Solving adaptive game music transitions from a composer centred perspective

Samuel Gillespie and **Oliver Bown**
Faculty of Art and Design, University of New South Wales, Sydney, Australia
samuel.gillespie@student.unsw.edu.au
o.bown@unsw.edu.au

Abstract

Transitioning from one section of music to another is a core problem for game music. As such, developing approaches to game music composition which can systematically execute a transition would greatly simplify the complexity of a composer's work. To answer this question we have undertaken the composition and analysis of a simple score, looking at the compositional techniques utilised in points of transition. This practice-led process is then used to inform the development of a prototype system for the real-time generation of music to accompany a game. The output of the generative system is then compared to the human composed music to further understand key features of musical transitions applicable to real-time systems. From this comparison, key features, such as tonality and contour, are highlighted. The techniques used to manipulate these features are explained in the course of discussing the generative system. Through this research a greater understanding is developed of enacting musical transitions in a generative system for game music.

Introduction

The topic of music in video games has received significant interest and is already a fitting field for the application of generative techniques (See Collins 2009). However multiple writers raise the issue of usability of generative systems by the composer of a game's soundtrack. Examples of challenges include questions of conceptual understanding as composers work with a non-linear series of events (Collins 2009) and the need to make effective use of a composer's time (Scott 2014). Professional composers working in the industry also see a need for generative tools which are usable by composers (Weir 2015).

Additional challenges are also presented by the requirements of game developers, the primary being that a system is consistent and secondly that such a process is manageable as part of a game engine (Weir 2015, Collins 2009, Farnell 2007). Of these two significant concerns this paper will be focusing on system consistency and will not directly address performance.

This paper investigates important properties of well crafted state transitions in video game sound tracks encapsulated in an algorithmic approach for performing transitions in real-time. To identify important properties a practice-led process was employed, conducted by the first author, of writing music for video captures of game-play followed by reflections and analysis on the composed works.

One key feature identified in this process was a change of pitch structure. To facilitate this process a systematic representation of intervalic spaces was developed. The representational model draws heavily from contemporary discourses in music theory in the vein of Lewin's *Generalised Musical Intervals and Transformations* (Lewin 1987). In terms of representing musical structure, Tymoczko's development of Lewin's framework into a geometric model (Tymoczko 2011) will form the foundation of our representations. Combined with this representation system a novel application of rule constrained, probabilistic, counterpoint generation is used to direct phrases and control chord choice.

The practice of composing scores for recorded footage of games provides an opportunity to observe how a human composer might approach problems with perfect foresight of a player's actions. While this is somewhat removed from the context in which a video game's sound track exists, it provides an opportunity to examine effective treatment in a near perfect knowledge setting. In this paper the transition between states of combat and non-combat will be the primary focus.

Through applying a robust musical framework to the task of creating generative music systems for games, this paper aims to address the conceptual understanding of a composer as a first step towards developing usable systems for composers and consistent systems of generation which may be well understood due to a systematic understanding of the musical space in which they operate. To demonstrate this a prototype system has been developed and will be used to provide examples supporting the concepts outlined in this paper.

Existing techniques

High-quality audio has been the expected norm in video games for well over a decade now (Collins 2009). Despite common audio formats such as WAV and mp3 only supporting sequential playback from sample to sample, many game

scores exhibit branching, layering and transitions which allow them to adapt to the current game state (Collins 2009). Indeed the challenge of maintaining high quality audio while pursuing adaptability has posed an immense challenge for video games and audio designers (see Liebe 2013 and Farnell 2007). Evidence to this can be seen in the prevalent use of easy to implement techniques, such as cross-fading (for examples consider *Skyrim*, Bethesda Softworks 2011 and *Rebel Galaxy*, Double Damage Games 2015, among others). Prechtl argues that these cross-fades are clumsy and aesthetically displeasing and has prototyped and examined a dynamic transition system. His system performs linear interpolation between two Markov models to a continuous probability space between states (Prechtl 2016). Each state has a parameter matrix which alters the chord selection matrix and synthesis parameters. As such the space between two states is a linear space consisting of a matrix of values between the two states. Within the limits of his system, we can see an example of more involved a solution for creating more fluid and dynamic cross-fades. However, as discussed later, from exploring the creation of exemplar scores, some important elements of an effective musical transition require more clearly enunciated differences. As such there are deficiencies in Prechtl's system for realising structural functions of game music. From this we can primarily understand Prechtl's system as contributing primarily towards concepts of player interaction (see Richard Stevens and McDermott, 2015, for more on this area).

