AEGIS AUDIO ENGINE: INTEGRATING A REAL-TIME ANALOG SIGNAL PROCESSING, PATTERN RECOGNITION, AND A PROCEDURAL SOUNDTRACK IN A LIVE TWELVE-PERFOMER SPECTACLE WITH CROWD PARTICIPATION

Ivica Ico Bukvic

Virginia Tech,
SOPA, DISIS, ICAT
Blacksburg, VA, 24073, USA
ico@vt.edu

Michael Matthews

Virginia Tech,
SOPA, DISIS, ICAT
Blacksburg, VA, 24073, USA
michael.matthews@techmdinc.com

ABSTRACT

In the following paper we present Aegis: a procedural networked soundtrack engine driven by real-time analog signal analysis and pattern recognition. Aegis was originally conceived as part of Drummer Game, a game-performance-spectacle hybrid research project focusing on the depiction of a battle portrayed using terracotta soldiers. In it, each of the twelve cohorts—divided into two armies of six—are led by a drummer-performer who issues commands by accurately drumming precomposed rhythmic patterns on an original Chinese war drum. The ensuing spectacle is envisioned to also accommodate large audience participation whose input determines the morale of the two armies. An analog signal analyzer utilizes efficient pattern recognition to decipher the desired action and feed it both into the game and the soundtrack engine. The soundtrack engine then uses this action, as well as messages from the gaming simulation, to determine the most appropriate soundtrack parameters while ensuring minimal repetition and seamless transitions between various clips that account for tempo, meter, and key changes. The ensuing simulation offers a comprehensive system for pattern-driven input, holistic situation assessment, and a soundtrack engine that aims to generate a seamless musical experience without having to resort to cross-fades and other simplistic transitions that tend to disrupt a soundtrack's continuity.

1. INTRODUCTION

There are generally two basic models for audio engine design in video games. In one model, sound layers are mixed together to create the audio track. When the game becomes more intense, additional sound layers are added to reflect this while the audio that was already present remains audible. In the other model, all of the game audio is in one file with markers indicating locations of key start points for different sounds and music. As the game changes and calls for different sound cues, the appropriate marker in the audio file is played. In order to avoid gaps or jarring changes, a sound file specifically designed to be a transition piece is often played during the switch to a new marker in the main audio file [1].

"If we are to coin a term to highlight … the current [procedural] way of doing [game audio it would be] the 'data model'." [2] Large amounts of sound clips are recorded, edited, and arranged, much like the way that sample libraries have been produced for music samplers. They are matched to code in the game engine logic using an event-buffer manager like the FMOD [3] or WWise [4] audio system. [2] Both FMOD and WWise present a DAW-like approach to programming sound for video games, and include elements of procedural audio, such as game-driven music scores, real-time effect processing, and editing of audio while the game is running. However, neither system allows for the flexibility of dynamic sound creation in the way that MaxMSP [5], the programming environment used to build Aegis, is able to provide.

Some of the drawbacks of the procedural audio model, as outlined by Farnell, are that aesthetic decisions usually must be made early, and revisions later in the development process can be both difficult and expensive. Sound has historically been placed below visual and interactive elements in games [6], and considering it early in the development cycle can be challenging.

Benefits of the procedural audio model include the ability to closely reflect actions and events within the in-game world. Each object and action can, and indeed usually does, have its own unique sound associated with it. The game music is also less strict in terms of time and form, allowing the composer to focus more on the overall shape of the music, and how it will contribute to and affect gameplay and performance throughout the course of the game [2].

In recent years, "audiogames" and "mostly-audio" games have been built on the idea of using audio to supplement or completely replace a visual representation of the game [7][8][9]. These games use audio to guide and prompt the user to interact with the game engine through what are still traditional methods of control: tactile buttons, styli, gamepads, keyboard and mouse [8][10].

2. DRUMMER GAME

Drummer Game is a transdisciplinary education-centric research project conducted in the summer 2011 that aims to integrate digital signal processing, music, visual arts, digital storytelling, gaming, computer sciences, and engineering. Funded by United States' National Science Foundation [11], its educational focus is on the Chinese history, and specifically terracotta soldiers [12] that was in part inspired by the

This material is based upon work supported by the US National Science Foundation (NSF) under Grant No. 0940723. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation.

