Musica ex machina: a history of video game music. Presented at: In Cahoots Conference, Aberdeen, 23-24 October 2015.

Author Dr Kenny McAlpine, Abertay University

Good morning everybody. My name's Kenny, and I work at the University of Abertay in Dundee, where I head up teaching and research on game audio, and I'm here today to talk to you about game music. Now I'm guessing from the fact that you're all here, rather than next door finding out about new music and theatre, that you're all at least interested in interactive screen media, and some of you might also enjoy playing video games. Could I just see a quick show of hands? How many gamers do we have in the audience?

Great. Now today, what I'd really like to focus on is the role the music plays in video games, and some of the challenges and opportunities that presents for composers. We're going to begin that story right back at the dawn of video gaming, because much of what we think of as video game music was born out of the technical limitations of the sound chips that were installed in the first generation of home computers and game consoles, and it was here that many of the gaming codes and conventions, as well as working practices were established.

But, if technical constraint was a driver for creativity, so too was its reciprocal, technological progress. We'll look at, first of all, how FM synthesis and wavetables replaced programmable sound generators, using MIDI to encode performance, and how this, in turn was replaced by audio streaming from CD-ROM and ultimately high fidelity adaptive music loops, which allows for interactive music tracks that sync to gameplay and reconfigure themselves in real-time in response to player input.

We'll spend a bit of time unpicking this, because adaptive game soundtracks are something of a hot topic at the moment, and there are some very interesting open questions that — I think — need musical rather than technical solutions. Interestingly, though, the industry has come round almost full circle, and we have a growing and vibrant indie scene typified by small development teams and quick turnarounds on lo-fi products that focus on engaging gameplay over flashy graphics and sound. The days of MIDI soundtracks and chiptunes are back, and, for the moment, at least, they look like they're here to stay.

We'll round off by looking at routes into the industry, followed up by a Q&A. I can't guarantee to have answers to your questions, but I'll certainly have a go.

Let's begin, however, with a little bit about me.

I started my musical life as a classical pianist, and along the way, picked up some clarinet, guitar, bass, drums, clawhammer banjo, ukulele, bagpipes and kazoo. I studied piano through to diploma level, but decided to study maths at uni, primariliy because I wanted to keep music as something solely for enjoyment — there are few things as satisfying as bashing the hell out of a piano after a hard day. I'm not sure I would have had quite the same sense of release if I'd been bashing the hell out of a piano throughout the day too.

In the end, I decided to combine my love of the two things, bringing together maths and music for my PhD, with a project on algorithmic composition. During that time, I entered a competition run by the then Scottish Games Alliance to find the best amateur games developers in the UK, and won the audio section. Off the back of that, I began to pick up some freelance and contract work, scoring some PC and Playstation games, and a few demos for E3, and landed my current job at Abertay.

Most of my day job is taken up with teaching, but my research interests cover everything from algorithmic and procedural music, through tactile and enactive musical interfaces to heritage applications of music technology, and I keep an active interest in professional practice, having written interactive scores for games, promenade theatre, and producing live radio for the BBC. What I'm going to try and do is distill some of that experience down and give you a perspective on the point where gaming and new music overlap.

In the beginning

It all started with Pong.

Pong wasn't the first arcade game. It wasn't even the first arcade game with sound. But it managed to muscle its way into the public consciousness in a way that none of its predecessors had. Before Pong came along, the domain of the video game was the university computing lab, because the hardware required to run games ran to tens or hundreds of thousands of dollars.

Al Alcorn, the engineer hired by Atari to develop their new game, by contrast, bought a black-and-white television for $75 from Walgreens and added some crude sound effects. His bosses had wanted the roar of a crowd of hundreds, but Alcorn had neither the electronic components or the knowhow to create them. Instead, he poked around inside the sync generator and repurposed the waveforms that were already there.

<PLAY CLIP OF PONG>

So those sounds that came to define the voice of gaming, at least in those early days, happened — essentially — by accident.