Contemporary audio engines do have alternative approaches for managing transitions, for example consider Wwise's transition segment system (Audio Kinetic 2016). A composer can layer tracks and transitions to create a convincing transition between two different cues. Fmod, another popular game audio middle-ware, outlines a similar system for managing branching and looping music (Firelight Technologies 2016). The key difference between Prechtl's approach and what appears to be the predominant approach to game music is, organising material by intrinsic properties material, and a style of scripted transitions, respectively. In Prechtl's system the next chord depends upon the transition of the current chord selected via the probability matrix of the Markov model, which is altered by the game's state (Prechtl 2015). In contrast the Wwise approach requires the composer to directly manage the content of the transition (Audio Kinetic 2016) which takes both time and consideration. Additionally as the complexity of the score increases such a hard coded system of transitions becomes rapidly more cumbersome to maintain; An unfortunate sign of an approach which scales poorly for use in a dynamic system, due to the rapidly increasing number of situations for which a transition must be specified. The limitations imposed through organising a score through pre-determined transitions appears laborious and confining when compared to the promise of systems utilising intrinsic properties of musical material. However to their merit, they provide a system where a composer maintains substantial control over how transitions occur, musically, in a game. As such, developing a stronger understanding of which features are important to composers when creating a transition is for the future of generative music in games.

Challenges of complexity

In order to illustrate the issues of increasing complexity outlined above, we suggest a small example game and consider how the soundtrack may be approached. For our example let us consider a game where a player must solve puzzles and defeat foes to traverse a maze. Let us assume that the dungeon has been crafted with increasing difficulty as players approach a final challenge. Each challenge is in a unique room with its own characteristic aesthetic within the aesthetic of the overarching game. Taking apart our initial premise the game director can see that they need content for at least puzzle and combat sections with an increase of dramatic tension as the player progresses. The first unknown quantity with which we must deal is how long the player may spend in each room of the dungeon, either in the challenge or perhaps exploring. While we have a relatively clear guide of the overall dynamic structure, the fact that we do not know how long a player will be present makes the structure of each room's music problematic to plan. Traditionally this problem is solved through various looping and layering techniques or by playing for a fixed duration followed by no music (Collins 2009).

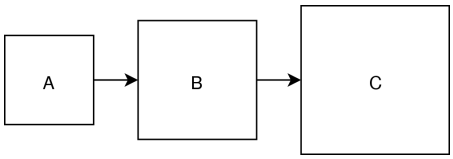

Figure 1: Illustration of linear progression.

Through utilising generative algorithms in scoring rooms we can build far more robust soundtracks (see Scirea et al. 2016 and Houge 2012). Another question that is unanswered by the above brief is how a player may traverse from one room to another. If the rooms are sequential in arrangement we could utilise predetermined transitions from one cue to the next (see Figure 1).

However if we imagine that each room connected to two or more other rooms, as required for our maze, the amount of transitions rapidly increases (see Figure 2).

This may be further complicated if the game generates levels such that each time the game is played it would have different sequences of connections between rooms (For more on procedural content generation see Liapis, Yannakakis, and Togelius 2014).

As shown through this small problem we can see that as unknown variables increase, so too does the complexity of dealing with them when relying on a more traditional, linear conception, of music composition. The complexity then either leads to a heavy workload for a composer or as is more common, the use of naive techniques such as crossfading between sections. Given the growing demands on an end user to maintain such a system of transitions, it seems feasible to adopt alternative approaches which scale more favourably as the number of unknown variables increases. The solutions

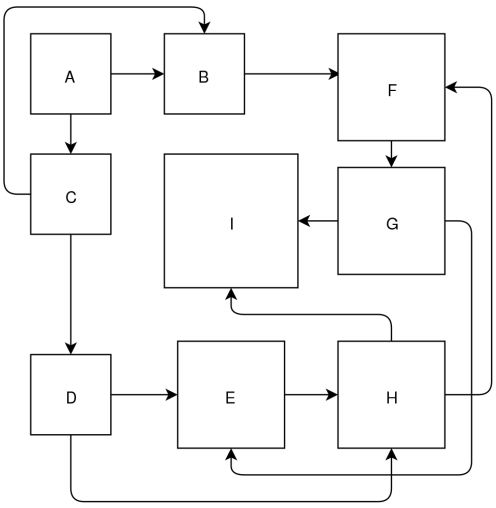

Figure 2: Illustration of maze progression.

presented by both Wwise and Fmod would fall within the less favourable category.

