UNIVERSITY OF CALIFORNIA
RIVERSIDE

Gameplay as Discrete Form: Leveraging Procedural Audio for Greater Adaptability in Video Game Music

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Music

by

Christiaan Aaron Clark

September 2021

Dissertation Committee:
Dr. Ian Dicke, Chairperson
Dr. Paulo C. Chagas
Dr. Dana Kaufman

The Dissertation of Christiaan Aaron Clark is approved:

_____

_____

_____

Committee Chairperson

University of California, Riverside

ABSTRACT OF THE DISSERTATION

Gameplay as Discrete Form: Leveraging Procedural Audio for Greater Adaptability in Video Game Music

by

Christiaan Aaron Clark

Doctor of Philosophy, Graduate Program in Music
University of California, Riverside, September 2021
Dr. Ian Dicke, Chairperson

Video game music has finally begun to be taught in academic multimedia composition programs. This is an exciting prospect, but video game music is then consistently lumped into the same category as film and television. This is an egregious error. The types of music needed for the game industry is more extensive than the traditional orchestrations that currently dominate linear media - video game music can range from death metal, to electronic dance music, to synth pop. This is not the only classification problem. Video game design creates an intrinsic problem in the music-making process - since game designers will never know *when* a player will do an action or move to a new section, the composer cannot write pieces with discrete temporal form like the silver screen. Instead, they have to leave certain musical elements up to an inherent level of indeterminacy. Fixed audio recordings have made this flexibility almost impossible throughout the past two decades. As the level of immersivity increases for game design at an exponential level with the development of comprehensive game engine visuals, adaptive video game music has begun to plateau due to the lack of improvement

in audio engines over the last decade. This is where the significance of procedural audio becomes relevant. This dissertation discovers and programs new methods of allowing future video game composers to procedurally generate their musical vision inside of game engines so that elements such as timbre, melodic density, and rhythmic intensity (among others) can be driven and altered by player choices rather than relying on the stiff nature of pre-recorded audio.

Table of Contents

List of Figures

I. Introduction

II. Limitations of Game Audio in the Current Generation

III. Exceptional Examples of Adaptive Game Music

IV. Technical Demonstrations

I. Introduction

A. Setting the Stage

Video game adaptive music has reached a creative stalemate caused by technical limitations. It is superficial - soundtracks that seem novel at first quickly become banal and repetitive after just a few hours of similar interactions. It reconciles rather than responds - even if composers understand the dilemma that nonlinear gameplay creates, they have to write music that will always sound the same except for a stretched temporal form. It establishes a limiting definition of what "adaptive" means - music always latches on to a single gameplay state despite the myriad complex structures that are present in these virtual worlds.

This dissertation postulates how sequencing music in real-time with procedural audio can establish greater levels of adaptability in video game soundtracks that are currently limited by audio files. Audio files hold back adaptive music in two main ways: they are rigid because you cannot alter one aspect of the file without affecting the rest, and they are unsustainable due to a limit to the file size that a game can accommodate for sound assets. Procedural audio is being generated in real-time so it can change in any way that a composer dictates through gameplay parameters, and it requires no extensive sample data that would bog down memory storage. Real-time synthesis provides an elaborate timbral palette, but even melodic, harmonic, and rhythmic alterations can be made in real-time, because composers will sequence musical information through the audio engine rather than exporting a fixed file.

This dissertation does not postulate how composers should use this added flexibility to shape a game's musical aesthetic. Artistic decisions such as a soundtrack's instrumentation, genre, and connection to the game are problems that will be solved during the initial design stage. Two composers could represent the same gameplay state in different ways - tension could be created by adding rhythmic complexity, but it could also be expressed through a shift to grittier timbres. Those examples are still arbitrary though - composers will invent their own salient musical parameters that are informed by each game's special mechanics. The statements put forth in this dissertation are meant only to explain why procedural audio tools are necessary in adaptive music creation, not dictate how they should be used.

B. Toys vs. Puzzles vs. Games

Video games are an inherently performative art form, and watching people play them is moving closer to the forefront of society as a popular visual medium. Streaming websites such as Twitch and video-sharing domains like YouTube and Vimeo allow users to create their own live or fixed "performances" of a game for others to watch. This informal content has even been elevated to more official events in recent years. Production company Rooster Teeth launched a live touring concert at the Moody Theater in Austin, Texas in 2015 known as *Let's Play Live*. At this event, the audience attended in order to watch the six main personalities from Rooster Teeth's subgroup Achievement Hunter play video games on a giant projected screen. These performances all have one

thing in common – because they are playing a game, performers create new outcomes structured by predetermined rules.

Imagine a separate event where performers are equipped only with action figures and stuffed animals. Arguably, to make this a successful event, a great deal of care would need to go into establishing the content of the performance. The performers could walk out and ad lib the entire show, but even the world's greatest improvisers value structure. The television show *Whose Line is it Anyway?* quite literally pits its performers against one another for abstract "points that don't matter." Each skit is presented as a game to give the performers a goal that needs to be fulfilled before they end. Prior to each *Let's Play Live*, the members of Achievement Hunter are equally devoid of expectations of what will be said or happen during each event, and are only aware of what games will be played.

Chris Crawford's (veteran video game designer and the founder of the Game Developers' Conference [GDC]) classification of games as a medium helps explain why this level of structure can assist performance. In his "taxonomy of creative expressions," he categorizes interactive entertainment in three ways: "toys," "puzzles," and "games." A toy specifically refers to anything that is interactive but that does not inherently have a goal. A puzzle is anything that has a goal and a fixed outcome, but that does not have active competitors. A game establishes a goal, but does not have a fixed outcome because it adds active competitors.[1] These designations are made to explain the technical constraints that an interactive object places on its users, not to designate how the user

1 Chris Crawford, *chris crawford on game design* (Indianapolis: New Riders, 2003), 6-9.

actually intends to use them. Computers are toys, but users can make puzzles and games on them.

Toys do not have an ending aside from when users stop interacting with it, so they do not have strict form. To connect this to music, this closely resembles the way that an audience can interact with sound installations. Some installations allow the audience to interact with and influence the piece, as well as decide how long they actually want to engage with it. They are often created with an open form in mind, just as toys are created for exploratory purposes. Musical instruments are toys as well - while they may be designed with an expected purpose in mind, users can do anything they want with them. Nam June Paik exemplifies this experimental freedom in his 1962 piece *One for Violin Solo* where a performer is instructed to smash a violin after slowly raising it above their head for about five minutes.[2]

If this is true, any piece of music played from a written score is a puzzle. There is a clear expectation of what the piece should sound like in the end, but the performer puts it together in a new way each time. Most *puzzle canons* (such as the "Crab Canon" from Bach's *The Musical Offering*) clearly fall into this designation as well. Even though only one line of the canon is notated, the composers have an understanding of what the solution should be, and often provide clues through cryptic titles or unusually placed clefs and key signatures.

Indeterminate composers of the New York School such as John Cage, Morton Feldman, and Earle Brown wrote music that frequently blurred the line between toys and

2 "Paik, Nam June: One for Violin Solo," Media Art Net, accessed February 18, 2021, http://www.medienkunstnetz.de/works/one-for-violin-solo/.

puzzles. John Cage's *One* for solo piano removes standard metric notation such as time signatures and measure barlines, but he still provides a left to right order of pitches with temporal guidelines on how long a performer should ruminate on a particular vertical sonority.[3] This most closely resembles a puzzle, because the outcome will always sound the same with the exception of deliberate rubato.

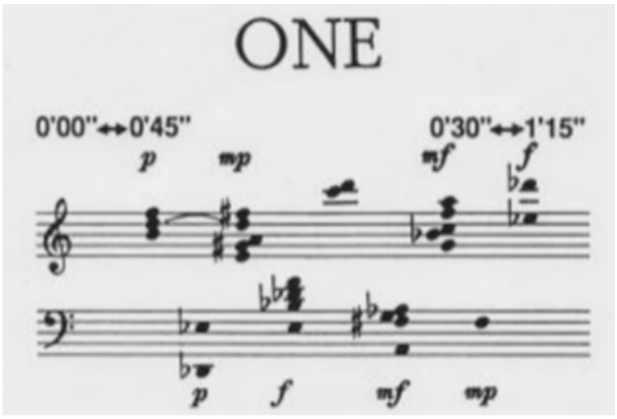

Figure 1: Excerpt from *One* for solo piano by John Cage

Figure 2: *Folio II* by Earle Brown

3 John Cage, "John Cage - One (1987) audio+sheet music," Welcome to sale, April 30, 2020, YouTube video, 11:52, https://youtu.be/TfLvAhEyJ48.

Earle Brown's *Folio II* on the other hand is better analyzed as a toy since players are greeted with a score that dictates no clear structure. The removal of a traditional grand staff prevents performers from knowing when any of the individual motifs should happen, because they all blend together horizontally. The sharps and flats are also irrelevant, because without the staff they could ostensibly be referring to enharmonically equivalent naturals such as E-sharp or F-flat.[4]

The emergence of competition means that an interaction will no longer unfold in exactly the same way. It is no longer enough that players memorize and duplicate an intricate sequence of button pressing and human expression - they must now adapt to the presence of another organic or artificial intelligence. Single-player video games still count as games, because the computer acts as the competitor, but video games in the rhythm genre such as *Guitar Hero* and *Rock Band* arguably fit better into the puzzle category of interactive media. The lack of active competition means that level design exists on a fixed timeline and will always unfold in the same way. Musical autorunner video games like *BIT.TRIP Runner* and *Geometry Dash* have a similar fixed level progression system and move players forward at a constant velocity in a method reminiscent of a moving playhead. In this sense, players are learning the motions of specific levels as if they were practicing a solo piano composition. Not all rhythm video games are structured like this - Brace Yourself Games' *Crypt of the Necrodancer* challenges players by forcing them to perform actions as the main protagonist only when

4 Earle Brown, *Folio II* (Leipzig: Edition Peters, 1982), http://www.earle-brown.org/images/file/media/Folio%20II%20as%20of%2012-12-2013.pdf.

downbeats of the soundtrack happen. While this may seem like a tedious restriction at first, it adds to the strategy that players need to conjure to escape the game's dungeons.

It can be difficult to connect the concept of games to music - musicians do not usually compete against one another, at least not consciously. Despite this, there are many examples throughout the history of Western music that connect gameplay to music such as *Musikalisches Würfelspiel*. These musical dice games (often associated with Wolfgang Amadeus Mozart) establish musical structure from precomposed material by rolling dice.[5] In this particular instance, the dice can be viewed as the competitor in the same way that it is in the tabletop role-playing game *Dungeons and Dragons*.[6] Puzzles and games both have much stricter form than toys because they both have a clear ending, but puzzles leave the final form up to the original creator while games leave it up to the competitors.

Starting in the late 1970s, composer and saxophone improviser John Zorn began to write pieces that were literally games. To place an emphasis on performers rather than composers, his goal was to "deal with *form*, not with *content*, with *relationships*, not with *sound*."[7] Zorn's belief about creating relationships between performers in music directly parallels Crawford's designation of competitors in a game. In fact, his 1984 piece *Cobra*

5 Wolfgang Amadeus Mozart, *Musikalisches Wurfelspiel, K.516f* (Bonn: N. Simrock, 1793), https://imslp.org/wiki/Musikalisches_W%C3%BCrfelspiel,_K.516f_(Mozart,_Wolfgang_Amadeus).

6 The official *Dungeons and Dragons* website explains that "Players roll dice to determine whether their attacks hit or miss and whether their characters can scale a cliff, roll away from the strike of a magical lightning bolt, or pull off some other dangerous task." "Basics of Play: Learn About the Game." Dungeons & Dragons. Wizards of the Coast, 2021, https://dnd.wizards.com/basics-play.

7 John Zorn, "The Game Pieces," In *Audio Culture: Readings in Modern Music*, edited by Christoph Cox and Daniel Warner (New York: Continuum, 2010) 196-200.

allows performers to make decisions in real-time with hand gestures that directly affect the other performers. “For example, one musician might hold one hand over his mouth while displaying four fingers with the other hand, meaning that he wants the ‘sub crossfade’ event to take place.”[8] The conductor (Zorn calls this the prompter) will then lift a card from a deck to trigger its corresponding musical event. As a composition, *Cobra* exists only as a set of rules to structure improvisers, but these rules give each performance an identity that can only be associated with *Cobra*.

The way Steve Reich defines “process” music in his essay *Music as a Gradual Process* seems to imply that he writes music in the style of a game as well. He establishes his role as the composer as the one who creates material and sets up the rules of the process. He then admits that “I completely control all that results, but also that I accept all that results without changes.”[9] This is his way of handing over the piece’s final form to a game and its implicit competitors. Reich demonstrates this beautifully with his popular phasing compositional technique. In his piece for tape *Come Out*, he creates competition between two tapes of the same voice recording by establishing the rule that one tape is slightly faster than the other. Brian Eno took this one step further to produce the track *2/1* for his album *Music for Airports* by recording a different pitch on each of seven different varying length magnetic tapes.[10] John Cage established his own process when he

8 Dylan van der Schyff, “The Free Improvisation Game: Performing John Zorn’s *Cobra*,” *Journal of Research in Music Performance* (Spring 2013): 1, accessed February 20, 2021, https://scholar.lib.vt.edu/ejournals/JRMP/2013/schyff.pdf.

9 Steve Reich, “Music as a Gradual Process,” In *Audio Culture: Readings in Modern Music*, edited by Christoph Cox and Daniel Warner (New York: Continuum, 2010) 304-306.

10 Tero Parviainen, “How Generative Music Works,” last modified 2017, https://teropa.info/loop/#/airports.

composed *Music of Changes*. All parameters of the piece were determined using chance operations from the Chinese *I Ching* (*Book of Changes)*. This compositional method taught Cage how to relinquish his artistic integrity and give in to the process as the final creator.[11]

Moving forward, it will be beneficial to maintain the classifications of how certain music exists as interactive media (toy, puzzle, game) to better understand how it is created, not how it is experienced. Just as Steve Reich mentions that musical processes determine discrete form, so do games. The performers on *Whose Line is it Anyway?* and the *Let's Play Live* events also accept all that results without change.This provides an exciting framework for how and why video game adaptive music should be created with competition in mind in order to unravel a fresh experience every time gameplay initiates a discrete form. Contrary to this idea however, most current video game composers have decided to structure their music like a puzzle rather than a game. That is to say, their mission is to find an audio implementation method that allows their musical vision to be heard exactly as they hear it in their head. This means that they veer around and disregard game logic to make sure that the music's final form is determined by them as the creator. It also exists to ask why composers do not embrace the game over the puzzle with the help of procedural audio, why they are reluctant to relinquish certain levels of control to the process, and how current video game audio software is contributing to and exacerbating the situation.

11 "Music of Changes," John Cage Complete Works, accessed February 19, 2021, https://johncage.org/pp/John-Cage-Work-Detail.cfm?work_ID=134.

C. Dilemma of the *Earcade*

This dissertation was initially outlined in the prospectus as a live exhibition of video game/sound installation hybrids called the *Earcade*. These installations would have demonstrated the power of creating indeterminate compositions through individual gameplay performances in an attempt to inspire the future of adaptive video game music. One demo relied on the dynamic nature of player strategy to build and upgrade new towers to defend against a continual onslaught of incoming enemies. The music would have been generated entirely through real-time synthesis, with timbral changes controlled through tower construction. This and all of the other experiences would have leveraged the player's desire to win the games against digital competition, creating an entirely discrete performance each time.

In theory, this could have been an engaging and illuminating experience. In practice, it was doomed from the start. Early research into the procedural capabilities of video game audio engines showed a myriad of obstacles that would need to be addressed. In regards to the tower building example, there were no tools that had the capability of dynamic synthesis in real-time because of current limitations of game audio middleware such as Wwise and FMod. This first dead end resulted in the discovery of a blossoming framework for real-time synthesis native to the popular video game engine Unreal Engine 4 (UE4). A new UE4 application programming interface[12] (API) was a lot more promising than the tools in the middleware programs Wwise and FMod, but it had one

12 An application programming interface connects multiple different sets of code together for simplified iterations of common functions in order to expedite software development.

major drawback - it lacked a proper way to read and sequence MIDI[13] files, so musical information sent to the software synthesizers would inevitably sound unquantized (meaning rhythms came in either early or late). Scott Bishel's *Procedural MIDI* plugin on the Unreal Engine Marketplace allows users to load and perform MIDI files, but it lacks a dedicated network to connect the musical information to UE4's procedural sound sources.[14]

When the mainstream software proved unhelpful, further research went into finding alternatives that could run the *Earcade*'s complex adaptive music. The software *Elias Adaptive Music*[15] has an interface remarkably similar to Ableton Live's Session View, but its MIDI capabilities are tied exclusively to launching audio from sample libraries, meaning synthesis and real-time timbral control is not possible. In addition, there are no features that allow MIDI files to be transformed in salient ways. Real-time synthesis means that musical information such as a piece's tonal center and metric drive do not need to stay fixed either, and can change in the same way as timbre. One of the primary benefits of starting a sonic data stream exclusively from digital information is that musical parameters can be shaped and developed dynamically by devices such as arpeggiators and step sequencers. Many procedural music plugins were found after rummaging through the Unity (a popular video game engine for independent developers) Asset Store, but they all rely on artificial intelligence and algorithms to generate the

13 MIDI (Musical Instrument Digital Interface) is a universal protocol that allows electronic music hardware to interface with computers.

14 Scott Bishel, "Unreal Engine Marketplace," *Unreal Engine Marketplace* (Epic Games), accessed February 21, 2021, https://www.unrealengine.com/marketplace/en-US/product/procedural-midi.

15 "Elias Adaptive Music: Adaptive Game Music and Sound," *Elias Adaptive Music: Adaptive Game Music and Sound* (Elias), accessed February 21, 2021, https://www.eliassoftware.com/elias-studio/.

musical material rather than providing elaborate sequencing tools for composers. Most of them, such as *psai Music Engine Pro*,[16] have become defunct and no longer have a promotional website. There is currently no software that can function as an all-in-one sequencing solution.

After nearly a year of fruitless (but useful) research and experimentation, it began to feel like the *Earcade* would be impossible to create. No applicable procedural audio toolsets had the necessary power for in-engine composition, and any functional sequencers were completely devoid of real-time synthesis. One early solution to these problems would have leveraged the OSC (Open Sound Control) protocol to connect between Ableton Live and Unreal Engine, allowing Live to handle all of the audio processing while Unreal Engine housed the actual experiences. To continue with the project, it began to feel like the *Earcade* would need to sacrifice some of its integrity and focus more on philosophical and creative possibilities rather than practical and technical applications. This would have defeated the point of the experience though - video game composers cannot embed their favorite digital audio workstations[17] (DAWs) into a game engine. After the initial roadblocks the *Earcade* endured, the most logical forward trajectory culminated into two convergent projects: the creation of intelligent and flexible music software for use with Ableton Live to showcase the idyllic potential of what adaptive music should strive to be and the development of rudimentary modular sequencing and synthesis tools inside of UE4 to act as a jumping off point for the future.

16 Periscope Studio, "Unity Asset Store," *Unity Asset Store* (Unity, August 3, 2016), https://assetstore.unity.com/packages/tools/audio/psai-music-engine-pro-24788.

17 Digital audio workstations are audio production software used to edit recorded sound or generate new audio with sequencing and procedural audio.

D. Introduction to Procedural Audio

Procedural audio is the key to unlocking unlimited flexibility in video game adaptive music, so it is important to understand its benefits and drawbacks. Andy Farnell defines procedural audio as "a living sound effect that can run as computer code and be changed in real time according to unpredictable events."[18] His book *Designing Sound* has a chapter that outlines numerous benefits of procedural audio for creating sound effects in video games, and provides practical examples on how to achieve these common soundscapes. One advantage compares the central processing unit[19] (CPU) cost of iterating pre-recorded samples to the cost of running procedural sounds. Procedurally generating wind sound effects requires less CPU resources than a recording of wind. In addition, procedural sounds can utilize a flexible and dynamic amount of resources in contrast to the static amount that sample data requires.

If a game sound such as footsteps is dimensionalized in a video game to emulate the real world, it should lose clarity the further away it is. Farnell notes that accomplishing this with sample data will rely on the same amount of CPU resources no matter where the sound is in the world, but a procedural sound could switch synthesis methods dynamically to demand fewer resources as the footsteps move further away. Game engine visual renderers achieve a similar goal by lowering the polygon count on objects that are far away from sight. If gamers can barely see the objects, there is no reason for them to have the same high level of detail as the objects in their immediate vicinity.

