JAMES NEWMAN
Bath Spa University, FHEA
j.newman@bathspa.ac.uk

# Driving the SID chip

## Assembly Language, Composition, and Sound Design for the C64

**ABSTRACT**

The MOS6581, more commonly known as the Sound Interface Device, or SID chip, was the sonic heart of the Commodore 64 home computer. By considering the chip's development, specification, uses and creative abuses by composers and programmers, alongside its continuing legacy, this paper argues that, more than any other device, the SID chip is responsible for shaping the sound of videogame music. Compared with the brutal atonality of chips such as Atari's TIA, the SID chip offers a complex 3-channel synthesizer with dynamic waveform selection, per-channel ADSR envelopes, multi-mode filter, ring and cross modulation. However, while the specification is sophisticated, the exploitation of the vagaries and imperfections of the chip are just as significant to its sonic character. As such, the compositional, sound design and programming techniques developed by 1980s composer-coders like Rob Hubbard and Martin Galway are central in defining the distinctive sound of C64 gameplay. Exploring the affordances of the chip and the distinctive ways they were harnessed, the argument of this paper centers on the inexorable link between the technological and the musical. Crucially, composers like Hubbard et al. developed their own bespoke low-level drivers to interface with the SID chip to create pseudo-polyphony through rapid arpeggiation and channel sharing, drum synthesis through waveform manipulation, portamento, and even sample playback. This paper analyses the indivisibility of sound design, synthesis and composition in the birth of these musical forms and aesthetics, and assesses their impact on what would go on to be defined as chiptunes.

**KEYWORDS:***SID chip; driver affordance; design potential: Rob Hubbard; Martin Galway*

### INTRODUCTION

Released in 1982, the Commodore 64 (C64) would go on to sell approximately 17 million units, becoming "the best-selling single personal computer model of all time" (CHM, 2007). Among its catalogue of approximately 10,000 com-

mercial programs, games were prominent, and the C64 became a key gaming platform with magazines such as *Zzap!64* (UK, Newsfield Publishing, 1985-1994), offering reviews and features on the ever-expanding catalogue. As Barton and Loguidice (2007) note, the C64's impact on the nascent home gaming space was significant, and its low cost and inherent flexibility of application when compared with single-function gaming devices are often cited as contributory factors to the early 1980s US market "crash" (see Wolf, 2012). The provision of two joystick ports and high-resolution graphics modes ensured that the computer was well suited to gaming. As such, like the BBC Model B and Commodore's later Amiga series, although originally conceived as a general-purpose home computer, the C64 was and continues to sit alongside dedicated videogame consoles in popular and scholarly discourse (see Gazzard, 2016; Maher, 2012).

Sound was especially crucial to the platform's success and, as Collins (2006a) notes, C64 game music has a "unique aesthetic" that makes use of "screaming guitar-like square wave solos, full-length songs, [and] attempts to re-create traditional 'rock band' line-ups in its use of tone channels" (online). Such was the popularity of C64 soundtracks that, in addition to rating the music and effects of the titles under review, *Zzap!64* regularly featured interviews with composers including Rob Hubbard and Martin Galway who gained a celebrity equaling the designers of the games their music accompanied. The magazine even compiled monthly Top 10 lists of readers' favorite soundtracks. The popularity of this music remains today with archives such as the online *High Voltage SID Collection* (*HVSC*) gathering files for replay on emulators dedicated to the singular task of reproducing C64 music and sound (to the exclusion of graphics or gameplay).

The unique aesthetic of C64 music is partly a function of the system's sound chip. The *Sound Interface Device* (or "SID chip", as it is more commonly known) is widely celebrated in gaming, electronic music and general computing discourse (Laing, 2004; Byte, 1995). It is known for the sophistication of its design and for being the inaugural project of the designer who would go onto to found Ensoniq, a company which low-cost sampling keyboards proved commercially and technically impactful in the musical instrument space (Vail, 2014). As Viens (2012: 47) notes, "This chip really requires no introduction. It's by *far* the most famous of all music chips in the world. It is also by far the most complex analog/hybrid chip of the lot" (original italics). Some caution must be exercised, however. The comparative sophistication of the SID chip's specification perhaps encourages the adoption of a broadly technologically deterministic approach to conceiving of videogame music production, in which aesthetics and specification are not only related to one another but also are historically mapped in terms of identifiable generations:

> "As with the visual side, the history of video game music is highlighted by the type of technology available at that time. As a result, we have the 8-bit, 16-bit, 64-bit, and the 128-bit eras. The first video games lacked a sound component, included

> only a brief theme, a few sound effects or were limited to simple melodies by early sound synthesizer technology" (Marquez, 2014: 68).

Eschewing a simple linear timeline, some commentators point to the distinctiveness of specific hardware devices and platforms. For Joseph P Beuckman (2001), "(c)omputers have personalities, shapes and architectures like a canvas that influence what we make" (Beige 0036, 2001: online). If we are to appreciate the unique aesthetic of the C64 music, disentangling the specific contribution of the SID chip and the compositional and sound design techniques that harnessed it is essential. This is especially the case given the tendency of contemporary *chiptune* practitioners and fans to conflate such differences. As Altice (2015) notes:

> "The output of the GameBoy, NES and Commodore 64 are now subsumed under the chiptune moniker, but the sonic character of those machines are far more unique than the Xbox 360, PlayStation 3, or Nintendo Wii. Games ported across those platforms will exhibit visual differences, but their soundtracks will remain the same. There is no "sound" of the Xbox 360 any more than there is a "sound" of an Onkyo CD player" (Altice, 2015: 277).

Moreover, given the particular characteristics of these chips' programming interfaces, I argue that we must extend our analysis beyond silicon design and specification. To appreciate the distinctive forms and aesthetics of C64 music, we must consider the creation of the music player routine or "driver", because it is in the creation of this interpretative software layer that we see the interplay between musical inventiveness, sound design, and the hidden and revealed affordances of the SID chip.

Ultimately, I argue that the creation of the driver is both a response to, as well as an investigative revelation of, the SID chip's affordances. In considering the chip in terms of a suite of affordances rather than a definitive or stable specification, I draw on Gibson (1979) and, in particular, Norman's (1999) exploration of the relationship between the attributes of a (design) object and an actor using it. In part due to Norman's popularization of the concept, the concept of *affordances* has been widely adopted in Human Computer Interaction (HCI) research where it is used to refer to the functional properties of objects that allow particular uses (Murray, 2011). As such, my analysis is concerned with the capabilities of the chip, which are revealed by and codified in the production of specific software tools for creating music and sound, and which reflect particular technical and musical sensibilities and contexts. For instance, the SID chip's ability to replay samples is not part of its specification per se, given that it is not a feature intended or documented but, rather, a "hidden affordance", uncovered through experimentation and the exploitation of a "bug", or quirk, of the SID chip a few years after its initial release.