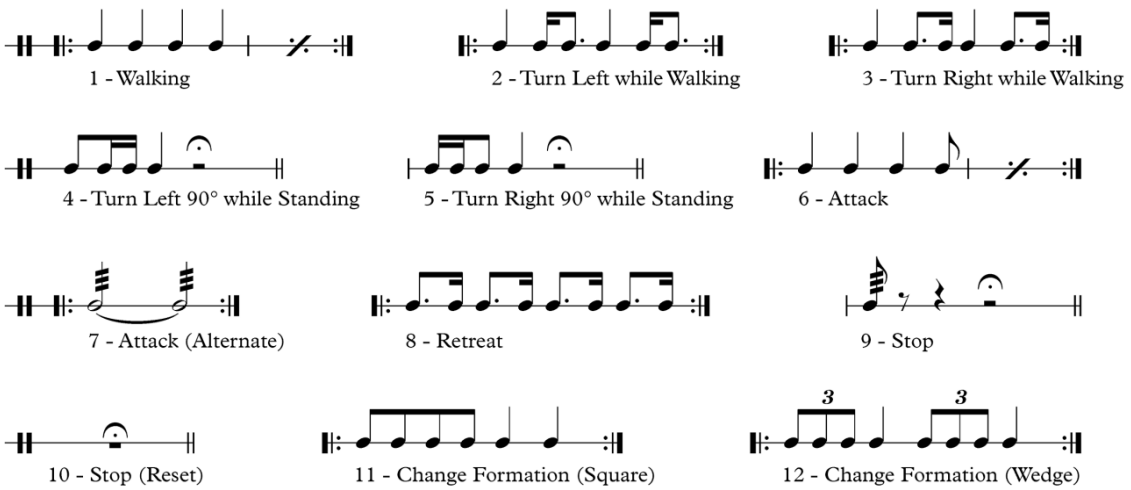


Figure 1: The twelve rhythmic patterns and the corresponding commands.

inaugural world tour of this historical and cultural wonder. Drummer Game's original aim was to produce a spectacle-like event where two teams would face off in front of a large audience. Each team consisted of six cohorts of terracotta soldiers led by six players with the ultimate goal of either destroying or forcing all enemy soldiers to retreat. Unlike traditional gaming paradigms where players typically control their units using various mainstream controllers, in this case players are trained musicians, percussionists whose task is to play one of the six authentic Chinese war drums of varying sizes. By accurately playing and repeating precomposed rhythmic patterns, each performer issues one of the twelve commands to their cohort of terracotta soldiers (Fig. 1). Upon being destroyed, a terracotta soldier turns into clay, essentially becoming an obstacle for the rest of the cohort and its desired formation. During the battle, audience cheering for two sides can further contribute to their respective army's morale, with possible approaches including directional amplitude and brightness (e.g. via smartphone flashes) monitoring, and/or embedded applications running on smartphones. As part of its educational focus, the game also integrated authentic historical data associated with Chinese culture pertaining directly to the clay terracotta soldiers.

This paper focuses solely on the audio-related component of this project, namely the Aegis engine consisting of an analog signal capture from the Chinese war drums, its analysis and interpretation within the context of preexisting patterns, and the networked game audio engine that supports audio cues (special effects), and more importantly a procedural soundtrack that is driven by the analysis of cues from the players, as well as cumulative situational awareness. For the purpose of analysis we have divided the engine into two components, the analog signal analysis and pattern recognition, and the networked procedural soundtrack engine. While there is also a third component pertaining to sound effects, this component was not fully developed as part of this project and as such it is not included in the following analysis.

3. THE AEGIS ENGINE

Unlike other audio engines on the market, Aegis provides integration of analog audio driven controller with a pattern recognition, and a procedural soundtrack capable of keeping tempo, meter, and overall structural integrity amidst fluid change of game states, as observed through captured controller data and the overall game's situational awareness. Given that Drummer Game, the catalyst for the creation of Aegis, is controlled entirely through audio, or more specifically detected acoustic rhythms, Aegis is designed to analyze these patterns and generate control information for the game engine from a musical instrument--the authentic Chinese war drums. Aegis also builds upon the elements of a procedural audio model, and utilizes the same to create a dynamic and organized aural experience for the game players. Its design also defies traditional production cycle--because of the crucial role the sound and music played in Drummer Game, Aegis and the soundtrack it controlled were among the first elements of the game platform to be developed.

3.1. Analog signal analysis and pattern recognition

3.1.1. Capturing drum input

The project called for authentic Chinese war drums with water buffalo skin membranes on both the top and the bottom of the drum frame. There were two sets of six drums of unique sizes ranging from approx. 10 to 48 inches (Fig. 2).

Although there is a level of subjectivity associated with this process, rhythmic patterns assigned to various actions were composed to best reflect musically the desired in-game action. For instance, a simple "square" 4/4 rhythm was chosen to characterize a change in army stance to a "square" formation, whereas a change to a "wedge" (or triangle) formation was represented with a triplet rhythm (Fig. 1).