18 Andy Farnell, *Designing sound* (Cambridge, Massachusetts: MIT Press, 2010), 1.

19 All code is executed in the central processing unit of a computer.

Designing Sound is an excellent resource for learning how to produce real-time sound effects, many of the listed pros and cons it presents do not actually relate to or affect procedurally generated music. Traditionally, background music is nondiegetic and therefore does not require dimensionalization calculations as players move around the world. Another disadvantage the book notes is that the counterpoint of multiplayer gameplay could result in too much variable cost - "if ten players all suddenly decide to smash the windows in a building and an explosion of particles is spawned, the audio engine must cap the resources." This anecdote shares no relation to generating music with procedural audio, because the necessary maximum amount of CPU resources needed for instruments and digital signal processing[20] (DSP) effects would be calculated well ahead of time. In the same way that audio teams currently receive an allocation of how many CPU resources can be allowed for audio, the music can be composed and optimized in a way that limits its digital footprint on the game. This already happens when composers choose to implement their soundtracks with *vertical remixing*, a production technique that allows stems of a larger project to fade in and out for dynamic orchestration. With proper implementation, certain audio channels will be turned off to optimize performance if only a fourth of the ensemble is heard. If the players are not hearing a track, there is no reason for it to be processing empty audio.

Using procedural audio for music does present its own obstacles compared to a game's evolving soundscape. Many VST (Virtual Studio Technology) plugins that allow digital synthesis in DAWs are known for consuming a large chunk of CPU resources, but

20 In audio production, digital signal processing specifically refers to operations that modify incoming audio signals such as compression, equalization, and reverb.

there are many forces at play that cause this. First, the graphical user interfaces that musicians interact with are often taxing processes. Luckily, once all of the production choices have been made for a game's soundtrack, the interface would be completely stripped away and have no influence on the final product. Additionally, audio plugin developers build VSTs for use in advanced music production, so optimization comes second as a development goal compared to out-of-the-box tone quality. Some software synthesizers come with a fixed amount of polyphony that is usually inevitably underutilized. If a synthesizer comes with sixteen voice fixed polyphony, the same number of virtual tone generators is calculating the value of its next sample even if half of them are producing nothing. In order to embrace procedural audio plugins for adaptive video game music, optimization will need to be the first development goal. This means that while computationally expensive synthesis methods such as physical modeling and additive synthesis may initially be avoided for projects, more efficient methods such as wavetable synthesis and FM synthesis are all equally viable solutions. Traditional additive synthesis requires an extra calculation everytime a new sine wave adds to the complexity of a timbre – this could become computationally taxing once eight or more separate oscillators are combined. Conversely, wavetabtle synthesis only requires the calculation of one portion of a buffer against its next chunk, and FM synthesis only requires the calculation of the modulating oscillator that then affects the carrier oscillator. Understanding this CPU economy will be crucial if procedural audio becomes a part of video game soundtracks.

Farnell presents a troublesome opinion of the way adaptive game music should be composed. At the end of the game audio chapter, he asserts that "some believe it yields soulless and two-dimensional results" because "although many algorithmic composition techniques exist and musical rules can be formalised in code, it is impossible to replace the human creativity a real composer brings to interpreting and augmenting a scene."[21] There are many problematic elements in this quote. First, Farnell seemingly equates adaptive game music to algorithmic composition, but using procedural audio to compose is not the same as algorithmic composition (sometimes also referred to as "generative music"). Wholly synthesized chiptune soundtracks for retro-inspired independent games start entirely as procedural audio - the DAW has to translate MIDI information to generators before it can even exist in a concrete form. This statement also neglects more than two decades of video game music that was procedurally generated on programmable sound generators. Sound chips in the early years of video gaming exclusively used procedural audio to perform its soundtracks, because audio files were too large to be loaded onto the machines.

The quote also makes generalizations about the "human creativity" of a composer, implying that all worthwhile artistic elements must be predetermined by a single entity. This fails to understand the role of the player as a performer, and how that can be used to unfold their own unique story. Supergiant Games' video game *Bastion* uses nonlinear dialogue to narrate player actions, eventually constructing their own shuffled verbal

21 Andy Farnell, *Designing sound* (Cambridge, Massachusetts: MIT Press, 2010), 326.

account of the way the game's events actually happened.[22] In-game actions such as breaking boxes and clumsily falling off the map are each accompanied by their own sarcastic remark from the narrator. One player could break the boxes, and then accidentally fall off the map while a different player could do those actions in reverse. The narrator's tense implies that the story is being told in the future about past events, so these two gameplay performances yield a completely different verbal account of the tale. This phenomenon is similar to the way jazz musicians interpret lead sheet symbols. If three different performers read the same harmony, one could potentially arpeggiate chords in an ascending pattern, one could arpeggiate them in a descending pattern, and the last could just play all the notes together as a chord. The order of events has changed, but the overall structure has not. In other words, it is not the composer's job to completely "interpret" and "augment" the music. Instead, jazz composers write out "musical rules" that are essentially "formalised in code."

Research over the past three decades has culminated in a robust list of open-source and commercial softwares for procedural audio. Arguably the most popular open-source procedural audio software, *Pure Data* (*Pd*), has already been embedded into game engines. In fact, Enzien Audio's Python framework *Heavy Compiler Collection* has made the conversion from *Pd* patch to C/C++ source code an expedited process.[23] This is necessary for two reasons: the code needs to be formatted in a particular way to fit the

22 "Supergiant Games," (San Francisco: Supergiant Games, August 11, 2011), https://www.supergiantgames.com/games/bastion/.

23 *Pd* patches always start with a node-based visual programming interface that will be irrelevant to gameplay, so *Heavy* can remove all but the essential code. "GitHub," *GitHub* (Enzien Audio, September 21, 2018), https://github.com/enzienaudio/hvcc.

desired game or audio engine, and the visual baggage of a *Pd* patch needs to be removed completely. At Unite 2013, Zach Aikman showed his integration of James McCartney's *SuperCollider* (another popular open-source software) in the Unity game engine.[24] The method relied on communicating between Unity and *scsynth* (*SuperCollider's* real-time coding server) with OSC messages. Realistically, neither this or the Pure Data example yield ideal results for games that will be released on multiple platforms. Both examples have an inherent lack of proper optimization, making them impossible to be used in triple-A video games[25] where CPU resources are a luxury that audio designers rarely receive.

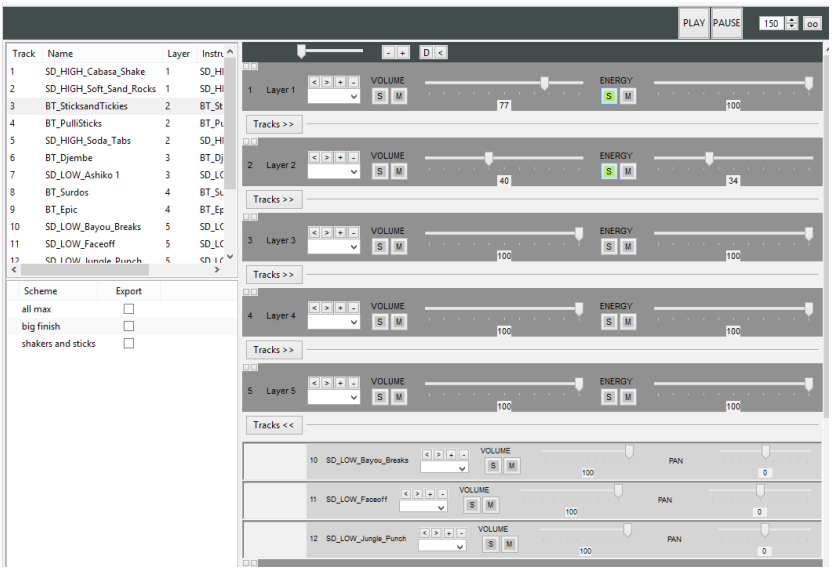

Figure 3: Screenshot of *INTELLIGENT MUSIC SYSTEMS*

24 Zach Aikman, "Unite 2013 - Real-time Audio Synthesis with SuperCollider," Unity, September 16, 2013, YouTube video, 29:01, https://youtu.be/4uvClBGhJyA.

25 Similar to the film industry, the video game industry has different class designations of video games based on budget, indie being the lowest and triple-A being the highest.

The program *INTELLIGENT MUSIC SYSTEMS* (*IMS)* from composer and software engineer Daniel Brown is currently one of the only commercial procedural audio tools used for triple-A video game soundtracks, and it is still limited to composition with unpitched percussion. Debuted in *Rise of the Tomb Raider*, *IMS* runs as a single DSP unit in any game's audio engine and analyzes a piece's existing MIDI files to sensibly generate new rhythmic motifs. Composers can load their sample libraries into the Designer Tool (shown in Figure 3) and alter a specific layer's "Energy" level, which makes a percussion track more or less active based on the precomposed music. This way, the composition can more subtly transition between different game states without creating abrupt "audio artifacts."[26] In addition to the obvious musical benefits, *IMS* also improves how this music implementation would affect gameplay performance in a positive way. Rather than stemming out each percussion layer and running them from separate channels, all percussion can exist on a single lane in the audio engine to drastically reduce CPU usage.

To make procedural audio accessible to game development, tools should be built from scratch in each popular engine to mesh with the existing code. Unity's audio plugin software development kit (SDK) and UE4's SynthComponent both allow for the development of native procedural audio plugins. Since they both utilize the same internal code as the rest of the engine, it can be assured that the audio tools will run properly on either a high-performance console or a mobile device. The currently available commercial plugins built with these SDKs have focused exclusively on developing DSP

26 Daniel Brown, "Intelligent Music Systems," *Intelligent Music Systems* (Intelligent Music Systems, 2016), http://www.intelligentmusicsystems.com/.

effects to process pre-existing audio files. Tools such as convolution reverb help establish a realistic and convincing atmosphere of sound effects, but do nothing to promote procedurally generated music.

II. Limitations of Game Audio in the Current Generation

A. Finding Where Things Went Wrong

The history of video game audio is a complicated tapestry to detangle. In recent decades, sound is the medium's only element that is consistently non-synthetic. While all aspects of a game's visuals from 3D meshes to physics simulations to particle effects are usually calculated and generated by the computer, sound effects and music are consistently reproductions of information captured from the real world such as foley or orchestral recordings. This was not always the case - the first few generations of video game audio shared in the glory of presenting itself in a purely synthetic manner. This chapter investigates how recent creative trends and technical barriers have effectively stifled the evolution of adaptive video game music.

B. Lack of Musical Interpolation

In computing, larger bit depth allows any function to have greater possible outcomes by increasing the amount of bits (1s and 0s) in a particular binary stream. A bit depth of 1 allows only one number in the stream, so there can only be either a 1 or a 0 (essentially on or off, true or false) - this type of information is also referred to as a *boolean*. A bit depth of 2 allows the binary stream to contain two numbers, so there are

now four (two to the power of two) possible options: 11, 10, 01, and 00. Differences in bit depth start as a completely abstract concept on paper, but are essential in helping computers contextualize digital information in ways that humans can comprehend. Disparities in resolution are easy to witness when looking at varying qualities of the same image on a computer. Lower image bit depths have a limited capacity for color: 1-bit means the image will only have black and white (also known as monochrome), 4-bit means that there are only sixteen (two to the power of four) colors between and including black and white, and 24-bit exponentially increases that to 16,777,216 colors. Figure 4 below demonstrates this phenomenon with lowered resolution on two images (of the Anaheim skyline) to the left while maintaining the bit depth of the original picture on the right.

Figure 4: Image Pixel Bit Depth from left to right: 1-Bit...4-Bit...24-Bit.

Technological advancement has metaphorically been driven throughout history by a desire to increase the "bit depth" resolution of a machine's capabilities. While bit depth literally relates to digital technologies, the same concept is paralleled in the development of mechanical and analog devices of the past. The first light bulb was boolean and had two options, off and on (0 and 1). In the modern age, many light fixtures have dimmers that allow users to adjust the brightness, which enhances the bit depth of luminescence.

These same technological parallels have existed in the design of musical instruments for over a millennium, especially in the realm of keyboard design. All advancements in keyboard design over the past five centuries can be explained as occurring as a result of musicians wanting to improve their metaphorical *emotive bit depth*.

The harpsichord was an iconic instrument, but it had a few drawbacks. It "cannot produce any appreciable change in loudness in response to a change in the force with which the key is struck, since, regardless of force, the string is displaced virtually the same amount by the plectrum."[27] Essentially, the harpsichord had almost no ability to establish dynamic contrast, except for the possibility to add another set of strings (though this primarily acted as a timbral enhancer by plucking the second strings at a different place and adding harmonics). With only two volume levels (one set of strings and two sets of strings), the harpsichord was boolean and had a volume bit depth of 1 (soft, and not as soft).

This limitation could be why the harpsichord fell out of fashion and lost the capitalistic battle against the fortepiano, the predecessor to today's modern concert pianos. The articulation method for these instruments involves the use of hammers that strike the strings instead of plucking them - it is basically a dulcimer with keyboard control. This one mechanical change allowed the instrument to not only have a much wider dynamic range than the harpsichord, but also one with a higher emotive bit depth similar to the dimming of a light bulb. It is worth noting that as a result of this choice the

27 *Grove Music Online*, s.v. "Harpsichord," accessed February 18, 2021, https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000012420.

specific timbre that the public associated with keyboards as a popular instrument completely changed, shaping entire eras of Western music.

Throughout the history of keyboard design before the 20th century, one musical trait has consistently suffered from low emotive bit depth - timbre. The main exception to this rule, the pipe organ, has the ability of combining various different types of pipes (completely metal, reed pipes, etc.) for unique timbral combinations. This design has its limitations though - organs are massive instruments that are housed in specific locations, so this control is not even remotely portable. Many composers became frustrated with the lack of a comprehensive timbral palette on keyboards, and developed extended techniques that seeked to enhance it. Henry Cowell's 1925 composition *The Banshee* bypassed most mechanisms inside the piano by coercing performers to use their hands to articulate the strings.[28] The advantage of hands over felt hammers meant that Cowell could utilize both flesh and nail for rounder or harsher tones, and use fingers to strike or pluck the strings in the manner of a harpsichord. A decade later, John Cage's works for prepared piano combined the existing mechanics of the piano with objects such as screws and rubber in between strings to interfere with their natural resonance.[29]

At the same time Cage was experimenting with acoustic pianos, the progression of analog synthesis was well underway. Subtractive synthesis became a turning point for higher timbral resolution. Taking a highly complex sound and filtering out harmonics to

28 Anthony McDonald, "Beyond the Score: Henry Cowell's *The Banshee*," New York Public Library for the Performing Arts, December 9, 2020, https://www.nypl.org/blog/2020/12/09/beyond-score-henry-cowells-banshee.

29 John Cage, "How the Piano Came to be Prepared," John Cage, 2012, https://johncage.org/prepared_piano_essay.html.

produce pure sine waves is remarkably similar to the act of dimming a light. This potential has only grown since the advent of digital synthesis. Wavetable synthesis allows sound designers to take two completely different waveforms (which initially presents itself as a timbral bit depth of 1) and smoothly shift between them using calculations known as "interpolation."

In mathematics, interpolation refers to "the process of calculating an approximate value based on values that are already known."[30] Imagine an x/y graph that has point A at (0,0) and point B at (1,1) - given the information of a new point on the x-axis at 0.6, linear interpolation (also known as "lerp") can be used to approximate its y-value along the line created by points A and B to approximate that y will also equal 0.6. Thanks to a continual current of electricity, the dimmer of a light fixture linearly interpolates from no light (0) to the brightest possible light (1), allowing users to adjust their luminescence to their desired level. A computer can only interpolate to values inside of its bit depth however - as previously mentioned, a bit depth of 2 has four possible levels of variance that can be imagined for any given parameter: levels 0, 1, 2, and 3. This can be scaled from 0 to 1 as such: 0.0, 0.33, 0.66, and 1.0. Interpolating a value of a point on the x-axis at 0.6 would simply round up to its nearest level of 0.66 because of the low bit depth resolution.

This variance (known as *quantization error*) can be seen in Figure 5 where the blue line represents linear interpolation and the red line represents the available bits. Humans have the ability to extrapolate the necessary information when only certain data

30 *Merriam-Webster.com Dictionary*, s.v. "interpolation," accessed February 21, 2021, https://www.merriam-webster.com/dictionary/interpolation.

is present, but computers cannot accomplish this without the proper bit depth. In fact, musical scores written with Western notation demonstrate this dichotomy. Despite the fact that volume has an infinite range in nature, composers frequently limit themselves to six different dynamic markings: pianissimo, piano, mezzopiano, mezzoforte, forte, and fortissimo. Veteran composerss are skilled enough to read between the lines and can express themselves between these directions, giving dynamic markings a somewhat capricious existence. Computers, on the other hand, cannot act on personal judgement to move slightly above or below these suggestions. This is why the performance of pieces from computer-notation software such as *Sibelius* or *Finale* frequently sound unimaginative and stiff. Instead of working with just a few dynamic markings (between 1 and 2-bit depth), MIDI performance has 7-bit volume control known as *velocity* that allows computers to express volume on a scale from 0 to 127.

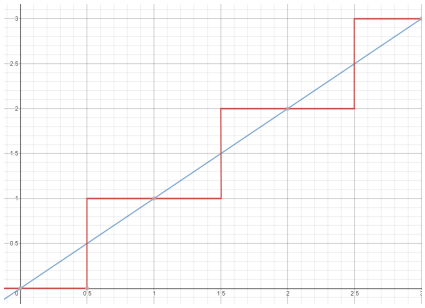

Figure 5: Visual Representation of 2-Bit Quantization Error

Composers have notated the interpolation of musical traits for centuries to combat the lack of expressiveness presented in the quantization error of these markings, and have devised ways to notate these techniques as well. Performing a crescendo from pianissimo to fortissimo is a volume interpolation method that convinces musicians to gradually build their dynamics from their instrument's softest possible volume to its loudest

possible volume. Performing a glissando on a violin from notes C4 to C5 is essentially the same as telling the performer to interpolate from the frequency 261.63 Hz to 523.25 Hz. Analog and digital synthesizers use low-frequency oscillators (LFOs) to interpolate almost every musical parameter possible from pitch to volume to timbre.

Smooth *musical interpolation* such as this is only possible if an instrument's emotive bit depth has a high enough resolution. This does not mean that composers have to use a large emotive bit depth in a conjunct manner to extend expressivity in music. Composers interested in total serialism such as Pierre Boulez and Luigi Nono ordered their music with emotive bit depth in mind without acknowledging it in that way. Boulez's piano composition *Structures 1* doubled the possibilities of written dynamics beyond the conservative range of pianissimo to fortissimo (which would represent only six dynamics) by mapping twelve different dynamics each to a different chromatic pitch. Boulez obviously felt that the standard notation dynamic-scale was lacking in emotive bit depth and increased it accordingly.[31] These composers organized other musical traits such as pitch class and rhythmic values to determine the emotive bit depth of each musical spectra. This ensured that the "interpolation" methods and patterns used to move along the musical spectra established a piece's identity.

Any reader with reservations about the previous section's discussion on emotive bit depth and musical interpolation in relation to musical parameters may be unaware of the technical limitations video game composers faced at the end of the 20th century. These dual composer-programmers had the monumentally tedious task of translating all

31 *Grove Music Online*, s.v. "Serialism," accessed February 18, 2021, https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000025459.

of their music into numbers that would then be sequenced back into a low-fidelity version of where they started.[32] "8-bit music" and "chiptunes" are common phrases used to summarize all of the music written for this medium using programmable sound generators together. The former categorization has led to the misconception that the music is produced with a bit depth of 8, when in reality it is highly misleading and unrepresentative of the many varying sound chips that graced consoles for generations.

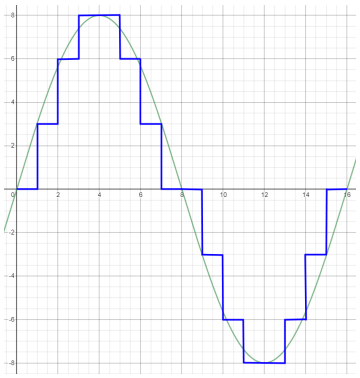

Figure 6: Approximation of a Sine Wave with 4-Bit in Blue, 16-Bit in Green

The sound chip for the Atari 2600, which was released in 1977, can be found in its Television Interface Adapter (TIA). The TIA had 1-bit polyphony (meaning only two monophonic voices were possible at a time), 4-bit volume control (meaning there is a dynamic range of sixteen different possible volumes), 5-bit frequency control (meaning there are only 32 possible pitches) and 4-bit timbral control (meaning that only sixteen different waveforms were possible).[33] The sound chip's timbral control was 4-bit, so it would also have to approximate its waveforms based on that restriction. The blue

32 "The Rise of VGM | Diggin' in the Carts," Red Bull Music, September 11, 2014, YouTube video, 29:01, https://youtu.be/4uvClBGhJyA.