From the above example we can begin to infer that issues related to musical form, at its various levels, will be relevant to developing systems for the generation of video-game music. The dynamic nature of the timing of a player's interactions with a game make this a challenging task, as the form may need to change in relation to the players actions or in-action. In the following section we will being to unpack some techniques a composer might use to transition from music for one state to another and the forms used to do so.

Observations from human composed transitions

In order to better understand approaches a human composer might use to transition between non-combat and combat game states, we undertook the task of composing a brief score for a video capture of a game-play sequence. The video game used was *Rebel Galaxy* (Double Damage Games 2015) and the game-play was captured expressly for this research. To allow for the integration of a new score the in-game music was muted via the games settings, leaving sound effects and dialogue in the audio mix recorded with the in game footage. From this practice lead inquiry we concluded four effective techniques which where used in our created examples to delineate between combat and non-combat sections.

As heard in the first example (`https://drive.google.com/file/d/0BykcGYFSAweyVEVfdnRreFdTMWM/view?usp=sharing`), showing a transition from non-combat to combat states, there is a change in instrumentation, tonal structure, tonal centre and tempo. The instrumentation transitions from a softer synthesizer pad and arco cello to heavier synthesised bass and marcato violins. The drum rhythm also changes when in a combat state, and carries the accelerando from 80bpm up to 144bpm. Tonality is used in two ways to contrast the two sections. The tonal centre moves from an F up to a C which emphasises the lydian tonality of the non-combat section against the predominantly aeolian character of the combat scoring. Additionally we can consider the change in gestures from the smooth flowing character to a sharper more accented character. In many ways this change in character is related to the change in tempo as it supports the change in pace.

In the second example (`https://drive.google.com/file/d/0BykcGYFSAweya3dLMFVVZnNsMlE/view?usp=sharing`) we notice similar techniques used to transition out of combat, back to a non-combat state. This time we hear the drums drop out completely and while the bass holds a drone for a few bars we hear a bridging passage from the cellos again. The piano is then joined by a choir in a more amorphous tonal setting favouring E as a tonal centre. This use of tonality as a foundation of form is rather Classical in nature, resonating with similar motivations which drove its stylistic development (see Rosen 2005, p. 43-53).

While various other techniques do exist for communicating structure, the four main differences outlined provide a set of parameters to change when approaching a real-time generation context. As such, instrumentation, tonality (transposition and structure) and pace will be the varied parameters of our prototype real-time generative system. The key challenges we then expect are questions such as how to maintain direction across points of change and when to apply specific changes to maintain a logic musical structure.

Prototype system design

The design of our prototype real-time scoring system utilises the same captured footage from *Rebel Galaxy* (Double Damage Games 2015) as our human composed scores. As shown in figure 3 annotations have been created for the captured game-play which specify at which frame an event, such as entering combat, occurs in the game. The timing of these events may not be identical to the timing of events in the game however they are as close an approximation as can be derived without access to the game internals.

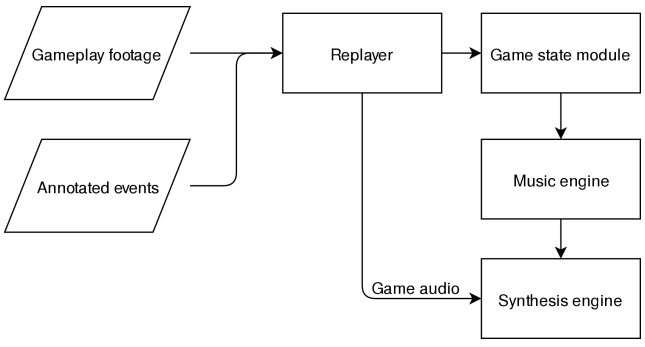

Figure 3: System design used to simulate a game environment.

