twentieth-century music 4/2, 209–227 © 2008 Cambridge University Press
doi:10.1017/S1478572208000510 Printed in the United Kingdom

In the Loop: Creativity and Constraint in 8-bit Video Game Audio

KAREN COLLINS

Abstract

This article explores the sound capabilities of video game consoles of the 8-bit era (*c.*1975–85) in order to discuss the impact that technological constraints had on shaping aesthetic decisions in the composition of music for the early generation of games. Comparing examples from the Commodore 64 (C64), the Nintendo Entertainment System (NES), the Atari VCS, and the arcade consoles, I examine various approaches and responses (in particular the use of looping) to similar technological problems, and illustrate how these responses are as much a decision made by the composer as a matter of technical necessity.

> Another example would be when you're faced with a guitar that only has five strings. You don't say, 'Oh God, I can't play anything on this.' You say, 'I'll play something that only uses five strings, and I'll make a strength of that. That will become part of the skeleton of the composition.
>
> Brian Eno[1]

Introduction

Technological constraints are not new to musical composition, although most explorations of the subject have focused on twentieth-century concerns. Mark Katz, for instance, discusses how the 78-rpm record led to a standard time limit for pop songs, and how Stravinsky famously tailor-made *Sérénade en LA* to the length of an LP, even though the composer, in doing this, may have been driven by 'his penchant for self-imposed limitations'.[2] Critiques of 'hard' technological determinism as it relates to musical technologies have dominated the literature.[3] In its place has arisen a softer approach, notes Paul Théberge, in which 'traditional instrument technologies can sometimes be little more than a field of possibility within which the innovative musician chooses to operate. The particular ''sound'' produced in such instances is as intimately tied to personal style and technique as it is to the characteristics of the instrument's sound-producing mechanism.'[4] However, video games audio has been strangely absent from nearly all accounts of popular music technology. By examining video games of the 8-bit[5] era (approximately 1975 to 1985), we can see how technological constraint has shaped aesthetic decisions, some of which have impacted upon other gaming

1 In Aikin, 'Brian Eno'.
2 Katz, *Capturing Sound*, 3–5; cf. McLuhan, *Understanding Media*, 283.
3 See, for example, Taylor, *Strange Sounds*, 27; Théberge, *Any Sound You Can Imagine*, 160; and Katz, *Capturing Sound*.
4 Théberge, *Any Sound You Can Imagine*, 187.
5 'Bit' refers to the smallest unit of digital data. An 8-bit machine could simultaneously process eight binary digits (bits) of data.

platforms and even other genres of music. As a number of recent studies of music technology have argued, I maintain that the relationship between technology and aesthetics is one of symbiosis rather than dominance – what Barry Salt refers to as a 'loose pressure on what is done, rather than a rigid constraint'.[6] I shall show how, while some choices may be predetermined by technology, creative composers have invented ways in which to overcome or even to aestheticize those limitations.

The influence of early games audio has been underestimated. The continuing prominence of one particular sound, *Pac-Man's* 'waca waca' (the sound the character makes when eating), is clear evidence of the significance of this influence. In the 1980s the sound was incorporated into popular songs such as 'Weird Al' Yankovic's 'Pacman' and Buckner and Garcia's 'Pac-Man Fever', but it has more recently been used by Aphex Twin, Bloodhound Gang, DMX, Lil' Flip, and many more.[7] There has also been an increase in the popularity of 'micromusic', sometimes known as 'chiptunes' – songs based on 8-bit games machine sound. Malcolm McLaren has begun promoting the scene, and Beck released an EP of 8-bit remixes of some of his songs in 2005.[8] A small but dedicated scene of micromusicians has developed, making music from otherwise obsolete computers and games consoles, with bands like 8bitpeoples, Teamtendo, and the Reverse Engineers. Some might suggest that this is Generation-X retro-gaming nostalgia, as Reverse Engineers' Edward Jones seems to imply: 'When I was a kid, I didn't listen to pop music on the radio. I listened to music on computer games. My friends and I, we would make tapes of the songs from computer games and we would listen to those. They were the tapes we'd copy and pass around.'[9] But many of the composers in the scene are too young to have spent childhoods with the machines. Rather, they like the challenge of the limitations ingrained in the technology, as Teamtendo intimates: 'Working with this limited harmonic vocabulary forces you to be creative, and there are some very pleasant discoveries along the way';[10] or, as Goto80 says, 'it's fun working with such hardcore limits, forcing you to realize your ideas in other ways'.[11] When the constraints are looked upon with the attitude of these artists, or, for instance, with that of Brian Eno in the quotation at the beginning of this article, the results become part of a new paradigm for creating music.

The concern of this paper, however, is original games music rather than contemporary micromusic, since numerous micromusicians have incorporated more modern technology into their work. Many 8-bit machines are worth studying, but I shall limit my discussion to the most successful consoles – the Atari VCS and the Nintendo Entertainment System (NES) – and the first major home-computer game system – the Commodore 64 (C64) – as well as early coin-operated arcade machines. Many of the machines that I have excluded (such as the Intellivision, ColecoVision, the ZX Spectrum, and the BBC Micro) used the same

6 Salt, 'The Evolution of Sound Technology', 37.
7 For details of recordings see Discography below.
8 *Hell Yes Remix*; see Discography.
9 Jones, 'Thingo of the Week: Micromusic', *Mondo Thingo* (2005), <http://www.abc.net.au/thingo/txt/s1083340.htm> (accessed 10 November 2007).
10 Katigbak, 'Game on'.
11 Carr, 'An Interview with Anders Carlsson AKA GOTO80'.

or similar sound chips to those discussed here.[12] I focus on two groups of games. The first consists of games that crossed platforms and spanned genres: *Arkanoid*, *California Games*, *Donkey Kong*, *Frogger*, *Ghosts 'n Goblins*, *Karate Champ*, *Maniac Mansion*, *Pac-Man*, *Tetris*, and *Ultima 3: Exodus*.[13] For the systems that did not have a port[14] of these games, comparable games were chosen. The second group comprises games designed specifically for each system, and so includes the best-known games from each, chosen on fan and industry ratings.[15] In several cases, each of the games studied also had a particular influence on the development of games in general. For instance, *Metroid* was one of the first examples of non-linear gaming on a home console; and *Super Mario Bros.* was one of the most successful franchises in games history, selling over 180 million games and making popular many features that are now standard.

Unlike film audio, which is strictly linear and commonly operates as 'diegetic' or 'non-diegetic',[16] games audio exhibits varying degrees of gameplay response, from the fairly simplistic song that remains constant throughout a game level but changes on completion of that level, to more complex relations of sounds and music that continue to respond to a player throughout a game. It is worth distinguishing between these types of interactivity. *Interactive* audio refers to sound events that occur in reaction to gameplay, responding to the player directly. For instance, a player presses a button, and the character on the screen jumps and makes a 'bleep' noise; pressing the button again will cause a recurrence of this sound. The 'bleep' is an interactive sound effect. *Adaptive* audio, on the other hand, responds to the gameplay environment. For example, the music might change as a timer in a game increases to a certain point. I use the term *dynamic* audio to encompass both of these types of interactivity – audio that reacts to changes in the gameplay environment or to the activity of a user.[17] In 8-bit audio there are several distinct degrees of dynamic sound:[18]

1 *Adaptive non-diegetic sounds.* These occur in reaction to gameplay but are unaffected by users once the game has begun, until a major new event triggers a new linear track, such as

12 The AY series of chips, for instance, was used in the Sinclair ZX Spectrum, Amstrad CPC, BBC Micro, Atari ST, Sega Master System, and many arcade machines.

