Games Without Frontiers: Audio Games for Music Production and Performance

Jason Hockman[1,2] and Joseph Thibodeau[1,3]

[1] Detuned Transmissions (DTND), Montreal, Canada/Birmingham, United Kindgom
[2] DMT Lab, Birmingham City University, United Kingdom
[3] Concordia University, Montreal, Canada
jason.hockman@bcu.ac.uk, joseph.thibodeau@concordia.ca

Abstract. The production of electronic dance music most often takes the form of sound-events arranged on a timeline, rendered as a stereo recording and mixed with other recordings in the context of mixes. Live electronic dance music on the other hand involves interaction with the sound-events in real time, but nonetheless subject to a master timekeeper. A similar comparison exists between sound for film and sound for (video) games, however in the case of games there is no assumption of a master timekeeper, and the timing of events is relative to the actions of the player and the state of the game environment. The authors explain a method by which electronic dance music can be produced in a similar manner to producing game audio, and in fact that an interactive or live piece can be considered an audio game. Musical structure is composed conceptually as scenes in a film, where interacting sound-machines generate rhythmic patterns of sound-events that place the performer/player in a virtual space. The performer/player pursues musical goals in non-linear time while maintaining the ability to sequentially arrange pieces in a coherent mix.

Keywords: interactive, virtual spaces, procedural audio, sound design

1 Introduction

The emergence of the gaming industry has played a crucial role in the development of visual aesthetics and narratives in the pre-existing film and television industry. The creation of synergetic film and game releases such as *Lara Croft: Tomb Raider* (2001) and the *Resident Evil* series (2002–16) is indicative of the confluence of two industries that share the primarily visual perspective. A second modality—no less relevant to both industries—is that of the sonic domain.

Film sound exists mostly as a post-production process, and is intended to represent on-screen cues that exist in *linear time*—that is to say sound-events occur and vary in a predetermined sequence. Sounds are usually built up through meticulous layering of several recordings for a single event. Game sound exists in *non-linear time*, where sound-events occur and vary relative to the player's interaction with a system. Again sounds are most often the result of several

layers of recordings; however, depending on the state of the game environment and the player within it, a game must deliver and modify these layered sounds appropriately. Another approach is to design sounds procedurally as in [1], which has the advantage of parametrically modelling the way a sound is generated in real-time, albeit at the expense of processing resources.

Audio games are a subset of video games that forego visual representation altogether, and focus chiefly on the auditory modality. These games have been developed for a variety of educative and entertainment uses, and are a key component in the development of an accessible gaming culture for the visually impaired [2], [3]. As in audio-visual games, sound in audio games must be delivered and modified dynamically, but since sound is the sole modality for representing the game state and providing the player with feedback, audio games present a unique set of challenges in the use of sound [4].

The aim of this paper is to explain how sound design techniques used in game development may be similarly applied for the purpose of non-linear music creation. The resultant pieces can be thought of as audio games in their own right, with the performers as players who interact with a virtual space that is simultaneously an abstract musical structure with musical goals.

The process that led the authors to develop these techniques serves to illustrate the process of building audio games for music production: layering and editing of sounds on a linear timeline, arranging sound collections into a filmic "scene", programming causal relationships between sounds, and interacting with the system.

2 Motivation

We began our journey into electronic dance music (EDM) through the use of tools designed for linear audio playback. As with many EDM producers, our earliest releases were produced through the use of digital audio workstations (DAW) such as MOTU Digital Performer[1] and Emagic Logic (later Apple Logic[2]). As much of our output had been intended for playback in DJ sets, the form and structure of the subgenres we were attempting to work within was paramount to the development of the music we created, as was its manifestation as static compositions (in vinyl or sound-file format). Under our production name DAAT, we began to focus on thematic concepts related to cyclic events, such as the interplay of mechanistic components (e.g., gears) within vehicles, which we implemented largely through the manipulation of field recordings. By 2014 (when we composed the album *HVAC*[3]), software sampling technology had improved sufficiently to allow for a vast amount of modulation capabilities that we used to add more and more detail to the components and to imply interactions between them. The complexity of our machine concepts grew into a film-like approach of representing the same machine from different points of view and at different points