As early as 1975, Atari had begun work on a home console, which launched in 1977 as the VCS, although you might know it better as the 2600. Back then, RAM was phenomenally expensive — tens of thousands of dollars per megabyte, in fact — and so to keep costs down, the engineers had to devise a way of minimising the memory requirements, and they looked to the display. Most consoles and computers use something called a framebuffer, an area of RAM to cache a full image of the screen display in memory before uploading it in its entirety to the display. Even for a relatively small 128 x 64 monochrome display, the VCS would have needed a 1kB framebuffer, which would have pushed the cost of the device beyond its target price point. The solution was a graphics chip that removed the need for a framebuffer completely by building the image directly, a line-at-a-time, on the cathode-ray television (CRT) displays that the VCS was designed to interface with.

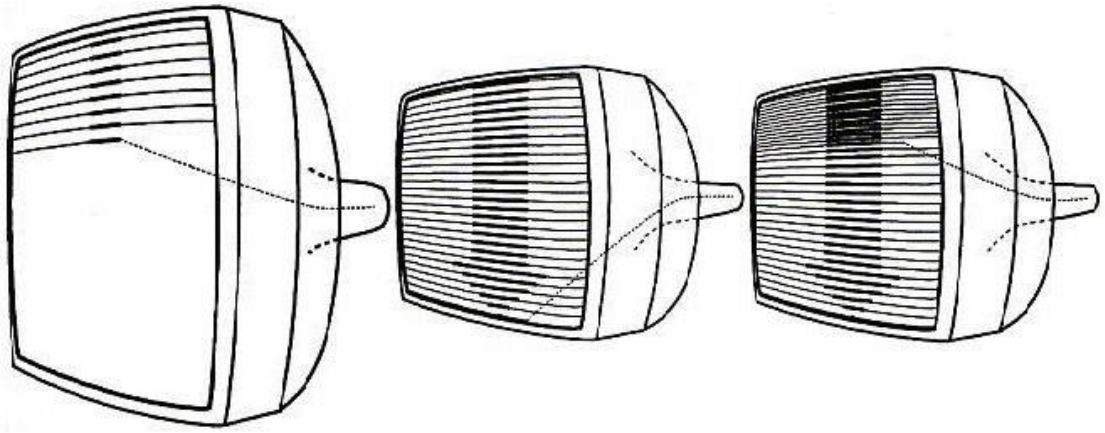

Figure 1 - Electron gun builds up the image a line at a time

At the heart of a CRT is an electron gun, which fires a stream of electrons towards a phosphorescent screen, which glows to create the visible picture. The screen image isn't drawn all at once, but line-by-line, as the electron gun pans from side to side across the screen. At the end of each line, the electron beam is turned off, and the gun returns to its starting position ready to fire the electrons to the next line of the image. The VCS's graphics system built each frame of the display like this, line-by-line, directly within the software. The cost saving on video memory shifted the responsibility for the video display from hardware to the game code, using the horizontal and vertical blanks, the time the electron gun is turned off as it transits to the start of each line and the top of the screen, as downtime to carry out all of the other game processes — including sound and music — in the fraction of a second while the gun reset.

Now you might be asking yourself, what has all this got to do with music? Well, on this chip, alongisde this complex display mechanism were two independent audio circuits, also synced to the display, whose output was multiplexed with the display components and sent to the television speakers via an RF modulator. This meant that the sounds that the VCS could create were actually tied to the display.

The oscillators were pretty basic, providing just 1-bit of resolution, but by sending the chip complex sequences of 1s and 0s, generated using something called a linear feedback shift register, they were capable of producing a range of different pitched and noise tones.

The rate at which these sequences were generated was sync'd to the horizontal video sync counter. A PAL system, with 312 video scanlines operating at 50 Hz gave a system clock of 31.2 kHz, while NTSC systems, which had 262 scanlines operating at 60 Hz gave a system clock of 31.44 kHz, and so PAL consoles would play slightly flatter than NTSC consoles.