The current implementation reacts to two game events, combat started and combated ended. These events are con-

trolled by the replayer application which sends messages at specified frames, according to the data file associated with the video replay. From these two messages we can derive the prototypes binary state, in combat and out of combat. Upon receiving a message a set of actions are triggered such as changing the pitch space, metric space, chord set and audio parameter adjustment, see the section below for an explanation of these terms. The changes result in the next phrase segment being realised in the new space, changing the tempo and tonality (see Figure 4). Instruments and gestures are played based on the current state when the phrase segment to which they belong starts playing. The original game audio, recorded without music, is mixed back into the final audio output, so as to most closely simulate the real game environment.

Phrases calculate their next segment, according to the current curve and pitch space while the current segment is being played. This includes selecting a chord, however currently instrumentation and gestures are selected when the phrase segment is triggered by the phrase pulse metronome. In this manner only the bare minimum is defined ahead of time, leaving the system with a great level of flexibility to accommodate the change of environment.

Representational model

Tymoczko suggests utilising geometric representations to model and constrain numeric relationships, such as intervals, in a manner more in line with our intuitive understanding of musical operations (Tymoczko 2009). The key concepts important for our work are vector representation of pitch structures, such as chords or melodies, and mapping into metricised spaces, as a representation of scales. In line with Tymoczko's representations we will use pitch vectors and interval vectors, metricised spaces will be expressed as an expansion of a vector's intervalic pattern.

Pitch vectors are defined as a finite collection of pitches, more generally it is a representation of fixed points on a rational number line. Interval vectors are a finite collection of intervals, as such they are relative to their surrounding pitch content. As will be illustrated, such structures exist within a geometric context which begins to establish their relevance to the music they represent. These abstractions help to accommodate generalised descriptions into the current context of the music generation, such as a gesture to a scale.

Scales

Many musicians would be familiar with the interval vector of the western major scale, $\vec{i} = [2, 2, 1, 2, 2, 2, 1]$ (often rendered as T, T, s, T, T, T, s). Starting at 0 the pitch vector of the major scale is thus $\vec{p} = [0, 2, 4, 5, 7, 9, 11, 12]$. The intervalic pattern laid out in the vector is expanded to become an infinite metricised space. Accordingly for the metricised space of $\vec{p}$, we have the ordered set M where, $M_0 = 0$, $M_1 = 2$, $M_2 = 4$, $M_3 = 5$ and so on, both ascending and descending as per our major scale intervalic pattern. This vector format is used for defining many elements of the generative system including scales, chords and interval based rhythms.

Chords

The concept of mapping structures into other structures can be applied to chords. Consider the pitch vectors $[0, 2, 4], [-2, 0, 3], [-3, -1, 1, 3], [0, 2, 4]$ (I, IV^6, V^7, I). The pitch vector representation above is unusual compared to more common musical set theory notation (see Forte 1973, p. 4-6 for an explanation of common practice) for two primary reasons. Firstly we have preserved the voice leadings instead of immediately utilising pitch classes and secondly because we have recorded diatonic scales degrees, not he standard chromatic values. However as we have alluded to above, we can map these pitch vectors into a scale structure such as a major or minor scale. The result of mapping the above structure into a harmonic minor scale, built on a twelve tone equal tempered system and preserving voice leadings, would then render as: $[0, 3, 7], [-4, 0, 5], [-5, -1, 2, 5], [0, 3, 7]$. Through relying on such a system of mappings we can express pitch structures in a diatonic context, yet understand their broader contexts, such as chromatic, without having to limit ourselves to a specific representational context.

While scales commonly repeat at the sum of their interval vector, chord intervals often do not sum to their repeating interval. For example consider a major triad interval vector from 0 $\vec{i} = [4, 3]$, the sum of which $\sum \vec{i} = 7$ suggests that if we were to map a vector $\vec{a} = [0, 1, 2, 3, 4]$ into $\vec{i}$ the resulting vector $\vec{a} \rightarrow \vec{i} = \vec{b} = [0, 4, 7, 11, 14]$ which assuming a twelve tone foundation turns our triad into a pentad (see Figure 5). However if we assert that $\vec{i}$ repeats around the interval 12 (the octave in our twelve tone system) $\vec{a} \rightarrow (\vec{i} \updownarrow 12) = \vec{b} = [0, 4, 7, 12, 16]$ our structure repeats around the octave, preserving it's triadic nature (see Figure 5).