33 Karen Collins, "Flat Twos & The Musical Aesthetic of the Atari VCS," Popular Musicology Online, 2006, http://www.popular-musicology-online.com/issues/01/collins-01.html.

waveform in Figure 6 shows an approximation of a sine wave within these parameters, with a hefty amount of jagged edges that will prevent it from truly sounding like what it is trying to emulate. The 5-bit frequency control made it difficult for composers to program precise pitch designations, because it divided the spectrum equally by a specific amount. The pitch-to-frequency relationship is best represented by a logarithmic function, so the computer chip's linear interpolation caused generators to perform the music out-of-tune compared to standard equal temperament. These restrictions were short lived - just over a decade later, sound chips in newer consoles such as the Super Nintendo Entertainment System (released in 1990) upgraded to 16-bit audio, drastically enhancing timbral possibilities and removing the difficulties associated with pitch and volume control.[34]

The use of programmable sound generators in video game sound design and music mimicked the aforementioned development of keyboards in the past, but lasted less than three decades in contrast to multiple centuries. These computer chips that procedurally generated audio in real-time met their end during the fifth generation of gaming in the mid-1990s (which included the Sony Playstation and the Sega Saturn consoles) with the introduction of sound cards in consoles that used pulse-code modulation[35] (PCM) to decode digital audio files on a games' CD-Rom. This

34 Rodrigo Copetti, "Super Nintendo Architecture: A Practical Analysis," Rodrigo's Stuff: At the service of good ideas, September 23, 2020, https://www.copetti.org/writings/consoles/super-nintendo/.

35 Pulse-code modulation uses square waves to convert analog signals into digital information. Computers need a sample rate to tell the DAW how fast to check for a new signal - a sample rate of 48,000 means it will check that many times a second.

development allowed composers to import fully orchestrated and produced tracks from their DAWs directly into game scenes.

A drastic shift in video game composition began as soon as this was a possibility. Composers can now utilize professional studio musicians in order to have unlimited expressive control over a recorded live performance, and then edit and balance the individual stems in digital workspaces that they are more familiar with. This all becomes completely recontextualized once it is exported as an audio file, however. A piece of recorded audio takes what was once a discrete initial performance and turns it into a concrete object where those parameters are no longer controllable independent of one another. Just as keyboard designers of the past sacrificed certain musical traits to enhance others, the introduction of recorded audio in video gaming replaced the control of primitive procedural audio with higher resolution audio fidelity.

Pierre Schaeffer and Abraham Moles refer to the concrete object as *l'objet sonore*, or *the sound object*. Moles determined that *sound objects* "contained three dimensions: amplitude (loudness); frequency (tone); and time (duration)," and that any piece of music was actually just a "sequence of sound objects."[36] While sound objects are multidimensional in their sonic structure, those dimensions become directly tied together once it is recorded. Increasing the speed of an audio file will naturally increase its frequency, and displace its amplitude peaks at different locations in time.

This was not necessarily a problem. Up to this point, video game music stuck to fixed musical forms that looped, so it made complete sense to do all of the laborious

36 Thom Holmes, *Electronic and Experimental Music: Technology, Music, and Culture* (New York: Routledge, 2012), 49.

work in a DAW instead of dealing with the time-consuming methods that existed before. Unfortunately, this also means that the emotive bit depth of just about every musical parameter will revert to 1 - the track will either be playing (on), or it will not be (off). While the track's volume can be adjusted, this has limited effect on the overall emotional impact of the piece and is only done as a means to mix it against sound effects and voiceover. An anecdote of this decrease in musical control can be heard in the soundtracks of the *Mario Kart* franchise from the third game (*Mario Kart: Super Circuit*) to the fourth game (*Mario Kart: Double Dash*). *Mario Kart: Super Circuit* and the previous two games in the series sequenced music through the programmable sound generators.[37] *Mario Kart: Double Dash*, released two years later for the GameCube (Nintendo's first console to run games from CD-Roms), marked the series' first foray into using recorded audio files.[38]

Games in the *Mario Kart* franchise amplify tension by speeding up the music's tempo during the final lap of the race. In the first three iterations, the pieces would speed up while no other musical parameters were altered thanks to the flexibility of programmable sound generators. When the music is sped up during the final laps in *Mario Kart: Double Dash* and all recent iterations of the franchise however, the overall pitch of the entire composition is also shifted as a result of time-stretching the audio file, and therefore an initially unintended harmonic modulation is forced to occur. The faster speed of the audio files leads to highly unrealistic perceptions of the initial performance,

37 "Mario Kart: Super Circuit," (Kyoto: Intelligent Systems, July 21, 2001).

38 "Mario Kart: Double Dash!!," (Kyoto: Nintendo Entertainment Analysis & Development, November 7, 2003).

and the presence of a musical uncanny valley emerges. On one hand, this raised modulation can be seen as a "feature" of the game and has now become somewhat iconic. Also, games in the *Super Mario* franchise boast a cartoony and surreal aesthetic where realism is not an immediate goal. Seeing how this was never the intention in the original three games of the franchise raises two questions: how did the developers feel about this new limitation, and would this type of audio manipulation benefit or hinder other games that strive for a more hyper-realistic setting? Many modern video games do their best to seem as realistic as possible – in fact, removing *ludonarrative dissonance* is an important part of deepening immersion into video game worlds. Ludonarrative dissonance is the intellectual clash that is created between what the game narrative asks the player to do, versus how the gameplay actually plays out the narrative. Unrealistic transformations of audio files and incessant repetitions of the same tracks throughout a game aggravate ludonarrative dissonance, and only serve to remind players that they are playing a game.

If the developers at Nintendo disliked this creative barrier, there is at least one hypothetical solution. *Mario Kart* composers would need to create a second audio file of each music track at the desired faster tempo and switch to that second file in the final lap so that no digital modulation occurs. This is not a bulletproof solution though - the overall music file size on the game cartridge or disc would almost double, a luxury that most games cannot accommodate when sharing space with visual assets.

Composers quickly saw the drawbacks of using audio files in game music, and began to develop new methods that are now referred to as *adaptive music*. This is a broad term that refers to different compositional techniques such as *vertical remixing* and

horizontal resequencing that make use of more than one simultaneous audio file to combat the disparity of 1-bit emotive bit depth.[39] Vertical remixing is accomplished by exporting each track separetely from a DAW (rather than all at once) and fading different tracks in and out with gameplay. This means that orchestrational decisions can be made in real-time without disrupting the structural flow of the piece. An orchestral piece could begin with a contrabass and violoncello ostinato, but then wait for the ideal time during gameplay to fade in the higher string layers. The use of stems gives the game's audio engine freedom to process and mix individual tracks separately to create gradual orchestration shifts over a fixed musical idea, but is not as useful for making urgent structural changes. Horizontal resequencing crossfades or immediately shifts between two different completely produced music tracks. The hypothetical *Mario Kart* solution is an example of horizontal resequencing. This technique has many advantages and works well for dynamically switching between different formal sections of a substantial piece, but horizontally resequenced music must bake all DSP effects such as reverb and delays into each audio file. Otherwise, any processing in the game's audio engine will be applied to the entire master track.

Both of these techniques fix the burden of 1-bit emotive bit depth from a single audio file, and greatly increase it to a more desirable resolution. The advances of adaptive music correlate to how Abraham Moles referred to music in general as a sequence of sound objects. Composers realized that limiting video game music to a single sound object also limited the musical potential, so vertical remixing and horizontal resquencing

39 Winifred Phillips, *A Composer's Guide to Game Music* (Cambridge, Massachusetts: MIT Press, 2007) 188-198.

both rely on more smaller sound objects to sequence new possibilites. Both techniques present the same problem as the hypothetical *Mario Kart* dilemma though - more individual audio files means composers and producers exponentially increase their needed file size capacity for music. If a project is broken into eight different layers and all of those layers are exported individually, the emotive bit depth has increased to 8 (256 different mixing possibilities), but the necessary file size has multiplied by eight if they are exported at the same quality. The latest installment in the *Final Fantasy* franchise, *Final Fantasy 7: Remake* (*FF7:R*) uses horizontal resequencing to switch between variations that represent a shift from casual exploration to thrilling battles as a way to prevent the looped structures from getting too repetitive.[40] Some locations even augment this duality by establishing orchestrational identities for certain characters in a space while maintaining melodic leitmotifs that refer to the macrolevel. The Wall Market section of the game, for instance, has three distinct variations to represent a "Trio" of non-player characters that look over each section of the town. The game's original soundtrack is seven to eight hours in length, due in large part to the different musical variations that were produced. This begs the question: how much did the massive soundtrack contribute to the game's massive 85 gigabyte download size, and could this have been avoided?

While composers have continued to push the boundaries of what is possible with adaptive music over the past two decades, few have postulated whether strictly adhering to the use of audio files is a sustainable model. When the evolution first occurred, it made sense to embrace higher audio fidelity over the flexibility of procedural audio. Despite

40 "Final Fantasy 7: Remake," (Osaka: Square-Enix Creative Business Unit 1, April 20, 2021).

this, the need for sufficient memory optimization is now an important roadblock to truly dynamic music systems. Some audio designers initially expected that the transition to audio files would only be temporary, and waited for CPU power to reach a point where procedural audio was viable again. Sony Computer Entertainment developers Jason Page and Michael Kelly informed the audience at GDC 2007 that "more developers will be using MIDI-based technology to power their in-game tunes," touting that "file sizes would no doubt be smaller and load times would be exponentially quicker."[41] At the same convention, veteran video game composer Koji Kondo (famous from *Super Mario Bros.* and *The Legend of Zelda*) lamented pre-recorded music and said that it "might as well be piped from a source outside of the room."[42] Despite these prophecies and griefs, no advancements have been made in the past fourteen years to usher in this new era. While many of the barriers to this era are still highly technical, audio designers' complacency with the existing workflow is exacerbating the situation.

C. Stagnation of Audio Middleware

"You must have an extensive knowledge of audio middleware options such as Wwise or FMod." Almost every single video game audio job application in the past decade has listed this skill as a requirement. This quote specifically references the two most popular audio middleware softwares on the market: Audiokinetic's *Wwise* and

41 Eugene Huang, "GDC: next-gen audio will rely on MIDI, say Sony," GemePro, March 16, 2007, https://web.archive.org/web/20080511031148/http://www.gamepro.com/news.cfm?article_id=106508.

42 Koji Kondo, "Painting an Interactive Musical Landscape," GDC Vault, March 2008, https://www.gdcvault.com/play/754/Painting-an-Interactive-Musical.

Firelight Technologies' *Fmod*. Audio middleware is an individual sound engine that can be integrated with a company's proprietary game engine to do all of the heavy lifting for any necessary DSP or musical timing events. There are myriad advantages to the use of audio middleware, but software updates in recent years have focused on maintaining seamless integration with the most popular game engines (UE4 and Unity) instead of building or evolving tools for composers and sound designers.

Game engine developers typically focus on designing and enhancing visual attributes such as gameplay physics, 3D rendering, animation interpolation, etc., so removing the need for these developers to also focus on audio programming is an attractive prospect. These sound engines must exist, because all visual aspects of a game are rendered at frame rate which commonly idles between 30 and 60 frames-per-second. On the other hand, professional quality audio is currently processed at sample rates of 48,000 Hz, meaning that the computer needs to calculate any necessary audio changes that many times a second. These differences exist because our eyes filter natural stimuli slower than our ears. According to research, it takes "our brain at least one-quarter of a second to process visual recognition" while we "can recognize a sound in 0.05 seconds."[43]

Optimization is a critical aspect of audio middleware. The tools are used by both composers and sound designers, so differences in asset metadata such as file type (WAV vs. FLAC) and exported sample rate or bit-depth may occur. Audio middleware handles this problem by compressing all audio files down into the same format to improve game performance and free up as much space as possible for visual game assets. It also

43 Molly Webster, "Ears don't lie," Radiolab (WNYC Studios, December 28, 2012), https://www.wnycstudios.org/podcasts/radiolab/articles/257870-ears-dont-lie.

optimizes the creative process - the user interface of each software bears a much closer resemblance to DAWs than game engines, ensuring that the creative personnel are not intimidated by a completely unfamiliar user interface. Industry-standard software allows freelance composers and sound designers to move from one project to the next and implement their audio in a familiar way each time instead of having to learn a studio's proprietary engine. *Wwise*[44] and *FMod*[45] even have readily available plugins that smoothly integrate with UE4 and Unity.

Currently, audio middleware is absolutely necessary for composers who want to implement adaptive music. To vertically layer or horizontally sequence their music, an audio clip cueing system moving at audio sample rate is essential to ensure that new layers launch at the appropriate time without perceptible gaps. Programming this sort of system in UE4 blueprints would cause significant auditory problems. All actions triggered through this system would be called on the *game thread*, so clip launching would be exclusively tied to frame rate rather than sample rate. Epic Games' audio programmers introduced an experimental audio component called *UTimeSynth* in Unreal Engine update 4.22 that allows this level of synchronicity, but it is still currently in beta and is a rather convoluted method compared to the way it can be accomplished in Wwise or FMod.[46] A full Blueprint library known as *Quartz* has been in development since 2020

44 "Wwise SDK 2019.2.9," *Audiokinetic* (Sony Interactive Entertainment, February 1, 2021), https://www.audiokinetic.com/library/edge/?source=SDK&id=releasenotes.html.

45 "FMOD Studio 2.01.08," *FMod* (Firelight Technologies, February 2, 2021), https://www.fmod.com/download.

46 "Unreal Engine 4.22 Release Notes," Unreal Engine, accessed February 18, 2021, https://docs.unrealengine.com/en-US/Support/Builds/ReleaseNotes/4_22/index.html.

that will simplify *UTimeSynth*'s interface and allow all audio clips (including sound effects) to be triggered and quantized at standard rhythmic note values.[47]

These softwares are not without problems - as previously stated, all musical expression and variance is completely tied to its individual audio file. In an effort to maintain compatibility with UE4 and Unity, both Wwise and FMod have significantly neglected the development of procedural audio tools such as digital synthesizers and sequencers that could solve these issues and give composers greater musical freedom. Of the two, Wwise has more thoroughly promoted this style of composition compared to FMod. As part of its "Interactive Music" work unit, composers can load MIDI files and connect individual tracks to a specific sound source.[48] Through this, entire sample libraries could conceivably be loaded into Wwise and sequenced note-for-note. This is a powerful tool, but if composers want the sound source to be a synthesizer they have a dearth of options. Wwise's most artistically functional audio generators are *SoundSeed Grain* and *Synth One*.

SoundSeed Grain[49] is a granular synthesizer that will always generate unique soundscapes, but has limited functionality when matching MIDI pitch to a specific frequency. Granular synthesis still relies on an audio file to retrieve its microscopic grains, so this method requires both the use of sample data and procedural audio, which

47 Jill Ramsay, "Quartz," last modified June 2020, https://portal.productboard.com/epicgames/1-unreal-engine-public-roadmap/c/199-quartz.

48 FMod includes even less control to the average composer. According to Firelight Studios' CEO Brett Paterson, MIDI files must be accessed and read through the C++ API and attached to "Programmer Sounds" which are essentially just samplers. Brett Paterson, "MIDI Files," last modified November 2016, https://qa.fmod.com/t/midi-files/12804.

49 "Wwise SoundSeed Grain," accessed February 22, 2021, https://www.audiokinetic.com/library/edge/?source=Help&id=wwise_soundseed_grain_plug_in

could be too demanding on the CPU. *SynthOne*[50] works perfectly as a subtractive synthesizer, but it only includes two rudimentary tone generators that each have a pool of four waveforms: sine, triangle, sawtooth, and square. It does allow frequency modulation and pulse-width modulation of the first generator, but assigning Wwise's real-time parameter control (RTPC) to gameplay does not immediately present the highest fidelity due to limited generator polyphony and lack of analog-style unison note doublings. This tool could be great for a procedurally generated chiptune video game soundtrack, but may fall short in a triple-A game with a cyberpunk setting.

Guy Whitmore (composer and proponent of enhanced adaptive music) gave a presentation sponsored by Audiokinetic at the beginning of 2019 to demonstrate some of his current methods for achieving greater control despite the software's limitations. He showed how a violoncello sample library built in minor thirds could promote a much more desirable performance of the game's music by providing higher-precision note cutoffs at the ends of phrases. Building a sample library with files that are each a minor third apart allows for better memory conservation than if every note was recorded, but will not create noticeably artificial time-stretching when they are repitched. Procedural reverb in Wwise will naturally decay, but will falsely stop if an audio file with pre-rendered reverb is halted from a note off MIDI message. He also made a point to acknowledge Wwise's poor selection of software synthesizers. His vision for the future (which he calls "a dream that I thought would be here by now")[51] of adaptive music

50 "Wwise SynthOne," accessed February 22, 2021, https://www.audiokinetic.com/library/edge/?source=Help&id=wwise_synth_one_plug_in.

51 Guy Whitmore, "Getting Ahead of the Game Utilizing Music Design Templates in Wwise," Audiokinetic, February 5, 2019, YouTube video, 1:10:20, https://youtu.be/_FWuiha_dw4.

would allow for real-time timbral control through dynamic filtering and other parameter changes. Whitmore admitted that his ideal situation would be to import his existing VST synthesizers and samplers, but there are many reasons why that could never happen with existing licensing rules. If a company's software synthesizer was purchased for use by a singular composer or music producer, the user license allows them to use it only for personal creation within a DAW, not its embedding into another software. Before this can happen, entirely new licensing agreements will need to exist that stipulate how and where the code can be implemented.

Epic Games is currently trying to enter the procedural audio world in UE4 with its new proprietary sound engine introduced in update 4.16.[52] Since its release in May 2017, continual updates have added four different audio generators built with the new *USynthComponent*[53] class: *UModularSynthComponent* (a subtractive synthesizer with deep patching capabilities) and *UGranularSynth* (a granular synthesizer) function similar to Wwise's *SynthOne* and *SoundSeed Grain* respectively, while *USynthSamplePlayer* functions as a standard sampler. The most recent tone generator, *USynthComponentMonoWaveTable*, is a wavetable synthesizer that provides extensive timbral control through waveform interpolation of a preloaded table. All of these components can be added to Actors in the game level and have blueprint libraries that make parameter control a much smoother process than moving back and forth between

---

52 Alexander Paschall, "Unreal Engine 4.16 Released!," last modified May 24, 2017, https://www.unrealengine.com/en-US/blog/unreal-engine-4-16-released.

53 All applicable subclasses can be found here. "USynthComponent," Unreal Engine, accessed February 22, 2021, https://docs.unrealengine.com/en-US/API/Runtime/AudioMixer/Components/USynthComponent/index.html.

audio middleware and the game engine. Sadly, all of these components currently suffer from two main problems: they are all monophonic, and cannot be sequenced in a musical way through the game engine.

These game engine advances are the brainchild of Epic Games' Lead Audio Programmer, Aaron McLeran. McLeran started his video game audio career in 2008 when he assisted Brian Eno in the creation of *Spore's* generative music score, so procedural audio has been an interest of his for more than a decade. McLeran and the rest of the UE4 audio programming team hosted an informal virtual conference called *State of Audio* in April 2020 amidst the COVID-19 pandemic where they discussed the recent advances and possibilities for the future. During this event, they encouraged other developers to dive into their constantly growing C++ API that has pre-built classes and functions for common DSP and synthesis techniques such as envelopes, filters, and low-frequency oscillators.[54] While none of these classes have been implemented in a way that can unfold sequenced procedural music at sample rate yet, it creates exciting speculation about future plans for their tools and what third-party developers could integrate directly into the game engine itself. In fact, with just a bit more time and resources, these developments could essentially render game audio middleware useless for UE4 projects.

As of the end of 2020, the game industry has moved into a completely new generation of hardware with the release of the PlayStation 5, the Xbox Series X, and new high-powered graphics-processing units (GPU) for gaming PCs by Nvidia and AMD. This hardware could make some of the concerns that audio middleware is supposed to

54 "State of Audio in 4.25 | Inside Unreal," Unreal Engine, April 2, 2020, YouTube video, 2:21:12, https://youtu.be/wux2TZHwmck.

tackle irrelevant. The PlayStation 5, for instance, has a dedicated sound chip called the "Tempest Engine" that aims to provide deeper immersion through the auditory landscape.[55] Currently, most ambient soundscapes in video games are accomplished through a few layered stereo tracks to keep game performance stable. If you hear idle city chatter, those voices are probably coming from a single stereo track rather than every non-player character in the game. This means that the audio is not truly three-dimensional - it is not reacting to where the player is. The Tempest Engine can allow "hundreds of advanced sound sources"[56] so that sound designers no longer have to choose what sounds can be three-dimensional. That number directly correlates with complex sound calculations such as fast-fourier transforms (FFTs) that enhance dimensionalization and location, so the potential for less computationally expensive real-time synthesis and sequencing for two-dimensional music is palpable. If this is the case, will the CPU optimization that audio middleware can provide really be necessary anymore?