Accordingly, my focus on the production of the driver sees it as both a response to the perceived affordances of the SID chip and an exploration of its less obvious affordances. Through experimentation and targeted investigation, channeled via musical imagination, these hidden affordances (whether intended, or quirks of the design and manufacture) are made accessible and perceptible and become part of the chip's functional repertoire. The investigative work that is undertaken by composer-programmers bears many similarities to other hacking practices (see Marquez, 2014; Danet, 2001; Raymond, 1996), as well as to gameplay. As such, I position the exploration and revelation of the chip's affordances as a "configurative" ludic practice dedicated to "the manipulation of dynamic systems that develop in unpredictable or emergent ways" (Moulthrop, 2004, pp. 63-64).

The argument concludes by examining the contemporary availability of the SID chip to musicians in software or within a hardware system such as the Elektron SidStation or Twisted Electrons Therapsid. Although these instruments support integration into modern *Digital Audio Workstations* (DAWs) and performance-oriented workflows, their new interfaces disallow techniques available via 1980s drivers. The SID chip is, therefore, constructed as a different instrument, and the suite of affordances is materially altered.

## SOUND INTERFACE DEVICE

In the early 1980s, computer and arcade game sound chips were typically uncomplicated affairs. As Collins notes, "PSGs [Programmable Sound Generators] offered little control over the timbre of a sound, usually limiting sounds to single (often square) waveforms, without much ability to manipulate that waveform" (Collins, 2006b). The Atari VCS' *Television Interface Adapter* (TIA) is a case in point. Handling both the system's visual and audio output, and extensively analyzed by Montfort and Bogost (2009), the TIA is a sound chip that employed a 5-bit frequency divider which generated a finite number of mathematically-related, but often musically unrelated, pitches. Although refreshing in their honesty, the opening lines of the *Atari 2600 Music And Sound Programming Guide* hardly instill confidence in the budding TIA musician. "It is difficult to do music on the Atari 2600 due to the limited pitch and two voices… many of the pitch values are not in-tune with others" (Slocum, 2003).

Indeed, much TIA music, such as the 1984 *Gyruss* soundtrack, is discordant almost to the point of comedy. However, as Driscoll and Diaz (2009) note, to compensate for the TIA's esoteric tuning, Garry Kitchen, the developer of Activision's 1983 *Pressure Cooker*, "determined a set of pitches that the Atari TIA could reliably reproduce. He then hired a professional jingle writer to compose theme music using only those available pitches" (par. 2.3). Excepting such inventive solutions, the majority of Atari's TIA-generated music has a decidedly atonal quality to it. Collin's (2006) analysis of Atari 2600 soundtracks notes an absence of harmony and an unusually high incidence of "flattened seconds" both of which are attributable to the TIA's unique tuning table and eschewing of equal

temperament. By contrast, Commodore hired Bob Yannes, an engineer with a musical background, to design the SID chip. As Yannes notes, "One of the reasons I was hired was my knowledge of music synthesis was deemed valuable for future MOS/Commodore products" (Yannes interviewed in Varga 1996).

As a musician, that Yannes was unimpressed with the state of computer PSGs might not surprise us. "I thought the sound chips on the market, including those in the Atari computers, were primitive and obviously had been designed by people who knew nothing about music" (Bagnall, 2011: 372). However, what is notable is that Yannes' ambition was not to make an incrementally better PSG than the TIA and comparable sound chips: "I really wanted to do a multi-track, polyphonic music synthesizer" (Bagnall, 2011: 373). This desire to create a different order of PSG that not only drew inspiration from the features and functionality of professional synthesizers but that also might be used to power a subsequent generation of such musical instruments, goes some way to explaining the sophistication of Yannes' design. As Charles Winterble, manager of MOS Technology (the company that manufactured SID), observed, "This thing is already 10 times better than anything else out there and 20 times better than it needs to be" (Bagnall, 2011: 374).

Fundamentally, the SID is built around a core of three discrete channels or "voices" that are combined prior to the final output stage for processing. Each voice is, in essence, an individually addressable monophonic, subtractive synthesizer offering flexible sound generation, modulation and shaping features. Where the TIA offered a series of tuning tables comprising of 32 often jarring pitches, SID's oscillators offered precise control over an eight-octave range (Vogel & Scrimshaw 1983: 5). Even the later *Nintendo Entertainment System* (NES) offered a limited palette of waveforms. Two of its four voices were hardwired to playback a variable pulse waveform, the third was limited to just a triangle, while the fourth offered noise. Typically, as with Koji Kondo's *Super Mario Bros.* soundtrack (Schartmann, 2015), these voices, and their waveforms, were assigned to particular musical duties with the two pulse waveforms handling melodic and harmonic lines, the triangle assigned to the bassline, and the noise channel used for percussion. As Troise (2015) observes, this "ensemble" form of composition has a history dating back many hundreds of years, although the "Famichord" (Akesson, 2011), with its omission of the dominant from a major or minor 7th chord, is a distinctive response to the voice architecture of the NES (or Famicom as the system was known in Japan).

Each of SID's oscillators offers four independently selectable waveforms: sawtooth, triangle, pulse and noise. Crucially, waveforms are not tied to channels and can be altered on a per-cycle basis as well as being used to cross-modulate one another giving rise to sync- and ring-modulation effects. The pulse waveform offers continuous duty cycle control, making possible the phasing, animated *Pulse Width Modulation* (PWM) effect familiar from analogue synthesizers and described by composer Chris Huelsbeck as "the holy grail of its [the

SID chip's] power" (Carr, 2001a). Further replicating the topologies of subtractive synthesizers, the three channels of harmonically rich waveforms pass into a multimode, state-variable resonant filter with dynamic cutoff control[1].

Additionally, SID offers per-channel ADSR (Attack-Decay-Sustain-Release) envelopes for controlling the volume contour of each voice. "The Commodore 64 sound generator allows us to control the rates at which the sound output will build up and die away, and by carefully manipulating these rates we can produce a wide range of instrument effects" (Money, 1984: 169). With attack times as short as a couple of milliseconds and release times approaching 30 seconds (see CBM 1983a), SID's envelopes, like its PWM and filter, directly reference the instruments Yannes took inspiration from and sought to contribute to with his design. By contrast, the NES's sound chip offers limited amplitude control over its pulse and noise channels, and no control at all over the amplitude of the triangle.

In the context of its contemporaries, SID is a sophisticated sound generator. The *Owner's Manual* for the recent Elektron SidStation (which puts an original 6581 chip under the control of a modern, digital musical interface) is more poetic: "SID is the classic synthesizer that never had a case built around it" (Elektron 1999: 5). In this article, I suggest that by producing bespoke software routines for sound design and composition that harness the perceived and revealed affordances, each composer-programmer effectively "encases" the SID chip. Each software driver constructs it as a specific instrument and music-making system, (de)privileging facets and codifying techniques that become inexorably entangled with the specification, capability and "sound" of the hardware.