Incorporating this information into our representation of a chord allows us to use the chord's definition to reason about how gestures should be mapped to chords, even over multiple octaves.

Phrase planning

Planning is an important aspect of music, and while plans may change, without planning, music begins to lose direction. In the example compositions, the structure was already known to the composer, however our system must operate in a real-time setting. As such, to accommodate the variability of duration in games, we have chosen to adopt an approach to music featuring the planning of phrases by following a projected curve. These curves are segmented and pitches are fitted to them in a first species counterpoint inspired selection algorithm. By approaching phrases in this way we balance long term direction and flexibility.

Contrapuntal voice-leading generation

Considering species counterpoint in a computational creative context we could broadly classify it as a heuristic algorithm, (for a detailed implementation see Schottstaedt 1984). While the generation of counterpoint is of limited interest, the structures generated from the process form a strong scaf-

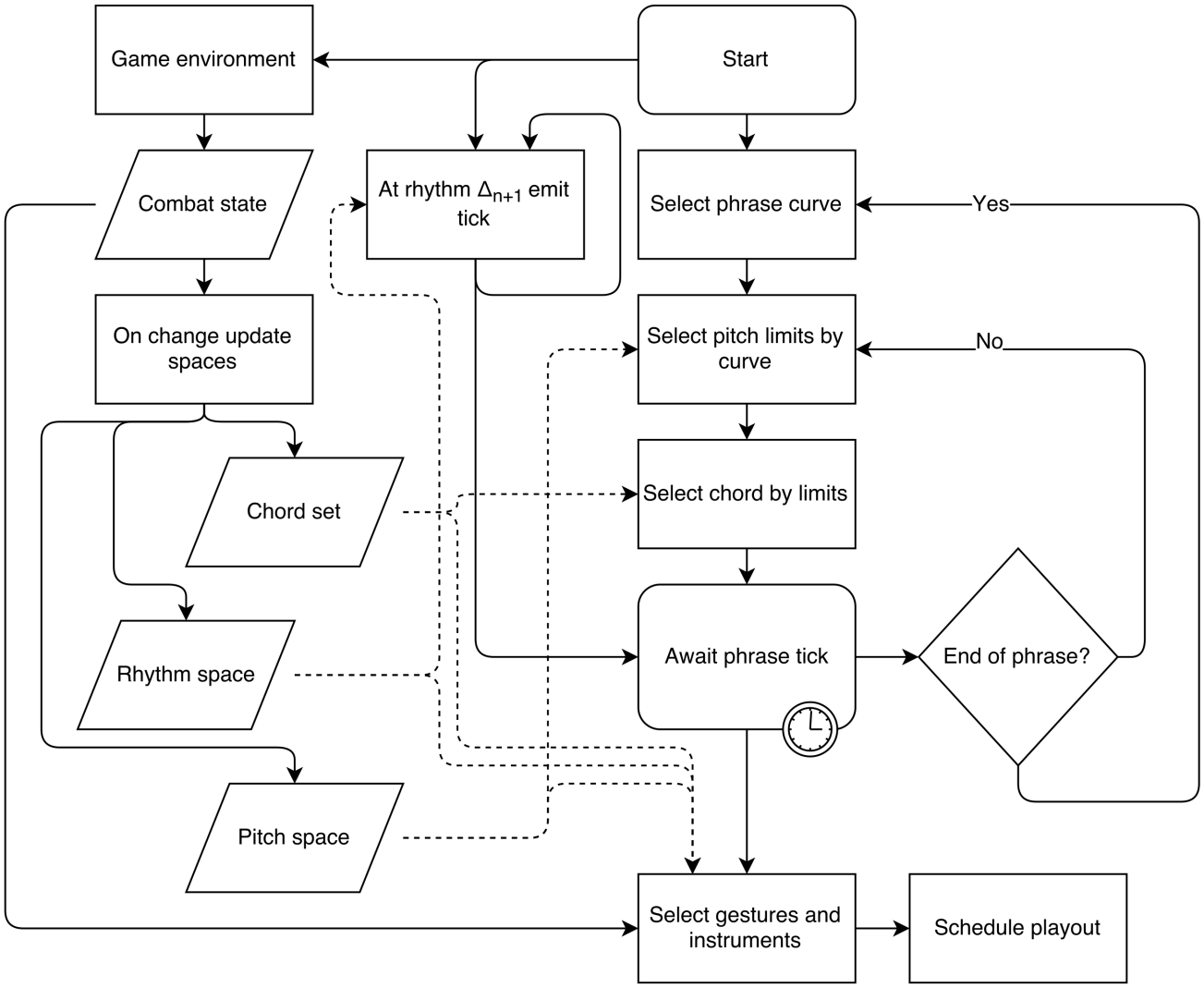

Figure 4: Flow chart of system design.

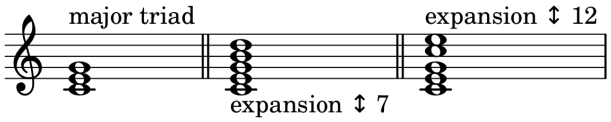

Figure 5: Chord expansion examples.

fold of musical structure. Our implementation is predominantly informed by the rules set out for first species counterpoint, defined as a note against note harmonisation featuring no dissonant intervals (Swindale 1962, p.4-11). One notable departure from species counter point is instead of composing a counterpoint to an existing melody, the *cantus firmus*, we fit our counterpoint to an upper and lower curve. First a pitch is selected for the upper voice. The pitch must sit along the possible steps between the previous note and the direct mapping of the next selected point on the upper voice phrase curve. While leaps are a useful device for adding character to a melody, here we are still creating a higher level structure and as such a smooth step within the bounds of the curve is desirable. For the lower voice smoothness is less of a concern, so a gradually expanding search for a consonant pitch is performed around the area of our last lower voice choice and the next point along the lower curve. Through this process a pitch structure is created adhering to the provided rules of dissonant, consonant, and perfect intervals. From this point chords can be selected which contain the upper and lower pitches, setting up structural tones for the current moment in the phrase.