Given that in the production setting, up to twelve drums were to be in close proximity to each other, the ensuing bleed-over of the amplitude spikes rendered traditional microphone techniques inadequate. This was particularly apparent with

larger drums that had a pronounced residual envelope and, due to a powerful dynamic range, they tended to cross-pollute nearby drums whose membranes' sympathetic vibrations

Figure 2: Zhangu Chinese drums Bian Gu and Tong, two of the six war drums used in the Drummer Game.

could be easily misinterpreted as false positives. Similarly the problem was apparent at faster tempi where a residual envelope's duration could easily elide with subsequent attacks causing practically no observable attenuation in the overall signal amplitude. For these reasons, we resorted to utilizing contact piezo transducers from Radio Shack [13] (Fig. 3) that were to be mounted onto the drums themselves. While these transducers offer a very limited frequency response, they proved adequate for the envelope tracking purposes.

Piezo transducer placement proved challenging due to the amount of vibration of both the membrane and the drum frame, particularly on larger drums. Upon investigating optimal placement strategies it was determined that for larger drums the best placement was on the side of the drum frame away from both top and bottom membranes (Fig. 4). Smaller drums allowed for a less conservative placement on the edge of the drum membrane where it stretches over the frame, in part because their resonance was less pronounced, and in part because the curvature of the frame did not allow for flush mounting (Fig. 4). The piezo elements were fastened with tape to prevent damage to the drum frames.

To further dampen the residual envelope, rather than placing the piezo element in direct contact with the drum, the transducer was kept in its plastic casing that served as a dampener of the vibrations. Further dampening was achieved using a soft cardboard material placed between the transducer and the drum frame. The ensuing setup allowed for a reasonable isolation of individual attack from their respective residual envelopes (Fig. 4).

Figure 3: The contact piezo transducer used for the Aegis analog signal capture.

3.1.2. Software signal filtering

Audio signal was fed into Max/MSP via a multichannel USB soundcard (Fig. 5) where a simple hysteresis [14] amplitude

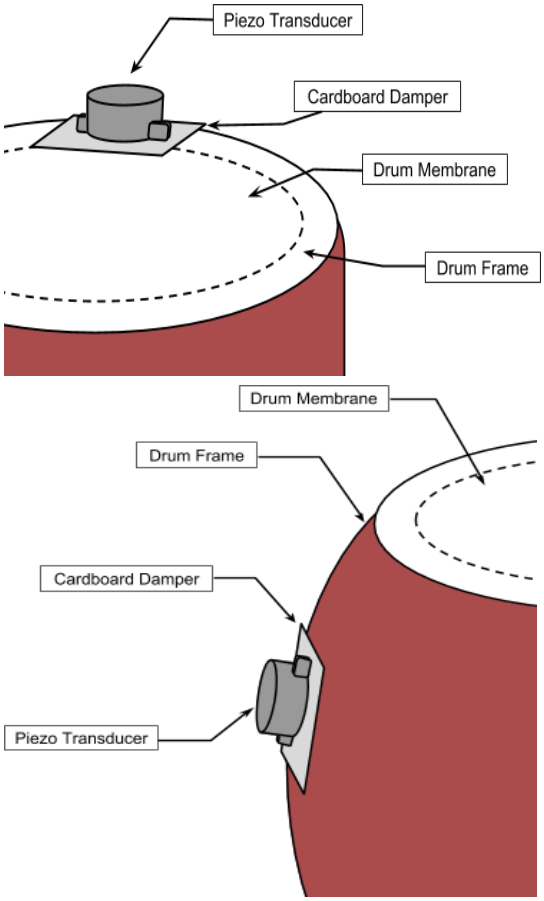
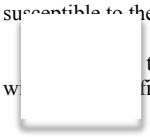


Figure 4: Diagram showing placement of piezo and damping element on the edge of the drum frame for the smaller drums (top), and on the side of the larger drums (bottom).

envelope follower (a.k.a. noise gate) with a high (hit detection) and a low (hit release) threshold was utilized to observe individual impacts. While analyzing signal, it was determined that despite the use of piezo transducers, sympathetic vibrations continued to pollute signal fidelity to the point where it was impossible to rely on the hysteresis alone. Even though the drums differed significantly in size, and each drum size offered a unique frequency response range where its amplitude spike was most pronounced, it is worth noting that it was mostly lower frequencies that were affected by sympathetic vibrations and the transfer of energy via the floor and air. Higher frequencies, while reflecting the actual hit, offered a much quicker attenuation of the residual envelope, thus posing as an equally effective frequency range for the hit detection as that of the entire audible range, while being less susceptible to the transfer of energy among different drums.



the software envelope follower was retrofitted w figurable FFT-based bandpass filter we hereby