13 For further details see Games list below. For a discussion of the video game genre see Wolf, 'Genre and the Video Game'.

14 That is, no copy for the new system was released.

15 'IGN's Top 100 Games', <http://top100.ign.com/2005/index.html>; 'Game Music Poll Results', <http://top100.ign.com/2005/>; 'Best Games on the NES', <http://www.amazon.com/gp/richpub/listmania/fullview/3CQFUX11P973Y/103-3964422-8647012?ie=UTF8&%2AVersion%2A=1&%2Aentries%2A=0>; 'Top 20 NES Games' <http://www.nintendoland.com/home2.htm?charts/nes.htm> (all accessed 10 November 2007).

16 Though, of course, not without disagreement or further subdivisions (see, for example, Gorbman, *Unheard Melodies*).

17 For a more detailed discussion of interactivity and dynamic audio see Collins, 'An Introduction to the Participatory and Non-Linear Aspects of Video Games Audio'.

18 These are my own distinctions and are not commonly used by games composers, or in early programming languages such as BASIC, which typically distinguished only between 'foreground' and 'background' sound. There are also a few cases of what I have termed 'kinetically gestural interactive sound' in 8-bit games, although this is extremely rare and so is left out of the discussion here. This refers to sound or music that occurs when the player, along with the character, physically interacts with the sound, such as in *Duck Hunt*, in which the player fires a gun at ducks, making shooting noises, or in *Dance Aerobics*, which required a 'power pad' on which the player would stand, interacting with the music.

when a timer triggers new music. For example, in *Frogger* a song plays until the player manoeuvres his/her character up to a 'home base', or the time runs out, at which point a new track is triggered.

2 *Non-dynamic diegetic sound*. This occurs when a sound event takes place in a character's space but the character does not interact with it. An example is the thunder and rain sounds in the background of *Ghosts 'n Goblins*.
3 *Interactive diegetic sound*. This, on the other hand, refers to sound in a character's space with which the player's character interacts. For example, in *Maniac Mansion* a character must record music on a cassette recorder and play it back during the game.
4 *Non-dynamic non-diegetic audio*. This refers to sound that is part of an underscore that is unaffected by a player's movements, such as the cut-scene[19] after the second level of *Pac-Man*, in which gameplay is stopped so a quick 'film' can play.
5 *Interactive non-diegetic sound*. This refers to sound events occurring in gameplay that can react or adapt to the user, such as jump sounds, or music events warning of approaching danger, which can be switched off by moving in a new direction. Examples of these would be *Super Mario*'s jumping sound, or the 'boss' tune in *Mayhem in Monsterland*, which plays when a character is in the vicinity of a boss character; if the player backs up, the music returns to the 'normal'-level gameplay music.

Levels 4 and 5 are the most common of these categories of musical interaction, and are the focus of this paper.

The concept of 'the loop' presents another issue. Here, I take loops to mean 'self-contained units that may or may not be combined with other loops or non-looping material in a larger structure [. . .] In music production and sound design circles, a "loop" is a bit of audio – usually, though not exclusively, of short length – that can be played back repeatedly and potentially endlessly without noticeable gaps or disruptions between one instance of the loop and the rest'.[20] Loops, however, occur in different shapes and sizes, as we shall see below. At this point it is perhaps useful to engage with Middleton's concept of 'musematic repetition' and 'discursive repetition' – the 'riff' versus the 'phrase' in the discourse about loops.[21] We see both forms of repetition occurring in games audio, often at the same time, along with a longer loop of the entire 'song' or a sequence of loops at a larger-than-phrase level. For the sake of brevity I have termed these 'microloops', 'mesoloops', and 'macroloops'. Thus a two-note bass line may provide a (musematic) microloop which repeats twice in a two-bar (discursive) mesoloop, which is part of a longer, eight-bar macroloop 'song', which is looped throughout a level of gameplay.

The Technology of 8-bit Audio

The majority of 8-bit machines (and early arcade and pinball machines) used what is known as programmable sound generators (PSGs), which were subtractive synthesis chips that

19 A cut-scene is a movie that plays in a game, in which the player does not control any actions.
20 Stillar, 'Loops as Genre Resources', 199.
21 See Middleton, ' "Over and Over" '.

offered little control over timbre and were usually restricted to square waveforms, with limited possibility for manipulation. Many of these PSGs were created by Texas Instruments or General Instruments, but some companies, such as Atari and Commodore, designed their own sound chips to improve sound quality.

Many coin-operated arcade games of the 8-bit era used the General Instruments AY-8910 chip, which contained three tone-generators and a white-noise generator. Each channel allowed for individual control of frequency, volume, and envelope. Pitch was controlled by the somewhat limited frequency-division method, but the AY chip used a larger bit register to set the divisor, allowing for 4096 possible tones instead of the 1024 pitches on the other popular arcade chip, the Texas Instruments SN76489.[22] Another common chip, designed by Atari, was the Pokey. Each channel had only a square-waveform option, and had a three-octave frequency register and control register (setting the distortion and its volume). One of the more advanced chips, the Curtis CEM3394, was introduced in 1984, but was used in only a handful of games. It had a single voice per chip, but that voice had selectable pulse, triangle, and sawtooth waveforms, rather than just the square waves of other PSG chips. Most games that used the chip had several of them on board – up to six, as in *Chicken Shift*, for example.[23]

Early games also used sound samples created with digital-to-analog converters (DACs) or pulse code modulation (PCM).[24] The downside to PCM sampling was the amount of space required to store the samples: as a result, most PCM samples were limited to those sounds with a short envelope, such as percussion and sound effects, which were suitable, as mentioned, for arcades requiring lots of explosive sounds to attract players, but unsuitable at the time for musical purposes. This could be one reason why we see advanced sound effects at a time of limited musical capabilities.[25] Speech chips, which could be used for short vocal samples or for sound effects, also became more prominent in the mid-1980s.[26] With separate chips to handle sound effects and voice, the primary sound-chip's noise channel could be

22 For a basic introduction to waveforms and bit depths see Adobe's 'A Digital Audio Primer' at <http://www.adobe.com/products/audition/pdfs/audaudioprimer.pdf> (accessed 10 November 2007).

23 The technical specs for sound chips and consoles have been published in many places on the Internet. Many of these specs were originally released to programmers or in manuals so that users could program the chip, and they have since been published by fans, chiptunes creators, or those designing emulators – software versions of the consoles or computers for today's PC users. The MAME arcade emulator allows users to see the technical data of what chips were originally used for each game, and it gives accurate information on the emulation.