[1] http://www.motu.com/products/software/dp
[2] https://www.apple.com/logic-pro/
[3] https://www.beatport.com/release/hvac/1320332

in time. As the scenes became more intricate, we needed these machines and environments to truly interact, rather than seeming to interact through clever editing. Once this motivation became apparent to us, we wanted to add more high-level control to the developed mechanisms—such as changing the speed of a vehicle and having this variable reflect upon the resultant environmental effects on the vehicle (e.g., hull vibrations).

As we were attempting to do this in a standard DAW environment, we began to run into problems for which programming solutions would be more suited (e.g., variable sharing across functions, global modulation parameterisation). At this point, we began to look into alternative production environments such as programming languages (e.g., Max/MSP, Puredata, CSound, SuperCollider), however we found that removing the DAW from the process entirely removed many of the audio processing mechanisms and workflows that formed the basis of our existing practice, not to mention removing elements of an EDM aesthetic that stem from a basis in DAW production.

Eventually we started using Ableton Live[4] and Max for Live[5] as a way of bridging DAWs and programming languages. Over the course of the past two years we have developed a method of production that is analogous to game audio production, but rooted firmly in EDM production and film sound design. In the subsequent sections, we will describe some of the techniques we have employed through providing examples from our initial work from within a DAW, along with examples from our more current methodology.

3 Dynamism

In offline production with a DAW, sounds are typically fixed in time. The score progresses from left to right, triggering events until the song is finished. For electronic dance music, the events tend to reinforce metrically relevant points in time, much the same as events in a film soundtrack reinforce relevant visual cues. Shaping the form of an event involves editing a potentially vast number of parameters depending on the level of detail. Multiplying this by the number of individual sounds in a project as well as the controls involved in mixing, the total parameter-space of the piece is practically endless.

The attention, time and energy of a producer is limited, and different people will focus on different things. In our case the detail-work moved away from tonal composition toward a focus on percussion and soundscape. Eventually the soundscape took over, with events still reinforcing metrically relevant positions to preserve the "mixability" of the music. One type of event that we have often used is the sound of something passing the listener, such as one vehicle passing another on a highway.

In *Apache* (2014)[6], a futuristic helicopter is presented from an evolving perspective from inside the cabin. Air rushes past vents and windows while sensor

[4] https://www.ableton.com/en/live/
[5] https://www.ableton.com/en/live/max-for-live/
[6] https://www.beatport.com/track/apache-original-mix/5509952

indicators and unintelligible radio chatter infect the environment. Multiple rotor sounds then emerge at different timescales, further placing the scene above the land.

In addition to a conceptual height suggested by these sounds, the helicopter is given width and depth perspectives to provide additional dynamism associated with navigation of the vehicle. The helicopter passes other airborne vehicles either to the left or the right of the aircraft. This is achieved through a simple Doppler-like effect that adjusts modulation envelopes controlling panning, filtering and amplitude across multiple field recordings for each vehicle passed. In *Apache*, and other tracks (e.g., *Running Man* (2014)[7] and *Sixth Gear* (2014)[8]) the events were laboriously hard-coded in the DAW with MIDI, and automation data was used to adjust the pitch of the samples independently to affect the size of the airborne vehicles.

The soundscape created in *Apache* was a first step towards adding dynamism to an envisaged flying machine, however there were several issues associated with the created representation that did not meet our criteria for the intended effect. While the soundscape had achieved a somewhat three-dimensional affect, the absent sensation was that of varying speed. To avoid this stasis, both *internal* and *external* sounds would need to be mutable. Internal sounds were those associated with the machine's functionality. In the case of *Apache*, the helicopter had relatively simple sounds associated with it (i.e., engine noises, rotor blades). External sounds were those sounds generated from the machine's interaction with an outside environment (e.g., wind noise, structural integrity). To make the model more convincing, both internal and external sounds would have to be modifiable, and placed under a mapping strategy that simulated a high-level variable such as *speed*. Passing such a variable throughout the various tracks in *Apache* was not possible (or at best extremely laborious) due chiefly to the track being created in a standard DAW.