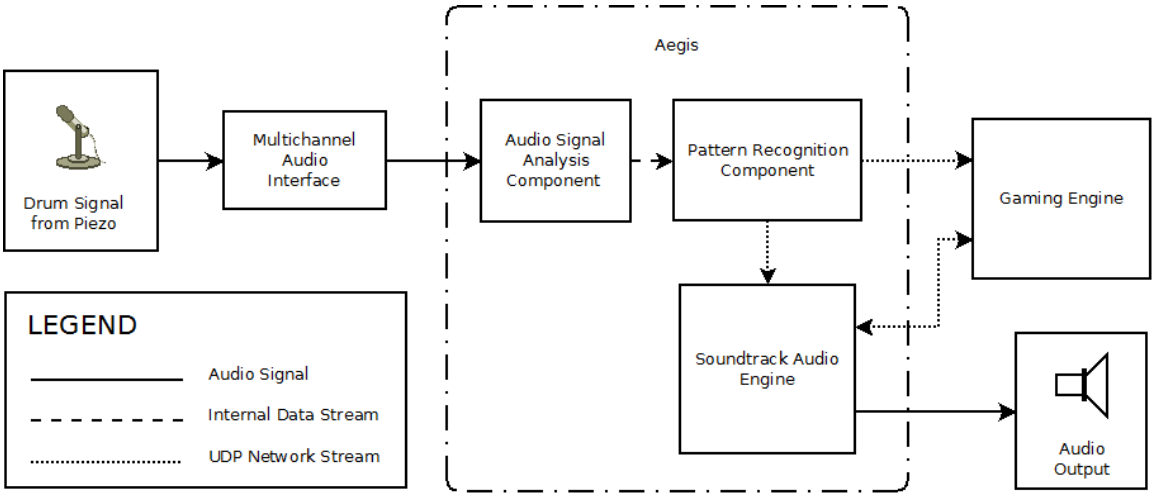


Figure 5: The Aegis signal network.

refer to simply as a "binpass" filter that offered unique frequency ranges to each of the audio inputs and consequently their respective drums (Fig.6). As observed in our tests, the binpass when coupled by a hysteresis amplitude envelope follower and hit detection algorithm, allowed for a significantly more reliable hit detection than when using the hysteresis approach by itself. The ensuing system allowed for the detection of hits at two varying levels of loudness primarily because it was determined that under the anticipated relatively noisy production conditions that in many ways mimicked a noisy war environment, any further granulation of the dynamic range would have limited impact on both performers and the audience while requiring a significant increase in complexity and potential for amplitude-based misclassification.

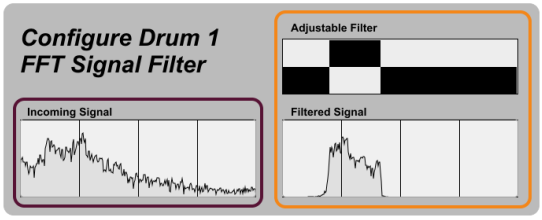


Figure 6: Signal analysis and user-configurable per-channel FFT binpass filter.

3.1.3. Pattern recognition

The ensuing signal analysis is broadcast to the pattern recognition module in the form of either a soft or a hard hit, both of which were essentially a "bang" signal dispatched to different inlets. Inside the pattern recognition module are twelve iterations of a pattern recognition algorithm, each preloaded with a specific coll Max-external-formatted text file reflecting a unique pattern. Each hit within this text file was defined by its ordinal position, hit intensity (1=soft, 2=hard), and a duration ratio in respect to the longest lasting hit in the pattern. For instance, a right-turn-while-standing order is portrayed in Fig.7.

Therefore, every incoming hit was dispatched to an abstraction and stored locally into an array the size of the pattern it was being compared against. This design allowed for the ensuing recorded pattern to be compared against the reference pattern in all its permutations (e.g. 123 would be compared against 123, 231, and 312 permutations of the original pattern, thus allowing for the fastest possible recognition of a given pattern, particularly in cases where a hit may have been misinterpreted). In situations where a patterns' permutations shared too much in common with other similar patterns, the shorter pattern of the two was duplicated, thus extending its length and through its increased complexity essentially eradicating a chance of a false positive. In order to provide a contextually meaningful comparison, like the coll-formatted text file containing the reference pattern, each hit was given an ordinal number from 1 to the pattern length, strength (1=soft, 2=hard), and a timestamp. The timestamp data was converted to ratios, using the longest event as a

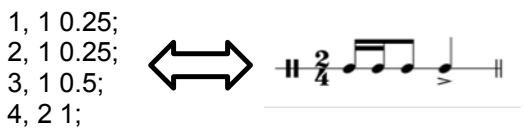


Figure 7: Coll-formatted rhythmic pattern for the right-turn-while-standing order and the corresponding score.

normalized value of 1. The ensuing data could then produce a dataset ready to be compared with reference pattern.