By fitting notes to curves we can take an abstract structure and realise it as a sequence of voice-leadings. Not only are the contour of bass and treble lines established but through selecting acceptable pitches for each voice, the complexity of selecting chords is greatly reduced. Indeed if one were to be working purely with triads this leaves only two possible

diatonic options for the third note. Building in the generated scaffold structure we can start filling in details by fitting gestures to our structure, rhythm and pitch space.

Scoring

With our systematic framework for intervalic spaces established it is now possible to examine the process of scoring a game's soundtrack. The first step is to describe the spaces which our tonalities and rhythms will populate. Next we can begin creating gestures of melody, rhythm and harmony which are enacted within our sounding space. These gestures could be created through a generative process, however currently they are manually defined.

Spaces

Spaces are structures, expressed as vectors, which define intervalic relationships between numeric values. Examples of what may be defined as a space include scales and pulse, e.g. quaver groupings (see Gauldin 2004, p. 318). The term space is adopted as conceptually elements such as motifs and chord progressions exists within the context of scales and pulse. Furthermore while keys may change within a passage, we can still be operating in the same space and as such it is beneficial to possess a term for a background layer. Following from the above we can additionally relate spaces to the concept of a state in a game environment.

Prototype spaces

In the prototype two spaces are defined which encapsulate the tempo for each space, one for combat, equivalent to a tempo marking of minim = 144 and one for peaceful states equivalent to minim = 54. The pitch space for combat expresses an octatonic scale, an eight note scale, $[0, 2, 3, 5, 6, 8, 9, 11, 12]$ (see Gauldin 2004, p. 739) and peaceful states are set in a pentatonic scale, five notes, $[0, 2, 5, 7, 9, 12]$.

When the prototype receives a change of state, combat to peaceful or peaceful to combat, the pitch and tempo spaces are changed to their corresponding space. Any gestures are then rendered according to the intervalic layout of their new space (see figures 6, 7 and 8).

In addition to two tonal spaces there are corresponding chord sets for each state. The chord sets are mapped into the current pitch space when play-out is scheduled (see Figure 4) as such the chord sets can be applied to any pitch space however not all chords may be desirable in any space.

Through the definition of spaces, chord sets and gestures a score can be generated in real-time, fitted to the voice-leading structure generated from the contrapuntal curve fitting. Following generation the score can be synthesised and mixed into the rest of the game's audio.