In *11D* (2017)[9]—one of our first ventures in a programmatic music environment—a similar vehicle is presented, however this machine also incorporates user control (via game controller) for several variables, including the selected route and engine revolutions per minute (RPM).

11D depicts a scene from the vantage point of a flying vehicle travelling through airborne motorways on the periphery of a city. Cyclic motor sounds and combustion noise output from the engines filter into a semi-permeable cabin under pressure from high-speed travel. As the machine accelerates onto the main route it overtakes several other vehicles. An afterburner initialises the final sequence speeding the craft into a system malfunction overheating the vehicle and ultimately leading to a fiery crash.

In a programmatic environment, the vehicle becomes a dynamic sound-machine: a virtual construction of related procedural audio generators (as in [1]) whose motion relative to its surrounding environment is indicated by generated per-

[7] https://www.beatport.com/track/signs-original-mix/5509956
[8] https://www.beatport.com/track/sixth-gear-original-mix/5509960
[9] https://www.beatport.com/track/11d-original-mix/9468067

ceptual cues (as in [5]), which carries the listener's ear. The mechanical sound of the engine is created through several short segments of field recordings of gears. A single rotation of the cylinder is achieved by cyclical switching between these recordings. The speed variable (here called RPM), which is directly controlled by the user, may then be used to adjust the cycle rate of the engine. At higher speeds, low-frequency bass tones emerge to represent sympathetic resonances of the chassis to the engine. Musical rhythm is represented through an effect on the structural integrity of the craft. This is negative space achieved by interruptions of the bass resonance by external wind gusts.

To achieve the Doppler-like effect, a similar approach to *Apache* is adopted. The panning, filtering, and amplitude variables are still controlled by envelopes, however these are dynamically changed based on the speed of the user's vehicle and the chosen route type (see Fig. 1). The motorway vehicles are randomly generated with four densities and speeds based on (arbitrary) motorway route types: *outskirts*, *city limits*, *main route*, *city roads*. Alternatively, the speed and density of these vehicles may be data-driven; information from a traffic website (e.g., Traffic England[10]) is scraped and sent via OSC into the Max for Live object. The user's RPM variable is then associated with the vehicle to be passed and appropriate values for the envelopes are generated for each vehicle being passed.

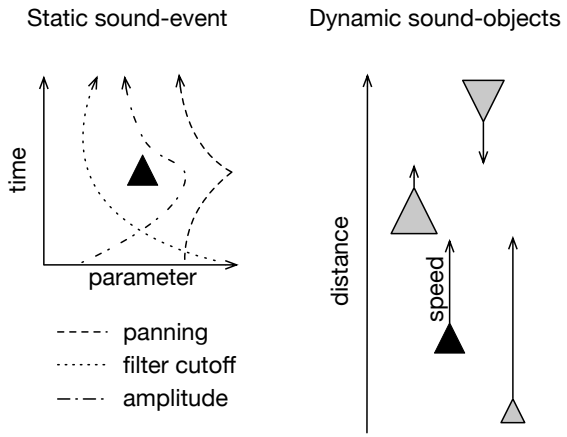

Fig. 1. A comparison between static sound-events like those used in *Apache* (2014) and dynamic sound-objects like those used in *11D* (2017). In the static example automation curves control parameters such as panning, filtering, and amplitude. In the dynamic example the size, relative position, and motion of the other vehicles is included in the resultant sound-events. In both examples the black triangle represents the listener's perspective.