The use of ratios allowed for the tempo-agnostic pattern recognition akin to Dynamic Time Warping (DTW) [15], except applied in an arguably more computationally efficient format essentially as real-time as conceivably possible. This has allowed for the signal from all twelve performers to be analyzed and run through twelve parallel iterations of a pattern recognition algorithm. Despite running 144 iterations (12 per performer) of the pattern recognition in conjunction with 12 FFT binpass filters and supporting envelope tracking algorithm, the overall system's CPU footprint remained negligible.

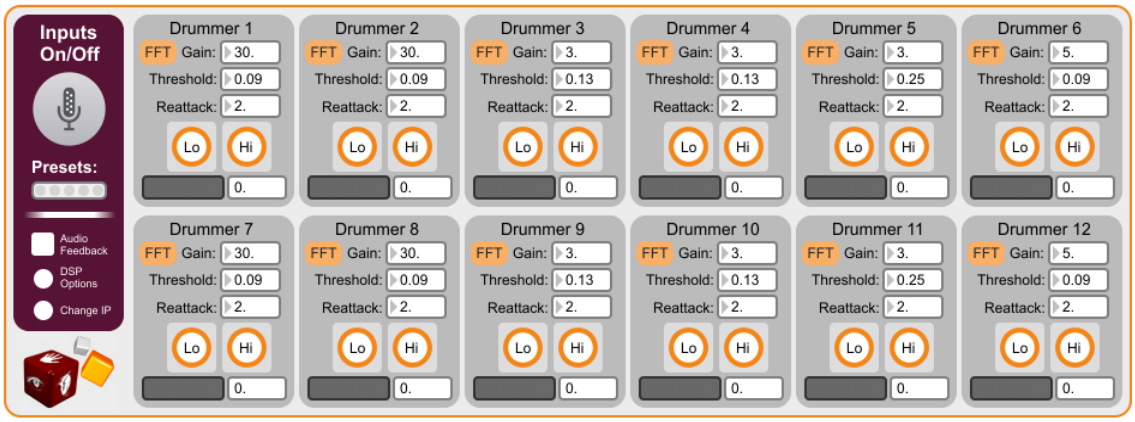


Figure 8: Analog Signal Analysis and Pattern Recognition Component Configuration and Monitoring Graphical User Interface

Two out of twelve patterns were given special treatment. One, with no hits detected, was treated as suggesting holding still. It was automatically invoked after two seconds of performer inactivity. The second, a drum roll, had a unique effect on the analog signal analysis that was not capable of accurately capturing every individual event due to residual envelope, despite considerable attenuation that was achieved through aforesaid physical and software filtering techniques. This event was therefore treated as a special case that was detected as a pattern of an unusually high tempo and whose contextual role was either of stillness (soft short roll; pattern 9), or attack (loud continuous roll; pattern 6), with their roles being mapped to satisfy desirable musical characteristics within the context of the overall performance.

3.1.4. Latency in pattern recognition

It is worth noting that the ensuing pattern analysis did inherently bear a minimum latency equivalent to the amount of time required for a performer to perform the entire pattern. One of the implemented optimizations to the pattern detection was pattern elimination–the pattern recorded from the performer's input was reset as soon as it could not match the reference pattern in any of its permutations, upon which the pattern recording was restarted, starting with the onset of the following event. The research team also noted the possibility of further optimizing the incoming stream by identifying a unique partially matching pattern through the aforesaid approach to pattern elimination. However, given such an approach could also potentially accept partially incorrect patterns and therefore promote inaccurate performance, it was dismissed as a viable option. Besides, the ensuing latency within the scope of this project seemed perfectly reasonable–on the battlefield, cohorts would need to hear the entire pattern at least once before they could successfully execute formation and orientation adjustments to stay in sync with each other.

3.1.5. Output

When coupled with analog signal analysis, the pattern recognition engine generated a numeric value pair, one reflecting the detected pattern, and another reflecting the detected pattern's observed tempo expressed in beats per minute (BPM). The latter was used to adjust the speed at which the cohort was executing a particular move, thus promoting showmanship of one's technical prowess as a performer by which they could also more efficiently direct their cohort across the battlefield. The aforesaid value pairs were fed over a network socket to the procedural audio engine for further processing. The ensuing analog signal analysis and pattern recognition component was coupled by a graphical user interface (Fig.8) to allow for easy configuration of the twelve inputs, as well as provide basic troubleshooting tools, like audible hit detection and visual monitoring of FFT signals (Fig.6).