This clock frequency was then sent to a frequency divider circuit, which used a 5-bit frequency value, giving 32 possible pitch values.

The actual frequency of each tone depended on two factors, the frequency register and the length of the bit sequence — longer sequences increase the period of the wave and reduce the frequency of the tone. This table shows the tuning chart for one of the VCS's tones, relative to true pitch. What it shows is that although there were, in principle, 32 pitch values to choose from, some of these were so badly out of tune that they couldn't really be used for music — the VCS sound chip was really intended more for effects than for musical expression.

So in exactly the same way that the composers of the renaissance understood the limitations of the meantone tuning system, and worked to its strengths, avoiding pitches and key signatures that sounded dreadful, Atari musicians quickly learned to use musical patterns that avoided notes that sounded 'off'. In other words, the limitations of the system began to directly influence the aesthetics of the game music.

In fact, during a presentation at the Game Developer Conference in 2011, David Crane explained that developer of Activision's Pressure Cooker, Garry Kitchen, calculated the nine notes that could be reasonably approximated by the VCS, and marked them on a Casio keyboard. They hired a professional jingle composer to write a melody for the game using only the marked notes. The two-channel intro is played during the game's title screen, with a single-channel two-bar bed looping continuously during the game, leaving the other channel free to play sound effects.

<PLAY CLIP FROM PRESSURE COOKER>

The Atari thrived on arcade ports, and one of the best examples, from a musical point of view is Atari's port of Namco's classic, Dig Dug, which sees the titular character tunnelling in a subterranean world, fighting the fire-breathing, dragon-like Fygars and Pookas, cute little creatures that resemble beachballs in sunglasses, with little more than a bicycle pump.

The music in the arcade version begins with a catchy intro sting as Dig Dug enters the playfield from the top right of the screen and tunnels down to begin play.

<PLAY CLIP FROM ARCADE DIG DUG>

Both the arcade and the VCS versions tie the in-game soundtrack to the player movements, playing the music back only when the player moves the joystick, and vamping with a warning siren sound when the player stands still for any length of time. There is one important difference, though. The arcade music maintains musical continuity, picking up the theme from where it left off when the player moves. The VCS version restarts the theme from the beginning.

<PLAY CLIP FROM VCS DIG DUG>

There is a real art to arranging music like this. It's a minimalistic distillation of the essence of the original track into just one or two tracks, and I think Dig Dug manages to capture much of the spirit of the original given the limitations of the hardware. Although it simplifies the original music, it manages to hit the Goldilocks spot, neither so much simplification that the musical detail and character of the original is lost, nor so little that the tracks are overly complex and too taxing for the system to handle — the music is just right for the chip. It uses both channels of the TIA to maintain a continuous two-channel soundtrack throughout the game, maintaining the sense of driving bass line and chirpy melody from the arcade, using 'voice-stealing' to handle the sound effects.

The ZX Spectrum came along in 1982, and took Britain by storm. Like the Atari, it had just 1-bit of resolution, but it was more restrictive, it had just one channel of sound, and to compound matters, there was no sound hardware, all of the sound was handled by the main processor, so while the Spectrum was beeping, it couldn't, without some clever coding, do anything else. So on the face of it, it looked like in-game music wasn't possible, because playing a tune during the game would have left no processor time for any gameplay.

One of the first games to challenge that idea was Manic Miner, released in 1983. The game begins with this, delightfully clangorous, and — if I'm honest — barely recognisable rendition of Strauss's Blue Danube.

