of fitting a 'real world' piece of music with drums and bass and strings and everything into three voices'' (cited in Pouladi 2004). In other words, in many early cases of cover songs in games, the original music had to be significantly altered in order to suit the limitations of the technology. Classical music was common in Commodore 64 games, such as Chopin's *Funeral March* (*Sonata no. 2 in B-flat minor*) in *Zak McKracken* (Chris Grigg, LucasArts, 1988), or Holst's *Mars, Bringer of War* and Bach's *Prelude no. 2 in C Minor* in *Wicked* (Electric Dreams, 1989). More interesting, perhaps, was the mingling of popular and classical songs in games like *Frantic Freddie* (Commercial Data Systems, 1983), which used the Sylvers' ''Boogie Fever,'' several Scott Joplin songs, Paul Simon's ''Kodachrome,'' Beethoven's *Fifth Symphony*, ''Crazy Little Thing Called Love'' (Queen), and ELO's ''Don't Bring Me

Down.'' Traditional American folksongs of the nineteenth century were also very common in games, including for instance *Buffalo Bill's Wild West Show*, which used the ''Star Spangled Banner'' (John Stafford Smith), ''Oh Susanna'' (Stephen Foster), ''Yankee Doodle'' (Richard Schuckburgh), and ''Camptown Races'' (Stephen Foster), among others. As well, many title themes were based on film music, such as *International Karate*'s (System 3, 1986) use of *Merry Christmas Mr. Lawrence* (Ryuichi Sakamoto, 1983), and *Super Pipeline*'s (Taskset, 1983) use of Miklós Rózsa and Walter Schumann's *Dragnet* theme (1967). Cover tunes seemed largely the result of the whims of the games' producers, and there was little or no concern for copyright infringements. As Martin Galway tells the story of one selection: ''I was still freelancing when I worked on that music first in 1984, and I just said to [programmer] Tony Pomfret, 'what do you want for the music?' to which he replied 'I want the B-side from the Limahl single ''Neverending Story'' I bought the other day.'''[30] Composer Mark Cooksey elaborates, ''I think I would have preferred to compose my own music most of the time, however the programmers reigned supreme when it came to many of the decisions especially where it concerned the lowly musician. A lot of the games were arcade conversions or licensed titles and hence the music was copied from the original source, when an original product was produced the programmer would often want some of their favourite music in the game (such as J. M. Jarre)'' (cited in Carr 2002b).

It was rare to see the original composers credited in these games, however, and there was little credit given or information on licensing at the time, although there were some exceptions, such as Devo's ''Some Things Never Change,'' used in *Neuromancer* (Electronic Arts, 1988) and *California Games'* use of the Kingsmen's ''Louie Louie'' (Chris Grigg, Epyx, 1987), both of which include a credit in the manuals.[31] As is discussed further in chapter 6, the music industry had not yet discovered the potentials of the medium as a form of promotion and sales, and there were apparently no lawsuits from these early covers.

By the late 1980s and early 1990s, facing more competition from home consoles, there were several Commodore games that clearly tried to copy the Nintendo aesthetic, such as *Mayhem in Monsterland* (Apex Computer Productions, 1993), or *The Great Giana Sisters* (Time Warp, 1987), which was so similar to *Super Mario Bros.* that Nintendo successfully sued to have it pulled from stores. Particularly notable is the fact that these games adopted a distinctly NES-style music and sound design. Each contained similar background loops for overworlds, underworlds, and boss music, in Nintendo style, with longer overworld music, and short boss music loops. *Mayhem in Monsterland*, for instance, followed Nintendo's looping style with a short four-bar boss music pattern. Sound effects were more ''cheery'' and ''poppy'' than Commodore's earlier games. This approach to game audio suggests that what had developed even in the 8-bit era was as much a result of aesthetic as it was technology.

CONCLUSION

The audio of the 8-bit era games represents an interesting tension between game sound aesthetics and the series of pressures and constraints exerted by technology, by industry, by genre, and by the very nature of games themselves. Each machine had a slightly different aesthetic that grew, in part, from the technology that was available. Explains composer Rob Hubbard of the sound of the C64, ''Well, you know, part of that [sound aesthetic] is dictated by the fact that you have such limited resources. The way that you have to write, in order to create rich textures, you have to write a lot of rhythmic kinds of stuff ... it's easier to try to make it sound a lot fuller and like you're doing a lot more if you use much shorter, more rhythmic sounds.''[32]

The persistent practice of looping is particularly illustrative of the tensions between technology and aesthetic. As was discussed, various approaches to looping were certainly available—and, at times, used—although straight short looping remained the most prominent response to limited memory. Looping was not the only choice, however, and other approaches such as random sequencing were also seen. Looping, then, appears to be as much an aesthetic choice as a predetermined factor led by technology.