3.2. Networked procedural audio engine

The networked procedural audio engine was designed to receive networked data from both the analog signal analysis and pattern recognition component (the desired pattern and its tempo), and the gaming engine (the overall status of the battle and the two teams, overall game status–most notably play, pause, beginning, and ending, and finally general spatialized sound effects and cues) (Fig.5). The two data streams were combined together to generate a holistic image of the simulation status in order to select the most appropriate clips for the two soundtracks, one for each team, that reflected primarily their respective army's overall status. A hierarchy of possible states was established with certain states taking precedence over others (Fig. 9) to allow for a single overall soundtrack to reflect the state of the entire team (six cohorts) and their potentially divergent orders. For instance, neutral states (e.g. a cohort standing still) were given a value of 1. As soon as at least one cohort of the army began moving, the movement state (2) would take precedence. Similarly, if any of the cohorts engaged in a battle, a battle state (3) would take precedence, and finally if any of the cohorts retreated, the state was computed based on the dominant number of battling versus retreating armies to be either battle (3) or retreat (4). This choice allowed for strategic regrouping that should not be necessarily regarded as a negative turn of events but rather as a momentary realignment in the overall course of the battle. This state collection was replicated twice for a total of three state sets (Fig. 9). One was a negative set where an army may engage in any of the four states with the overall situation leaning towards defeat. The second state set focused on the positive four states where the overall situation suggested

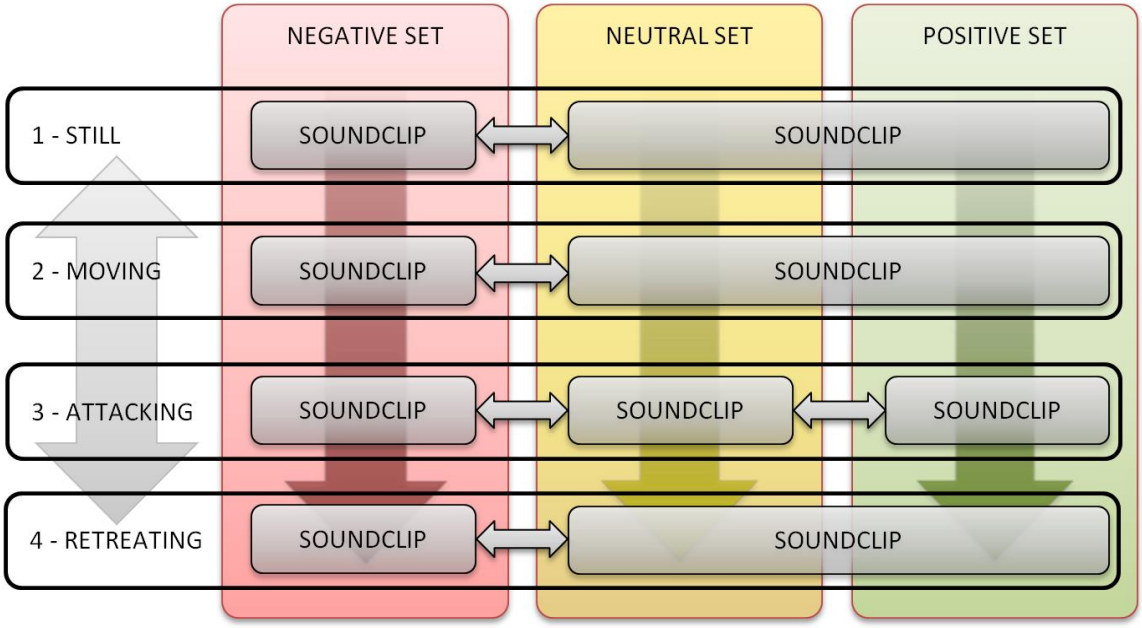


Figure 9: The hierarchy of states and state sets and a transition diagram.

positive outcome. This set was also used for neutral, neither negative nor positive state for still, moving, and retreating, whereas the third incomplete set came with a dedicated battle state depicting neither positive nor negative outcome. This was done to ensure additional granularity of the battle state whereby performers would be able to hear subtle yet potentially critical changes in the overall situation on the battlefield. Additional placeholders were provided for monitoring of the audience's participation that was to be fed into the gaming engine where overall morale computation took place. In return, the gaming engine provided the ending value in a form of a single integer depicting a team's morale as positive, neutral, or negative.