<PLAY MANIC MINER TITLE SCREEN>

You'll notice, though, that this is quite a different — a more complex sound — than the simple square or pulse waves you might expect. What programmer Matthew Smith realised was that by toggling the speaker on and then off again in quick succession, he could create rich, harmonic tones. This is called a binary impulse train, and it was implemented by using a timer, and toggling the speaker whenever it ticked down to 0. He realised that if he ran two timers simultaneously, he could create two interleaved impulse trains, and create two channels of sound. For single notes, he created a phasing effect by interleaving two timers set to very similar values. For chords, he used two separate timers. Now you can hear that the effect isn't terribly successful on the chords. The reason for that is that there is no amplitude enveloping, and the harmonic content of the sound is such that the individual tones contain all harmonics at equal strength, meaning that there's no real sense of a dominant fundamental to help us determine pitch. The result is a complex buzz. However, it's an important moment in game music, because it showed that the Spectrum could achieve feats of musicality that it was never designed to do, and the technique was refined over time, particularly by Tim Follin, who eventually managed to squeeze five or six channels of music from the Spectrum:

<PLAY CHRONOS>

More successful was the in-game music, appropriately enough, In the Hall of the Mountain King from Greig's Peer Gynt:

<PLAY MANIC MINER IN-GAME MUSIC>

To make this work, Smith used an interrupt in the main game loop. Every time it executed, he played a fragment of a note for a fraction of a second — 20ms to be exact. In the remaining 70ms of the game loop, all of the other game code was executed to update the screen, read the player input, move the characters, check for collisions and so on. In effect, Smith was using granular synthesis to granularise the music sequentially, relying on our brains to fill in the gaps between the sound grains.

<SHOW MANIC MINER THEME>

Smith went one step further, though. He realised that he could run sound effects alongside the music by similarly granularising those and slotting them into the spaces between the music grains.

<SHOW MANIC MINER THEME WITH SOUND EFFECT>

It's a novel solution, but, of course, it's not without cost, because the effect of breaking down the sound into grains like that gives it a disjoint, sort of bubbly quality. That was the sound that came to define the sound of Spectrum music.

The Commodore 64 was a very different beast. It had a sound chip called the SID, which was more like a hardware synth with a free computer built around it. In fact, it was designed by Bob Yannes, who went on to found Ensoniq. It was far more capable than anything that had been included in a console or computer before, with proper ADSR envelopes and filters on top of multiple waveforms. However, it had only three channels of sound, so it was still something of a challenge to create rich arrangements.

Martin Galway, one of the first video game career musicians, created a technique that simulated chords on one channel by rapidly arpeggiating the notes of a chord in sequence, collapsing down perhaps three or four channels into one, in the same way that bouncing down tracks on a portastudio could be used to free up tracks for use later. He first used this technique on the raggy soundtrack to the Donkey Kong clone, Kong Strikes Back!, and the sound very rapidly became a fundamental part of the video game music sound.

<PLAY KONG STTRIKES BACK>

All of this is a very truncated history of the early days of video game music, but it's important because it demonstrates two key things: Firstly that video game music, like film music, is functional. Its form and structure, and, to an extent, the way it sounds, is determined by factors outside the music itself: the hardware platform, the game code, the gameplay experience. Secondly, game music has always been a music that is defined by constraint, but as a result, there is huge potential, even now, to innovate and be creative, both with the music itself and also how it is implemented.

The modern era

If technological constraint was a driver for creativity in the sense that it forced people to innovate to overcome limitations, then its reciprocal, technological progress was too, by giving access to new techniques and higher resolution. The first wave of SoundBlaster cards had onboard FM synthesis, which was quickly superseded by higher quality wavetable synths, which could be controlled by MIDI, making game composition more accessible — before it had been very much a programmer's job — and also allowing for the music to be more complex and more musical.

With the arrival of the Sony Playstation in the mid-1990s, it became possible to stream music directly from CD-ROM, and all of a sudden game designers fell over themselves to license big name artists on soundtracks. Wipeout, one of the Playstation launch titles, featured a high-octane electronic soundtrack that was mostly composed by video game music composer Tim Wright under the alias CoLD SToRAGE, but some tracks were licensed from Leftfield, the Chemical Brothers and

Orbital. Sony also licensed music from some non-mainstream electronica acts to create an original soundtrack album that was released to promote the game in 1996.