This case offers a clear example of the interplay between perceived and hidden affordances and the exploratory work undertaken by composer-programmers. Delving deeper into Hubbard's driver, we find specific subroutines designed to facilitate the musical exploitation of the SID chip oscillators' precise pitch control.

## THE DRIVER

While Yannes' design may have explicitly referenced musical instruments such as the Minimoog in its functionality and topology, it is important to note that the SID chip's interface was quite unlike any performance-oriented musical instrument. In common with chips of its era, the whole of the functional potential of SID was accessed through 29 8-bit registers. These could be read and manipulated either with BASIC PEEK and POKE commands or more directly through Assembly code. Where the Minimoog offered an inviting and accessible rotary potentiometer to control its filter's cutoff frequency and a piano-style keyboard with a pitch bend wheel for triggering and manipulating notes, the SID chip presents "FC Lo" and "FC Hi" registers for cutoff and "Freq Lo" and "Freq Hi" registers per voice for setting oscillator pitch all of which are set in Hexadecimal.

This is an interface squarely located in the world of code with none of the concessions to accessibility or musician-friendliness that inform the workflow and meticulous labeling of a Minimoog control panel (see Pinch & Trocco,

**1.** The design was flawed as Yannes observes. 'I knew it wouldn't work very well, but it was better than nothing and I didn't have time to make it better.' (Varga 1996). That the filter design was compromised was problematic but, as composer Ben Daglish notes, the performance variability between SID batches was a yet more frustrating issue, 'you never had ANY idea how it was gonna sound on another machine.' (Flat Four 2005; see also Collins 2008). Indeed, 'The game BeachHead even allowed the user to change the filter settings, to try to compensate for this.' (Judd 1996).

2002). As Collins notes, "(t)his meant that most early games composers were in fact programmers working on other aspects of a game, or at best in rare cases, in-house programmer-musicians who had to work closely with programmers" (Collins, 2006b). As Rob Hubbard, whose compositions we will explore in detail below, notes:

> "There were no MIDI sequencers, no Trackers. We coded everything just in an Assembler. I used to load up a machine code monitor and literally display the bytes in real time. The music was all triggered on the raster interrupt and I would start changing the numbers in real time to alter the synth settings and musical notes. So, I would tend to work on four bar chunks that I would tend to repeat and I would sit on that Hex editor, changing things. I would sit and tweak all those numbers until I had the four bars pretty much the way that I wanted them to sound and that would let me continue on for another 16 bars…" (Hubbard, 2002).

There were consumer-facing products such as Commodore Music Maker (1982), which used standard notation and a musical keyboard overlay that sat atop the C64's QWERTY keyboard for note entry. The system was inventive but, as Carlsson (2008) notes, it was too inefficient for use in game development and the output too difficult to integrate. As composer Thomas Petersen (2006) notes, even tools such as Chris Huelsbeck's more professionally-oriented SoundMonitor program suffered from high memory usage and CPU spikes, making it less advantageous than using Assembly code.

In the absence of available or suitable tools, composer-programmers typically created custom "music players", or drivers, to control synthesis and sequencing. Being bespoke, the coverage of the driver might constitute a subset of the possible SID chip feature set as deemed significant to the composer or composition in question. As such, while the SID chip offers the affordance of PWM on each voice via six addressable registers, this feature need not be implemented in a driver routine. Similarly, the manner in which it is implemented is driver-specific and reflects compositional, sound design and aesthetic intentions.

McSweeney's (1993) investigation of a ripped (extracted) version of one of Hubbard's early SID drivers reveals the extent of the work performed by this comparatively small, but absolutely essential, software routine. Occupying just 900-1000 bytes of code, the driver controls every aspect of the sound from silencing and initializing the SID chip, through defining instrument settings such as waveform, envelope times and levels; gathering note durations and pitches; applying effects such as portamento, vibrato or arpeggiation; altering pulse-width depth and modulation time; to the selection of new sequences each comprising patterns of notes for playback. By separating a "song" into three "tracks", each comprising "pattern" numbers that refer to sequences of "notes", the structure of a given piece of music can be expressed. At the note level, the driver is separated into "notework" and "soundwork" routines governing

pitch and effect parameters. The main loop runs three times (once per voice). Clocked by the system counter (running at 50Hz on a European PAL machine and 60Hz in NTSC regions such as the US), the control of note and synthesis parameters has the precision of a single frame of raster time. The speed of these changes is of vital importance in generating some of the specific and distinctive sound design and compositional motifs found in the output of Hubbard, and of other C64 musicians. Importantly, we note already that timings are based on computational and audiovisual system contingencies rather than musical relevance. Hubbard (2005) recalls that:

> "I was writing my own assembly-language music routines. The music routines were basically controlling the chip, and whatever you could do with the software to get some more sounds, or anything more interesting… you would write the music to take advantage of that. The two basically went hand in hand… You would come up with an idea for something the software could do, you would write that, and then write the music to take advantage of it. In most cases the two things happened simultaneously. You can think of something to do with the software and you immediately know what the musical implication of that is gonna be" (Hubbard, 2005).

This case offers a clear example of the interplay between perceived and hidden affordances (Norman 1988) and the exploratory work undertaken by composer-programmers. Delving deeper into Hubbard's driver, we find specific subroutines designed to facilitate the musical exploitation of the SID chip oscillators' precise pitch control. "Instruments" (specific sounds such as the simulation of a bass guitar or piano) are set in an 8-byte data structure. Examining Byte 7, McSweeney (1993) notes that:

> "Bit#1 signals a 'skydive'. This is a slower frequency down, that I think sounds like somebody yelling as they fall out of a plane. […] Bit#2 signals an octave arpeggio. It's a very limited arpeggio routine in this song. Listen for the arpeggio and the skydive when combined, which is used alot (sic) in Hubbard songs" (McSweeney 1993).

Indeed, these effects are manifestly evident in Hubbard's *Monty on the Run* or *Crazy Comets* (Martech, 1985) soundtracks among others. As McSweeney notes, this combination of a smoothly descending pitch bend (the "skydive") with octave arpeggiation (alternating between the original pitch and +1 octave every 50 cycles) is a distinctive musical and sound design trait in Hubbard's output. And so, while this composite sonic effect is made possible by the stability of the SID chip's oscillators and the precision of the control over their pitch in their "hi" and "lo" registers, it is rendered musically executable through the coding of the driver. And, just as Galway incorporated PWM into a revised driver to provide greater synthesis sophistication and animation, so Hubbard's

updated routines add further pitch and sound design effects to their instrument definitions via the manipulation of different registers in different combinations.