3.2.1. Soundtrack design

Each of the nine possible states was assigned a loopable precomposed track in duration between 32 and 96 seconds. Due to time and budgetary constraints, loops were composed and produced using a collection of commercial samplers and a selection of thematically related sounds, resulting in a style that combines western orchestral music with traditional Chinese instruments. In addition, each state was accompanied with an arbitrary number of transitions, each marked with its state of origin and destination and variant (e.g. a transition that goes from retreat to still in a negative set would be labeled 4-1[0], with [0] referring to the first possible variant of that transition)

3.2.2. Procedural transitions

Arguably, the beat-synchronized transitions and modular treatment of individual loops through the use of a series of metafiles offers one of the more advanced approaches to procedural soundtrack generation. Below we present the most important aspects of this implementation. To ensure a seamless continuous soundtrack that retains structural integrity across transitions, all loops were coupled with an associated coll-formatted cue metafile that reflects possible entry points to which a soundtrack could transition into following a playback of an appropriate transition clip, ensuring that no two entrances are the same, and thus further adding to a possible variance. For this reason the procedural audio engine keeps history of previous states against which it can then compare potential transitions into the next desired state. At the point of transition, the engine randomly picks from a pool of auto-detected transition audio files that have an associated metafile reflecting their compatible point of origin and destination, as well as their duration. In order to ensure that each transition takes place at a desirable moment (e.g. a downbeat), transitions are delayed until they coincide with one of the possible transitioning points, as reflected by a supporting tempo and meter metafile. The downbeat is audio signal driven and, at the time the engine was designed, limited to the signal vector size (default 64 bytes at 48,000Hz sampling rate, or a maximum delay of 1.33ms) inside Max/MSP. With the recent introduction of Gen for Max/MSP [16], this accuracy can be theoretically brought down to a single sample regardless of sampling rate. As a result, all transitions are triggered with signal vector size accuracy. In our tests, even for a steady beat-pattern-driven soundtrack, 64 bytes proved adequate and below the perceivable threshold even of a musically trained listener.

Apart from the cue file, an additional metafile associated with each loopable sound clip contained tempo and meter information, allowing for smooth transitions between their respective settings, as well as accounting for changes in terms of downbeat positions. Although a gradual speed-up and/or on-the-fly playback speed change both with and without resampling was one of the features desired by the project and the current implementation has the necessary framework to support it, the final implementation, in part due to time constraints and in part because it was deemed unnecessary by the Drummer Game simulation, did not include this option.

Between the virtually unlimited variance in terms of possible transitions, the number of loops with equally

unlimited number of valid transition entry/exit points, as well as loop granularity and length, the ensuing system offers a malleable setup for more advanced procedural manipulation of precomposed musical snippets, their layering, and further advanced processing (e.g. glissando and/or a key change).

3.2.3. Soundtrack latency

As a result of the aforesaid architectural choices, the worst case scenario delay of a transition is the full bar. While this certainly is not ideal and does not match responsiveness of a precomposed and post-produced movie soundtrack, in our real-world tests we found the music to be adequately responsive to the changing conditions. More so, this arguably slower response also allowed for momentary misinterpretations of a particular command to be dropped before requiring rapid changes between two potentially dramatically different states that could compromise the soundtrack's overall structure and musicality. Therefore, the ensuing pace can more closely reflect that of a narrative-driven pre-composed soundtrack.

4. REAL-WORLD TESTING

Although the final simulation did not test all the features provided in the audio engine, tests using drums and the audio engine alone offered for an exhilarating experience where performer's accuracy was rewarded accordingly by a responsive system. There are lingering questions, however, particularly within the context of the Drummer Game project.

One of the questions is centered on generating convergence of musical ideas between the soundtrack and live performers, something that proved particularly challenging within the context of Drummer Game's arguably unique gaming paradigm where performance is subservient to the simulation, rather than the soundtrack. We anticipate that with an extended use of phase vocoding to provide time stretching as well as pitch shifting of the soundtrack material, this could prove an interesting symbiotic experiment between a live performer and the presented audio engine, although such an approach would require a different treatment of the compositional material and as such within the context of the Drummer Game project it proved undesirable. More so, with potentially six disjunct tempi and rhythmic patterns, establishing common pulse is essentially impossible.

Another concern is how the overall aural fabric would sound, even within the context of a single army where six performers play varying patterns at likely disparate tempi. While one conclusion would be that of cacophony, another suggests a more immersive experience of what such drums may sound on a battlefield, while also offering those unsuspecting moments of perfect sync in part because all cohorts may want to advance towards the enemy at the same pace, but also possibly for that dramatic effect that instills fear in their enemies (and at the same time excites the audience observing the spectacle and cheering for their team).