<PLAY WIPEOUT>

This is one of those aspects of video game music in the UK that's perhaps not as enlightened as it might be. Unlike the film industry, which, perhaps has had longer to respond, the UK Games industry has never really seen the intrinsic value of game soundtracks, and has never really sought to exploit that value through things like soundtrack albums, something, by contrast, that's absolutely huge in Japan. On the face of it, that might not seem like such a bad thing if you're getting paid reasonably well for writing the music, but generally, the sort of work-for-hire contracts that are typically used in the games industry are quite restrictive, and most games companies want to retain exclusive rights to all the work that you create while you're working for them. Certainly it was the case for all of my contracts that I had to sign all of my IP over to the companies I worked for, and one had a clause that said that *any* music that created while I was working under contract was the legal property of the company. In other words, even if I were to write original music for an EP, say, technically, I should have given that to the company who had contracted me to work on that game soundtrack, and certainly, the way the contracts were written would have restricted how I could have used my own work to promote myself.

This, I think, is one of the areas in which the games industry still has to update and modernise. Some companies are very good, but some don't see music as anything other than a game commodity to be bought, owned and used, and I think it's incumbent on all of us who value music and what it can do to work to raise its status and profile, and also to push to really unlock its commercial potential.

Once the Playstation normalised the idea that music in games could sound as good as the music you might actually listen to, things moved on apace. You've maybe heard the term convergence before, which relates to the way that different technological platforms evolve over time to perform the same sorts of tasks — look at how the mobile phone has evolved to become both a handheld personal computer and an music player and a video player, for example. Games very quickly began to converge with movies, particularly with first-person shooter games, and the challenge was to create music systems that could give a cinematic experience, but within the confines of an interactive system — we're back again to this idea of constraint.

Now it turns out that a big part of my day job involves teaching exactly this to groups of students who are really very good composers and musicians, and who want to write game music, but who find, when they try to do it, they don't really understand those technical or structural edifices, and more importantly how they shape the approach to composition.

Although film music and game music share some common qualities, writing for interactive screen media is really quite a different thing to writing music for a fixed screen medium, like film or television. And the main difference boils down to the fact that computer games are interactive media. In a film, once the rushes have been edited together, a composer has a pretty fixed framework to work to. They can work out when particular points in the story will take place, and they can use this to work out how much music they need to write, and how fast or slow it has to be to sync up to the visuals.

In a computer game, composers don't have this sort of framework. People play games very differently – some people are really goal-focused and want to get through each level to the boss as quickly as they can, while others, like me, want to explore every detail of a level, collect every item and uncover easter eggs and hidden areas – so first of all, the times that events occur during the gameplay are going to differ depending on who's playing, but the order in which events take place will also depend on the player, and so the composer who's tasked with trying to write music for every eventuality would find themselves having to score uncountably many different music cues and stings.

The approach that most games take is, rather than burn out composers, to use adaptive music engines, which monitor the gameplay and restructure or recompose the music in real time in response to what is happening in the game, or what is likely to happen in the game, at that particular point in time, and that's what we're going on to look at now — the different ways in which game engines reprovision music on the fly.

Adaptive Music

We'll start with one of the most obvious ways this can work – the event driven cue. Have you all played Super Mario Bros at some point? Well, one of the game events in Mario occurs when you collect 100 coins, and you get an extra life. What happens in the music? It triggers a little musical sting.

<Play Super Mario Bros Clip 1>

That little musical cue is triggered every time that game event takes place, hence the name event-driven cue. But you can probably imagine quite easily that it would be possible to collect those hundred gold coins at just about any point in the game, and so the event driven cue has to work with the rest of the soundtrack, no matter what else it's doing. So technically, the event-driven cue isn't difficult to achieve, logically it can be expressed as 'if this happens, then play that'. The challenge to making it work well is entirely a compositional one.