24 DACs typically used 6- or 8-bit digital values, with up to 256 different voltage levels. The programmer chose a sequence of numbers between 1 and 256 to determine the speed (and therefore the resulting frequency) of the sound. With PCM, essentially, an analog sound is converted into digital sound by taking many small samples of an analog waveform. The data is stored in binary, which is then decoded and played back. The fidelity of the sound depends on the sample rate or quantization – the number of bits representing the amplitude. The method is still used, for instance for DVDs or CDs, where the sample rate is 44 kHz, or 16-bit, but most early games could sample at a maximum of 22 kHz only.

25 One solution to the size issue was what is known as adaptive difference PCM (ADPCM). With the ADPCM method, the difference between two adjacent sample values is quantified, reducing or increasing the pitch slightly so as to decrease the amount of data required; this is somewhat similar to the way in which JPEG compresses a visual image. The fidelity of the sound, however, is reduced as compression rates are increased.

26 A variety of different types of speech chips were used, including ADPCM, PCM, and linear predictive coding (LPC). In several games Atari included a Texas Instruments TMS5220 (LPC) chip, which had been used in 'Speak 'n' Spell', the popular family electronic game.

freed up, allowing for the use of more complex percussion in the music, such as in *Discs of Tron*.

The sound chip in the Atari VCS, which also controlled graphics, was manufactured specifically by Atari and was known as the TIA chip. The audio portion had just two channels, so that music and sound effects could only be heard on two simultaneous voices mixed into a mono output. Each channel had a 4-bit waveform-control selector, but, of the sixteen possible settings, several were the same as or similar to others. Typically, the voice options were two square waves (one high, one bass), one sine wave, one saw wave, and several noise-based sound options useful for effects or percussion. The trouble with the tonal sounds, however, was that each had a different tuning (although two of the square-wave sounds were almost the same tuning), so the pitch value of the bass and the lead voice would often be different. Tuning sets could be quite variable, then, 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, for most tonal compositional purposes, quite useless. To compound the problem, there were variations between the sound on the NTSC and PAL versions of the machine.[27] At times, pitches were awry by as much as fifty cents (half a semitone) between the European and American models.

The Nintendo Entertainment System's (NES) sound chip, created by composer Yukio Kaneoka and more advanced than that of the Atari, used a built-in five-channel PSG delivering two pulse waves, a triangle wave, a noise channel, and a sample channel. The pulse channels had an 11-bit frequency control, capable of about eight octaves, and four duty-cycle options (altering the harmonics to create a sound that could be quite smooth, but also, with adjustments, 'fat', or thin and 'raspy'). In addition, the channels had a 4-bit amplitude-envelope function, and one of the pulse-wave channels had a frequency-sweep function that could create portamento-like effects. The triangle-wave channel was one octave lower than that of the pulse waves, had only a 4-bit frequency control, and had no volume or envelope control. The fifth channel was a sampler, also known as the Delta Modulation Channel (DMC). There were two basic methods of sampling using this channel: PCM, used for speech, such as in *Mike Tyson's Punchout*; and direct memory access (DMA), which was only 1-bit and was most frequently used for percussion and sound effects.

While Nintendo represented the peak of console audio in the 8-bit era, the Commodore 64 (C64) was far in advance of other home PCs, having been originally conceived as a games computer, with sophisticated (for the time) graphics and sound designed to entice consumers frightened off by the more business-like IBMs. The sound chip (called the Sound Interface Device, or SID) was a three-plus-one generator chip, created by Bob Yannes in

27 Eckhard Stolberg explains the difference as follows: 'The Atari 2600 VCS produces it's [*sic*] sound with some shift registers of various lengths. The speed at which it shifts the registers can be set to either the pixelclock/114 or to the CPUclock/114, which means two shifts per scanline or a third of this. Since a NTSC VCS produces 262 scanlines per frame and 60 frames per second this results in output rates of 31440 Hz and 10480 Hz. PAL TVs produce only 312 scanlines and 50 frames, so the output rates are only 31200 Hz and 10400 Hz' (Stolberg, 'Atari 2600 VCS Sound Frequency and Waveform Guide').

1981.[28] Each tone on the chip could be selected from a range of waveforms – sawtooth, triangle, variable pulse (square wave), and noise. An independent ADSR envelope generator for each channel enabled the SID to imitate traditional instruments more accurately than did previous chips.[29] Each tone could also be subjected to a variety of effects and programmable filters, including ring modulation, commonly for sound effects like bells, chimes, and gongs. The noise channel of the C64 could also operate as a simple pulse-width modulation (PWM) sampler. PWM was used for sampling and to simulate a low-frequency oscillator (LFO) and could simulate a tremolo effect. The ability to sample sounds led to the inclusion of more realistic sound effects in many C64 games tunes. *Turbo Outrun*, for instance, included a 'scratch' sound and voice samples, as well as samples of pitched percussion instruments.

Early sound programmers and musicians needed to understand assembly language to engage the chips, which meant that most of the early composers for games were, in fact, programmers working on other aspects of a game. It is unsurprising, then, that many early games tended to employ pre-existing popular or classical music rather than specially composed music.[30] The fact that the audio programmers were often not musicians could also explain why, even after some of the earliest constraints were lifted, music was thrust to the side in favour of more advanced sound effects. It was, therefore, a combination of the technological constraints of the time and the social constraints surrounding the specialized knowledge required to engage the chips that helped to create the unique aesthetic of early games audio.

The clearest evidence of these constraints was the lack of music in many of the earliest games. Typically, arcade games of the late 1970s had short title and game-over music. Otherwise the music would usually play only when there was no game action, and would be composed with simple DAC samples, as *Circus* and *Rip Cord* illustrate. Gameplay music was introduced as early as 1978, when continuous sound began to be added, as in *Space Invaders* and *Asteroids*. Arguably, these games represent the first examples of dynamic non-diegetic music in a game: the music of *Asteroids* was limited to an accelerating two-note melody, and *Space Invaders*' descending, looping four-tone marching of alien feet sped up as the game progressed.

By 1980 arcade manufacturers were including PSG chips in their machines more often, and more melodic music began to take off. The earliest examples of musical loops in games date from this time: for instance, *Rally X*, which had a six-bar loop, and *Carnival*, which used Juventino Rosas's 'Over the Waves' waltz (1889). Most arcade games now had co-processors specifically to control sound, allowing for the simultaneous playing of action and music, although the majority still had no background music. By 1982 it became increasingly common to use more than one sound chip in a coin-operated game. In this way the music could continue to play without being interrupted by the sound effects having to use the same

28 Yannes had helped engineer Commodore's VIC-20, and would later go on to create the DOC chip for the 16-bit Apple IIGS and to found Ensoniq keyboards.

29 The tone oscillators had a range of between 0 and 3995 Hz, approximately the same range as a piano. There were two 8-bit registers for each channel, controlling frequency (meaning 16 bits in total, or 65,536 frequency possibilities for each voice, so that composers could 'detune' notes if they wished).