As software and hardware continue to develop, audio middleware programs will need to upgrade their capabilities to stay relevant if they hope to compete. If Unreal Engine introduces completely functional adaptive music tools along with its extensive library of procedural audio tools, composers such as Guy Whitmore may completely leave middleware behind with certain projects. Unless middleware companies switch their business models from focusing on upkeep to evolving the current state of audio, the

55 Roland Moore-Colyer, "PS5 and 3D audio: Everything you need to know," Tom's Guide (Tom's Hardware, October 6, 2020), https://www.tomsguide.com/features/ps5-and-3d-audio-everything-you-need-to-know.

56 Mark Cerny, "The Road to PS5," PlayStation, March 18, 2020, YouTube video, 52:44, https://youtu.be/ph8LyNIT9sg...The Road to PS5.

game industry may see a drastic shift in the holistic sound design process that has existed at studios for more than a decade.

D. Soundtracks Should Respond to Multidimensional State Machines

Finite-state machines are critical in video game programming. Matyas Lancelot Bors defines a finite-state machine as "any device storing the state of something at a given time...The state will change based on inputs, providing the resulting output for the implemented changes."[57] Briefly returning to the analogy of keyboard development throughout the past three centuries, the harpsichord is a perfect example of an instrument that changes the state of its timbre through *stops*. Performers can change the disposition of a harpsichord by adding an extra choir of strings that sound an octave higher. This moves the keyboard state from a unison timbre with each key press to a more elaborate collection of harmonics.

Adaptive music soundtracks use finite-state machines to switch or blend between audio files that represent different flavors or developments of the same piece. Ben Prunty uses this technique in his soundtrack for *FTL: Faster Than Light*, a game about deep space travel. As players are exploring the cosmos, they are lulled into a false sense of security through a plethora of dreamy synthesizer soundscapes.[58] When hostile extraterrestrials inevitably begin to attack the ship, a layer of percussion is added that enhances the overall intensity to acknowledge a change of game state from "exploration"

57 Matyas Lancelot Bors, "What is a Finite State Machine?," last modified March 10, 2018, https://medium.com/@mlbors/what-is-a-finite-state-machine-6d8dec727e2c.

58 "FTL: Faster Than Light," (Shanghai: Subset Games, September 14, 2012), https://store.steampowered.com/app/212680/FTL_Faster_Than_Light/

to “battle.” This is not a rare use of adaptive music - most modern video games have some sort of musical variation that occurs when a battle begins. This is an improvement from the past though - in the first few decades of video games, the music would pivot to an entirely different composition when battles started rather than developing the currently playing one. This new trend shows a deeper understanding of how to increase emotive bit-depth of existing music.

Adaptive soundtracks tend to only adjust musical state based on the level of danger, somewhat insinuating that peril is the only game state that should relate to music. *Spyro: Reignited Trilogy* has an innovative and novel method of connecting the soundtrack to game state machines other than violence.[59] In this remastered version of the first three games from the series, players take control of a teenage, purple dragon that traverses his world by charging head first into action. This means that there are three different movement states: he is either standing still, walking, or charging. Each of these states has distinct musical variations in the soundtrack to reflect its variable physical tension. The walking variations are recreations of the original trilogy music composed by Stewart Copeland (more famously known as the drummer from the popular music trio The Police). The charging variations add new percussive layers that change the rhythmic feel of the pieces to double-time, which represent a sense of urgency that was not in the original games. The standing still variations put the music through low-pass filters that allow the player to stop focusing on the action, and take in the scenery.

59 “Spyro: Reignited Trilogy,” (Novato, CA: Toys for Bob, November 13, 2018), https://store.steampowered.com/app/996580/Spyro_Reignited_Trilogy/.

It is refreshing to see composers connect music to game states other than inherent danger, but use of rigid audio files severely limits this potential while visual elements of a game are not limited in the same way. Developments in the animation field now allow 3D models to be rigged with internal skeletons - each limb on a skeleton can be shifted along predetermined joints to emulate real-world movement. This is being done digitally, so animators can capture frozen character positions and then use keyframes to interpolate between those different poses. Game engines such as Unreal Engine 4 have taken this one step further. In UE4's Animation Blueprint, designers can interpolate between entire pre-existing animations, a process they call "*animation blending*."[60]

Discrete movement states such as walking and running can all have animations attached to them, and this interpolation allows those transitions to exist without appearing jarring to the player. This also allows characters to have different speeds other than just walking or running - just like on a lightbulb with a dimmer, this lets players move at their own speed based on joystick axis control which typically has a bit-depth of 16, not just 1. Walking and running animations insinuate that the player is currently standing, but if a moment of gameplay requires stealth they will be required to crouch for quieter footsteps. Some games even allow players to go prone in tall grass to completely avoid visual detection. In combination with smooth changes in forward velocity, these physical stances create a *multidimensional state machine* - player stance is independent of player speed.

60 "Animation Blueprints," Unreal Engine, accessed February 22, 2021, https://docs.unrealengine.com/en-US/AnimatingObjects/SkeletalMeshAnimation/AnimBlueprints/index.html.

Imagine a new racing game where the adaptive music reacts exclusively to the player's driving. The composer's first mission is to determine what the different states are: forward acceleration, backwards acceleration, idling, turning, and braking. Unlike in *Spyro: Reignited Trilogy*, the unchanged thematic material could be present in the idle position with variations being made based on the vehicle's dynamic momentum. Forward acceleration could add rhythmic layers to the music just like in the *Spyro* charging movement state example. Thanks to multidimensional state machines, this could be more varied than the charging example. It is a gradual process to move from a car's lowest speed to its top speed - achieving maximum velocity could unleash a variation that properly emulates the vehicular adrenaline rush. Backwards acceleration (which usually acts as a brief alignment correction) could trigger an audio reversal sound effect. Turning either left or right could quite literally take advantage of stereo audio by dynamically panning certain musical elements in those respective directions. Lastly, sudden braking could trigger a drum break - a technique used in popular music that isolates the drums for a more virtuosic moment.

All of the above examples specifically relate to nonlinear, gameplay actions that dynamically respond to the player, but even linear cutscenes in recent games allow for visual states to change. To animate dramatic moments of a narrative with the highest graphical fidelity, early developers would prerender cutscenes as video files. This changed in the past decade - enhanced hardware and graphics rendering algorithms allow these scenes to be completely procedurally generated in real-time. This allows certain player choices to be seen in the cutscene, and some moments even unfold while the

player is still in control of their character. Leveraging multidimensional state machines means that while the narrative of a game in a particular moment may be completely linear, other elements of a game are not required to be. In Insomniac Games' 2018 video game *Marvel's Spider-Man*, players are given the opportunity to put on different superhero suits that each have special powers.[61] The in-game cutscenes show Spider-Man in the player's most recently chosen suit, preventing any inconsistencies between narrative and the gameplay. These inconsistencies are examples of ludonarrative dissonance.

This level of visual flexibility continues to bolster as developers embrace procedural generation for graphics, which makes an even more convincing case to move audio in this direction as well. Some software is even being developed to deal with the problems that fixed recordings cause. JALI Research in Toronto, Ontario's eponymous application forces 3D facial animations to conform to spoken words through analysis of vowels and consonants in the computer graphics simulator Maya. JALI listens to and tracks the placement of phonemes of eight different languages in order to adjust facial tics such as tongue placement and cheek height.[62] While this is not an indictment against pre-recorded voice-over, it does present a concern that developers are growing complacent with the current state of audio and could become reluctant to implement any drastic changes in the future while the workflow stays the same. This complacency can be seen in regards to music with *Just Shapes and Beats*, a game where players move

61 "Marvel's Spider-Man," (Burbank: Insomniac Games, September 7, 2018).

62 "How JALI Research drives CYBERPUNK 2077 Facial Animation & Multibilingual Speech - Night City Wire," Jali Research Inc, November 26, 2020, YouTube video, 2:48, https://youtu.be/UNGvHkGv5Qk.

around and dodge projectiles that spawn in relation to the beat of the music. Levels are created in conjunction with an existing electronic dance music track, so gameplay events are locked onto a restrictive timeline. The level designers cleverly built levels where replayability is assured through the randomization of projectile spawn points, but their temporal placement does not have the same malleability.[63] Essentially, the event triggering and animation of each level is beholden to one musician's vision. It would be more compelling if gameplay decisions had a tangible effect on the musical output, thus producing a music-to-animation feedback loop where each playthrough guarantees a fresh experience.

Adaptive music is triggered by state changes, but there is always a clear distinction drawn between when the same piece should have variations, and when it should just change to a new composition altogether. Video games commonly switch to a new piece of music when the player has changed their setting in some way. Nintendo's popular franchise *Animal Crossing* tasks players with building a village in a lone enclosed area, so in lieu of changing music through level transitions it switches based on real-world time. Using the console's internal clock, the game detects when a new hour has started and changes game states to affect the position of the in-game lighting (through the moving sun) as well as the currently playing piece of music. This is a calculated choice so that players do not become exhausted with any given melody.

The time-progression mechanic has persisted in each new installment for nearly two decades. The most recent iteration, *Animal Crossing: New Horizons*, takes it one step further by allowing the music to react to in-game weather as well. To achieve this, the

63 "Just Shapes and Beats," (Quebec City: Berzerk Studio, May 31, 2018).

jovial thematic material of each track removes percussive elements when rain or snow establishes a mellower mood. The music changes based on more than one game state (time and weather), so the soundtrack is responding to a multidimensional state machine.[64] Even though there are multiple dimensions of game states, it is important to realize that it is still finite: 24 pieces (time) each with 3 variations (weather) leads to 72 different states. The tracks are varied with vertical remixing – certain instruments are in each variation, while others swap out for different tracks. This method limits the impact music has on the file size so that it is not exactly tripled, but it still does require a larger amount of space.

If music is procedurally generated rather than played from an audio file, the whole gamut of gameplay states could be symbolized in ways that truly orchestrate player actions. Composers of linear multimedia such as film and television have the luxury of a fixed timeline to precisely attach impactful instrumental flourishes to a story's narrative. Without a fixed timeline, there is currently no way for a video game composer to add these decorations since they will never know the underlying harmony of an audio file as it is looping during gameplay. If the music is being procedurally generated in some way instead, the MIDI chord data can be analyzed in real-time allowing these sonic ornaments to occur whenever some exciting visual gesture happens. In fact, Nintendo's 2017 game *Super Mario Odyssey* does this to many of the sound effects throughout its abstract worlds. When the main protagonist Mario transforms into unusual creatures throughout the game, these sound effects respond to the game state of his current physical form. If a

64 "Animal Crossing: New Horizons," (Kyoto: Nintendo Entertainment Planning & Development, March 20, 2020).

sound effect is represented by a musical arpeggio, the arpeggio directly reacts to both the underlying harmony and the tempo of the background music.[65]

Slight expansion on this idea could push these motives from diegetic sound effects to a non-diegetic soundtrack. Turn-based battle systems are amenable guinea pigs to conceptualize these adaptive music concepts. Games from the *Final Fantasy* and *Dragon Quest* series pit players against enemies through conflicts that wait for actions to be taken on each side, similar to a game of chess. The combat is structured entirely on these player decisions, so background music needs to loop to some degree in order to reflect the unpredictable ending time.

Allowing soundtracks to adapt to every aspect of a game's ludonarrative provides a compelling solution to enhancing adaptivity in video game music. Similar to how procedural generation has allowed fixed timeline cutscenes to respond to player choices, procedurally generated soundtracks could allow looped pieces of music with fixed forms to conform in the same way. Without this, soundtracks will aggravate the ludonarrative dissonance that critics continuously point out in contemporary games. As gameplay experiences become more expansive, dynamic, and multidimensional, music will not be able to maintain the pace of all other elements without moving from intractable and unsustainable audio files.

65 Joey Lopes, "Sounds in Super Mario Odyssey Harmonize with the Background Music," JalopesTL, November 4, 2017, YouTube video, 3:49, https://youtu.be/U5-YDxH6It8.

E. “Repetition Fatigue”

Winifred Phillips mentions in her book *A Composer’s Guide to Game Music* that the most common critique of looped video game music is the problem of “*repetition fatigue*,” a musical dilemma that resembles the law of diminishing returns in economics.[66] There are just a couple more anecdotes from the past decade and a half that are of special cause for concern and show the full scope of how “repetition fatigue” has affected the industry. *Auditorium*, released in 2008 as developer duo Cipher Prime’s (started by developer William Stallwood and composer Dain Saint) first project, demonstrates the low resolution of emotive bit depth that this restriction can bring. This puzzle game tasks players with refracting different streams of colored light into their corresponding light buckets using user-adjustable bubbles. As the buckets fill up, stemmed layers of the orchestrated soundtrack increase to their max volume. This could have been a textbook example of vertical remixing, but the level design made it impossible. Each piece of music corresponds to a larger collection of microlevels, with each microlevel introducing new instruments as the player progresses. These microlevels end when players fill in all of the buckets and hear the current fleshed-out orchestration in its full glory, but the transitions create a fragmented musical experience by lowering all stem volume levels back to zero.

In a 2010 interview with Indiepub, creative director William Stallwood explained that the beauty of the game was that the level design allows players to create their own

66 Winifred Phillips, *A Composer’s Guide to Game Music*, 66.

solutions.[67] This meant there was more than one way to refract the light into the buckets. The music only responds to the light's final destination though, so there is no conceivable way that the adaptive soundtrack can respond to a player's unique solution. This shows a clear dichotomy between the concept of a puzzle and a game - puzzles can have more than one solution, but they will still inevitably end in the same way. Their second game *Fractal: Make Blooms Not War* learned from many of *Auditorium*'s problems, and used DSP filtering to introduce the new orchestrations as the levels unfolded rather than automating to an empty volume. Unlike their first game which relied on the music as structure for level design, *Fractal* tasked players with combining hexagons into much larger ones. There was a caveat - players had a limited number of moves before they lost the level, adding an element of competition into the mix. To accommodate this freer gameplay form, the music starts as a drum beat and then slowly introduces extra stemmed layers of harmony. Sadly, the form of each piece is typically limited to eight-measure phrases and there is never any foreground to give the pieces any clear identity or feeling of accomplishment. If a company advertises that a game has reactive music, there is an expectation that those changes will ultimately be tangibly heard and comprehended, but *Fractal* never quite hits that mark.

These creative drawbacks are ultimately the cause of weak video game audio tools. For music to remain consonant in regards to traditional Western harmony, one stem of a vertically remixed composition ultimately determines the length of all the others. If one of the eight measure-long loops establishes a chord progression, all other harmonic

67 Dain Saint and William Stallwood, "Interview with Cipher Prime about Auditorium HD." Indiepub, October 12, 2010, YouTube video, 3:59, https://youtu.be/I_BdWjnO0ew.

and melodic elements have to conform to the base layer's rhythmic position. In addition, the novelty of an eight measure-long chord progression will wear off after just a few iterations, since the first harmonic layer establishes the only harmonic tension that will be present for the rest of the composition and relies on additive orchestration to simulate tension. After a handful of levels, this technique grows tiresome. If the music was procedurally generated, it could dramatically shift to new keys while maintaining the same harmonic structure. These transformations are only possible if the music is being sequenced – direct transposition would be an easy feat to accomplish by simply adding or subtracting the same pitch value to each note. Throughout the past decade, Cipher Prime's games moved from being tightly connected to adaptive music to eventually moving away from the concept altogether.

Critics of Cipher Prime's technology could chalk the limitations up to their status as an indie game studio and lack of human or monetary resources, but one of the industry's most profitable intellectual properties is not immune from this ailment either. Released in 2018, *Red Dead Redemption 2*, is an open-world game that allows players to do just about anything they want in the Wild West: hunt, tame horses, hijack trains, etc. Open-world structures pose a unique challenge to video game music, because there may not be immediate urgency or progression in the narrative to score in a more traditional way - players create their own stories. Composer Woody Jackson solves this with a technique he refers to as "human granular synthesis."[68] This means that he first viewed every piece he wrote in a fixed musical form, and then blended measures together in the

68 Woody Jackson, "Composer Woody Jackson on crafting the sound for *Red Dead Redemption 2*," AV Club, March 18, 2019, Video, 10:22, https://games.avclub.com/composer-woody-jackson-on-crafting-the-sound-for-red-de-1833376805.

same way that grains of an audio file merge in granular synthesis. If he wrote eight interrupted measures of material, the second musician would start playing measure two before the first musician finishes the first measure. During the recording sessions for the soundtrack, he would perform a singular motif on the guitar and let it ring out for an extended period, and then the other studio musicians would respond with a mutation of Jackson's material. There is no immediate structure in the gameplay, so he matched it with ambient music that has no immediate structure.

The musical issues arise in story missions. Jackson scores the gunslinging by introducing a clear tempo in contrast with the ambient music, so players know when the transition occurs between freeplay and action. In an interview with The A.V. Club in 2019, he admits that when composing this music he uses "all one tempo and one key." This same process was used in the first game of the series as well. All of the rhythmic music in the first *Red Dead Redemption* was composed with a tempo of 130 BPM (or a common division of that such as 65 BPM) and all 24 hours of music for the expansion pack *Undead Nightmare* were composed in the key of A-minor.[69] According to Jackson, advances in technology that were developed in the eight years between the two games made it so that the audio engine went from being able to vertically layer eight stems to fifteen stems.[70] In order to allow for a deep adaptive experience, he recorded as many isolated sound objects as he possibly could. Unfortunately, this meant that each sound

69 Jeriaska, "Myths, Mavericks, And Music of *Red Dead Redemption*," Gamasutra, November 4, 2011, https://www.gamasutra.com/view/news/127900/Myths_Mavericks_And_Music_Of_Red_Dead_Redemption.php.

70 Lily Moayeri, "Woody Jackson and Vox Studios," (Mix, January 11, 2019), https://www.mixonline.com/recording/woody-jackson-and-vox-studios-red-dead-redemption-2.

object had to be in the same key and tempo to ensure that they meshed properly with one another - the emotive bit depth of tonality and tempo was lowered to 1.

If the studio musicians' performances were recorded as digital information rather than concrete audio, the soundtrack could have been implemented to render in real-time. This would have many advantages on the final product. First, the hundreds of hours of recorded audio in the game could have had an exponentially smaller imprint on the download size. The soundtrack uses only a few instruments such as acoustic guitar and violin to achieve a folksy quality, only adding electric guitar and percussion in the more intense moments of the game. These instruments could be sequenced with comprehensive sample libraries, similar to what Guy Whitmore suggested in his 2019 Wwise conference talk. The power of sequencing also means that the actual performance could have more flexible tonality by transposing to any of the twelve main keys, and fully-automatable tempo to increase or decrease intensity as action scenes unfold.

"Repetition fatigue" is not an unsolvable quandary, but without embracing the technical need for procedural audio to generate adaptive music soundtracks in real-time, this part of the medium will reach an artistic stalemate. The multitude of creative and technical drawbacks of recorded audio should be abundantly clear by now, so it is up to a new generation of game audio developers to build the tools of the future if companies in the current era continue to refuse to evolve.

III. Exceptional Examples of Adaptive Game Music

A. "Literature Review"

It is encouraging to see that video game composers are pushing the limits of adaptive video game music, but they can only work within the hardware and software constraints that are provided to them. If a composer joins a project halfway through a game's development, the studio will have already chosen the audio middleware that is being used or they may already have a proprietary engine of their own. In some rare cases, composers work extensively with developers to build a new music system that fits their needs or discover the best implementation method to maximize adaptivity within the existing systems. This chapter investigates a few of these exceptional examples.

B. Using Timbre and Orchestration to Establish Setting in *Donkey Kong 64*

The use of musical signifiers has become a common technique for video game composers who want to establish a time or place in the medium. The presence of non-Western instruments such as kalimbas or sitars often alludes to a distant forest or a desert locale respectively. The presence of sleigh bells in a score commonly signifies that the setting exists in a cold, winter climate. Composer Grant Kirkhope shirked these Exoticism cliches while writing the score for Rare's 1999 game *Donkey Kong 64* (*DK64*) and established the setting through methods similar to word painting in the Renaissance and programmatic orchestration in the Classical and Romantic periods.