Results of prototype system

The results of the prototype system perform well in transitioning from one state to the other. Within the context of the music's function of communicating the change from peace to combat and the reverse, the output of the system clearly delineates these sections. This base level off functionality

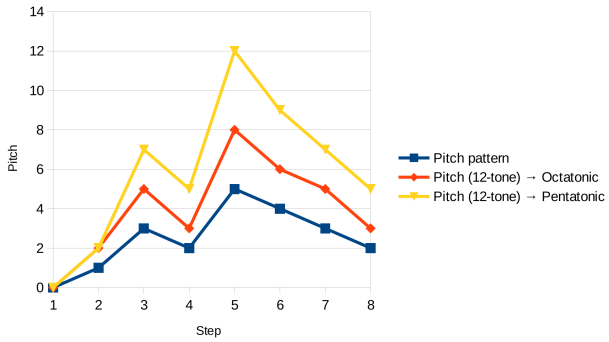

Figure 6: Example output from mapping a pitch pattern (blue) into different scale structures (yellow and red). See Figure 7 and Figure 8 for notation of pentatonic and octatonic mappings.

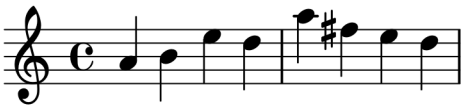

Figure 7: Notated version of pentatonic mapping shown in figure 6.

is achieved through applying the principals gathered from the compositional studies, utilising parameters of tonality, instrumentation and pace to realise this effect.

For example, in the output of the prototype transitioning from non-combat to combat (`https://drive.google.com/file/d/0BykcGYFSAweyWnJ4bHY2S00xbkk/view?usp=sharing`), although perhaps not as refined as the human composed examples (Section Observations from human composed transitions), there is a distinct change in score as the player switches into combat. Unlike in the human composed example, the tempo change is a hard jump as the next phrase segment is reached. However given that we do not have a very active rhythmic texture an accelerando would likely go unnoticed. Instead, in the generated examples, a more prominent use of audio effects is used to build the current phrase segment towards the upcoming change. The change in timbre has the additional benefit of somewhat linking between the change in instrumentation, which for this limited prototype was a predetermined selection. Although there is a hard transition between one pitch space to the next, the transition is relatively smooth as due to the nature of the phrase and chord algorithm, chordal voice leadings progress as smoothly as possible, even across pitch spaces.

Leaving combat, (as shown here: `https://drive.google.com/file/d/0BykcGYFSAweya1NaaWh2dXFtNms/view?usp=sharing`) is rougher than the human composed example, however a sense of consistency is still maintained. In the moments before the combat ends, the phrase begins

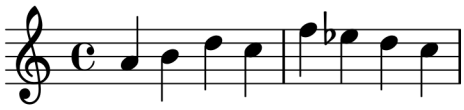

Figure 8: Notated version of octatonic mapping shown in figure 6.

descending down and lands on a final chord played by the synthesized peaceful state timbre. Following this we have a small breath before picking up again in the peaceful state, starting strongly from the last chord into it's new phrase.

All of these musical effects are achievable with current common practice, however as discussed above this would be a case of manually creating these transitions. In contrast here we have a system which through a change of parameters on a given event, generates this behaviour in real-time. This parameter based generation sets this system apart from common practice approaches and with further development could provide an alternative approach for composers and game designers.

Summary

The prototype system developed in this research demonstrates its ability to perform transitions between states in real-time. Built upon a systematic approach to intervalic spaces, states can be defined by these spaces and, as they utilise a common foundation, changing from one space to another can be done smoothly. Furthermore through projecting structural aspects of phrase contours elements of harmony can be selected with a logic as to the direction the music is required to take. The combination of these two techniques provides a strong foundation upon which the prototype system is built.

There is still much work to be done in regards to important questions such as how composers might interact with the system, such as developing a capacity for the system to learn features, for example phrase contours, from examples provided by a composer. Given the use of conceptual models, such as spaces, chord set and gestures, this system does move towards being approachable for composers. Additionally, the problem of consistent results is also partially addressed through this conceptual framework as, although the output varies as different chord possibilities are utilised, they are still from a controllable set and operate in a defined pitch space. As such providing a complex but finite set of possible outputs from the system.