The parallels between responses to these constraints of technology in games with those of early film sound are remarkable. Looping, for example, was also historically viewed as a way to get around certain problems in film sound. James Lastra (2000, p. 210), for instance, discusses W. A. Mueller's use of background loops to create smooth ambience and reduce the cost of production in film as early as 1935. Although the constraints in this case were financial rather than technical, it is interesting that both film and games turned to looping quite early on, and would later abandon the idea. Likewise, early transitioning between musical segments to accompany silent film often occurred in *hard cuts* (abruptly cutting between musical segments). Lastra examines Carl Dreher's criticism of early film audiences, who were not yet ''sophisticated enough to complain about 'synchronized pictures in which, as the scenes [shots] change, one musical selection is abruptly broken off and another starts will [*sic*] full volume in the middle of a bar''' (Lastra 2000, p. 176). As Roy Prendergast indicates, in the era of silent film, when music was commonly played from composition cue sheets, ''Sometimes single sections were played side by side with no transitions at all, although short modulations were written when the contrast of styles appeared too crude'' (Prendergast 1977, p. 11).[33] Later, audiences would expect smoother transitions and a closer relationship between picture and image, but in the early stages of both film and games, music was more of an accompaniment than a true integration with the image.

The limitations imposed on early game composers and sound designers were not just limited to technology, however. The nature of gameplay and a

game's genre also led to various constraints, as in the game *Frogger*, which had a specific time constraint in each level to which the music had to conform. It may also be the case that the prominence toward science-fiction or fantasy-oriented games (which do not depend on realistic sound effects or graphics, and which had not been common in mechanical games) was partially determined by the constrained nature of the technology. With no equivalent to realistic or naturalistic representation within our scope of experience, disbelief may be suspended regardless of the low-fidelity sound effects. In fact, despite the suggestion of flyers that professed a degree of realism, there were very few attempts during this time at creating a truly realistic sound design aesthetic. Rather, graphics developed toward cartoon-like figures, and audio responded with futuristic, alien, and obviously synthesized sounds. After all, in such cases ''there is no need to conform to any performance standards. Is someone going to criticize your rendition of 'A Lawnmower Travelling through the center of the earth'?'' (Righter and Mercuri 1985, p. 236).

The social constraints of specialized knowledge required to program the software—and to develop audio and write music—also influenced the developing aesthetic, with many composers lacking formal musical training, which is not to say lacking in talent. Even when musicians were involved, they were generally separated completely from the integration of the music; as Linda Law, manager of game composer George ''The Fat Man'' Sanger described, ''Game music in those days was usually delivered as note lists or on manuscript paper. This resulted in the composer being completely at the mercy of the programmer (a scary notion). But the upside for the composer was that the skills he needed to create this music were solely conventional composing skills and the ability to work within very specific parameters. A composer needed no special software or recording gear. The boatload of labor involved in getting the sound into the game was all done by the programmer'' (Law 2003).

The impact of this division between the specialized knowledge of music and programming could well be one of the reasons why sound effects were so prominent in these early games, and why some of the audio in the 8-bit era was unconventional in many ways, since many companies did not employ sound or music people but, rather, relied on programmers. Some games appeared to rely on merely copying music from piano sheet music.[34] Rather than their being detrimental to game audio's development, however, I would suggest that one of the reasons for some of the innovations in approaches to composition, such as the use of random generation and algorithmic composition, was the very fact that many of the game composers were often programmers. Many early games were produced by a single author who programmed gameplay, drew graphics, and designed the sound effects and music, and they were rarely specialists in any of these areas. Later, other innovative approaches, such as in the sound and music to *Metroid*, were in part driven by the industry and a desire to be distinguished

from the growing codification of stylistic traits. The resulting approaches to composition, I would argue, directly affected the development of music not only in games, but electronic music in general, as Rob Hubbard's module format, taken up in the following chapter, was certainly a likely influence on later sequencing software.

As shown, games were also a driving force in the development of microprocessor-based sound technology. Advanced sound chips were developed for computers specifically for the purpose of games. This technology, of course, would spill over into the music instrument industry. As Bob Yannes, creator of the Commodore 64 SID chip, who would later leave to form Ensoniq, a music keyboard manufacturer, elaborates, ''Actually, I was an electronic music hobbyist before I started working for MOS Technology (one of Commodore's chip divisions at the time) and before I knew anything at all about VLSI chip design. One of the reasons I was hired was [that] my knowledge of music synthesis was deemed valuable for future MOS/Commodore products. When I designed the SID chip, I was attempting to create a single-chip synthesizer voice which hopefully would find it's [*sic*] way into polyphonic/polytimbral synthesizers'' (cited in Varga 1996).

Game audio was, then, to some extent, instrumental in developing techniques and technologies that would carry over into the popular music and audio production sphere. This important historical role of games has been remarkably absent from histories of musical instrument technology and popular music studies.[35] Indeed, the relationship between games and some other industries has always been quite closely knit. Not only did games cross over with the musical instrument industry, but, as shown, elements of games were marketed alongside popular music and film. These relationships and their impacts are taken up in chapter 6.

Many of the developments of the 8-bit era would help to define a video game audio aesthetic that is still present today. Certainly, games were not the first to see some of the more advanced ideas introduced in this chapter: random sequencing goes back to early musical dice games such as that of Mozart's *Musikalisches Würfelspiel*, and had more recently been used by the aleatoric and mobile form compositions of Stockhausen, Boulez, and others—an idea that is taken up in chapter 8. Similarly, repetition and the use of loops had been seen previously in film, had been a staple of the 1960s minimalist movement, and had coincided with the rise of disco and drum machine–based popular music. Thus, although few of the important concepts introduced in the 8-bit era were necessarily new, they did help to introduce many of these ideas (consciously or not) to a mass audience and laid the groundwork for the game audio of the 16-bit era.