30 See Collins, 'Loops and Bloops'.

chip. But the additional sound chips tended to be used for more advanced sound effects rather than for an increase in the textural complexity of the music, for reasons to do with the arcade environment (as mentioned previously, subtleties of polyphony are unlikely to be captured in such an atmosphere), or with the fact that most early programmers were not musicians and were therefore more comfortable with sound effects. The trouble with using multiple chips became evident in some of these games: in *Chicken Shift*, for example, which used diegetic as well as non-diegetic music, the various sound sources often conflicted with each other, as when one song continued under lose/win cues and sound effects, resulting in a hilariously chaotic cacophony of noise. Generally speaking, the inclusion of background music, however simple, was not well established until long after the capabilities existed.

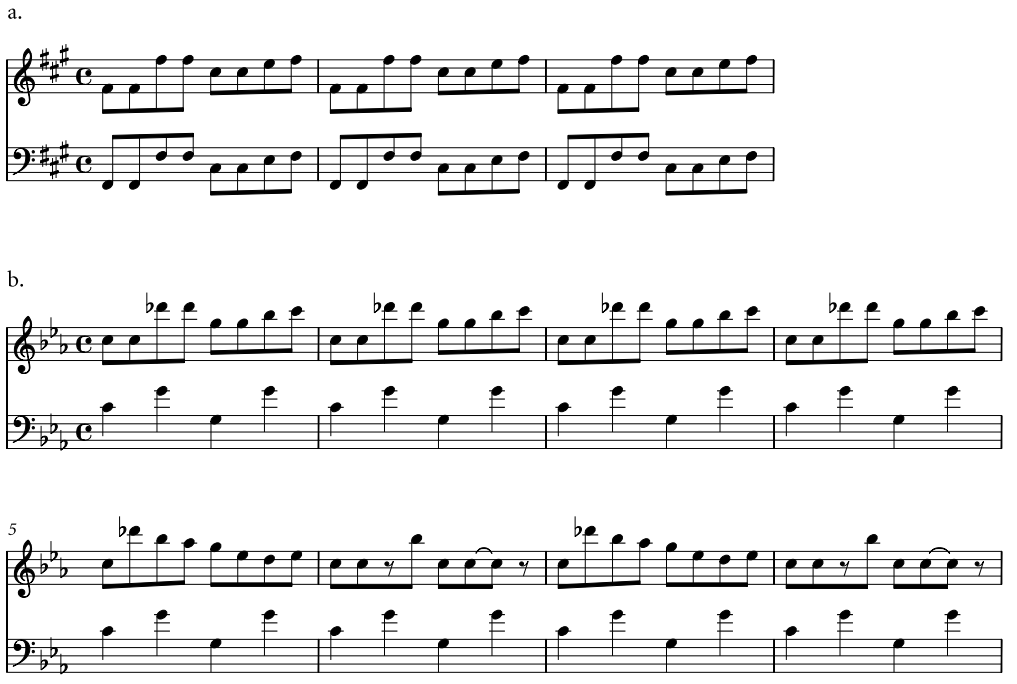
As with the arcade machines, the music of the domestic games consoles reflects their technological constrictions. The most obvious consequence of the tuning problems of the Atari VCS (discussed above) was the avoidance of harmony. The fact that the tuning could differ between different voices (for instance, a G may be available in the bass, but only a G♯ in the treble) complicated the programming of harmony, and it is little wonder that few games had songs with both bass and treble voices. Another effect was the inclusion of less common intervals, such as the prominence of minor seconds in the songs, which was due to the reduced number of notes available.[31] A comparison of the same games on the VCS console with those on the arcade machines shows these constraints in action. Most Atari songs would drop the bass line, as in *Qbert* and *Crazy Climber*. The tunes for *Up 'n Down* had a noticeably different flavour (see Example 1), suggesting that the Atari's tunings may have had a significant role to play in the prominence of minor seconds: the tune changes from a bluesy F♯-minor groove to a very unsettling version based in C minor, with a D♭ in the upper register.

The Nintendo Entertainment System (NES), by comparison, had a much wider tonal range, although the use of the tone channels reflects the constraints of its chip: typically, the two pulse channels provided chords or a solo lead, with the triangle channel acting as a bass accompaniment. The most obvious rationale for using the triangle as the bass was the limitations of the channel: the lower pitch, the reduced frequencies, and the lack of volume control. These limitations meant that many of the effects that could be simulated with the pulse waves were unavailable for the triangle wave, such as the vibrato (pitch modulation), tremolo (volume modulation), slides, portamento, and so on. Also, by altering the volume and adjusting the timing between the two pulse channels, echo effects could be simulated, as in *Metroid*, a science-fiction game with an unusually eerie soundtrack.

As was the case with the other 8-bit systems, most Commodore 64 games of the first few years had very little music, and almost no background music. In fact, in contrast with the NES, some of the popular games, even those launched in the late 1980s, had little or no music of any sort. Of the 'Top 100 C64 games',[32] about ten percent had no background music at all, and the earliest examples of games – those dating from about 1983 to 1985 – had the least

31 See Collins, 'Flat Twos and the Musical Aesthetic of the Atari VCS'.

32 According to downloads from <http://www.c64.com/> (accessed 30 October 2007).



Example 1 Music for *Up 'n Down*: (a) Arcade version (Sega 1983: 2 × SN76496); (b) Atari VCS version (Sega 1983).

amount of music. Typically, music was only used outside of gameplay. The clearest examples of this are in the *California Games* series, in which there were simple background-music introductions to events, but the music stopped as soon as the event (and therefore gameplay) began. By the late 1980s and early 1990s, when the 16-bit machines (Super NES, Sega Mega Drive/Genesis) began entering the market, a change occurred, and the games were increasingly likely to have more non-diegetic music. This suggests that the pressure of the games systems market led Commodore to attempt to emulate or adapt to the games audio aesthetic of the other available systems.

Two major factors played a part in the creation of audio for the C64: the music on the Commodore was coded in assembly language, which was harder to program than NES's BASIC-based language, and the availability of memory. The fact that systems such as the Nintendo had an (arguably) inferior sound chip to that of the Commodore, but had more gameplay music, suggests that there is some correlation between the provision of music and the storage capacity of the games cartridges. Whereas most C64 games averaged about 30 kB (on cassette), 10 kB (on cartridges), or 60 kB (to a maximum of 170 kB) on a floppy disk before having to go to a multi-disk game, Nintendo's cartridges held up to 512 kB; and with the use of memory management chips, some games (such as *Kirby's Adventure*) could be expanded to 768 kB. The issue of memory was clearly linked to the amount of looping in the games. As one Commodore programmer/composer explains of a cover version of the *Short Circuit* movie theme used in a game, 'the [song conversion] was a nightmare since it's the

tune right from the beginning of the movie 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.'[33]

Despite a few early exceptions, it was not until about 1984 that looping began to gain prominence in games. This is most obvious when examining the ColecoVision games: there is a clear division between the mostly non-looping games of 1982 and 1983[34] and the games of 1984, most of which have loops.[35] A similar effect is evident in Nintendo's early games: the very first games – *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-bar melody at different pitches), whereas later games have much more extensive loops.[36] It would appear, then, that rather than being the consequence of the limited memory available on the systems, loops were, at least *in part*, an aesthetic that grew as the games became more popular and more complex. It could be that they were also reflecting the popular music of the time, as techno and hip hop were beginning to see mainstream exposure.