The next technique we're going to look at is something called horizontal resequencing. The horizontal part of the title comes from imagining the timeline of

the game as a horizontal line stretching out along the x-axis. At some points along that time line, things are going to happen that require the music to change – that could be that your health drops below a certain threshold level, or it could be that you collect a power up – and at that point, the game stops doing whatever it's doing in the soundtrack at that point and swaps it for a different piece of music – it resequences the track. Again, to pull an example from Super Mario Brothers, think of Mario collecting an invincibility star. When he does it, the music changes from the Overland theme to the Starman theme. When the invincibility runs out, the music segues, via the intro, back into the main game theme.

<Play Mario Clip 2>

This technique is very difficult to achieve seamlessly. From any one track there may be multiple musical branches. The transitions should happen seamlessly so as not to break musical continuity, and yet those transitions have to happen near-instantaneously to keep a strong sense of synchronisation. You can achieve that by using tempo-matching and key-matching to ensure that you don't get musical discontinuities at these transition points, but then how do you write music that has a sense of musical direction and momentum? Striking this balance is a tremendously difficult thing to achieve.

Next, is a technique called vertical reorchestration.

<Show Garageband screenshot>

Now, if you've ever used Logic, Pro Tools, Garageband or another music sequencer, you'll have come across the idea of the music timeline extending across the horizontal axis, and all of the different instruments or music tracks being arranged vertically. What this technique does, then, is monitor the gameplay, and, when a particular gameplay element changes – proximity to an end-of-level boss, perhaps – different elements of the music are faded in or faded out to vary the intensity of the music, so the closer you get to the boss, the more elements are added, and the more intense the music becomes. Again, the challenge here is largely creative, because the composer has to create layers of sound that can work in multiple combinations with transitions happening, potentially, at any point in the game.

Have any of you played Parappa the Rapper? It's a rhythm-based noodle game, probably the best rhythm-based noodle game, in fact. But it monitors your gameplay and if you're doing badly, the game drops elements of the music out, making it a bit easier to keep time, and then, as you start to improve, it reintroduces them.

<Play Parappa the Rapper>

Finally, there is a technique called procedural music, or algorithmic music, which, in effect, composes music in real time in response to how the player plays the game.

This is probably the most complex to achieve, because generating music that works as music requires programming some sort of 'musical intelligence', or at least some rudimentary music theory into the game engine, which isn't a trivial thing to do, and which takes up valuable memory and processing power. However, quite a few games have experimented with this, including a Nintendo game called Otocky. Here, the player controls a ship that fires round balls at enemy shapes and flying musical notes. With a simple two-note bass line, the player's firing actions and in-game mechanics become the melody part, quantized in real-time to the beat.

So most games will use some sort of combination of these techniques, and actually, once you understand the mechanisms behind them, they're not really all that tricky. You can imagine, for example, writing a quick bit of code that looked at player health and said 'if health drops below 30%, stop playing the main in-game music, and play the you're gonna die soon music'. Easy.

Sadly, it's not quite as straightforward as that. Although the basic mechanics are quite simple, the detail is where the devil is. Think about it for a minute. Take two of your favourite music tracks, and imagine switching at random from one to another. That switch isn't going to be smooth and seamless is it? In fact, it's probably going sound a bit naff, and as soon as you draw attention to that sort of thing, even unintentionally, you take the player out of the game.

There are other problems too. Good music in film doesn't just follow the action on screen, it leads it. So in the run up to a big action sequence, the music will build and build ahead of the action and will peak just before the climax so that the audience are all pumped up and in a heightened emotional state when it happens. It's like a good club DJ – they create the atmosphere on the dance floor, they don't react to it. They read the mood, and if the crowd needs warmed up, they'll put on tracks that lift the tempo and mood and get people moving more, and then they'll pull things back and put on some tracks that chill, and they'll ride that mood roller-coaster all night.

So these are the structural elements of game music. But the story doesn't end there. We also need to think about the mechanics of the music.