In discussing the compositions of other composers he particularly revered, Galway's focus on Hubbard's work reveals his personal aesthetic preferences and speaks to the position that Hubbard held, and continues to hold, among his peers and fans. However, and more importantly, it also points to a contributing quality of the driver, as Galway points out to Carr (2001b):

> "[Neil Carr] *Is there a SID tune that wasn't your own that you would have liked to have composed yourself?*
> [Martin Galway] Absolutely, they're usually Rob's!!! But I suppose I was simply wanting to use his program to get all that percussion. I wouldn't have minded having a go with his software and doing music in my style with his percussion. Perhaps we should have swapped drivers for a game just to see what we could have come up with. Or perhaps, collaborated on the same tune, each other's driver contributing simultaneously" (Carr 2001b).

What we are reminded of here is that the driver is bespoke code that combines a particular understanding of the SID chip and its capabilities, a suite of distinctive musical and aesthetic priorities that both reflect and inflect the driver, and an approach to coding that sets out the musical and synthesis capabilities in exploitable configurations. Unsurprisingly, given their importance in constituting the SID as an instrument, drivers were continually developed and refined. Discussing *Monty on the Run*, Hubbard (n.d.) remarks that, "The middle section was an excuse to use the new pitch bend code that I wrote for this project" while Martin Galway, in-house composer at Ocean Software notes, "I didn't develop pulse-width modulation until the next project, which at the time was called *Cyclone*. It later came out as *Helikopter Jagd*. On *Roland's Ratrace* I guess you could say I was still mastering the C64" (Galway n.d.). However, the "personality" of the driver remained. As Chris Abbott, C64 remixer and owner of the C64audio.com website and record label, observes, "Rob Hubbard sounded very different from Martin Galway because they had to write their own synthesizer engine, as well as the music" (Flat Four, 2005). As Galway intimates, Hubbard's routine places emphasis on rhythmic elements, both explicitly in terms of drum and percussion instrument settings where his own routine focused more on waveform manipulation, ring modulation and PWM (although, ironically, the discovery of a means of performing sample playback would later transform Galway's driver and musical output, as noted below––see also Tognon, 2003).

Further related to the idea of driver "personality", it is important also to note that the early era of C64 composition is rather different from contemporary discourse of videogame music, which often focuses on: dynamic and adaptive aspects of audio and the explicit relationship between the auditory, visual and gameplay (see Fritsch, 2012; Collins, 2007b); the healing of the separation

between composition and instrument (Herber, 2008); procedurality and elements of randomization (Stevens & Raybould, 2016); and the articulation of affect through sound (see Horowitz & Looney, 2014). C64 soundtracks were not typically interactive but, rather, accompanied gameplay while being interrupted only by synchronized sound effects (indeed, Rob Hubbard's *Kentilla* soundtrack was originally intended to adapt to changes in gameplay environment, but the idea was scrapped due to its technical complexity––Zzap,164 1986: 74). Also, as Hubbard notes:

> "The sense of freedom that people had in those days was just extraordinary. You could branch out and do pretty much whatever you wanted to do… They [publishers and developers] were letting me write anything I wanted to write" (Hubbard, 2002, online).

Surveying collections such as *HVSC*, we find many examples of "samplers": compositions written on spec rather than in response to a specific commission. Designed to showcase the composer's work and to be available for purchase, many of Hubbard's critically acclaimed pieces (*One Man and His Droid*, for instance) began life this way and were written long before the game was even conceived and released in 1985).

With so much artistic and compositional freedom and the liberty to create original, experimental SID music rather than accompaniments to particular titles or sound in the service of specific experiences or gameplay mechanics, the driver is revealed as a key site for study. Hubbard's routine, like Galway's, is an expression of particular approaches and intentions––of specific musical predilections and influences, and born of a specific understanding and revelation of the affordances of the chip, both influenced by and sometimes exceeding, the extent of the documentation and design.

## THE DRIVER AT PLAY: THE MUSIC OF ROB HUBBARD

For all the discussion of SID's synthesis sophistication, we should remember that it does not present unlimited opportunity. For instance, while Yannes' original plan had been to make use of "multiplexed" oscillators to provide up to 32 separate voices (notes), the final chip could play just three. So, while each voice offers flexibility in excess of the capabilities of contemporary sound chips, as composer Ben Daglish put it, SID still offered "limited polyphony, to say the least" (Carr, 2001c). To compound matters, as the sole sound-producing device in the C64, SID was responsible for delivering not only music but also sound effects.

To tackle the issue, a number of strategies were adopted. In some cases, such as *The Human Race*(Mastertronic 1985), Hubbard composed the music to use just two SID chip voices with the third dedicated to sound effects––or "interactive non-diegetic sound" as Collins (2007a, p. 212) puts it. In other instances, such as *Delta* (Thalamus, 1987), where the in-game composition utilizes all

three SID channels, a switch on the options screen allows the player to toggle between this music and the sound effects as the accompaniment to their gameplay. However, more common was for both music and sound effects to play concurrently resulting in "note-stealing" as one or more channels of music were temporarily silenced to make way for a spot sound effect. This is heard extensively in C64 soundtracks such as *Thing on a Spring* (Gremlin Graphics, 1985) and *Commando* (Elite, 1985), though it was no means limited to that system. Indeed, although it offered an additional voice, most NES in-game soundtracks exhibit similar interplay between music and sound effects. Koji Kondo's *Super Mario Bros.*(Nintendo, 1985) theme is a case in point and appears to "duck" in volume as Mario jumps, for instance.

Notwithstanding the deleterious and unpredictable impact of player-triggered spot effects, Hubbard et al. devised a series of compositional and sound design strategies to deal with the limited polyphony and enrich the complexity of their soundtracks. "Most of it was simply done by multiplexing the three channels. If the lead line has two beats rest, put a fill or some effect in there" (Hubbard, n.d.). Hubbard's use of the term *multiplexing* is interesting here and while it attempts to achieve something of the same audible effect it technically differs from Yannes' use in relation to sharing the SID's oscillator to generate a greater number of simultaneous voices. What Hubbard refers to here is a combination of compositional and sound design techniques that make use of a specific and distinctive affordance of SID's oscillators. Put simply, rather than assign a channel to a specific sound or musical part that plays throughout the piece (such as a bassline, lead guitar, and drums, as with the ensemble approach to writing on the NES), any given musical sound might be performed on any of the three channels depending on their "availability". As Troise (2015) observes, the composer "looks for 'gaps' in the melody in which to fill out the composition."