Approaches to Looping in 8-bit Audio

There were some interesting responses to the technology's limits. Composer Rob Hubbard partially overcame the memory constraints of the C64 by arranging code for the music in a series of modules containing a set of songs.[37] Each module may contain title music, in-game music, and game-over music using the same source code to share instrument tables and thus save space (each different timbre used in the game was set out in a table in advance and called upon when needed). Typically, each song had three separate instrumental sounds (one for each channel), and each of these was made up of a list of patterns (sequences) in the order in which they were to be played. The code would then refer to specific sections of the module, which were called upon when necessary, reducing the need to repeat coding and thus save valuable space. This module format, emulated by other composers, lent itself well to looping. Despite the fact that looping was often a result of technological constraint, there were many different approaches to the loop in 8-bit games. These can be summarized into five categories, which I shall now explore in further detail: accumulative form, random loops, pattern repeats in different registers, a mesoloop-built song, and variations in the order and length of loops.

Accumulative form (the gradual building up of a groove by adding sequential units cumulatively)[38] was reliant on smaller formal units (micro- or mesoloops) within larger

33 Martin Galway interview, <http://www.c64hq.com/> (accessed 30 October 2007).

34 For instance, *Tutankhamun* (Parker Bros), *Jungle Hunt* (Atari), or *Choplifter* (Coleco).

35 For instance, *Gyruss* (Parker Bros), *Burger Time* (Coleco), or *Up 'n Down* (Sega).

36 All were released by Nintendo. For the NES, *Donkey Kong*, *Donkey Kong Jr*, and *Popeye* were released in 1986, but on the Famicom (the Japanese version), *Donkey Kong* and *Jr* were released in 1982, *Popeye* in 1983, and *Devil World* in 1984.

37 By using the RanSID Analyzer, it is possible to disassemble the code of SID tunes to see the way in which songs were written.

38 For an examination of accumulative form in popular music see Spicer, '(Ac)cumulative Form in Pop-Rock Music'.

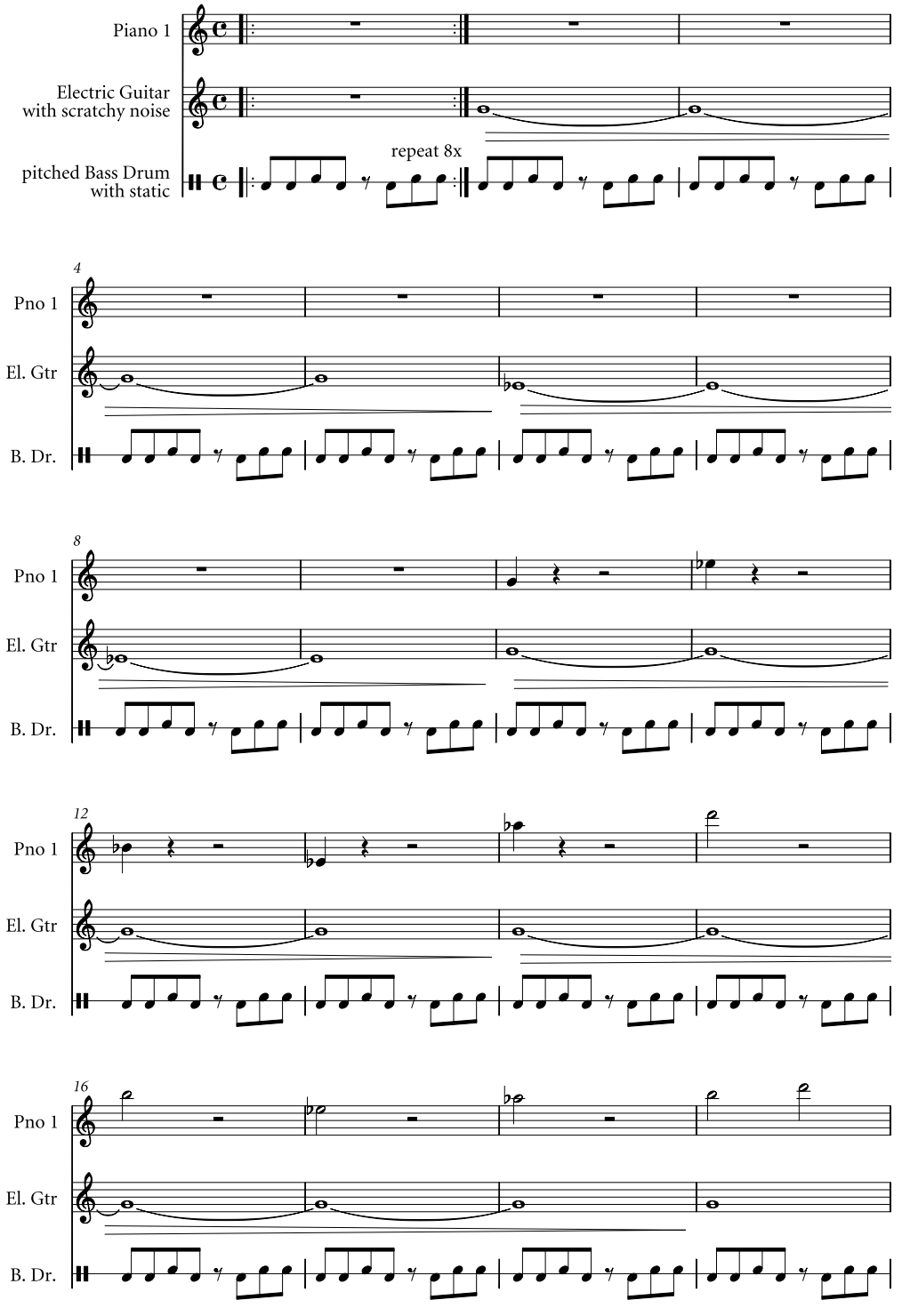
compositions. Each small unit could be called up once and then repeated, in terraced fashion, so that it would not tax the memory of the machine. Evidence of this approach is seen in Commodore 64's *Tetris*. Not having the selectable looping background music that was an option on the NES, Wally Beben composed original music – one very long track (about 26 minutes: 13 kB) of smaller loops (see Example 2). In order to save space, certain micro- and mesoloops repeat (the bass/percussion line that begins the song repeats just one bar for about half the song, for instance) while a number of melodies are superimposed on to various accompaniments.

A second approach to space constraints – random generation – was seen occasionally in the arcades and on the Commodore. The coin-operated game *Frogger*, for instance, 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 songs. The game begins in what I shall term the 'home' song, and when the player successfully guides a frog into a safe house, the song will switch to another song quite abruptly and then continue until the frog has either moved successfully into another safe house, at which point there is a new song, or died, in which case the music returns to the 'home' song. Since the maximum time that gameplay could continue before arriving at a safe house or dying was 27 seconds (much less as the levels increased and gameplay sped up), the songs did not rely on loops. Similarly, Jetsoft's *Cavelon* used what appears to be a random looping of sequences. The player moves about the screen capturing pieces of a door. Each time the player stops moving, the music stops. When the player obtains a piece of door, a brief win-item cue is heard and the loop changes randomly into a new sequence.