Future avenues of enquiry involve expanding the number of states to explore how music for complex game structures might be made less complex for composers. As mentioned above how composers can use this system in a full game production environment is still an open question. Furthermore, the application of additional generative techniques to the creation and or variation of gestures and phrase structures appears to be a logical extension of the current system, allowing for greater variation and perhaps more highly integrated interaction with the detailed state of a game environment. Finally listener surveys are planned and will be utilised to provide a broader reference of the systems musical qualities and performance in tasks such as transitions.

Although relatively simple, the prototype's ability to perform transitions through the manipulation of tonality, pace and instrumentation addresses some notable challenges in the real-time generation of music in video-games. It provides a promising foundation for further work on improving the expressive capacity of composers in the art of video game soundtracks.

Acknowledgements

This research was carried out with support from the Australian government in the form of an Australian Government Research Training Program Scholarship.

References

[2016] Audio Kinetic. 2016. Wwise help 2016.2.2.6022: working with transitions. *https://www.audiokinetic.com/library/edge/?source=Help&id=working_with_transitions*, accessed 18/03/2017.

[2011] Bethesda Softworks. 2011. The elder scrolls v: Skyrim.

[2009] Collins, K. 2009. *An introduction to procedural music in video games*. Contemporary Music Review, 28(1), pp.5-15.

[2015] Double Damage Games. 2015. Rebel galaxy.

[2007] Farnell, A. 2007. An introduction to procedural audio and its application in computer games. *Audio mostly conference* 1–31.

[2016] Firelight Technologies. 2016. Adaptive music in fmod studio: Transition markers and logic. *http://www.fmod.org/adaptive-music-fmod-studio-transition-markers-logic*, accessed 18/03/2017.

[1973] Forte, A. 1973. *The structure of atonal music*, volume 304. Yale University Press.

[2004] Gauldin, R. 2004. *Harmonic practice in tonal music*. 2nd Edn, WW Norton.

[2012] Houge, B. 2012. Cell-based music organization in tom clancy's endwar. *Demo at the AIIDE 2012 Workshop on Musical Metacreation*.

[1987] Lewin, D. 1987. *Generalized Musical Intervals and Transformations*. Oxford University Press.

[2014] Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2014. Computational game creativity. In *Fifth International Conference on Computational Creativity*.

[2013] Liebe, M. 2013. *Interactivity and Music in Computer Games*. Springer Fachmedien Wiesbaden. 41–62.

[2015] Prechtl, A. 2015. A musical feature based approach to automatic music generation in computer games. *http://proceduralaudionow.com/anthony-prechtl-a-musical-feature-based-approach-to-automatic-music-generation-in-computer-games/*, accessed 07/05/2016.

[2016] Prechtl, A. 2016. *Adaptive music generation for computer games*. Ph.D. Dissertation, The Open University.

[2015] Richard Stevens, D. R., and McDermott, D. 2015. Extreme ninjas use windows, not doors: Addressing video game fidelity through ludo-narrative music in the stealth genre. In *Audio Engineering Society Conference: 56th International Conference: Audio for Games*. Audio Engineering Society.

[2005] Rosen, C. 2005. *The Classical style: Haydn, Mozart, Beethoven*. New Edn, WW Norton & Company.

[1984] Schottstaedt, B. 1984. *Automatic Species Counterpoint*. CCRMA, Stanford.

[2016] Scirea, M.; Togelius, J.; Eklund, P.; and Risi, S. 2016. Metacompose: A compositional evolutionary music composer. *Evolutionary and Biologically Inspired Music, Sound, Art and Design* 202–217.

[2014] Scott, N. 2014. Music to middleware: The growing challenges of the game music composer. In *2014 Conference on Interactive Entertainment*, 1–3.

[1962] Swindale, O. 1962. *Polyphonic Composition*. Oxford University Press.

[2009] Tymoczko, D. 2009. *Generalizing musical intervals*. Journal of Music Theory, 53(2), pp.227-254.

[2011] Tymoczko, D. 2011. *A geometry of music: harmony and counterpoint in the extended common practice*. Oxford University Press.

[2015] Weir, P. 2015. Never ending music. *http://proceduralaudionow.com/paul-weir-never-ending-music/*, accessed 06/05/2016.