Unlike the majority of video game soundtracks from this generation, the music was sequenced in real-time. Grant Kirkhope faced many hurdles while composing music

for *DK64*, and these challenges in turn became part of the overall creative process. Implementing audio with the Nintendo 64 (N64) hardware was a monumentally difficult task. For starters, the N64 was Nintendo's first home console that did not have its own sound chip. This meant that all of the audio had to be processed through either the main CPU or the Reality Co-Processor (RCP).[71] The same hardware units that handled gameplay and graphics had to process audio at the same time, and the first two design elements are always given priority over the latter for the final product. As if this was not already a tremendous barrier to overcome, *DK64* is notorious for tasking players to collect hundreds of items to fully complete the game. This was the first game that required consumers to enhance their consoles with the *Expansion Pak* that increased the N64's RAM by 4MB to manage the necessary computations and giant level sizes.

The console architecture posed a problem, but so did the cartridges that held the individual games. The main advantage of keeping the cartridge format was that they had a higher bandwidth, allowing games to load faster from level to level. However, failing to move to the CD-ROM format like their main competitors (the Sega Saturn and Sony PlayStation) meant that games shipped by Nintendo had to fit inside a much smaller file size limit. This meant that any game asset - including audio files - were heavily compressed. With limited storage space available on the cartridges, Kirkhope chose to work with the N64's built-in instrument sounds and a MIDI tracker instead of a low-

71 Rodrigo Copetti, "Nintendo 64 Architecture: A Practical Analysis," Rodrigo's Stuff: At the service of good ideas, September 23, 2020, https://www.copetti.org/projects/consoles/nintendo-64/.

quality audio file.[72] This decision was critical in establishing how the musical timbre sounded, and how he would eventually arrange all of his music.

The character Donkey Kong actually made his debut as a villain in the eponymous platformer video game, but later became the protagonist in Rare's 1994 *Donkey Kong Country* for the Super Nintendo Entertainment System. Works in the platformer video game genre challenge players by requiring them to jump or climb from one raised surface to the next without falling to their demise or succumbing to surrounding obstacles, usually while collecting context-sensitive objects. Many developers for the remainder of the decade (such as Insomniac Games with *Spyro the Dragon* and Naughty Dog with *Crash Bandicoot*) designed their platformer games by building smaller levels that players could complete in thirty minutes or less. In these cases, composers wrote new pieces of music for each level and sometimes even reused the same piece for multiple levels.

In direct contrast to this world design formula, Rare's games at the end of the 1990s contained massive open-world levels that players could explore without interruption for multiple hours. If Kirkhope followed in the footsteps of his contemporaries and wrote just one composition for each level, each track's originality would eventually diminish and the musical loops could potentially become grating during future repetitions. It would have also been unintuitive to have multiple pieces of music with contrasting material alternating in the same level when that level needed to maintain a sonic identity. To combat this design challenge, he adaptively shifted different

72 Tom Power, "As Donkey Kong 64 turns 20, the devs reflect on it design, the infamous DK Rap, and how a shocked Shigeru Miyamoto created the Coconut Shoot," GamesRadar+, (Future US: December 6, 2019), https://www.gamesradar.com/making-of-donkey-kong-64/.

arrangements of each level's singular material in real-time to maintain interest and prevent it from overstaying its welcome. Kirkhope first experimented with this technique in *Banjo-Kazooie* in 1998, but expanded on and perfected it in an exceptional way in *DK64*.

The original soundtrack for *DK64* includes multiple hours of different music, so this investigation will only explore how Kirkhope composed adaptive music for the first main level of the game - *Jungle Japes*. Players aware of his work on *Banjo-Kazooie* were familiar with Kirkhope's musical style that somehow managed to exude undeniable brightness despite copious amounts of nonfunctional harmony and rapid modal shifts. Fans of the earlier *Donkey Kong Country* series were more acquainted with "darker" thematic tones and more "moody" musical material from composer David Wise.[73] To ease the existing fandom into his personal sound, he borrowed a familiar tune from *Donkey Kong Country*'s level *Jungle Hijinx* for use in *DK64*'s introductory level.

Upon entry into the level, players are greeted with the eponymous *Jungle Japes* theme. Kirkhope arranged the piece with a swinging digital big band and tacked an extra C-section onto the end of the original AABA form. This section showcased improvisatory material that accentuated his personal musical style which is ripe with rapid modulations and harmonic rhythm. It is only present in the entrance theme though - the rest of the variations throughout the level conform to AABA and reorchestrate the material with instruments that signify their distinct locations. After solving a couple puzzles in the level's main entrance, players are briefly funnelled into a cavern where the

73 Tom Power, "As Donkey Kong 64 turns 20, the devs reflect on it design, the infamous DK Rap, and how a shocked Shigeru Miyamoto created the Coconut Shoot," GamesRadar+, (Future US: December 6, 2019), https://www.gamesradar.com/making-of-donkey-kong-64/.

music transforms. He minimalistically orchestrates the *Jungle Japes* material for solo marimba at a tempo more than half the speed of the original theme. Use of mallet percussion frequently acts as a signifier of stalagmites and stalactites in video game music, and the solo instrumentation helps represent momentary darkness and isolation in the scene.

Later in the level, Kirkhope demonstrates his understanding of existing orchestrational tropes from the history of Western music. When the setting's weather transitions from sunny and tropical immediately to a thunderstorm later in the level, he invokes the *sturm und drang* orchestrational style ("storm and stress") commonly used by Classical and Romantic composers. He uses tremolo contrabasses and violoncelli to emulate the rumble of thunder like in the fourth movement of Beethoven's *Pastoral Symphony* and the beginning of Haydn's *45th Symphony*. Level design in Rare's *Donkey Kong* games frequently require the protagonists to traverse the landscape by being shot out of cannon-like wooden barrels. Kirkhope iconizes this in a bonus stage in *DK64's Jungle Japes* by limiting the theme's orchestration to a flute duet. This is potentially another reference to the *Pastoral Symphony* as well as Prokofiev's *Peter and the Wolf* where flutes signify birds, and as a byproduct of flying at a high altitude.

The orchestral variations throughout *Jungle Japes* perfectly convey the scenes with which they are fused with, but the overall implementation is not as smooth as it realistically could be. When transitions are made between each variation, the theme returns to the very beginning of the musical form rather than continuing from where the previous variation currently was. This process could easily be improved if multiple MIDI

files were vertically layered in real-time. This way, the musical form would properly loop without becoming artificially fragmented. The soundtrack is procedurally generated, so smoother tempo shifts could also be made to sew the variations together more naturally without affecting the overall pitch of the entire track. An interpolation between swung and straight eighth notes would also be needed to ensure that the feel from the initial theme could translate properly to the variations.

Many of the orchestrational decisions that occur in the *Jungle Japes* level occur throughout the rest of the game: cave variations always occur as a lumbering marimba performance, barrel courses always harmonize the melody as a flute duet, and so on. This approach gave Kirkhope the opportunity to rapidly iterate each level's material through existing instrument schematics. The development of intelligent MIDI devices could expedite this process even further. By analyzing the harmonic, melodic, and rhythmic structures of a single theme, these devices could potentially reproduce each basic variation in real-time within the given parameters. None of these tools were at his disposal twenty years ago, but this primitive example of video game adaptive music is still quite extraordinary. Despite the surreal complexity of *DK64*'s landscape, Grant Kirkhope managed to reorchestrate music in a way that maintained a world's identity while transporting the audience to constantly shifting physical settings.

C. Adapting Debussy's *Preludes* in a Laborious Manner

In the fall of 2019, the internet was attacked with an influx of memes about the video game *Untitled Goose Game*. This digital satire was inspired by the silliness of the

actual game, where players quite literally took control of a goose on a mission to disrupt the daily lives of people in a small English town. While social media comedians continued this fad for several months, the game also garnered acclaim for its use of adaptive music techniques created by composer Dan Golding. To say that the game's music was composed by Golding is slightly misleading however - all of the musical material was taken from Claude Debussy's *Preludes* for solo piano. The soundtrack excels in its ability to deconstruct the pieces from the discrete proportional time structures found in their written scores, allowing for performances of the piece that never existed before. Despite the novelty, the method Golding used to achieve this effect perfectly exemplifies the shortcomings of the use of audio files in adaptive music.

For the most part, the game is relatively silent except for a few sound effects that relate to the goose's interactions. The music connects to three different game states: when the goose is doing nothing, when the goose is scheming, and when the goose is being chased.[74] Rather than unfolding just as the original score dictates, the flow completely stops when the game's state changes. The first of Debussy's *Preludes* to grace its presence is *Prelude No. 12, Minstrels*.[75] This feels like a calculated and deliberate choice - *Minstrels* is a jovial piece that disregards lyrical melodic lines for gestural piano ideas that thrive on secundal harmony and pentatonic scales. The benefit of this material lies in

74 Dami Lee, "How *Untitled Goose Game* adapted Debussy for its dynamic soundtrack," The Verge, (Vox Media: September 23, 2019), https://www.theverge.com/2019/9/23/20879792/untitled-goose-game-nintendo-switch-debussy.

75 Claude Debussy, *Preludes* (Paris: Durand et Cie., 1910), https://imslp.org/wiki/Pr%C3%A9ludes%2C_Livre_1_(Debussy%2C_Claude).

its ability to avoid musical expectations such as Common Practice cadences that may leave players with anticipation for some sort of resolution.

Another advantage *Minstrels* creates for adaptive music is its through-composed form. This allows the necessary musical time stretching that needs to be accommodated for player actions while remaining completely attached to the form of the players' gameplay. Since the player's goal is to progress through a checklist of different disturbances without ever returning to a previous one, the removal of musical connections between returning A and B-sections establishes a true connection between the score and the player's own narrative. The through-composed nature of the prelude also differentiates itself almost completely from other typical video game soundtracks - nothing ever loops back to the beginning. While previous video games used loop-based music to avoid the difficulty of gameplay temporality adjacent to standard musical form, Dan Golding tackled the issue head on to great acclaim.

Golding acted primarily as a performer and implementer for *Untitled Goose Games's* music, as he performed and cut all of the piano parts that can be heard in the game. When asked about how he decided to implement the audio, he "began by recording two versions of the 'Prelude': one played normally, and one with a much lower, softer energy. The tracks were then split up into different 'stems,' or sections, at the same parts. Though he began by splitting the song into about 60 stems, it didn't prove to be enough." With that initial amount of stems, he was deciding a specific structural emotive bit depth of how the music could connect to the game's actions. By admitting that this was not enough, he was already facing the challenge of using audio files and eventually had to

settle on a much larger amount of stems. Golding used the DAW Apple Logic Pro to split each recording into 400 individual stems, and used procedural reverb to prevent the stems from sounding choppy.[76]

While this method clearly met the goal it set out to achieve, it demonstrates multiple problems about how this level of adaptiveness could become increasingly convoluted for games on a much grander scale and budget than *Untitled Goose Game*. Since there are three distinct game states, two individual music recordings had to be sourced. The chase music relies on standard performances of the pieces that could be realized by any professional pianist, while the second versions are much slower interpretations for the calmer and more introspective moments. If a triple-A studio introduced this level of complexity to a soundtrack it would require unrealistic memory storage needs for experiences that are already reaching the 100 GB size. In addition to this method's convolution on a technical level, it is also unsustainable on a creative and personal level. The amount of busy work that Dan Golding endured as the audio implementer would need to be multiplied on a massive scale to meet the needs of a larger game.

An easier and more sustainable solution to this problem is already possible with currently available software. Since the music is exclusively piano, there is no logical reason why this soundtrack could not have been accomplished with MIDI files and a piano sample library. If the idea of implementing an entire sample library seems far-fetched, the video game *Unravel* did so for a completely unmusical effect. The only

76 Dami Lee, "How *Untitled Goose Game* adapted Debussy for its dynamic soundtrack," The Verge, (Vox Media: September 23, 2019), https://www.theverge.com/2019/9/23/20879792/untitled-goose-game-nintendo-switch-debussy.

reason the sample library exists in that game is to provide auditory feedback when the gamer (who roleplays as an anthropomorphic ball of yarn) steps on keys of a piano while platforming across a fictional living room. More recently, the video game *The Last of Part II* included an entire steel string guitar sample library for the sole purpose of letting players pretend to play chords on the guitar. Rather than recording individual chords, the Naughty Dog audio team realized the possibilities of pooling notes together for adaptive polyphony rather than limiting the performance potential to one recorded instance of each chord.

By sequencing the soundtrack with a piano sample library, Golding could have lowered the music's necessary file size, and the emotive bit depth connected to multidimensional game states could have increased through dynamic control of musical parameters in place of the two different recordings. Recording length information of the soundtrack shows that the fastest renditions of the five different preludes (*Minstrels, Hommage a S. Pickwick Esq. P.P.M.P.C., Les collines d'Anacapri, La serenade interrompue, and Feux d'artifice*) are collectively fifteen minutes long. When rendered as a stereo file at a sample rate of 48,000 Hz and a bit depth of 16, the overall file size is approximately 160 MB. For the sake of comparison, 88 piano note samples (the full range of the modern piano) rendered out individually adds up to only about 40 MB. Armed only with these samples and the corresponding MIDI files, the preludes could have been procedurally generated at a quarter of the size. That only accounts for the five fastest recordings - the second, slower recordings of each piece average at about 1.7 times the time length of the originals, adding about 250 MB to the file size for a total of close to

400 MB for music alone. This creates an even larger memory storage disparity between Golding's implementation method and the use of a piano sample library. With this in mind, even more samples could be added for each note to allow round-robin sample selections to further enhance the procedural realism.

These are the quantitative reasons why MIDI sequencing with sample libraries would have been a better choice for the soundtrack, but there are essential qualitative reasons as well. The two different piano performances exist for the sole purpose of accommodating multiple game states and variable levels of tension within the game. Breaking each recording into individual stems drastically increased the emotive bit depth of temporal flexibility, but that does not account for other expressive parameters such as tempo and dynamics. As it stands, the emotive bit depth of tempo and dynamics is 1 - fast and bombastic in the first recording, or slow and pensive in the second. If the pieces evolved through MIDI, separate stems would not even be necessary - the playhead of the pieces could simply start and stop as needed to follow the game's narrative. In addition, parameters such as tempo and dynamics could react to the multidimensional game states in more coherent ways. Rubato could be executed as the player moves in and out of danger, and piano dynamics could be established through adaptive velocity control.

The music of *Untitled Goose Game* is a glimmer of hope for what adaptive music can and should aspire to be, drifting from the mundanity of looped tracks into the promise of a score that will truly follow the player. At the same time, it is also a cautionary tale against the sustainability of using recordings for the entirety of a soundtrack. Music sequencing could have taken Golding's implementation of Debussy's piano *Preludes* to

new heights by drastically lowering the file size requirement and enhancing emotive bit depth through greater dynamic parameter control.

D. *Spore* - A Game About Evolution That Tried To Cause It

When the technical demonstration for the video game *Spore* debuted in 2005, viewers became enamored at the prospect of taking an alien species from inception as cellular microbes to full-fledged interplanetary conquest. A game with this level of complexity promised multiple different game genres in a single package - third-person role-playing, action-oriented fighting, and even city management all had a place in this experience. Accomplishing this task required the game developers and engineers to study the philosophical and tangible prospect of evolution. From the very beginning of development, director Will Wright knew that the best way to digitally replicate the phenomenon of evolution was extensive use of procedural generation to artificially develop enemy species variants, unique plant types, and entire planets. The *Spore* team built the game engine from the ground up, which allowed for most aspects of the game (including music) to be procedural.

Audio director Kent Jolly was reluctant to design a procedural music system, because he claimed that music generated through MIDI often "sounds like MIDI" and that it "just doesn't sound like good production."[77] This all changed when veteran generative music composer Brian Eno joined the project and imparted decades of relevant experience and wisdom. The audio software engineers developed a workflow that

77 Kent Jolly and Aaron McLeran, "Procedural Music in SPORE," GDC Vault, March 2008, Conference recording, 1:00:07, https://www.gdcvault.com/play/323/Procedural-Music-in.

integrated Miller S. Puckette's open-source software Pure Data into the *Spore* game engine and dubbed the result *EAPd* (EA is short for the game's publishing company Electronic Arts). The designers then programmed several *abstractions*[78] into *EAPd* to aid Eno's compositional process, allowing him to focus entirely on a particular scene's sound rather than the minute details of building things like sample libraries from the ground up.

Many of the *EAPd* music patches relied on random number *seed* generation to determine pitched and rhythmic musical information. When given a new seed vector,cde random number generators will reproduce the same pattern each time based on a specific form of probability. In a talk given by Kent Jolly and Aaron McLeran at the 2008 Game Developers Conference, Jolly points out that this means that a computer already has musical "presets"[79] baked into its basic architecture. This method was used instead of pure random number generation, because using a specific seed vector allows the same randomized pattern to return and create a more humanistic musical repetition as a result.

After building the random number generation systems, the team programmed specific musical scales into *EAPd* so that the algorithms did not output strict chromaticism. If the random number generator counted from 0 to 6 for a C major scale, individual pitch classes would be mapped accordingly: 0 == C, 1 == D, 2 == E, 3 == F, 4 == G, 5 == A, 6 == B. This same methodology was used to determine which drums would play at a given time from the percussion sample libraries. At Brian Eno's request, the engineers even programmed advanced DSP effects such as ring modulation into the

78 An *abstraction* is a chunk of code that hides all but the relevant arguments needed for the function to work to create ease in future uses.

79 Kent Jolly and Aaron McLeran, "Procedural Music in SPORE," GDC Vault, March 2008, Conference recording, 1:00:07, https://www.gdcvault.com/play/323/Procedural-Music-in.

patches that could transform the timbre of a specific track in order to increase the emotive bit depth of his orchestrational palette.

The main downside of procedural audio in video games is the need for computer chip bandwidth to send some of its power away from graphics and gameplay and focus it on sound. Not every piece of music in *Spore* is procedural - in fact, in order to compromise the necessary computer resources between different elements of the game, most of the procedural music occurs in the in-game editors. Throughout the game, players can influence their creatures' evolution with editors that change how the species looks. The editors were the least computationally taxing moments of the game, so they were the ideal places to introduce procedural music. The music adaptively followed player choices, creating stark effects such as dark modal shifts when designing menacing creatures with large fangs. Once a species reaches the Civilization Stage of evolution, players are even offered an interface to compose their own city anthems.

Unfortunately, keeping procedural music limited primarily to the in-game editors is its greatest downfall. While embedding a vast collection of music tools into the game engine was a novel concept for 2008, it still only responds to player interactions in the editors as if it were a toy or a sound installation. The editors have no goal aside from the users' artistic endeavors, so there is no competition or goal that can shape the overall compositional form or sound. This concept is reminiscent of generative music phone applications that Brian Eno has developed throughout the past decade. *Bloom: 10 Worlds* is a minimalistic application that generates music and colored spheres as users tap on their screens. Similarly, *Trope* generates a new piece as users draw amorphous shapes

with their fingers. These two applications/toys only have identity through their timbral choices - no harmonic progressions or melodies are predetermined.

At the 2008 GDC talk, Aaron McLeran explained that the "goal was to create music that wasn't distracting, didn't repeat itself, was playful, and responded subtly to how people were playing" so that users did not get irritated by the music when experimenting for long periods of time in the editors.[80] This is not inherently a problem, but it did nothing to evolve video game adaptive music because it did not respond to competition. As one of the few video game soundtracks that has tried tackling truly procedural music in recent years, it only serves as an example of how procedural music can be subtle.

As soon as players start a new game and enter the Cell Stage of evolution, they are greeted with the text "Throughout Spore, the choices you make impact your future" - this is true throughout this section of the game, though the pre-recorded music does nothing to exemplify it. In the Cell Stage, players are required to swim around the primordial soup, eat enough matter, and fight other organisms for the right to grow big enough to become a land creature. The music that plays during this scene transitions as players reach certain checkpoints, but rather than shaping the microbe's journey it simply maintains the same subtlety and ambience that is present in the editors' procedural music. Rather than conforming to the progress bar at the bottom of the screen, the music produces undesirable structural dissonance by barely changing or growing with the player's cell.

---

80 Dave 'Fargo' Kosak, "The Beat Goes on: Dynamic Music in *Spore*," GameSpy, (IGN: February 20, 2008), http://pc.gamespy.com/pc/spore/853810p2.html.

If the procedural music tools were utilized in this sequence, it could have thrived on the presence of danger and clear goals. In response to combat, the tools could account for different game states determined by the power differential between each microorganism in the ooze - one where the player's cell is close to a battle that will almost assuredly end in failure, one where the player is near a more closely matched foe, and another when the player will surely triumph. Music notes could have slid along a modality brightness spectrum (using the built-in scales) where Locrian represents the most futile player deaths while Dorian acts as the brightest potential for combat. The random seed vector does not need to change for this - the presets can simply remap the notes through scale quantization.

The procedural music could have acted on a meta-level, adding notes to the music's "cellular structure." When players eat enough matter, they unlock a new limb that enhances their cell in some way. As this literal evolution occurs to the microbe, the music could utilize additive principles similar to those found in the music of Philip Glass. The ambient music could initially limit the amount of notes that are heard in a particular random sequence with a note gate, and slowly allow them back in as players progress through the Cell Stage.