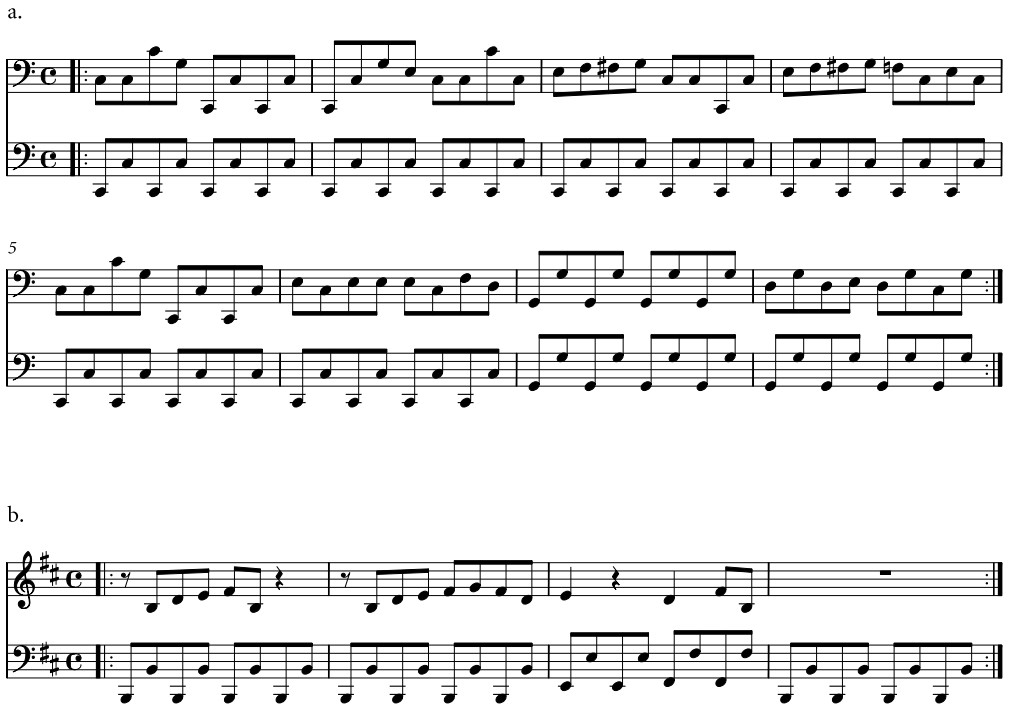
Some Commodore 64 composers were also adventurous with their coding, including the use of random number generators into the code to select from a group of loop options. For instance, *Times of Lore* used a random selection of guitar solos for the eleven-minute duration of the song. In this way the game's ten songs (comprising over thirty minutes of music) could fit into just 923 bytes, and the music sound a lot more varied than it otherwise would.[39] Random generation was also used in *Rock Star Ate my Hamster*, a rock management game in which a band practises, accompanied by a tune that draws from a random combination of sixteen sequences. In *California Games* random generation is built into the half-pipe event. Here, an opening sequence plays for seven seconds, followed (as long as the player stays up) by a random sequence drawing on a collection of sixteen possible choices, each one twenty-three seconds long. If the player falls, the first segment repeats.

The Atari VCS's *Clown Down Town* also offers a very unusual example of an approach to looping. Its nearly atonal effect results from a mesoloop that repeats in other registers and keys. Although looping was far less common on the VCS than on other systems of the time – most likely because of memory limitations – the character of this track suggests, perhaps, that if the gameplay was uncomplicated and therefore did not take too much memory, the capabilities for longer tracks did exist but were not exploited.

39 Martin Galway on *Times of Lore* in *SIDfind*, <http://www.c64.org/sidfind/> (accessed 30 October 2007).



Example 2 Music for the first part of *Tetris* (Wally Beben, Mirrorsoft 1987), Commodore 64.



Example 3 Music for *Lazy Jones* (Terminal Software 1984), Commodore 64: (a) 8-bar track 1 (main gameplay) (b) 4-bar track 21.

Another approach to looping was to use mesoloops that built up into one longer song, as in the C64 game *Lazy Jones*, which had twenty-one mesoloops that were selected in turn 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' or mesoloop, which actually played like a segment of one greater macroloop song (the title music). Even if the character left the room at, say, bar 3, the rest of the loop would play before it would transition seamlessly back into the theme song. Most of the loops worked well together, in part owing to the ragtime-like microlooped bass lines, the similarity of timbres employed, and the fact that the game used only two channels (see Example 3).

Finally, the length and order of loops were significantly different among games, platforms, and genres. As the NES was the 8-bit system with the most standardized background looping (both interactive and non-interactive), I focus on it in the following discussion. Loops on the NES were of varying lengths, depending on the genre: role-playing games and platform adventure games had the longest loops; fighting games and arcade-like games, short loops; and puzzle games, mid-length loops. Racing games had little non-diegetic music (most just had engine sounds during gameplay), and flight simulators tended to be silent. The loops on the NES ranged in length from very short (5–10 seconds), to medium length (10–30 seconds), to very long (the longest I found was about a minute and a half). Most loops averaged about thirty seconds and consisted of four or five different sections. The longest loops were present in the adventure-style games, probably because players would spend

longer on them than on other games, since they were designed to last for many hours before ending: a short loop would rapidly become irritating to the player.

Some other games looped only the last section in the song, rather than the entire song, as in the title themes to *Metroid* and *Lagrange Point*. In these cases there is therefore a mesoloop but no macroloop (A B C ‖:D:‖). The most obvious explanation for this type of loop is that the title themes were not meant to be left on: once the introduction and credits were played, the player would probably begin the game rather than finish listening to the music – a clear case of function leading the form. Some songs also had brief intros that were not included in the loop. So, for instance, *Castlevania* level 2 has a brief intro and then a looped segment that skips the intro (A ‖:B C D E:‖). The 'overworld' or game level loops (the longest sections of gameplay) were typically the longest loops (30–90 seconds), containing the most varied and the longest sections.[40] Battle music (or 'boss' music) usually had much shorter loops than other sections, with only one or two mesoloops, the likely reason being that the quick succession added to the tension, and the fact that this part of the game usually lasted only a few seconds. There were several variations at the point of looping: typically, either the loop was designed so that the last section would fit seamlessly with the return to the beginning, or a small transitional bridge was made in between loops. Transitions commonly used a glissando or a rising scale lasting one or two bars.

The loops were built in sections ranging from one to eight bars. These might repeat, but were usually found in the macroloop only once, one after the other, before progressing to a new section, and rarely returning to the original unless the entire loop was beginning anew. Looking in more detail, we can break down each section to see the looping aspect of several songs. For instance, the sixteen-bar level 1 music of *Castlevania* (see Figure 1a) has a one-bar intro (A) that loops itself once before moving to B, which has a two-bar mesoloop (B and B'); this two-bar pattern also loops once. The C and D sections also have two-bar mesoloops that repeat once, and then the E section has one bar that loops. This entire A–E element then repeats in a macroloop.