Accordingly, while the channels might primarily be dedicated to particular duties with drums and percussion predominantly performed on channel 3, for instance, additional drum fills and percussion flourishes might be added to channels 1 and 2 throughout the piece depending on their availability and the musical sense such additions would make. *Commando* is a case in point, with percussion spread, or multiplexed, across all three channels. Figure 1 shows a waveform display of the three SID channels with channel 1 (top) predominantly handling the lead melodic line, channel 2 (middle) performing pseudo-polyphonic chordal backing through rapid arpeggiation, and channel 3 (bottom) covering the bassline. As such, no single channel is responsible for the drums and percussion, which are a notable feature of the soundtrack's aesthetic. Instead, the rhythm track is performed across all three voices with noise and toms on channel 1, ring modulated percussion on channel 2, and a snare on the offbeat performed on channel 3. Notably, there is no kick drum on the downbeat and channel 3's bassline effectively covers this function alternating with the snare to provide the foundation of the rhythm section.

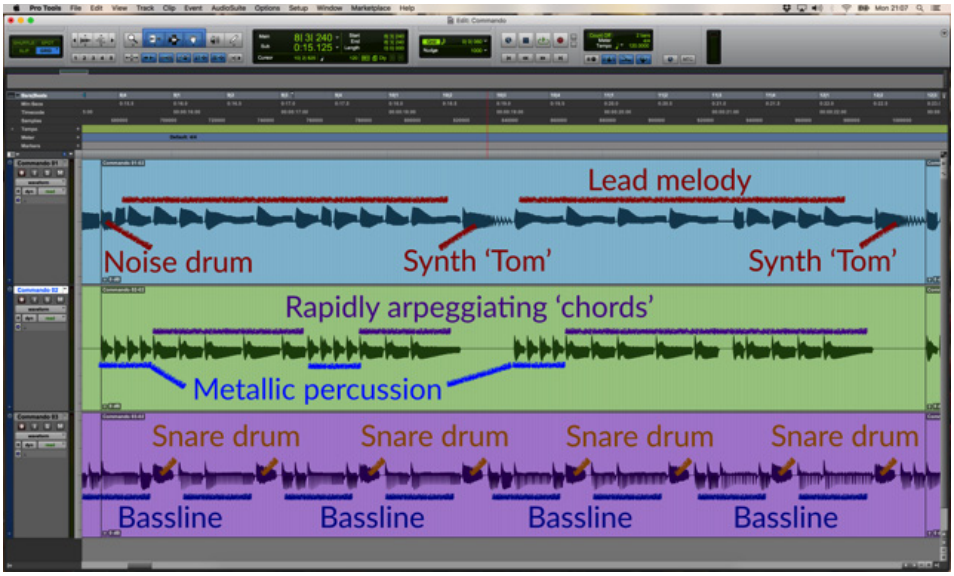


Figure 1: Multiplexing drums and percussion in Rob Hubbard‚Äôs Commando. Photo courtesy of the Author.

Examining *Commando* in this way, it is clear that there is very little sonic or compositional space remaining that speaks eloquently to the specificity of this as a composition for the SID chip, as well as reminds of the characteristic harmonic and rhythmic intensity of this chip music aesthetic. Drawing attention both to the specificity and historical continuity of these techniques, Guay and Arsenault (2012) note that chip composers often deploy the kinds of extreme ornamentation characteristic of the Baroque. As Akesson (2011) amusingly puts it, this is the compositional equivalent of a restless child who cannot sit still.

For Collins (2006a), the emphasis on building textures through the combination of rhythm elements is similarly characteristic of the C64 aesthetic. Certainly, it is not only in *Commando* that we find Hubbard's extensive use of pitched percussive instrumentation often using ring modulation and simultaneously performing rhythmic and contrapuntal duties. *The Last V8* (Mastertronic, 1985) and *Sanxion* (Thalamus, 1986), for instance, are both deeply complex, textural works. Heard in isolation, some of *Sanxion*'s heavily ring modulated lead and supporting lines sound utterly cacophonous; yet, in concert, they gel together reminding us of the novel, often transgressive character of game sound (Cheng, 2014) and of the integrated composition and sound design of these works as "SID music", rather than as transcribed pieces or even as "chip music".

The modulation of the SID chip's oscillators is found at work elsewhere albeit at an altogether different periodicity. Where the alternating bass-percussion patterns we have noted above allow for compositional multiplexing, accelerating the modulation of waveform output to the speed of the system clock allows for new sound design techniques and the creation of complex instrumentation. As we note in McSweeney's disassembly of Hubbard's driver, part of the Instrument setup is dedicated to create just this effect:

> "Bit#0 signals that this is a drum. Drums are made from a noise channel and also a fast frequency down, with fast decay. Bass drums use a square wave, and only the first 50th of a second is a noise channel. This is the tell-tale instrument that gives away a Rob Hubbard routine! Hihats (sic) and other drums use noise all the time" (McSweeney, 1993)

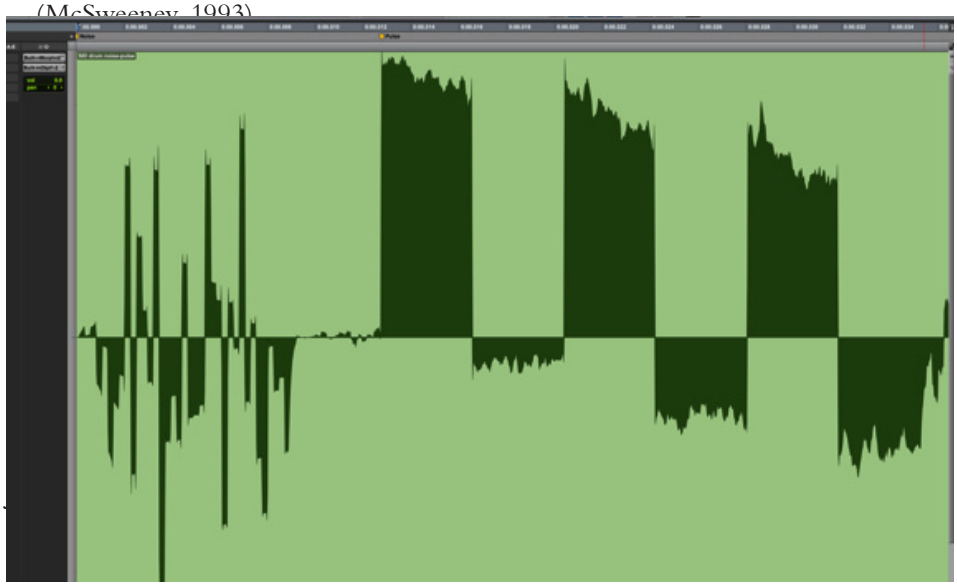

Figure 2: SID chip drum sound design showing dynamic waveform manipulation. Photo courtesy of the Author.