Procedural music in *Spore* could have revolutionized an entire industry, but none of its techniques have managed to gain mainstream appeal. *Spore* could have acted as a shining example of the flexibility of building a music system from the ground up, especially since it was released at a time when audio middleware was getting massively adopted by video game developers around the world (Wwise was first released in 2006).

The moments of procedural music in *Spore* occur during moments that are not reminiscent of gameplay in story-driven triple-A games. This meant that present developers did not find any parallels or advantages to creating real-time music systems. Luckily, *Spore*'s soundtrack is still a triumphant proof-of-concept that procedural music in video games is not only possible, but that it can produce charming results when the rules are properly set.

IV. Technical Demonstrations

A. Potential Solution

Procedural audio will not immediately solve every problem associated with adaptive video game music, nor should it. Looped music has been a staple in this medium for more than four decades because it is the most obvious response to indefinite temporality and open form. Today's adaptive music operates under the belief that the solution relies solely on building compositions that also do not have a fixed timeline, but sequencing music with procedural audio introduces multiple dimensions of adaptivity beyond piece duration. Exclusively playing from MIDI files is only the first step though, because they still dictate all pitch and rhythm-based aspects of a piece along a timeline. This is where *musical interpolation* comes into play.

If one channel of music is perpetually moving with an arpeggiator, an underlying harmonic progression can move at any pace while other parameters such as MIDI note length, rate, and pattern can all be changed to provide subtle changes in intensity. If that

same channel is generating its sound with a synthesizer, digital techniques such as subtractive synthesis filtering, FM synthesis, and phase distortion are now all viable possibilities for making timbral shifts. A piece does not have to be wholly procedural either - if the musical form of a piece is looped, this implies that there is potential for hybridity between sample data and procedural audio. This solution is desirable, because it allows composers to have as much control in their music as they could possibly want. If a composer hears a melody in their head one way and cannot fathom it with any specific variations, that layer could be recorded and played as audio. A single melody can be accompanied by a myriad of potential harmonies though, so each of those channels could be procedurally generated against the fixed material.

It will take a monumental effort from software developers to build a program that can power this level of video game music, so the first technical demonstration - inspired by *parametric design* - shows this hypothetical style of composition using Ableton Live. Ableton Live lives up to its name by providing music producers and DJs with a plethora of musical devices that are designed for real-time audio processing, and its integration of Max/MSP increases that potential even further. The second technical demonstration will show a generic implementation of new MIDI sequencing devices and synthesizers inside UE4 to show how composers can create more adaptive music using even the most basic tools.

B. Parametric Design - Explanation

Parametric design assists modern architects by automating the creative process of building design, ensuring that a property will be both structurally sound and visually idiosyncratic. Antoni Gaudi is considered to be the first architect who used this technique while conceptualizing the church of the Sagrada Familia in the late 19th century. To achieve this in an analog world, Gaudi attached differently-weighted birdshot to variable-length strings and let gravity unveil the model upside-down. The placement of the church's vertices was decided by the proportional relationships between all of the birdshot once the model reached equilibrium.[81]

After establishing a set of rules that must be followed to create a basic skeleton for a model, "parametric variations" can be made that alter the other more external elements of the model. In essence, each parameter variation only changes the way a specific building looks without ultimately altering its identity.[82] The same can be said about the human body. A person's weight may fluctuate, their hair length may change, and their body may scar from injury, but their overall visual identity never changes. Those are just parameters that are constantly varying the external elements of a person's physical appearance.

Kate Compton showed an example of parametric design at the 2017 Game Developers' Conference that generates a brand new flower everytime the "dice" button is

81 "What is Parametric Design?," Studio Robazzo, accessed February 18, 2021, https://robazzo.com/journal/what-is-parametric-design.

82 Ning Gu, Rongrong Yu, and Peiman Amini Behbahani, "Parametric Design: Theoretical Development and Algorithmic Foundation for Design Generation in Architecture," in *Handbook of the Mathematics of the Arts and Sciences*, ed. Michael J. Ostwald, Kyeong-Hwa Lee, Torsten Lindstrom, Gizem Karaali, and Ken Valente, (New York: Springer, 2018), 1-22.

pressed.[83] Once pressed, 32 different parameters such as hue, leaf shape, and spread are randomly changed to new values. All of the parameters can be randomized, but they can also be individually altered for smoother visual transformations through slider-based interpolation. In this particular example (shown in Figure 7), the consistent visual identity that remains is the plant's presence as a flower. At no point does the plant begin to look like a tree, a cactus, or a fern. This also happens to be the way that *SPORE*'s computer-controlled alien species are generated in real-time to smoothly simulate evolution from amoeba to sentience.

Figure 7: Parametric Design for Flower Modeling

Parametric design has aided visual artists throughout the past century and a half, but Western composers have also organically and unknowingly pursued this technique for centuries through the musical form "theme and variations." This form sonically establishes a piece's identity from the beginning with the initial theme, but then ornaments its external elements (such as melodic complexity, harmonic density, and articulation) in subsequent variations. While all variation parameters are controlled by the

83 Kate Compton, "Practical Procedural Generation for Everyone," GDC Vault, March 2018, Conference recording, 31:30, https://www.gdcvault.com/play/1024213/Practical-Procedural-Generation-for/.

composer, the entirety of a new variation can realistically compose itself after the first measure or phrase establishes the rules. A brief analysis of Ludwig van Beethoven's *Diabelli Variations* unearths different musical parameters that are being randomized in each variation as if he also pressed a metaphorical "dice" button.

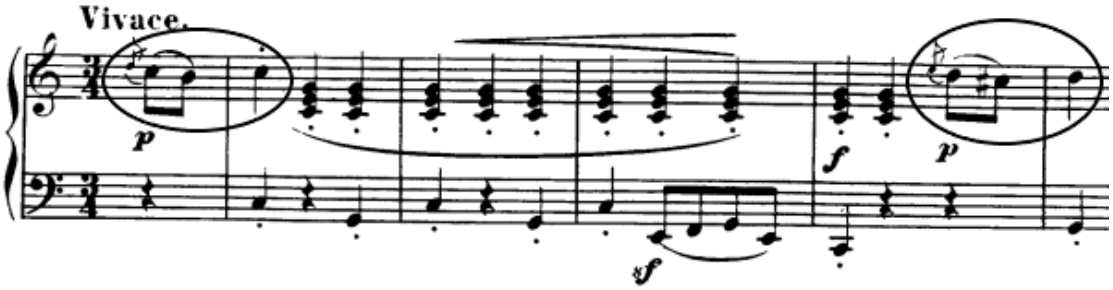

Figure 8: Theme of the *Diabelli Variations*

The theme of the *Diabelli Variations* (the first four measures are shown in Figure 8) is actually a waltz composed by Anton Diabelli, and many of the subsequent variations share the theme's binary form structure. This is akin to the skeleton of a building model before any alterations are made. If the overall structure is not altered, then other musical elements must have been to create 33 unique variations. The first two isolated moments of the melody (circled in Figure 8) have a similar contour: a grace note upper neighbor tone, followed by the main melody note, followed by a chromatic lower neighbor tone, followed by a final reiteration of the main melody note. Since the neighbor tones are embellishments on the C and D, they can be classified as ornamentation of the melody rather than essential melody notes. In between these moments the right-hand only plays block root-position C Major chords to engrain the tonic in the listener's ear. At the same time, the left-hand burns the tonic into the audience's mind with a sol-do bassline that

creates natural accents on beats 1 and 3 to establish the waltz rhythm and a hypermetric division of a half note followed by a quarter note.

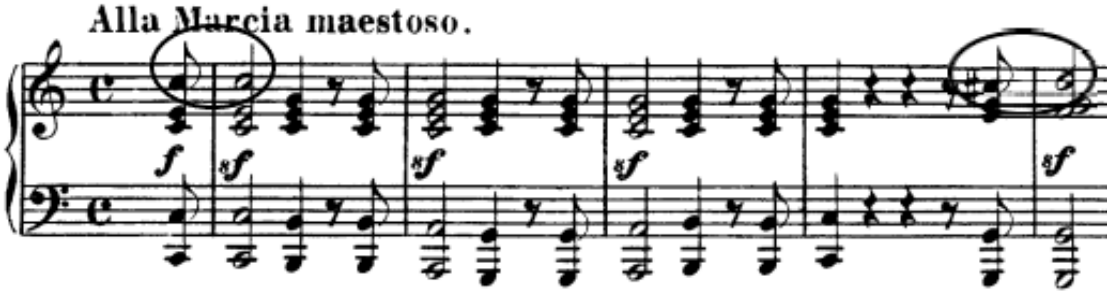

Figure 9: Variation 1 of the *Diabelli Variations*

Figure 9 shows how the first variation establishes a fresh set of rules while maintaining the theme's sonic identity. One striking change is the time signature's immediate move from 3/4 to 4/4. This shifts the mood of the work from a dance to a march-like pace. The time signature shift creates a wholly different momentum than the waltz because of the extra beat at the end of each measure. A deliberate choice was made to keep the natural accents on beats 1 and 3, but the fourth beat changes the hypermetric division to two half-notes. Variation 1 lowers the melodic complexity from the theme as well. The ornamental neighbor tones have been completely removed, except for the chromatic passing tone C# that leads up to D at the end of measure 4. The bassline is still outlining the sol-do relationship, but now that it is moving in scalar motion its density and complexity has slightly increased. The harmonic density and complexity has not changed from the theme, because the right-hand is still performing block chords. While Figure 9 only shows the first four measures, these changes are consistent throughout the

entire variation. This phrase alone introduces the rules for the remaining 28 measures, allowing Beethoven to fill out the rest in a computational manner.

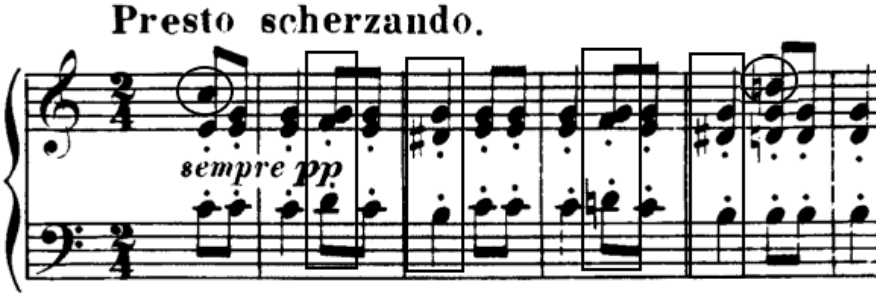

Figure 10*: Variation 15 of the Diabelli Variations*

Variation 15 shows how elaborate the musical adjustments become throughout the lengthy work. As seen in Figure 10, Beethoven alters the time signature yet again. Compressed down to 2/4 time, the hypermetric division now only consists of a single half note. The harmonic complexity drastically increases now to include passing dominant harmonies (outlined by rectangles) other than the previous C Major chords. In addition, the bassline increases complexity to match the original melodic complexity of the theme with the presence of neighbor tones that are only contextualized by the above harmony. Not every musical parameter has increased complexity though – this is the simplest the melody has been. This time, the only notes present in the melody are the necessary structural notes C and D.

Imagining each of these variations in the context of parametric design means that musical parameters can be arbitrarily measured on their individual spectra, and it is up to the composer to clearly dictate what the lowest and highest points of each attribute are. In an ideal setting, the ability to interpolate these parameters in real-time could allow video

game composers to directly attach musical development to game design elements such as weather, time of day, level of danger, and even movement. This first tech demonstration shows prototypes of how "instant" variations can be created on a composed theme (or skeleton) without deconstructing its sonic identity.

C. Parametric Design - Implementation

The following four Max for Live devices will exemplify how parametric design can be used to dynamically alter a piece of music that is being sequenced in Ableton Live. There are three MIDI effects: a Euclidean rhythm generator, an intelligent bassline generator that performs brand new lines from incoming blocked harmony, and a drum sequencer that can intelligently embellish an existing pattern. The bassline generator and drum sequencer are "intelligent," because they have been programmed to learn and adapt from existing material at a moment's notice. Random number generation is used to generate some of their material, but only through inconsequential additions to the predetermined skeleton. This is why they fit better under the umbrella of parametric design than as pure algorithmic composition or generative music. They will only work if some musical material has already been provided. The last device is a multidimensional FM synthesizer that uses wavetable interpolation for the carrier frequency, modulator frequency, and LFOs.

The bassline generator requires a chord detection algorithm in order to analyze the harmony in real-time. The foundation of the algorithm is built to work with tertial harmony and is partially taken from Ableton Live 10's *MIDI Monitor* Max for Live

device, but it has been improved in a number of ways. First, *MIDI Monitor* ignores or does not recognize a large number of complex chord qualities such as minor-major 7 or added extensions beyond 9ths. The algorithm also does not take into account the possibility that some notes such as the 5th are commonly omitted from chords when they are not crucial to determining the quality. This means that a major or minor 7 chord that omits the 5th goes completely undetected. After adding these crucial functions to the algorithm, it can now detect the majority of Western music's tertial harmonies and provide two separate essential variables to the generators: the root of the chord, and the quality.

Figure 11: Max for Live Bassline Generator

The bassline generator's user interface design is inspired by analog bassline synthesizers like Erica Synths' *Bassline DB-01* - there is a 16-step sequencer that provides the rhythmic trajectory for individual sixteenth notes. Rather than requiring the user to set each step's specific pitch like analog bassline synthesizers, this MIDI device determines what note it should play based entirely on notes from the chord it has detected. This prevents a creative drawback common to these types of machines. Ostinato patterns usually have to be saved within a limited number of slots and switched between during performances. Now, the machine's harmonic emotive bit depth is only limited by

a user's imagination. As seen in Figure 11, the line above the step sequencer allows users to choose whether or not the steps will play the root, third, or fifth of the chord. This ensures that the same intervallic and rhythmic pattern will happen regardless of any harmonic change.

"Rand" buttons on the top left of the device randomly change the rhythmic pattern and each step's chord tone. This means that unlike in its analog predecessors, rhythm and harmony now exist on two separate dimensions. Composers could automate these buttons to trigger after a given number of times to prevent patterns from becoming stale. The "Slide" button to the right of the step sequencer determines the outcome of greyed-out steps. If "Slide" is turned off as shown in Figure 11, the greyed-out step will act as a MIDI note-off to create a staccato-like effect, but will hold out for a more legato-esque sound if it is on.

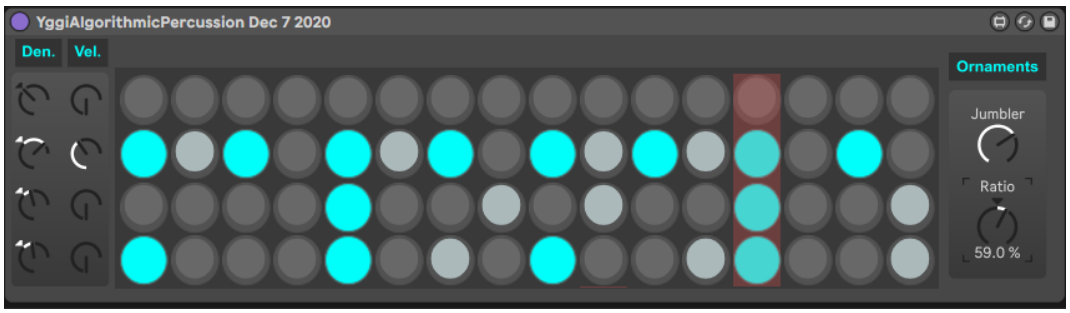

Figure 12: Max for Live Drum Sequencer

The drum sequencer was influenced by Reason Studios' algorithmic drum machine *Beat Map*, but was designed to give more control to the composer. On the surface, this sequencer works exactly like every other drum sequencer on the market that plays predetermined patterns. A crucial new element has been added though - the sequencer can ornament and fill steps around the existing pattern to allow for rhythmic

complexity interpolation. As shown in Figure 12, the blue dots represent the composer's pattern while the light-grey dots represent steps that the device has added. Much like *SPORE*'s procedural music, the ornaments are randomized with a specific random seed vector so that users can freeze and replicate any pattern they desire.

The "Jumbler" dial in the top right changes the seed number, and replaces ornaments with the new sequence. The "Ratio" dial underneath it determines the MIDI velocity difference between the embellishments and the main pattern: embellishments will be half the velocity of the main pattern if it is set to 50%, the same velocity if set to 100%, and simply won't play if set to 0%. The "Den" dials to the left of the sequencer determine a specific row's density - that is, how many notes are inside that one row. If set to 50%, the row will play exactly as it was written without any embellishments. Numbers between 50 - 100% add more embellishments to the pattern until the row is fully covered. Numbers between 0 - 50% create a gate that lowers the chance that a step in the main pattern will play when it should. The "Vel" dials help humanize the performance of the step sequencer by increasing the possible MIDI velocity range in every single step in a row. Rather than outputting MIDI notes at a fixed velocity such as 64, turning velocity dials up just slightly could allow notes to randomly output from 32 to 96. This function seeks to emulate the variability in a human drummer's hits which will never be exactly the same volume. Having a discrete dial for each row ensures multidimensionality among each rhythmic lane. The hi-hat and kick drum rhythms could become more or less dense, but the clap or snare rhythms could stay the same.

The Euclidean rhythm generator provides countless rhythms that can lead to an almost limitless number of chordal accompaniments. Defined by Godfried Toussaint, Euclidean rhythms "have the property that their onset patterns are distributed as evenly as possible" and "may be used to automatically generate, very efficiently, a large family of rhythms[84]." The algorithm places a certain amount of steps inside a specific grid size so that each step is spaced out as evenly as possible. In a grid of eight sixteenth notes, four steps would be equally displaced every other step and would therefore emulate standard eighth notes. If the number of steps is changed to three but the grid size stays the same, then the algorithm must respace the distance between steps so that as much symmetry as possible can be salvaged. These rhythms are realized in Figure 13.

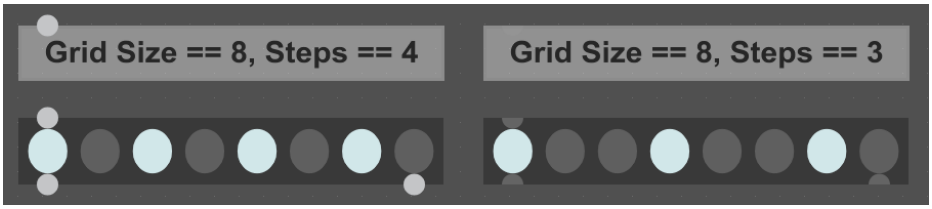

Figure 13: Euclidean Rhythms in a grid of 8 with variable step amount

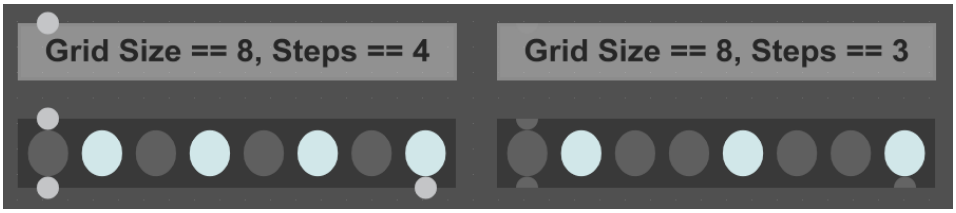

Figure 14: One Rotation to the Right

These patterns are usually shown with circles to display their radial symmetry, which demonstrates how they can be rotated to place emphasis on different points in a

84 Godfried Toussaint, "The *Euclidean* Algorithm Generates Traditional Musical Rhythms," in *Renaissance Banff: Mathematics, Music, Art, Culture*, ed. Reza Sarhangi and Robert V. Moody, (Alberta: Bridges Conference, 2005), 47-56. http://archive.bridgesmathart.org/2005/bridges2005-47.html.

measure. This is similar to the way musicians who study set theory use the metaphorical image of a clock to determine how many different ways a set can be transposed along the 12 chromatic pitches before returning to the same initial tonic. Most Euclidean rhythms can only be rotated a certain number of times before returning to the same pattern. This is similar to the scalar "modes of limited transposition" that were hypothesized by Olivier Messiaen. Figure 14 shows one rotation to the right of Figure 13's patterns.