4.1. Applicability in other scenarios and future work

Aegis offers a great deal of underexplored features–loops with odd meters, micro- and macro-loops, pitch shifting and other advanced techniques, options for immersive spatialization, and synchronized integration of performer and soundtrack parts (e.g. in performance contexts). With these in mind we very much look forward to building the next iteration of the engine with many of these features integrated as a turnkey solution.

4.2. Obtaining Aegis

For all enquiries regarding Aegis, contact Ivica Ico Bukvic <ico@vt.edu>.

5. CONCLUSION

Unlike traditional gaming audio solutions, the first iteration of Aegis consists of two components: the analog signal analysis and pattern recognition module, and the networked audio engine. Its audio engine design builds upon the foundations of procedural audio, software platforms such as FMOD and WWise, and audio-driven interaction, while also pushing the boundaries of a sample- and pulse-accurate procedural soundtrack and its implementation so as to allow for greatest malleability without sacrificing structural integrity. In its initial tests, the engine showed negligible CPU overhead and as such the research team posits it is ready for deployment in a broad range of scenarios. Despite its project-specific design, the engine has broad applicability potential, including games, immersive simulations, digital storytelling, as well as the less conventional performance- and installation-based scenarios.

6. REFERENCES

[1] Fortner, Stephen, "Interview with Tom Holkenborg," *Keyboard*, pp. 13–20, Mar-2012.

[2] A. Farnell, "An introduction to procedural audio and its application in computer games," in *Audio Mostly Conference*, 2007, pp. 1–31.

[3] "FMOD." [Online]. Available: http://www.fmod.org/. [Accessed: 28-Feb-2015].

[4] "Audiokinetic | Wwise." [Online]. Available: https://www.audiokinetic.com/products/wwise/. [Accessed: 28-Feb-2015].

[5] "What is Max? « Cycling 74." [Online]. Available: http://cycling74.com/whatismax/. [Accessed: 08-Feb-2012].

[6] G. W. Coleman, C. Hand, C. Macaulay, and A. F. Newell, "Approaches to auditory interface design-lessons from computer games," 2005.

[7] M. Liljedahl, N. Papworth, and S. Lindberg, "Beowulf: A game experience built on sound effects," 2007.

[8] N. Rober and M. Masuch, "Leaving the screen New perspectives in audio-only gaming," 2005.

[9] T. Westin, "Game accessibility case study: Terraformers–a real-time 3D graphic game," in *Proceedings of the 5th International Conference on Disability, Virtual Reality and Associated Technologies, ICDVRAT*, 2004.

[10] D. Avissar, C. N. Leider, C. Bennett, and R. Gailey, "An Audio Game App Using Interactive Movement

Sonification for Targeted Posture Control," in *Audio Engineering Society Convention 135*, 2013.

[11] "EAGER: Drummer Game: A Massive-Interactive Socially-Enabled Strategy Game | Computer Science at Virginia Tech." [Online]. Available: http://www.cs.vt.edu/node/4853. [Accessed: 28-Feb-2015].

[12] S. Wei, Q. Ma, and M. Schreiner, "Scientific investigation of the paint and adhesive materials used in the Western Han dynasty polychromy terracotta army, Qingzhou, China," *J. Archaeol. Sci.*, vol. 39, no. 5, pp. 1628–1633, 2012.

[13] "76dB Piezo Buzzer : Piezo Buzzers | RadioShack.com." [Online]. Available: http://www.radioshack.com/76db-piezo-buzzer/2730059.html#.VPIfJfnF98E. [Accessed: 28-Feb-2015].

[14] "Noise gate," *Wikipedia, the free encyclopedia*. 14-Sep-2014.

[15] *Information Retrieval for Music and Motion*. .

[16] "Category:Max And Gen - Cycling '74 Wiki." [Online]. Available: https://cycling74.com/wiki/index.php?title=Category:Max_And_Gen. [Accessed: 28-Feb-2015].

[12] S. Wei, Q. Ma, and M. Schreiner, "Scientific investigation of the paint and adhesive materials used in the Western Han dynasty polychromy terracotta army, Qingzhou, China," *J. Archaeol. Sci.*, vol. 39, no. 5, pp. 1628–1633, 2012.

[13] "76dB Piezo Buzzer : Piezo Buzzers | RadioShack.com." [Online]. Available: http://www.radioshack.com/76db-piezo-buzzer/2730059.html#.VPIfJfnF98E. [Accessed: 28-Feb-2015].

[14] "Noise gate," *Wikipedia, the free encyclopedia*. 14-Sep-2014.

[15] *Information Retrieval for Music and Motion*. .

[16] "Category:Max And Gen - Cycling '74 Wiki." [Online]. Available: https://cycling74.com/wiki/index.php?title=Category:Max_And_Gen. [Accessed: 28-Feb-2015].