Analysing Hubbard's SID compositions, one can identify an extensive use of frame-accurate manipulation of waveforms. In addition to those sounds obviously present as discrete "drums", compositions such as *Crazy Comets* and *Sanxion* add a short (as little as 1/50th of a second) burst of noise to a number of their otherwise pitched instruments. The result of this is to add an additional percussive sound running concurrently with the pitched element. In *Sanxion*, channel 3's pulse wave bass instrument is augmented with digital noise in its attack. The 16th-note sequence thus appears to create both a synthetic bassline and a continuous shaker or hi-hat sound from a single channel.

Resonating with Hubbard's technique, Martin Galway notes: "I just tried to make sure all 3 channels were getting used. A couple of techniques allowed it to sound like more was going on, like fast arpeggios, and chorusing/echoes." (Galway n.d.) The provision of arpeggiation in Hubbard's driver as we noted above speaks to another commonly utilized technique designed to compensate for the SID chip's limited polyphony, but that also makes use of its ability to be clocked at high rates. Because using the three channels for chordal composition is inefficient, composers often deployed rapid arpeggios cycling around two or more notes at 50Hz to simulate the effect of multiple notes playing simultaneously. As Akesson (2011) notes, while the necessity for the chip music composer's use of the technique is different and derives from interaction between the execution of code and television standards and refresh rates, the simulation of polyphony

through rapid arpeggiation and large interval skips was known to Bach and evident in the final movement of the *Partita No. 2 in D Minor*, BWV 1004.

For the chip composer, the combination of rapid arpeggiation, along with other effects such as pulsewidth modulation, creates an effect that not only harmonically rounds out compositions, yet also adds dynamism and animation. The sonic effect of this technique is a chordal "warbling" that is, perhaps more than any other, characteristic of chip music. As with the drum sound design example above, it is crucial to remember that the rate of this arpeggiation is in no way musically derived but, rather, is a function of the C64's processor and screen-update interrupt-system, which is taken advantage of in the driver. As with the other examples of sound design and composition we have noted above, it is the singular interaction between musical intention and opportunity, and the technical capability of SID (as a discrete device but also part of a larger computing system) that collides in the driver and gives rise to the identifiable quality and personality.

For the chip composer, the combination of rapid arpeggiation, along with other effects such as pulsewidth modulation, creates an effect that not only harmonically rounds out compositions, yet also adds dynamism and animation. It is the singular interaction between musical intention and opportunity, and the technical capability of SID (as a discrete device but also part of a larger computing system) that collides in the driver and gives rise to the identifiable quality and personality.

## LET THERE BE (MORE) DRUMS

Although Hubbard's driver included sound design routines for synthesizing drum sounds, it was the discovery of a glitch in the SID chip that would transform the performance of percussion on the C64.

> "I figured out how samples were played by hacking into someone else's code… I had no equipment for editing samples though, so my program synthesized the drums as a series of farts and burps! Later I was able to acquire some proper drum samples and by "Game Over" it got quite sophisticated" (Galway n.d.).

The ability to replay samples exploits an inconsistency arising from the chip's design and fabrication which gives a *DC offset* between the channels. By rapidly manipulating the chip's amplitude registers, it is possible to generate otherwise undesirable audible clicks that when clocked at the right rate and with suitable data can create a form of *Pulse Code Modulation* (Collins, 2006a). Importantly, the "fourth" voice is actually a phantom that appears across at the SID's output stage as a result of the interaction of the other channels. As the exploitation of a glitch codified and transformed into process in the driver, it is perhaps the best example of the interplay between technical and creative exploration that, in this case, yields affordances not only hidden but unintended. Indeed, the subsequent "8580" revision of the SID removed the DC offset making sample playback all but inaudible on machines equipped with the newer sound chip. As such, the affordance was not simply re-hidden but eradicated altogether.

Hubbard's workflow reveals the tight integration of coding, arrangement, and composition. The process is not merely about transcribing musical ideas into Hexadecimal code. As Troise (2015) notes, it is the difference between "…trying to recreate externally composed music on the PSG [and] writing music *for*the PSG" (original italics). What we see in Hubbard's driver is not only writing directly *to* the silicon by manipulating the 8-bit registers but writing directly *for* the silicon––effectively constructing the SID as a synthesizer in a case. Commenting on the longer-term influence of these techniques, composer Neil Baldwin notes that:

> "Another old C64 trick I employed a lot in James Bond Jr (and continued to do so for almost all the other NES projects I worked on) was to simulate echo but just using a single voice. Instead of using two voices, one playing a melody and the second at a lower volume but slightly delayed, the single-voice method used slightly truncated note lengths and in the gaps in between notes, play the same notes shifted later in time, quickly dropping the voice volume (and then restoring it to play the next melody note)" (Baldwin, 2009).

Exploration, and the dialogue between musical intention and utility, and technical capability and exploitation were central to the development of the driver. This was partly because, according to Yannes, the documentation was written before the completion of SID prototypes let alone finished silicon, thereby rendering it unintentionally inaccurate (in Bagnall, 2011). However, experimentation was also rewarded because the SID's registers behaved in sometimes unpredictable ways modified in combination or in ways that musicians found more useful than the chip designers who had masked such combinations or left them undocumented. As Hubbard (2002) suggests, the search for a distinctive sound, effect or technique, remained an ongoing process and drove experimentation and the refinement of the driver.

> "We were always looking for ways to try to find something that the machine could do that it really wasn't designed to do. You'd look in the manual on the C64 and it would say things like… don't set this bit whatever you do and we'd say "OK, I don't care, I'm going to try setting the bit and see if it does anything". We were always looking for ways to squeeze more out of this thing by doing things in assembler and tweaking around" (Hubbard, 2002).

Indeed, even Commodore's own *Programmer's Reference Guide* appears to endorse an investigative strategy that perhaps belies the lack of a specification covering the full range of SID's capabilities. "Only through experimentation on your own will you fully appreciate the capabilities of your machine. The examples in this section of the Programmer's Reference Guide merely scratch the surface". (CBM, 1982: 208).

## SID TODAY

The SID chip remains a popular device today with fan projects curating music files, databases and emulator applications to replay the work of Hubbard, Galway *et al*. The SID chip is also available to contemporary musicians in a number of forms and is a cornerstone of the contemporary chiptune scene or subculture (see Marquez, 2014; Bittanti, 2009; McLaren, 2003). Software emulations such as chipsounds and QuadraSID seek to replicate the chip's distinctive features either through sampling or circuit modeling (see Viens, 2012) while devices such as Therapsid and the SidStation eschew emulation and connect an original SID chip to a modern, tactile interface. Regardless of the approach, it is clear that each device whether software plugin or MIDI-equipped hardware, these modern SID implementations may be integrated into contemporary music production workflows in ways simply impossible in the 1980s. In one sense, Yannes' original vision for the SID chip as a more mainstream instrument is realized. However, while the accessibility of the chip is dramatically increased, other facets are diminished. Each new device attempts to replicate some of the features discussed above. The chipsounds plugin, for instance, incorporates a wavetable into each instrument patch. This allows per-cycle waveform and pitch variation thereby simulating the warbling chords so characteristic of C64 music and which may be triggered with a single key. However, with a fixed arpeggio per patch, and the contemporary DAW (Digital Audio Workstation) workflow's dissociation of the sound generation (plugin) from the compositional environment (the sequencer), the ability to continuously vary the groups of pitches over time becomes altogether more complex and comparatively impossible in the live performance contexts that these plugins and hardware repackagings also facilitate. Similarly, the ability to generate the virtual fourth channel is lost without the ability to manipulate the DC offset of the SID chip's output.