Game mechanics are all about the way in which the player relates to the game, and how the gameplay is mediated and fed back to the player. From a musical perspective, we need to invest a bit of time and thought into how to capture player input, how to turn that into a set of musical control signals, and how to use these musical controls to modify and update the music.

The more eagle-eyed amongst you will have noticed that I've got a bunch of buttons and boxes sitting on top of these plinths here, and what these represent is a series of proxies for different elements of gameplay. The button here is a simple trigger, which represents our game event for the event driven cue. Remember that the idea

behind that was simple. Any time the event happens — that is, any time the button is pressed — trigger the music, in this case a percussive, dubstep thump, which will sit nicely with the backing track, no matter when the event occurs.

The other boxes are all ultrasonic sensors, so they all use proximity as a proxy for a gameplay marker. This, I think, is a perfectly sensible one to use, because often proximity — to danger, to an end of level boss, to ammunition or a health pack — is often used for exactly this purpose, so there's a strong cognitive link. What you'll also maybe notice as I go round and demonstrate is that the associated gestures mimic, to an extent, those of a conductor, and so there's a natural cognitive link between performance and play there too.

This box here demonstrates vertical reorchestration, introducing an ambient layer of sound which you can fade in and out. This one demonstrates horizontal resequencing. In the background there are a series of key-matched arpeggiated sequences running. As you move your hand back and forward in the beam, you selectively solo these sequences, and so you effectively compose a new melodic line by resequencing fragmentary snatches of the others.

These last two boxes work in tandem. Essentially it works like a simplified theremin, using gesture to generate new musical ideas in real time. Unlike a theremin, however, it's easy to play. I've sampled the pitch space to create an invisible keyboard, and the pitches are quantised to the notes of the blues scale, so no matter where you stick your hand it's not going to sound dreadful, but there's still enough potential for creativity to create some interesting new expressions.

Casual gaming and the app store

One of the most interesting new developments has been the rise of the casual game and the app store as a commercial platform. This has given rise to a new wave of low-budget indie games, which takes video gaming back to its roots: simple graphics, simple gameplay and simple music. This time, though, the choice to go simple has more to do with aesthetics and a retro feel rather than because of any technical limitations, although given the quality of the speakers on most mobile phones, those are, of course, still there.

So there has probably never been a better time to be a game composer, or to think about becoming one. Indie and casual gaming has made it possible again for anyone who has a good idea and a little knowhow to put together a game quickly and get it out to market, and even handles all of the income gathering and distribution for you, although this comes at a price. For those who want to write video game style music, and embrace the challenge of writing efficiently and minimalistically, this is a great way to go. The financial rewards are variable. Some games become massive hits and bring in tens of thousands of pounds, others sink without a trace. For the former, it's

probably better to be part of the company with a profit share. For the latter, probably better to get paid a one-off fee and walk away.

For those more interested in the structural challenge of writing game music for big-budget triple-A titles, the barriers to entry are higher. Most big studios won't employ people who don't have previous experience of working on big titles, but if you've got equivalent experience in film or television, that can be a useful in.

With a growing and lucrative market, however, there are now, as you might expect, a growing number of specialist service studios who specialise in, for example, voice overs, music and sound effects for games, and periodically they advertise for new creative talent. In general, though, the industry is still fairly entrepreneurial, and so there is nothing stopping you from setting up your own studio and going out to pitch for work, perhaps alongside other film or advertising work.

If you'd prefer to become an employee, however, probably the best place to start would be one of the specialist recruitment agencies, Aardvark Swift, Amiqus and Datascope would be where I would head to.

For me, I realised that what I was really interested in was in understanding how it all worked. I was the sort of kid who took my joysticks to bits to try and figure out how they made Miner Willy move, only to have bits left over when I tried to stick them all back together again. That sense of playful inquiry is the thing that motivates me, rather than the desire to create a finished product to sell. So I became an academic.

Does anyone have any questions?