[10] http://www.trafficengland.com

4 Interaction

As in other genres, musical form in EDM often comprises several sections [6]. For example, a popular structure in Drum and Bass consists of the intro (A), first drop (B), breakdown (C), second drop (B2) and outro (A2). While this form may be appropriate for many tracks, it is often very useful for producers to investigate a variety of other possibilities in the development of their tracks. Due to the difficulties in managing a large amount of hard-coded MIDI and automation data, it is not always feasible to explore these possibilities within a traditional DAW environment. However, once the various components are connected through programmatic design, it is much easier to navigate the solution space to find a time-scale and form that is optimal for a piece.

Sub (2014)[11] opens in the dormant control room of an automated warship submerged in a waterway and staged for battle. A message is received and the submarine's pressure control system is activated. The vehicle moves upwards, where it surfaces and is met by waves splashing against the hull. Distant explosions are briefly heard above the cries of gulls before the submarine dives once again with a final sounding of its radar system.

The whole of *Sub* takes place in three states: *rest*, *surfacing* and *diving* (as can be seen in the left graph in Fig. 2). All three states were created using a DAW with a variety of field recordings as sound sources. The transitions between the *underwater* and *diving* states were achieved though DAW automation associated with filters and additional effects processing. The *rest* state, which takes place in the envisioned control room comprises several computer display indicator sounds. When the pressure control system is activated, this triggers a state change to move from the *rest* state to the *surfacing* state. After a series of transitions between the *surfacing* and *diving* states, the submarine returns from the *diving* to the *rest* state. Selecting an optimal rate of change and temporal positions for these transitions was a tedious process as a large amount of automation data needed to be created for each state change. Additional modulation and automation was included to ensure that a listener would not hear the same events and structure upon revisiting states. The result is a track that exhibits the dynamism of state transitions but exists solely within a linear timeline. *010005* (2017)[12] occupies the same conceptual background as *Sub* in that it is intended to be part of the same battle scene. Using causal links between events housed on different tracks in Ableton and Max for Live, state changes are triggered that navigate the narrative of the piece.

In *010005*, the slow combustion of a giant engine powers a well-seasoned warship into choppy water at the mouth of a river. The scene shifts below deck, where waves can be heard violently hitting the hull. The drone of a wave of attack aircraft can be heard overhead and another ship, already involved in the fray radios in. The ship's radar system is activated and targets the enemy; guided

[11] https://www.beatport.com/track/sub-original-mix/5509958
[12] https://www.beatport.com/track/010005-feat-books-original-mix/9468066

missiles are launched as the aircraft fly overhead. Once the enemy aircraft have been eliminated, resounding euphoric victory music is heard all around.

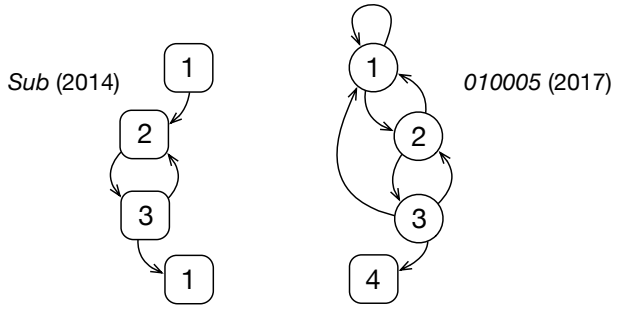

Fig. 2. State representations for the structure of *Sub* (2014) and *010005* (2017). Rounded rectangles denote states that unfold in linear time and circles denote states in which time is defined by the user. *Sub* (left) unfolds in a linear timeline and consists of three states *rest* (1), *surfacing* (2) and *diving* (3). *010005* (right) unfolds in a user-determined non-linear timeline and consists of four states: *cruise* (1), *scan* (2), *attack* (3) and *level complete* (4).