What we see in these modern interfaces to the SID chip is certainly a stylistic recovery of the fundamental oscillator sound, its routing and filter characteristic, combined with a degree of control over some key synthesis features, that integrates into contemporary workflows. However, […] there is far more to the SID chip. Its characteristic sound is also a product of the way it was harnessed (or in the terms of Elektron, "encased") by composer-programmers such as Hubbard and Galway

In light of such developments, while it is possible to find increasingly easy ways to access to the chip, they often offer a narrower set of the chip's capabilities than are available by directly writing to its data registers at 50Hz. In these accessible reworkings, certain features of the chip are necessarily curtailed, and many of its once revealed affordances become hidden again. What we see in these modern interfaces to the SID chip is certainly a stylistic recovery of the fundamental oscillator sound, its routing and filter characteristic, combined with a degree of control over some key synthesis features, that integrates into contemporary workflows. However, as important as the raw tonality of the

oscillators, the design of the envelopes, or the non-linear distortion characteristics of the filter, might be in fundamentally distinguishing the SID chip from other soundchips (as Altice rightly notes), the analysis above has demonstrated that there is far more to the SID chip. Its characteristic sound is also a product of the way it was harnessed (or in the terms of Elektron, "encased") by composer-programmers such as Hubbard and Galway. It is through the production of their bespoke driver software that the technical affordances are inexorably bound to the connected processes and principles of composition and synthesis. The driver is informed by both musical and sound design priorities along with a knowledge of the chip's affordances, as shown in the (imperfect) documentation and via the exploitation of its documented and undocumented capabilities, flaws and quirks. Controlled by Rob Hubbard's driver, the SID chip is constituted as a specific, bespoke compositional platform, related to, but different from its existence under the control of another driver or interface. By revealing and making musically usable what was designed (and, unintentionally, hidden) in the silicon and the 29 8-bit registers, the driver constitutes the SID chip as a complex suite of perceived affordances.

**REFERENCES**

*(all electronic resources and URLs accessed September 2017)*


Akesson, L. (2011) "Elements of Chip Music". *Revision 2011*, Saarbrücken, Germany. http://www.linusakesson.net/music/elements/

Altice, N. (2015) *I Am Error: The Nintendo Family Computer / Entertainment System Platform*. Cambridge, MA: The MIT Press.

Bagnall, B. (2011) *Commodore: A Company on the Edge*. Winnipeg: Variant Press.

Baldwin, N. (2009) "James Bond Jr (Eurocom/THQ 1991)". *DutyCycleGenerator* (29 March 2009) http://dutycyclegenerator.com

Barton, M. & Loguidice, B. (2007) 'A History of Gaming Platforms: The Commodore 64'. *Gamasutra*. http://www.gamasutra.com/view/feature/1991/a_history_of_gaming_platforms_the_.php/

Beige 0036 (2001, October, 19) http://www.post-data.org/beige/beige_make.html

Bittanti, M. (2009) 'So, When did Planned Obsolescence become an Artistic Practice?' In D. Quaranta (Ed.) *Playlist*, Gijón, Spain: LABoral: pp. 32-36

Byte (1995) 'Most Important Chips'. *Byte,* September 1995: pp.74-75.

Carlsson, A. (2008) 'Chip music: low-tech data music sharing'. In K. Collins (Ed.) *From Pac-Man to Pop Music: interactive audio in games and new media*. Aldershot: Ashgate, pp153-162.

CBM (1982) *Programmer's Reference Guide*, Indianapolis, Indiana: Commodore Business Machines, and Howard W. Sams & Co.

Cheng, W. (2014) *Sound Play: Video Games and the Musical Imagination*. Oxford: OUP.

CHM (2007) 'Commodore 64—25th Anniversary Celebration'. http://www.computerhistory.org/events/video/75/

Colbeck, J. (1985) *Keyfax: Julian Colbeck's Guide to Electronic Keyboards*. London: Virgin Books.

Collins, K. (2006a) 'Loops and bloops: Music of the Commodore 64 games'. *Soundscapes*, Volume 8. http://www.icce.rug.nl/~soundscapes/VOLUME08/Loops_and_bloops.shtml

Collins, K. (2006b) 'Flat Twos & the Musical Aesthetic of the Atari VCS'. *Popular Musicology Online*, Issue 1 (Musicological Critique). http://www.popular-musicology-online.com/issues/01/collins-01.html

Collins, K. (2007a) 'In the Loop: Creativity and Constraint in 8-bit Video Game Audio'. *Twentieth Century Music*, 4(02): pp. 209-227.

Collins, K. (2007b) 'An Introduction to the Participatory and Non-Linear Aspects of Video Games Audio', In S. Hawkins & J. Richardson (Eds.) *Essays on Sound and Vision*. Helsinki: Helsinki University Press: pp.263-298.

Collins, K. (2008) *Game Sound: An Introduction to the History, Theory and Practice of Video Game Music and Sound Design*, Cambridge, MA: The MIT Press.

Danet, B. (2001) *Cyberpl@y. Communicating Online*, Oxford: Berg.

Driscoll, K., & Diaz, J. (2009) 'Endless loop: A brief history of chiptunes'. *Transformative Works and Cultures*, No. 2. http://dx.doi.org/10.3983/twc.2009.0096

Elektron (1999) *SidStation Owner's Manual rev 2.2b*. https://www.elektron.se/wp-content/uploads/2016/05/Elektron_SID_Users_Manual_r22b_OS1.1.pdf

Franklin, P. (2011) *Seeing Through Music*, New York: Oxford University Press.

Fritsch, M. (2012) 'History of Video Game Music'. In Moormann (Ed.) *Music and Game: Perspectives on a Popular Alliance*, Springer VS: pp.11-41.

Gazzard, A. (2016) *Now the Chips are Down: The BBC Micro*, Cambridge, MA: The MIT Press.

Gibson, J. (1979) *The ecological approach to visual perception,* New York: Houghton Mifflin.

Guay,L-M. & Arsenault, D. (2012) 'Thumb-Bangers: Exploring the Cultural Bond between Video Games and Heavy Metal'. In A. R. Brown and K. Fellezs (Eds) *Heavy Metal Generations*, Oxford: Interdisciplinary Press, pp. 105-115.