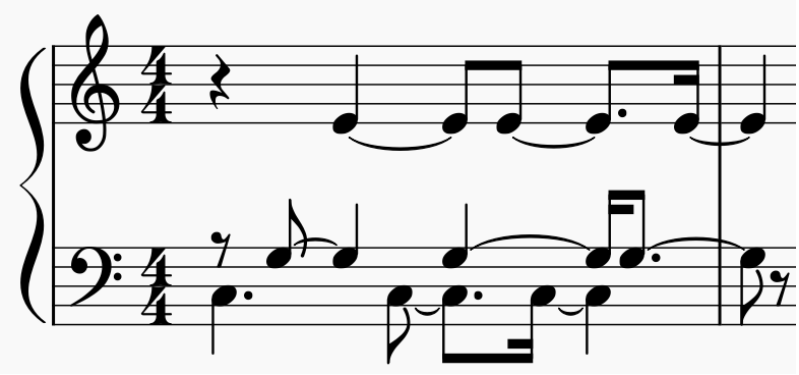

Figure 15: Realization of Euclidean Rhythm Three-Voice Accompaniment

Euclidean rhythms may generate intricately syncopated drum beats, but they can also be used with pitch to generate fresh harmonic accompaniments. The generator uses Euclidean rhythms to isolate and pump out each individual voice of a chord to determine when each voice will be heard like a sophisticated arpeggiator. If the generator is set to produce three voices and it is fed a C major triad, it will treat each pitch as separate lines, each with their own rhythm. Without adjusting the rotation of each voice, the instrument will produce an accompaniment that consists entirely of blocked chords. If users adjust the rotations, the sequencer will begin to churn out an accompaniment that more closely resembles an arpeggiation of the given harmony. Figure 15 shows a realization of how

this device can create an arpeggiation with a given grid size of 16, 3 steps, and with each voice rotated an eighth note away from the previous one.

Unlike the importance of row parameter independence with the drum sequencer, it is helpful to include a master dial (shown in Figure 16) that controls the number of divisions because each note of the arpeggiation belongs to the same chord. This allows harmonic density of the entire pattern to be increased and decreased in an instant. To achieve this, a percentage range is scaled with each row's grid size. If the master dial is set to 60% and a grid's size is 10, then there will be 6 steps in the sequence. Float-based scalings are rounded down with a floor function to ensure that the number of divisions is an integer - setting the dial to 60% with a grid size of 12 will result in 7 steps. The devices' rotation buttons still exist on a separate dimension, so different rhythmic patterns can be implemented without needing to adjust the harmonic density.

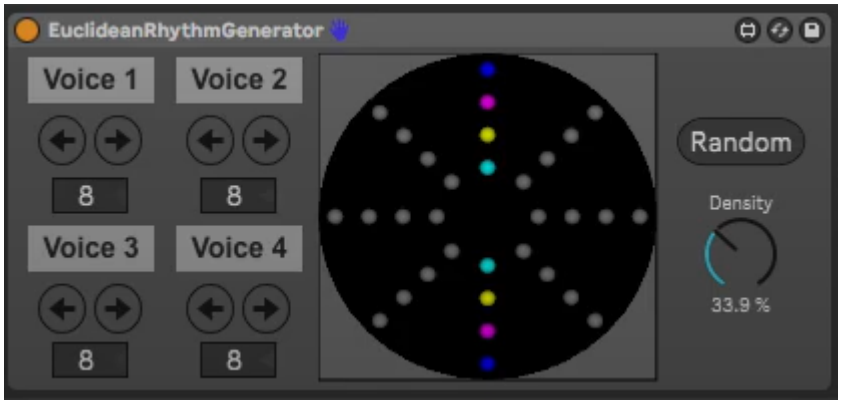

Figure 16: Max for Live Euclidean Rhythm Generator

The final device takes FM synthesis to its precipice, and is best understood as an augmentation of Ableton Live's *Wavetable* instrument. Instead of having two disconnected wavetable oscillators that amount to rudimentary additive synthesis, the

second oscillator acts as the first oscillator's modulating frequency for phase-modulation. In the same way that wavetable interpolation is used to sweep across an audio buffer, timbral complexity can be increased and decreased overtime with FM synthesis by automating the amount that the modulating frequency affects the carrier frequency. When the two are combined, the timbral palette becomes two-dimensional and has an exponentially larger amount of possibilities.

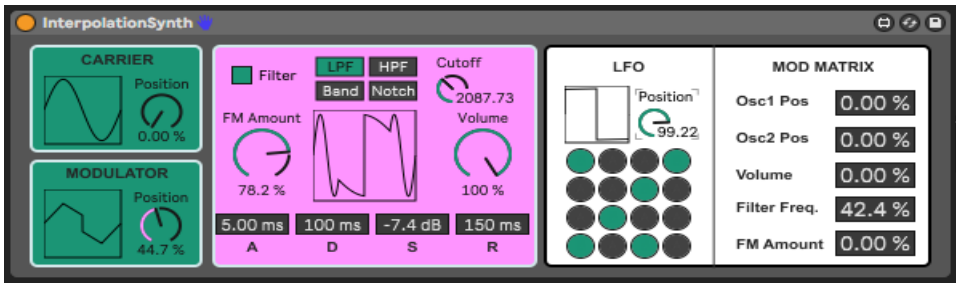

Figure 17: Max for Live FM Synthesizer

Vector synthesis hardware devices experimented with real-time two-dimensional timbral shifting by using joysticks to mix between four different waveforms on either side of an x/y axis. This Max for Live device's multidimensionality does not end there though - the modulating frequency is also a wavetable, meaning that the FM signal's interpolated waveform can influence timbre without changing its modulation depth. In a physical space, a joystick would no longer be able to handle all possible timbral dimensions. Now a better physical representation would be to envision an object moving around the inside of a cube and the introduction of a z-axis. Each point inside of this cube would result in a special sound. These smooth transitions are clearly heard but could be difficult to visualize with an oscilloscope, so the FM synthesizer (pictured in Figure 17)

draws an approximation of the resulting waveform at any point along the three dimensions.

The synthesizer's LFO is also a wavetable. Software synthesizers traditionally limit the available LFO shapes to the four basic waveforms (sine, triangle, sawtooth, square) without interpolation and sample-and-hold (to generate random numbers), but this only allows for a modulatory emotive bit depth between 2 and 3. Allowing LFO shapes to be interpolated would let sound designers move from smooth to janky parameter modulation at their own pace without having to instantly pick a new shape. LFO sync rate (when syncing modulation to a specific tempo) is also usually burdened by a drop-down menu or dial that picks the appropriate rhythmic value. This can be overcome with parameter automation, but automation is no longer an option if a piece is not moving along a fixed timeline. To compensate for this, a sixteen-step grid is drawn across four separate rows to quantize the LFO's rhythmic phase in relation to the spaces between the steps added together as sixteenth notes. Figure 17 shows steps 1, 4, 7, 10, 13, and 15 turned on, so the resulting rhythm will be four dotted-eighth notes followed by two eighth notes.

These Max for Live devices are used in conjunction in the *Terra* Ableton Live set to demonstrate their enhanced levels of emotive bit depth and dimensionality. By mapping specific device user-interface objects to four MultiMap Max for Live Devices, users can interpolate rhythmic density, synth articulation, timbral sharpness, and audio fidelity in the same way Kate Compton did with flowers at the 2017 GDC. Similar to her "dice" randomization button, users can randomize the dials to generate instant variations

on *Terra*. Introducing parametric design as a form of computer-aided composition can provide adaptive video game music with the necessary multidimensionality that it is currently missing.

Figure 18: Parametric Design Dials in *Terra* Ableton Live Set

D. The Power of Sequencing and Modular Synths - Explanation

The analysis of audio middleware earlier in this dissertation briefly explained why audio processing has to move exponentially faster than visual processing. These simultaneous procedures are digitally achieved with separate computer *threads* that asynchronously process tasks at the same time. In game engines, there are two individual threads: one for gameplay actions (which runs at frame rate) and one for audio rendering (which runs at sample rate). The threads have to communicate with one another to

properly trigger sound effects and musical layers when changes are made. If an explosion occurs in the game, the sound effect is triggered on the game thread and then processed on the audio thread. This creates a serious problem if composers want to procedurally generate new music that is entirely based on gameplay actions.

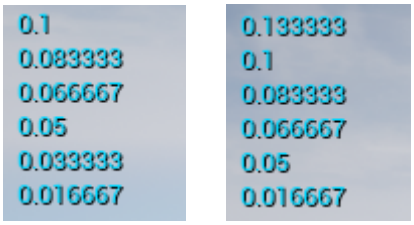

Figure 19: Event Tick times at 60 FPS/0.01667 secs (left) and 60 FPS/0.02 secs (right)

One demonstration on the Unreal Engine Marketplace (simply titled *DrumSequencer*) shows this technical dilemma perfectly. On the surface, this program looks and feels exactly like an analog drum sequencer. Upon closer inspection, it suffers from one fatal flaw: it triggers each consecutive step on the game thread. Built entirely with UE4's Blueprint System, the sequencer fires a new step when a new "Event Tick" is received, which happens every visual frame.[85] If a player is lucky, that speed will be locked[86] at 60 frames per second and a new frame will occur every 0.01667 seconds. If this is true, the drum sequencer will unravel perfectly at a tempo of 60 BPM, and a new sixteenth note would trigger every 15 frames, or every .25 seconds. If the composer changes the Event Tick interval to 0.02 seconds in an attempt to match a tempo of 50 BPM, a critical music quantization error happens. The event tick does not actually trigger

85 Martinsart, "Unreal Engine Marketplace," *Unreal Engine Marketplace* (Epic Games), accessed February 21, 2021, https://www.unrealengine.com/marketplace/en-US/product/drumsequencer.

86 In an effort to accommodate varying levels of GPU rendering complexity, some video games leave the framerate unlocked meaning that it can move at a variable rate within a given range.

every .02 seconds. Instead, it has to approximate to the closest .01667 second interval, since the game thread is still moving at 60 frames per second.

Our eyes cannot see this quantization error in animations, but our ears can easily detect this discrepancy. Music producers quantize MIDI information to the beat in a DAW so that musical rhythms are perceptible. This drum sequencer is attempting to quantize the steps in a similar way, but the slow speed of the game thread makes certain samples trigger either too early or too late. Figure 19 shows how inconsistent these rhythms will be when the tick interval is changed. How then, can this dilemma be solved?

Sequencer steps should always be triggered on the audio thread. Only the step sequencer's pattern should be changed on the game thread. If the composer adds a step on beat three while the music is currently at beat 1, that update does not need to be taken into account immediately in the same way as the step triggers. Similarly, if an arpeggiator is unfurling a specific harmony with rhythms tied to the audio thread, the harmony can be changed on the game thread without interfering with the persistent rhythmic drive. Developers of procedural audio software for video game engines will need to ponder which elements of music production should happen on each thread.

Imagine, for example, a digital FM (frequency modulation) synthesizer that could act as one of the many instruments at a composer's disposal. To achieve the FM effect, two oscillators will need to exist at the same time - the initial carrier frequency as well as the phase modulating frequency. While it is possible to modulate the carrier frequency on the game thread, doing so will result in inconsistent sound design that resembles coarse sample-and-hold modulation instead of the typical smooth waveforms. Instead, both

oscillators MUST be processed on the audio thread. The FM depth could be altered on the game thread through a modulation matrix though, along with other attributes such as volume, filter frequency, LFO rates and shapes, and envelope parameters.

Computer music pioneer Max Mathews tackled a similar artistic dilemma in the late 1960s at Bell Labs. Due to the limited processing speed of computer chips at the time, it was impossible for computer musicians to make performative changes in real-time. His solution was the creation of a digital/analog hybrid system called GROOVE - *Generated Real-time Output Operations on Voltage-controlled Equipment*. GROOVE still relied on analog synthesizers for all of the audio processing, but cleverly used the computer to send CV (control voltage) information to manipulate certain synthesizer parameters during a live performance. Changes were input with a joystick, rotary dials, and a regular computer keyboard.[87]

The computer was able to track and record a performance for future iterations, but Mathews was uninterested in pressing the play button and replicating the same performance each time. At one point he said that "the computer performer should not attempt to define the entire sound in real time...instead the computer should retain a score and the performer should influence the way in which the score is played."[88] GROOVE is an ideal exemplification of how to use asymmetric hybrid systems to bridge the gap between game engines that calculate at frame rate and procedural audio engines that will need to process at sample rate.

---

87 Thom Holmes, *Electronic and Experimental Music: Technology, Music, and Culture*, 4th edition (New York: Routledge, 2012), 484.

88 Thom Holmes, *Electronic and Experimental Music: Technology, Music, and Culture*, 4th edition (New York: Routledge, 2012), 275.

E. The Power of Sequencing and Modular Synths - Implementation

The following demonstrations implement these ideas into UE4 by borrowing methods consistent with modular synthesis. While an elaborate procedural music system will allow the possibility of loading and tampering with existing MIDI files and sample libraries, compelling adaptive soundtracks could be produced using only digital synthesis and basic sequencing devices. YouTube creator Jay Hosking demonstrates this potential with his "live, semi-improvised" analog synthesizer improvisations.[89] Prior to each performance, some musical material such as harmonic progressions and step sequences has been programmed into each piece of hardware, but the musical form and sound design are entirely controlled by pads and dials on his hardware in real-time. This results in performances that have a clear identity despite the flexible and editable musical parameters.

To demonstrate the strong creative potential of procedural audio for adaptive game music, I coded a few user-interactable modules to make this modular setup work in UE4. The first module was created as a subclass of the SynthComponent DSP processing block. This object will control all sample rate actions of any other connected modules - the brains of the operation. The SynthComponent has two important roles - it acts as the *clock* for generating musical material and the *mixer* for the final master audio. Two sequencer modules were created: an *arpeggiator* and a *step sequencer*. The sequencers are defined as any device that outputs musical material such as pitch or rhythm. To complement these material generators, two instrument modules were built: a *wavetable*

89 Jay Hosking, "Analog synth jam w/ Dreadbox Typhon & Abyss, Moog Matriarch & Minotaur, DrumBrute Impact, effects," Jay Hosking, November 19, 2020, YouTube video, 5:44, https://youtu.be/iWLBmAGR_jc.

synthesizer and a *sampler*. The instruments retrieve the sequencer information and transform it into audible sound through the use of either synthesis or sampling. Once the C++ classes were created, module-relevant functions were coded so that musical parameters can be changed on the game thread using UE4's visual programming Blueprint System.

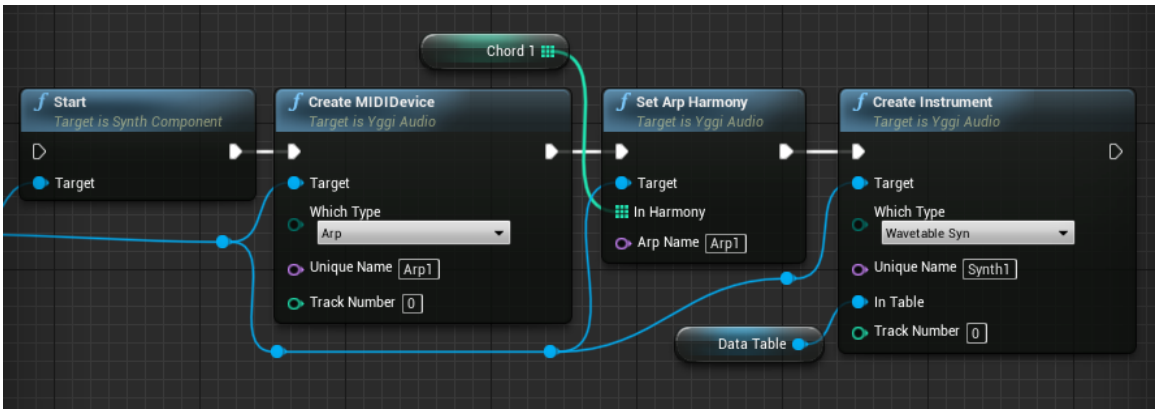

Figure 20: Example of Modular Patching System

In lieu of true modular synthesis patching with cords, this patching system relies on naming each individual device so that an array of pointers can find them in memory through proper tokenization. Upon creation, sequencers are given unique names and users must also set an integer value that defines the track signal path like in a DAW. Newly initialized instruments that intend to receive musical information from a specific sequencer copy the integer track number - this means that more than one instrument can receive information from the same lane. Device names are used in conjunction with specific UE4 Blueprint functions that can alter their parameters according to game logic such as changing a wavetable synthesizer's table position, an arpeggiator's existing harmony, or even a step sequencer's number of steps. Figure 20 shows an

implementation where an arpeggiator is initialized with an harmony dictated from the integer array called “Chord 1,” followed by the creation of a new wavetable synthesizer that is preloaded with memory buffer information from the “Data Table” variable.

Special Blueprint utility functions have been developed that give game developers and composers access to salient musical parameters that may not be immediately apparent. Commercial step sequencers usually have the ability to store at least four separate patterns to switch between on the fly, which allows for quick changes in musical form. Employing the concept of musical interpolation, a function has been built that interpolates between multiple stored patterns in the same way as wavetable interpolation. If composers want a piece’s percussive texture to grow at a variable rate, they could embed both a minimum texture and a maximum texture into the step sequencer and use probability to determine how much of the larger texture should bleed in over time. Similarly, rather than just immediately changing an arpeggiator’s chords, composers could embed two different harmonies into the arpeggiator’s memory and blend between them in a way that could imply cacophonous tone clusters or mellifluous extended chords.

Two separate levels in UE4 have been scored to show this introductory music sequencing API in action. The first level (which is pictured in Figure 21) is a free 3D-modelled fantasy village that is available as a free download on the Unreal Engine Marketplace. After some slight alterations, the level has been programmed to cycle between day and night in order to establish a more dynamic setting than its stock example. The level's accompanying composition demonstrates a tonal shift from major to

minor to properly convey these lighting shifts. A textbook four-measure long chord progression is used to show how certain aspects of harmony are multidimensional and should therefore be treated as such. Using Roman numeral analysis, the chord progression in the major key is I-IV-ii-V7, and in the minor key it is i-iv-ii°-V7. This outlines a key point in musical parametric design - the function of each chord in relation to its temporal placement does not change between the two progressions, only the chord quality does.

Figure 21: Fantasy Village...During the Day on left and At Night on the right

Scale quantization macro functions were built in UE4 to flawlessly move from one key to the next. If the time of day moved below a specific threshold, the third and the sixth scale degrees were lowered to move from major to harmonic minor, and vice versa. This means that the piece does not need to be written in two separate ways, and could conceivably be transposed to any new mode or key imaginable. Some other musical choices were made to score and insinuate that the scene moves from bright and cheery during the day to cold and distant during the night. The accompanimental pattern first appears light and bouncy during the day but becomes much sparser at night. Timbrally,

the melody's synthesizer wavetable position moves from a square wave to closer to a triangle wave at night, and the overall reverb on the UE4 audio submix is increased to make the space feel more melancholic.

The second UE4 level involves a more abstract concept, but it implements musical multidimensionality in ways that truly mimic a game - it is the digital equivalent of dice music. The momentary outcome of the piece will be determined by the result of two six-sided dice that are thrown into the air in UE4 using simulated physics. When they land, the combined value of the dice decides how long a two-line musical pattern will be. The longest pattern can be 12 notes long, and the shortest will be 2. This sets up an additive process similar to the early compositional style of Philip Glass. The dice are still separate, so each of their individual numbers can affect other musical parameters. One die determines the timbre of each synthesizer, and the other decides to which mode the patterns will be scale quantized.

This design perfectly demonstrates that while some elements of a piece may change, they should not all change at the same time like they do in most modern adaptive music. Another way to comprehend musical multidimensionality is through counterpoint terminology. Switching from one audio file to another implies continuous parallel or similar motion, which can result in an undesirable outcome. By embracing multidimensionality, music can have structural oblique or contrary motion and enhanced variability. Take two potential outcomes of this dice game for example - one result could be a 5 on dice one and a 2 on dice two, or a 3 and 4 instead. Both outcomes add up to 7,

so the pattern length will not change. The pattern's mode will change and the synthesizer timbres will change, creating structural oblique motion.