Unlike *11D*, which comprised a single state with a continuous variable associated with change in the environment, *010005* has four defined states: *cruise*, *scan*, *attack* and *level complete* (as can be seen in the right graph in Fig. 2). In addition, unlike *Sub*, *010005* exists in non-linear time, in that the unfolding of events relies on user input. In the initial *cruise* state, hull sounds are randomly generated at metrically-relevant positions at the front and rear of the ship. Once the *scan* state is triggered by the user, the radar sweep sound effect is activated, and the positions of enemy aircraft populate the stereo field. The track transitions directly to the *attack* state, during which missiles are launched from turrets. A distance model generates the duration and amplitude of the trail of each missile as it is fired, and provides an auditory indicator representing a heads-up display tracking the missile to its destination. If the explosion is too close to the ship, blowback causes powerful waves to collide with the hull. Once the enemy armada has been defeated, the track transitions to its end state, *level complete*, during which euphoric synthesiser pads play.

Movement between these states is facilitated by message passing between Max for Live patches in each track, which trigger events without the use of hard-coded MIDI or automation. In addition, subtle variation is achieved through adding either random or probabilistic control over environmental sounds and amplitudes. The removal of deliberate control over many processes is advantagous in this sense as it allows events to be put into action without knowing the patterns that will emerge. This can result in unintended consequences, which is very useful for reducing listener fatigue and creating a space for serendipitous discovery.

5 Games Without Frontiers

A major question that began to arise in the production of these pieces is how to perform them. Since they are internally interactive (the component sound-objects interact with each other) there are plenty of ways for one to "hook in" to certain parts of the system and influence the whole.

Over the course of creating a piece such as *11D* we would determine useful high-level parameters that we would assign to a gamepad or MIDI controller, effectively allowing us to dictate the behaviour of the system on a large scale. As we became more familiar with the process of creating these pieces, we wanted to explore them as engaged participants, rather than detached empyrean figures. From this point, thinking of them as *games* arose naturally.

In simulations or first-person games, players inhabit a sensorium within virtual worlds, and ideally are immersed in the experience. They have limited freedom to navigate the space, goals (whether explicit or implied) and a degree of risk. This is similar to performers who have musical freedoms, goals, and risks according to the specifics of their selves in relation to the performance environment. By merging the relationships of music/performer and game/player, we found ourselves able to design interactions that were simultaneously audio games and musical compositions.

AC3244 (2017)[13] tells the story of a lonely jet-gull who lives on aluminium cliffs by the sea. It is resting when it spots a predator in the distance, a gargantuan sky-whale. Taking flight, the unfortunate bird draws the predator's attention and flees as fast as its thrusters can handle. Its reactor is fundamentally unstable, and loss of focus can cause it to overload, falling to the ground to cool down. If it doesn't outrun its pursuer, it will be eaten whole and melted for scrap.

In *AC3244*, the performer takes the role of the jet-gull and must live through its life-threatening tale. The control scheme is simple yet challenging. One button ignites the power plant and starts the takeoff, after which two sliders are used to push the jet-gull to its limit. One slider controls the throttle, directing the available reactor power to the thrusters that in turn increase the speed. The other slider controls the reactant feed, increasing the available power. The state of the reactor is modeled as an inverted pendulum that the performer has to balance using the reactant feed. It becomes more unstable as the speed increases, and so becomes increasingly difficult to balance as the performer attempts to outfly the pursuer. If the reactor overheats, the jet-gull crashes into the soft sand by the water and daydreams about simpler times. After cooling down the player can re-ignite the engine, but by this time the sky-whale is understandably much closer and the task that much more difficult. It is worth noting that both of the flying controls are *indirectly* mapped to the higher-level parameter of speed. This serves both to mitigate the low-resolution control of MIDI messages and to create a sense of being a part of a larger system rather than its controller.

[13] Unreleased

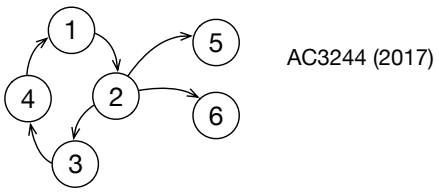

Fig. 3. State representations for the structure of the player's vehicle in *AC3244* (2017). It consists of six states: *rest* (1), *flying* (2), *overheated/crashing* (3), *cooldown* (4), *escape* (5), and *eaten* (6). The player traverses these states depending on their ability to maintain the vehicle's flight and escape their pursuer.