Herber, N. (2008) 'The Composition-instrument: emergence, improvisation and interaction in games and new media'. In Karen Collins (Ed.) *From Pac-Man to Pop Music: Interactive Audio in Games and New Media*, Aldershot: Ashgate: pp. 103-123.

Horowitz, S. & Looney, S. (2014) *The Essential Guide to Game Audio: the theory and practice of sound for games*, Abingdon: Focal Press.

*High Voltage SID Collection* (*HVSC*) (n.d.). Retrieved from www.hvsc.c64.org

Judd, S. (1996) 'SID Primer: The Working Man's Guide to SID'. *disC=overy: The Journal of the Commodore Enthusiast*, Issue 2 (1 October 1996). http://codebase64.org/doku.php?id=magazines:discovery2

Laing, G. (2004) *Digital Retro: The Evolution and Design of the Personal Computer.* Lewes: Ilex.

Maher, J. (2012) *The Future Was Here: The Commodore Amiga.* Cambridge, MA: The MIT Press.

Marquez, (2014) 'Playing new music with old games: The chiptune subculture'. *G|A|M|E Journal* 4: 67-79.

McLaren, M. (2003) '8-Bit Punk'. *Wired* (1 November 2003). available https://www.wired.com/2003/11/mclaren/

McSweeney, A. (1993) 'Rob Hubbard's music: Disassembled, commented and explained'. *C=Hacking*, No. 5 (March 7). http://www.ffd2.com/fridge/chacking/c=hacking5.txt

Money, S. (1984) *Commodore 64 Graphics and Sound.* London: Granada Technical Books.

Montfort, N. & Bogost, I. (2009) *Racing the Beam: The Atari Video Computer System.* Cambridge, MA: The MIT Press.

Moulthrop, S. (2004) 'From Work to Play: Molecular Culture in the Time of Deadly Games'. In N. Wardrip-Fruin & N. Harrigan (eds) *First Person: New Media as Story Performance, and Game.*Cambridge, MA: MIT Press: pp. 56–70.

Murray, J. (2011) *Inventing the Medium: Principles of Interaction Design as a Cultural Practice.*Cambridge, MA: The MIT Press.

Norman, D. (1999) *The Design of Everyday Things.* New York: Basic Books.

Petersen, T. (2006) 'Music Recollection'. *Recollection*, Issue 2. http://www.atlantis-prophecy.org/recollection/?load=online_issues&issue=1&sub=article&id=11

Pinch, T. & Trocco, F. (2002) *Analog Days.* Cambridge, MA and London: Harvard University Press.

Raymond, E. S. (1996) *The New Hackers' Dictionary.* Cambridge, MA: MIT Press.

Schartmann, A. (2015) *Koji Kondo's Super Mario Bros. Soundtrack.* London: Bloomsbury Academic.

Slocum, P. (2003) *Atari 2600 Music And Sound Programming Guide*. http://qotile.net/files/2600_music_guide.txt

Stevens, R. & Raybould, D. (2016) *Game Audio Implementation: A Practical Guide to Using the Unreal Engine.* New York: Focal Press.

Tognon, S. (2003) 'Martin Galway's Arkanoid Music Routine'. *SIDin*, Issue 4: pp. 23-98 http://digilander.libero.it/ice00/tsid/sidin/SIDin4.zip

Troise, B. (2015) 'Compositional Strategies For Programmable Sound Generators With Limited Polyphony'. http://www.ludomusicology.org/2015/07/16/compositional-strategies-for-programmable-sound-generators-with-limited-polyphony/

Vail, M. (2014) *The Synthesizer.* Oxford: Oxford University Press.

Viens, D. (2012) *chipsounds User's Guide (v1.877).* https://s3.amazonaws.com/chipsounds/chipsounds_guide.pdf

Vogel, J. & Scrimshaw, N. (1983) *The Commodore 64 Music Book: A Guide to Programming Music and Sound.* Boston: Birkhaüser.

Wolf. M. (2012) *Before the Crash: Early Video Game History.* Detroit, MI: Wayne State University Press.

Zzap!64 (1986) '*Kentilla* review'. *Zzap!64*, (June 1986): pp. 73-74.

**INTERVIEWS**

Carr, N. (2001a) 'An Interview with Chris Huelsbeck'. *Remix64.* http://www.remix64.com/interviews/interview-chris-huelsbeck.html

Carr, N. (2001b) An Interview with Martin Galway. *Remix64. http://www.remix64.com/interviews/interview-martin-galway.html*

Carr, N. (2001c) 'An Interview with Ben Daglish'. *Remix64*. http://www.remix64.com/interviews/interview-ben-daglish.html

Flat Four (2005) 'Programme 3: Commodore Music'. *Flat Four Radio*. http://www.mcld.co.uk/flatfour/chiptunes/commodore/

Galway, M. (n.d.) 'Martin Galway interview'. *C64.com*.http://www.c64.com/interviews/galway_part_2.html

Hubbard, R. (2002) 'The Golden Days of Computer Game Music'. *Assembly 2002*, Helsinki (video). https://www.youtube.com/watch?v=DiPdjbsiQqM

Hubbard, R. (n.d.) 'Rob Hubbard interview'. *C64.com*. http://www.c64.com/interviews/hubbard.html

Varga, A. (1996) 'Progenitor of the SID: An interview with Bob Yannes'. *disC=overy: The Journal of the Commodore Enthusiast*, Issue 2 (1 October 1996). http://codebase64.org/doku.php?id=magazines:discovery2

## LUDOGRAPHY

*Commando* (1985) Elite, C64.

*Commodore Music Maker* (1982) Commodore, C64.

*Crazy Comets* (1985) Martech, C64.

*Delta* (1987) Thalamus, C64.

*Game Over* (1987) Imagine, C64.

*Helikopter Jagd* (1986) Ocean, C64.

*Pressure Cooker* (1983) Activision, VCS.

*Monty on the Run* (1985) Gremlin Graphics, C64.

*One Man and his Droid* (1985) Mastertronic, C64.

*Roland's Ratrace* (1985) Ocean, C64.

*Sanxion* (1986) Thalamus, C64.

*Spellbound* (1986) Mastertronic, C64.

*Super Mario Bros.* (1985) Nintendo, NES.

*The Human Race* (1985) Mastertronic, C64.

*The Last V8* (1985) Mastertronic, C64.

*Thing on a Spring* (1985) Gremlin Graphics, C64.

## AUTHOR'S INFO:

James Newman is Professor of Digital Media at Bath Spa University and a researcher and curator at the UK's National Videogame Arcade. He has published widely on videogames, game history and culture, and game preservation and has curated major exhibitions at the NVA. He is currently writing books on gameplay spectatorship and on early videogame sound and music.