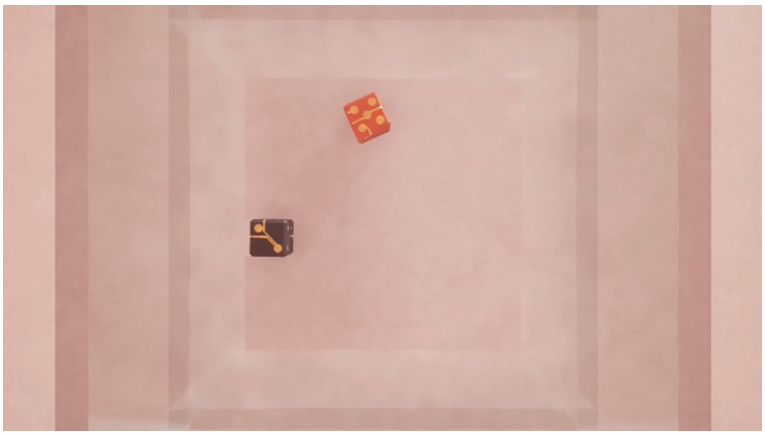

Figure 22: 21st Century Dice Music

F. Final Thoughts

The future of this system should focus both on expanding its existing API and building a user interface that is friendlier to composers and ideal for an accelerated workflow. Composers could have an interface similar to DAWs like *Reason* and *Bitwig* or the web-based application *Audiotool* where the majority of production occurs through user-connected modules. UE4 provides an extensive framework to develop plugins for the engine, but the main difficulty stems from how a standalone plugin window will be able to interact with the UE4 Blueprint system or C++ API. One attractive solution could resemble Ableton Live's Macro dial system. In order to organize parameter manipulation in what can sometimes be an extensive and daunting chain of DSP and MIDI effects, 16 macro dials can be mapped to any parameter along that chain for a central hub of control. After designing the skeleton of the music in the UE4 plugin, composers could assign

specific parameters to these macros and allow game designers to easily implement them into code or the Blueprint system like the parametric design demonstration.

While this system is suggested as a solution to obtain greater adaptivity in video game music, it has applications in other mediums as well. In its most recent update, Unreal Engine 4.26 integrated Digital Multiplex (DMX) data support and communication which lets the engine power live events by triggering lights, fog machines, and other special effects hardware.[90] Musician and actor Donald Glover used Unreal Engine 4 in November 2018 to create "a mesmerizing visual experience projected onto the dome's interior in real time" known as *Pharos*, which brought further attention to the use of video game technology in concert production.[91] With this in mind, procedural audio could round out UE4's multimedia design possibilities by providing a dedicated system for generating sound in live concerts and interactive art installations.

This implementation is not meant to completely define the future of video game adaptive music - it is intended as a jumping off point for conversations that will hopefully shape the next decade or two. The criticism throughout is not against the quality of music that is being composed for the medium - it is meant to illuminate problems of complacency with the state of video game adaptive music and how limited software improvements have exacerbated the situation. With powerful procedural audio tools, the dilemma that is currently preventing the full creation of the *Earcade* will no longer exist, and composers such as Guy Whitmore will finally be able to build soundtracks with

90 "DMX Overview," Unreal Engine, accessed February 23, 2021, https://docs.unrealengine.com/en-US/WorkingWithMedia/DMX/Overview/index.html.

91 Andy Blondin, "Childish Gambino mesmerizes fans with real-time animation," last modified March 11, 2019, https://www.unrealengine.com/en-US/spotlights/childish-gambino-mesmerizes-fans-with-real-time-animation.

methods for which they are already yearning. The rigidity of audio files that has aggravated adaptive music's creative stalemate does not need to hold the medium back any longer – leveraging procedural audio will prevent the sonic banality of exact musical repetitions and allow soundtracks to respond to and shape with a player's multidimensional gameplay interactions.

Bibliography

"Ableton Live 11." Computer software. Berlin: Ableton, February 23, 2021.

Aikman, Zach. "Unite 2013 - Real-time Audio Synthesis with SuperCollider." Unity, September 16, 2013. YouTube video, 29:01. https://youtu.be/4uvClBGhJyA.

"Animal Crossing: New Horizons." Computer software. Kyoto: Nintendo Entertainment Planning & Development, March 20, 2020.

"Animation Blueprints." Unreal Engine. Accessed February 22, 2021. https://docs.unrealengine.com/en-US/AnimatingObjects/SkeletalMeshAnimation/AnimBlueprints/index.html.

"Audiotool." Computer software. Germany: Audiotool GmbH, 2018.

"Auditorium." Computer software. Philadelphia: Cipher Prime, April, 2008.

"Banjo-Kazooie." Computer software. Twycross: Rare, June 29, 1998.

"Basics of Play: Learn About the Game." Dungeons & Dragons. Wizards of the Coast, 2021. https://dnd.wizards.com/basics-play.

"Bastion." Computer software. San Francisco: Supergiant Games, August 11, 2011. https://www.supergiantgames.com/games/bastion/.

Beethoven, Ludwig van. *Symphony No. 6*. Leipzig, Breitkopf und Hartel, 1863. https://imslp.org/wiki/Symphony_No.6%2C_Op.68(Beethoven%2C_Ludwig_van).

Beethoven, Ludwig van. *Variationen fur das Pianoforte, Nr. 165*. Leipzig: Breitkopf und Hartel, 1862. https://imslp.org/wiki/Ver%C3%A4nderungen_%C3%BCber_einen_Walzer%2C_Op.120_(Beethoven%2C_Ludwig_van).

Bishel, Scott. "Procedural MIDI." Computer software. *Unreal Engine Marketplace*. Epic Games. Accessed February 21, 2021. https://www.unrealengine.com/marketplace/en-US/product/procedural-midi.

Blondin, Andy. "Childish Gambino mesmerizes fans with real-time animation." Last modified March 11, 2019.

https://www.unrealengine.com/en-US/spotlights/childish-gambino-mesmerizes-fans-with-real-time-animation.

Bors, Matyas Lancelot. “What is a Finite State Machine?” Last modified March 10, 2018. https://medium.com/@mlbors/what-is-a-finite-state-machine-6d8dec727e2c.

Brown, Daniel. Computer software. *Intelligent Music Systems*. Intelligent Music Systems, 2016. http://www.intelligentmusicsystems.com/.

Brown, Earle. *Folio II*. Leipzig: Edition Peters, 1982. http://www.earle-brown.org/images/file/media/Folio%20II%20as%20of%2012-12
-2013.pdf

Cage, John. “How the Piano Came to be Prepared.” John Cage, 2012. https://johncage.org/prepared_piano_essay.html.

Cage, John. “John Cage - One (1987) audio+sheet music.” Welcome to sale, April 30, 2020. YouTube video, 11:52. https://youtu.be/TfLvAhEyJ48.

Cerny, Mark. “The Road to PS5.” PlayStation, March 18, 2020. YouTube video, 52:44. https://youtu.be/ph8LyNIT9sg.

Collins, Karen. “Flat Twos & The Musical Aesthetic of the Atari VCS.” Popular Musicology Online, 2006. http://www.popular-musicology-online.com/issues/01/collins-01.html.

Collins, Karen. *Game Sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design*. Cambridge, Massachusetts: MIT Press, 2007.

Compton, Kate. “Practical Procedural Generation for Everyone.” GDC Vault, March 2017. Conference recording, 31:30. https://www.gdcvault.com/play/1024213/Practical-Procedural-Generation-for/.

Copetti, Rodrigo. “Nintendo 64 Architecture: A Practical Analysis.” Rodrigo’s Stuff: At the service of good ideas. September 23, 2020. https://www.copetti.org/writings/consoles/nintendo-64/.

Copetti, Rodrigo. “Super Nintendo Architecture: A Practical Analysis.” Rodrigo’s Stuff: At the service of good ideas. September 23, 2020. https://www.copetti.org/writings/consoles/super-nintendo/.

Crawford, Chris. *chris crawford on game design.* Indianapolis: New Riders, 2003.

Debussy, Claude. *Preludes*. Paris: Durand et Cie., 1910. https://imslp.org/wiki/Pr%C3%A9ludes%2C_Livre_1_(Debussy%2C_Claude).

“DMX Overview.” Unreal Engine. Accessed February 23, 2021. https://docs.unrealengine.com/en-US/WorkingWithMedia/DMX/Overview/index.html.

Dodge, Charles and Thomas A. Jerse. *Computer Music: Synthesis, Composition and Performance*, 2nd edition. New York: Schirmer Books, 1997.

“Donkey Kong 64.” Computer software. Twycross: Rare, November 22, 1999.

“Elias Studio.” Computer software. *Elias Adaptive Music: Adaptive Game Music and Sound*. Elias. Accessed February 21, 2021. https://www.eliassoftware.com/elias-studio/.

Eno, Brian and Peter Chilvers. “Bloom: 10 Worlds.” Computer software. GenerativeMusic.Com. Accessed February 21, 2021. http://www.generativemusic.com/bloom.html.

Eno, Brian and Peter Chilvers. “Trope.” Computer software. GenerativeMusic.Com. Accessed February 21, 2021. http://www.generativemusic.com/trope.html.

“DB-01.” Analog synthesizer. Riga: Erica Synths, 2020.

Farnell, Andy. *Designing sound*. Cambridge, Massachusetts: MIT Press, 2010.

“Fractal: Make Blooms Not War.” Computer software. Philadelphia: Cipher Prime, May 26, 2010.

“FTL: Faster Than Light.” Computer software. Shanghai: Subset Games, September 14, 2012. https://store.steampowered.com/app/212680/FTL_Faster_Than_Light/.

“FMOD Studio 2.01.08.” Computer software. *FMod*. Firelight Technologies, February 2, 2021. https://www.fmod.com/download.

Gu, Ning, Rongrong Yu, and Peiman Amini Behbahani. “Parametric Design: Theoretical Development and Algorithmic Foundation for Design Generation in Architecture.” In *Handbook of the Mathematics of the Arts and Sciences*, edited by Michael J. Ostwald, Kyeong-Hwa Lee, Torsten Lindstrom, Gizem Karaali, and Ken Valente, 1-22. New York: Springer, 2018.

Haydn, Joseph. *Symphony No. 45 in F-sharp minor, Hob.1:45*. London: Cianchettini & Sperati, 1808. https://imslp.org/wiki/Symphony_No.45_in_F-sharp_minor,_Hob.I:45_(Haydn,_oseph).

“Heavy Compiler Collection.” Computer software. *GitHub*. Enzien Audio, September 21, 2018. https://github.com/enzienaudio/hvcc.

Holmes, Thom. *Electronic and Experimental Music: Technology, Music, and Culture*, 4th edition. New York: Routledge, 2012.

Hosking, Jay. “Analog synth jam w/ Dreadbox Typhon & Abyss, Moog Matriarch & Minotaur, DrumBrute Impact, effects.” Jay Hosking, November 19, 2020. YouTube video, 5:44. https://youtu.be/iWLBmAGR_jc.

“How JALI Research drives CYBERPUNK 2077 Facial Animation & Multilingual Speech - Night City Wire.” Jali Research Inc, November 26, 2020. YouTube video, 2:48. https://youtu.be/UNGvHkGv5Qk.

Huang, Eugene. “GDC: next-gen audio will rely on MIDI, says Sony.” GamePro, March 16, 2007. https://web.archive.org/web/20080511031148/http://www.gamepro.com/news.cfm?article_id=106508.

Jackson, Woody. “Composer Woody Jackson on crafting the sound for *Red Dead Redemption 2*.” AV Club, March 18, 2019. Video, 10:22. https://games.avclub.com/composer-woody-jackson-on-crafting-the-sound-for-red-de-1833376805.

Jeriaska. “Myths, Mavericks, And Music of *Red Dead Redemption*.” Gamasutra, November 4, 2011. https://www.gamasutra.com/view/news/127900/Myths_Mavericks_And_Music_Of_Red_Dead_Redemption.php.

Jolly, Kent and Aaron McLeran. “Procedural Music in SPORE.” GDC Vault, March 2008. Conference recording, 1:00:07. https://www.gdcvault.com/play/323/Procedural-Music-in.

“Just Shapes and Beats.” Computer software. Quebec City: Berzerk Studio, May 31, 2018.

Kondo, Koji. “Painting an Interactive Musical Landscape.” GDC Vault, March 2007. Conference recording, 29:54. https://www.gdcvault.com/play/754/Painting-an-Interactive-Musical.

Kosak, Dave ‘Fargo.’ “The Beat Goes on: Dynamic Music in *Spore*.” GameSpy. IGN, February 20, 2008. http://pc.gamespy.com/pc/spore/853810p1.html.

Lamperski, Philip and Bobby Tahouri. “Real-time Procedural Percussion Scoring in *Tomb Raider’s* Stealth Combat.” *GDC Vault*, March 2016. Conference recording, 31:40. https://www.gdcvault.com/play/1023215/Real-time-Procedural-Percussion-
Scoring.

“The Last of Us Part II.” Computer software. Los Angeles: Naughty Dog, June 19, 2020.

Lee, Dami. “How *Untitled Goose Game* adapted Debussy for its dynamic soundtrack.” The Verge. Vox Media, September 23, 2019. https://www.theverge.com/2019/9/23/20879792/untitled-goose-game-nintendo-switch-debussy.

Lopes, Joey. “Sounds in Super Mario Odyssey Harmonize with the Background Music.” JalopesTL, November 4, 2017. YouTube video, 3:49. https://youtu.be/U5-YDxH6It8.

Madhav, Sanjay. *Game Programming Algorithms and Techniques*. Upper Saddle River, NJ: Addison-Wesley, 2014.

“Mario Kart: Double Dash!!” Computer software. Kyoto: Nintendo Entertainment Analysis & Development, November 7, 2003.

“Mario Kart: Super Circuit.” Computer software. Kyoto: Intelligent Systems, July 21, 2001.

Martinsart. “DrumSequencer.” Computer software. *Unreal Engine Marketplace*. Epic Games. Accessed February 21, 2021. https://www.unrealengine.com/marketplace/en-US/product/drumsequencer.

“Marvel’s Spider-Man.” Computer software. Burbank: Insomniac Games, September 7, 2018.

McDonald, Anthony. “Beyond the Score: Henry Cowell’s *The Banshee*.” New York Public Library for the Performing Arts, December 9, 2020. https://www.nypl.org/blog/2020/12/09/beyond-score-henry-cowells-banshee.

Moayeri, Lily. “Woody Jackson and Vox Studios.” Mix, January 11, 2019. https://www.mixonline.com/recording/woody-jackson-and-vox-studios-red-dead-redemption-2.

Moore-Colyer, Roland. “PS5 and 3D audio: Everything you need to know.” Tom’s Guide. Tom’s Hardware, October 6, 2020. https://www.tomsguide.com/features/ps5-and-3d-audio-everything-you-need-to-know.

Mozart, Wolfgang Amadeus. *Musikalisches Wurfelspiel, K.516f*. Bonn: N. Simrock, 1793. https://imslp.org/wiki/Musikalisches_W%C3%BCrfelspiel,_K.516f_(Mozart,_Wolfgang_Amadeus).

“Music of Changes.” John Cage Complete Works. Accessed February 19, 2021. https://johncage.org/pp/John-Cage-Work-Detail.cfm?work_ID=134.

Paschall, Alexander. “Unreal Engine 4.16 Released!” Last modified May 24, 2017. https://www.unrealengine.com/en-US/blog/unreal-engine-4-16-released.

“Paik, Nam June: One for Violin Solo.” Media Art Net. Accessed February 18, 2021. http://www.medienkunstnetz.de/works/one-for-violin-solo/.

Parviainen, Tero. “How Generative Music Works.” Last modified 2017. https://teropa.info/loop/#/airports.

Paterson, Brett. “MIDI Files.” Last modified November 2016. https://qa.fmod.com/t/midi-files/12804.

Payne, Richard J. and Jeffrey J. Blackford. “Interpolation.” In *Merriam-Webster.com Dictionary*. Accessed February 21, 2021. https://www.merriam-webster.com/dictionary/interpolation.

Periscope Studio. “Psai Music Engine Pro.” Computer software. *Unity Asset Store*. Unity, August 3, 2016. https://assetstore.unity.com/packages/tools/audio/psai-music-engine-pro-24788.

Phillips, Winifred. *A Composer’s Guide to Game Music*. Cambridge, Massachusetts: MIT Press, 2007.

Power, Tom. “As Donkey Kong 64 turns 20, the devs reflect on its design, the infamous DK Rap, and how a shocked Shigeru Miyamoto created the Coconut Shoot.” GamesRadar+. Future US, December 6, 2019. https://www.gamesradar.com/making-of-donkey-kong-64/.

Ramsay, Jill. “Quartz.” Last modified June 2020. https://portal.productboard.com/epicgames/1-unreal-engine-public-roadmap/c/19-quartz.

“Reason Studios 11.” Computer software. Stockholm, Propellorhead, September 25, 2019.

“Red Dead Redemption II.” Computer software. San Diego: Rockstar Games, October 26, 2018.

Reich, Steve. “Music as a Gradual Process.” In *Audio Culture: Readings in Modern Music*, edited by Christoph Cox and Daniel Warner, 304-306. New York: Continuum, 2010.

Ripin, Edwin M., Howard Schott, and John Koster. “Harpsichord.” In *Grove Music Online*. Oxford Music Online, 2001. Accessed February 18, 2021. https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000012420.

Saint, Dain and William Stallwood. “Interview with Cipher Prime about Auditorium HD.” Indiepub, October 12, 2010, YouTube video, 3:59. https://youtu.be/I_BdWjnO0ew.

Schyff, Dylan van der. “The Free Improvisation Game: Performing John Zorn’s *Cobra*.” *Journal of Research in Music Performance* (Spring 2013) 1-11. Accessed February 20, 2021. https://scholar.lib.vt.edu/ejournals/JRMP/2013/schyff.pdf.

“Sergei Prokofiev | Peter and the Wolf Op. 67 (with score).” ClassicalMusicScores, December 21, 2019. YouTube video, 26:14. https://youtu.be/ki0xu6Gl9Nc.

“SPORE.” Computer software. Redwood Shores: Maxis, September 4, 2008.

“Spyro: Reignited Trilogy.” Computer software. Novato, CA: Toys for Bob, November 13, 2018. https://store.steampowered.com/app/996580/Spyro_Reignited_Trilogy/.

“State of Audio in 4.25 | Inside Unreal.” Unreal Engine, April 2, 2020. YouTube video, 2:21:12. https://youtu.be/wux2TZHwmck.

“The Rise of VGM | Diggin’ in the Carts | Red Bull Music.” Red Bull Music, September 11, 2014. YouTube video, 15:05. https://youtu.be/m8z8-SKg3WU.

Toussaint, Godfried. “The *Euclidean* Algorithm Generates Traditional Musical Rhythms.” In *Renaissance Banff: Mathematics, Music, Art, Culture*, edited by Sarhangi, Reza and Robert V. Moody, 47-56. Alberta: Bridges Conference, 2005. http://archive.bridgesmathart.org/2005/bridges2005-47.html.

“Unravel.” Computer software. Sweden: Coldwood Interactive, February 9, 2016.

“Unreal Engine 4.22 Release Notes.” Unreal Engine. Epic Games. Accessed February 18, 2021. https://docs.unrealengine.com/en-US/WhatsNew/Builds/ReleaseNotes/4_22/index.html.

“Untitled Goose Game.” Computer software. Melbourne: House House, September 20, 2019.

“USynthComponent.” Unreal Engine. Accessed February 22, 2021. https://docs.unrealengine.com/en-US/API/Runtime/AudioMixer/Components/USynthComponent/index.html.

Webster, Molly. “Ears don’t lie.” Radiolab. WNYC Studios, December 28, 2012. https://www.wnycstudios.org/podcasts/radiolab/articles/257870-ears-dont-lie.

“What is Parametric Design?” Studio Robazzo. Accessed February 18, 2021. https://robazzo.com/journal/what-is-parametric-design.

Whitmore, Guy. “Getting Ahead of the Game Utilizing Music Design Templates in Wwise.” Audiokinetic, February 5, 2019. YouTube video, 1:10:20. https://youtu.be/_FWuiha_dw4.

“Wwise SDK 2019.2.9.” Computer software. *Audiokinetic*. Sony Interactive Entertainment, February 1, 2021. https://www.audiokinetic.com/library/edge/?source=SDK&id=releasenotes.html.

“Wwise SoundSeed Grain.” Accessed February 22, 2021. https://www.audiokinetic.com/library/edge/?source=Help&id=wwise_soundseed_grain_plug_in.

“Wwise SynthOne.” Accessed February 22, 2021. https://www.audiokinetic.com/library/edge/?source=Help&id=wwise_synth_one_plug_in.

Zorn, John. “The Game Pieces.” In *Audio Culture: Readings in Modern Music*, edited by Christoph Cox and Daniel Warner, 196-200. New York: Continuum, 2010.

Appendix

Hyperlinks to Supplemental Videos

I. Max Devices in Ableton Live 11

a. Euclidean Rhythm Generator: https://youtu.be/YzZcURJx8YQ

b. Bassline Generator: https://youtu.be/7hvPw-sXabQ

c. Algorithmic Percussion: https://youtu.be/yX791B8cUzc

d. FM Interpolation Synth: https://youtu.be/MlY8eoZhhPk

e. Parametric Design in Ableton Live: https://youtu.be/7U6fIHjuuWM

II. Unreal Engine 4 Technical Demonstrations

a. Dynamic Mode Mixture in Unreal Engine 4 - https://youtu.be/aWhfArO4uv0

b. 21st Century Dice Music - https://youtu.be/GWZS4O5mOHk