Much like *11D* and *010005*, this piece comprises several states through which the player can progress, illustrated in Fig. 3. Unlike the previous examples, these states refer to the states of the jet-gull rather than states of the piece as a whole. The state of *flying* moves the player at a certain speed along a track, at the end of which is the reward of living another day. It includes the possibility of "losing"—being eaten by the predator before reaching the finish—and the performer has to balance these game goals with musical ones. Is it better to accept a timely death for the sake of double-dropping, and is it possible to keep the jet-gull's reactor stable long enough to mix in the next track?

Designing audio games is known to have unique challenges, such as informing the player about the game-state while conveying aesthetics and not cluttering the soundscape [4]. Designing audio games for electronic dance music brings further challenges in terms of how a given piece will fit into a larger set. There are technical complications that make this non-trivial. The primary difficulty is that each interactive piece or audio game is an intricate system requiring its own project file and controller mapping, so it cannot easily be played along with another project on the same computer. In practice, we have found it necessary to run two computers, one for the interactive pieces and one for mixing and playing prepared material or clips. This is also insurance against anything going wrong or "losing" one of the games—in which case it is possible to mix out and recover. Ensuring that interactive pieces mix well is also a challenge, as it requires designing the piece to reinforce an underlying metrical structure when desired, even from within non-linear time. Experimentation and practice are necessary to achieve an understanding of how to perform each piece, whose controls have their own unique nuances.

6 Conclusions and Future Work

Looking over the progression from detailed audio editing on a linear timeline to dynamically controlled sound-events to systems of interaction and audio games, we have described a process of identifying static, overly deterministic or linear structures of control and making them dynamic. At first the low-level parameters of a sound-event were static, so we coded models to instantiate and change

them based on context. We then modified the structure of a piece to make state progression non-linear. Finally, we detached direct control over global states by placing the performer inside the piece where their global influence changed based on context.

There are many fruitful directions in which to go from here. The state-space of individual sound-machines and their causal relationships could benefit from application of machine learning and AI such that their behaviours and interactions are not fully predictable. Further explorations into audio game design and immersive soundscapes will allow us to further enhance the sense of space and increase the subtleties of the game mechanics. By treating the entire performance space as an interactive installation with sensors and feedback devices, we could shift the performance away from the stage to include the immediate physical context of the concert in the game-space of a piece.

There are exciting applications to these techniques in a VR experience, and we are currently working with the National Film Board of Canada in the development of an open-ended game known as *Biocube*, [14] in which the player's long-term interactions with the environment serve to build empathy with plant life and to compose a personalized immersive soundscape.

References

1. Farnell, A.: Designing Sound. MIT Press, Cambridge (2010)
2. Archambault, D., and Olivier, D. How to Make Games for Visually Impaired Children. In: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, pp. 450–453. (2005)
3. Chakraborty, J., Chakraborty, S., Dehlinger, J., Hritz, J. Designing Video Games for The Blind: Results of An Empirical Study. In: Universal Access in the Information Society, pp. 1–10. (2016)
4. Röber, N., and Masuch, M.: Leaving the Screen: New Perspectives in Audio-only Gaming. In: Proceedings of the 11th Meeting of the International Conference on Auditory Display, pp. 92–98. Limerick, Ireland (2005)
5. Merer A., Aramaki M., Ystad S., Kronland-Martinet R. Perceptual Characterization of Motion Evoked by Sounds for Synthesis Control Purposes. In: ACM Transactions on Applied Perception, vol. 10, pp. 1–24. New York, USA (2013)
6. Butler, M. J.: Unlocking The Groove: Rhythm, Meter, and Musical Design in Electronic Dance Music. Indiana University Press, Indiana, USA (2006)

[14] http://onf-nfb.gc.ca/en/produce-with-the-nfb/english-program/projects-in-production-and-development-at-the-nfbs-english-program/