In contrast, the 'Ambrosia' level music of *Ultima 3* has an eight-bar mesoloop, A, which repeats, followed by a four-bar mesoloop, B, which repeats, and a four-bar C section, which we hear only once before the entire song loops (see Figure 1b). Because of this, the macrolooping segment (sixteen bars) feels a bit less repetitive than some of the other, more rigid and straightforward loops. *Metroid's* 'Brinstar' level, on the other hand, has taken a completely different approach to the looping of sections, with most sections being a two-bar mesoloop but repeating in odd rather than even segments (see Figure 1c). By alternating the length of the mesoloops and alternating mesoloops themselves, the looping becomes less obvious and monotonous. There are also games that use less conventional looping, such as *Super Mario Bros.*, which has four-bar sections of typically two-bar

40 Many adventure games have 'overworlds' and 'underworlds', a light-side, above-ground setting versus an underground, dark and more dangerous setting. Typically, in a game such as *Super Mario Bros.*, one level may contain three 'overworlds', where players increase basic skills before facing the fourth, 'underworld' section of the level.

a. *Castlevania*, 'level 1' music (Konami/Nintendo, 1986)

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

b. *Ultima III: Exodus*, 'Ambrosia' level music (Origin/Nintendo, 1983)

Bars	1–8	9–16	17–20	21–4	25–8
Section	A	A	B	B	C

c. *Metroid*, 'Brinstar' level music, composed by Hirokazu 'Hip' Tanaka (Nintendo, 1986)

Bars	1–12						13–20				21–8	29–30
Section	A	A	A′	A′	A′	A	B	B	B	B′	C	Bridge

d. *Super Mario Bros.*, 'level 1: overworld' music, composed by Koji Kondo (Nintendo, 1985)

Bars	1	2–5	6–9	10–13	14–17	18–21	22–5	26–9	30–3	34–7
Section	Intro	A	B	B	C	A	D	D	C	D

Figure 1 Analysis of looping sections in four different games songs.

mesoloops *nearly* repeating (with minor variation; see Figure 1d). This is one of the few games to repeat alternate sections before the entire song loops at the macro level.

Generally speaking, loops on the Commodore 64 were much longer than those of the NES, but this could be due to the fact that fewer songs were included overall, whereas the NES was more likely to have consistent interactive background music. It could be, however, that because the C64 developed separately in the USA rather than in Nintendo's Japanese home, its aesthetic tended towards full songs rather than short loops. In games that did have regular looping, these loop lengths were comparable to those of the NES, although there is a distinct lack of the transitional bridges seen on the NES in favour of seamless loops. In some cases, such as *Forbidden Forest*, the loops were abrupt, changing mid-stream depending on the interaction with gameplay.

By the late 1980s and early 1990s there were several Commodore games, such as *Mayhem in Monsterland* or the *Great Giana Sisters*, that clearly tried to copy the Nintendo game style. The latter was so similar to *Super Mario Bros.* that Nintendo successfully sued to have it pulled from stores. What is interesting is the fact that these games not only adopted NES-like gameplay and visuals, but also a distinctly NES style of music, suggesting that an associated, well-defined aesthetic was involved, rather than merely a technological component. Each game contained interactive background loops for 'overworlds', 'underworlds', and 'boss' music, in Nintendo style.

Conclusions

The year 1984 seems to be a key point in the culmination of an 8-bit aesthetic that saw the establishment of loops, dynamic music, and various forms of polyphony. As *Tetris* and *Clown Down Town*, among others, demonstrated, since alternatives to this aesthetic were available, the games audio aesthetic was *chosen* as much as determined by the limitations of the available technology, yet each machine had a slightly different aesthetic that grew, in part, from the technology available. The most notable difference, perhaps, was that between the Commodore 64 and the NES. The extended capabilities of the C64's SID chip helped create more of a traditional rock-song approach to music, while the NES relied more on a series of loops. As composer Rob Hubbard says:

> 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.[41]

The persistent practice of looping nicely illustrates the tensions between technology and aesthetic. As we have seen, various approaches to looping were certainly available and, at times, used. But straight short-looping remained the most prominent response to limited memory. Looping, then, appears to be as much an aesthetic choice as a factor predetermined by technology. But, as also mentioned, constraints were not just limited to the technology. The social constraints of the specialized knowledge required to programme the machines – and to write music – also impacted on the developing aesthetic, with many composers lacking formal musical training. The impact of this could very well be one of the reasons why sound effects were so prominent, and why some of the music in the 8-bit era was unconventional in many ways. After all,

> When musicians are faced with new pieces of equipment, they often tend to transfer their knowledge from previous experiences with similar equipment [. . .] Students often program 'outer space noises' or 'bubbling volcanos' [*sic*] because 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'? Certainly not, unless they are jealous because it sounds better than their 'Phase Dishwasher'.[42]

Sound aesthetics are the result of a culmination of knowledge, creativity, and constraint. Perhaps one of the reasons why we saw innovations such as the use of accumulative form and random generation was that the composers were programmers rather than musicians. We also saw that the constraints were not limited to these two factors, but also to genre and gameplay narrative. For instance, *Frogger* had a specific time constraint in each level, to which

41 Rob Hubbard interview, <http://www.freenetpages.co.uk/hp/tcworh/int_6581.htm> (accessed 23 November 2005).
42 Righter and Mercuri, 'The Yamaha DX-7 Synthesizer'.

the music had to keep. Sports games had little music, and role-playing games had a lot of lengthy loops.

It could also be argued that the looping elements of games audio could have been influenced by the popular music of the time. By 1984 – as we see, the real 'beginning' of the loop as a common aesthetic in games – hip hop had become mainstream, disco had come and gone, and techno was in the ascendancy. Although looping goes back further in time than these genres, by the mid-1980s looping and repetition were a distinct and important part of the popular repertoire because of the development of sequencers and drum machines. Repetition, therefore, was an aspect of games music that was as much an aesthetic choice as it was a result of the technology. It would appear, then, that despite some quite rigid technological constraints, as indicated in the Introduction, these were a 'loose pressure' on the development of games audio, rather than the deciding factor in 8-bit aesthetics.

Discography

Aphex Twin. *Pac-Man*. CD single, Ffrreedom 110. 1992.
Beck. *Hell Yes Remix*. EP, Interscope INTR-11325–1. 2005.
The Bloodhound Gang. *Hooray for Boobies*. CD, Interscope 90455. 2000.
Buckner and Garcia. *Pac-Man Fever*. CD, Columbia 37941. 1982.
DMX. *Flesh of my Flesh, Blood of my Blood*. CD, Def Jam 538640. 1998.
Lil' Flip. *U Gotta Feel Me*. CD, Sony 92411. 1998.
'Weird Al' Yancovic. *Dr Demento's Basement Tapes No. 4*. Cassette (fanclub-only release). 1981.

Games

Recordings, remixes, and MIDI versions of games music can be heard on many websites, including the Video Games Music Archive <http://VGMusic.com>, and VORC <http://www.vorc.org/>. Soundtracks featuring collections of early games sound also exist.

Arkanoid (coin-op, Nintendo Entertainment System (NES), Commodore 64). Taito, 1986. Nintendo sound composed by Hisayoshi Ogura; sound effects by Tadashi Kimijima; game designed by Akira Fujita. Commodore music composed by Martin Galway.
Asteroids (coin-op). Atari, 1979.
Burger Time (ColecoVision). Coleco, 1984.
California Games (Commodore 64). Epyx, 1987. Music composed by Chris Grigg.
Carnival (coin-op). Sega, 1980.
Castlevania (Commodore 64, NES). Konami, 1987.
Cavelon (coin-op). Jetsoft, 1983.
Chicken Shift (coin-op). Bally, 1984.
Choplifter (ColecoVision). Coleco, 1983.
Circus (coin-op). Exidy, 1977.
Clown Down Town (Atari VCS). Walker – Rainbow Vision, date unknown.
Crazy Climber (coin-op). Midway, 1982.
Dance Aerobics (NES). Nintendo, 1989.
Devil World (Nintendo Famicom). Nintendo, 1984.
Discs of Tron (coin-op). Bally, 1983.
Donkey Kong (Nintendo Famicom/NES). Nintendo, 1982.
Donkey Kong Jr (Nintendo Famicom/NES). Nintendo, 1982.
Duck Hunt (NES). Nintendo, 1986.
Forbidden Forest (Commodore 64). Kosmi, 1983.

Frogger (coin-op). Konami, 1981.
Ghosts 'n Goblins (NES, coin-op, Commodore 64). Capcom, 1985. Commodore music composed by Mark Cooksey.
Great Giana Sisters (Commodore 64). Rainbow Arts, 1987.
Gyruss (ColecoVision). Parker Bros, 1984.
Jungle Hunt (coin-op). Atari, 1983.
Karate Champ (NES, coin-op, Commodore 64, Atari VCS). Nihon Bussan, 1984.
Kirby's Adventure (NES). Nintendo, 1993.
Lagrange Point (Famicom). Nintendo, 1989.
Lazy Jones (Commodore 64). Terminal Software, 1984.
Maniac Mansion (Commodore 64, NES). LucasArts, 1987. Music composed by Chris Grigg and David Lawrence.
Mayhem in Monsterland (Commodore 64). Apex, 1993.
Metroid (NES). Nintendo, 1986. Directed by Yoshio Sakamoto; music composed by Hirokazu Tanaka.
Mike Tyson's Punchout (NES). Nintendo, 1987.
Pac-Man (coin-op). Namco, 1980.
Popeye (NES). Nintendo, 1983.
Qbert (coin-op, Atari VCS). Parker Bros, 1983.
Rally X (coin-op). Namco, 1980.
Rip Cord (coin-op). Exidy, 1979.
Rock Star Ate my Hamster (Commodore 64). CodeMasters, 1988.
Space Invaders (coin-op). Midway, 1978.
Super Mario Bros. (NES). Nintendo, 1985. Music composed by Koji Kondo.
Tetris (Commodore 64). Mirrorsoft, 1987. Music composed by Wally Beben.
Times of Lore (Commodore 64). Microprose, 1988. Music composed by Martin Galway.
Turbo Outrun (Commodore 64). Sega, 1989.
Tutankhamun (ColecoVision). Parker Bros, 1982.
Ultima 3: Exodus (NES, Commodore 64). Origin Systems, 1983. Commodore music composed by Kenneth Arnold.
Up 'n Down (coin-op, Atari VCS). Sega, 1983.

Bibliography

Aikin, Jim. 'Brian Eno'. *Keyboard* 7 (July 1981). Extract at <http://www.specht-h.at/interview.htm> (accessed 30 October 2007).
Carr, Neil. 'An Interview with Anders Carlsson AKA GOTO80'. <http://www.remix64.com/index.php?interview_anders_carlsson_aka_goto80> (accessed 30 October 2007).
Collins, Karen. 'Loops and Bloops: Music on the Commodore 64'. *Soundscapes: Journal on Media Culture* 8 (February 2006). <http://www.icce.rug.nl/~soundscapes/> (accessed 30 October 2007).
———. 'Flat Twos and the Musical Aesthetic of the Atari VCS'. *Popular Musicology Online*, Issue 1: *Musicological Critique* (June 2006). <http://www.popular-musicology-online.com/> (accessed 30 October 2007).
———. 'An Introduction to the Participatory and Non-Linear Aspects of Video Games Audio', in *Essays on Sound and Vision*, ed. Stan Hawkins and John Richardson. Helsinki: Helsinki University Press, 2007. 263–98.
Gorbman, Claudia. *Unheard Melodies: Narrative Film Music*. Indiana: Indiana University Press, 1987.
Katigbak, Raf. 'Game on'. *Montreal Mirror*, 27 October 2004. <http://www.montrealmirror.com/2004/102104/nightlife2.html> (accessed 30 October 2007).
Katz, Mark. *Capturing Sound: How Technology has Changed Music*. Berkeley: University of California Press, 2004.
McLuhan, Marshall. *Understanding Media: the Extensions of Man*. Toronto: McGraw-Hill, 1964.
Middleton, Richard. ' "Over and Over": Notes towards a Politics of Repetition. Surveying the Ground, Charting Some Routes'. Paper presented at the conference *Grounding Music for the Global Age*, Berlin, May 1996. Available at http://www2.rz.hu-berlin.de/fpm/texte/middle.htm> (accessed 30 October 2007).
Peters, Michael. 'The Birth of Loop' (1996, modified 2004, 2006). <http://www.loopers-delight.com/history/Loophist.html> (accessed 30 October 2007).
Pouladi, Ali. 'An Interview with Ben Daglish'. *Lemon* 64 (June 2004). <http://www.lemon64.com/interviews/ben_daglish.php> (accessed 30 October 2007).
Righter, Dennis, and Rebecca Mercuri. 'The Yamaha DX-7 Synthesizer: a New Tool for Teachers', in *Proceedings of the 5th Symposium on Small Computers in the Arts*. Philadelphia: IEEE Computer Society Press, 1985.

Salt, Barry. 'The Evolution of Sound Technology', in *Film Sound: Theory and Practice*, ed. Elisabeth Weis and John Belton. New York: Columbia University Press, 1985. 37–43.
Spicer, Mark. '(Ac)cumulative Form in Pop-Rock Music'. *twentieth-century music* 1/1 (2004), 29–64.
Stillar, Glenn. 'Loops as Genre Resources'. *Folia Linguistica* 39/1–2 (2005), 197–212.
Stolberg, Eckhard. 'Atari 2600 VCS Sound Frequency and Waveform Guide' (2000). <http://home.arcor.de/estolberg/texts/freqform.txt> (accessed 30 October 2007).
Taylor, Timothy D. *Strange Sounds: Music, Technology and Culture*. New York: Routledge, 2001.
Théberge, Paul. *Any Sound You Can Imagine: Making Music/Consuming Technology*. Hanover, NH: Wesleyan University Press, 1997.
Wolf, Mark J. P. 'Genre and the Video Game', in *The Medium of the Video Game*, ed. Wolf, foreword by Ralph H. Baer. Austin: University of Texas Press, 2001. 113–34.