FORM-AWARE, REAL-TIME ADAPTIVE MUSIC GENERATION FOR INTERACTIVE EXPERIENCES

Christodoulos Aspromallis
Department of Computer Science
University College London
c.aspromallis@cs.ucl.ac.uk

Nicolas E. Gold
Department of Computer Science &
UCL Centre for Digital Humanities
University College London
n.gold@ucl.ac.uk

ABSTRACT

Many experiences offered to the public through interactive theatre, theme parks, video games, and virtual environments use music to complement the participants' activity. There is a range of approaches to this, from straightforward playback of 'stings', to looped phrases, to on-the-fly note generation. Within the latter, traditional genres and forms are often not represented, with the music instead being typically loose in form and structure. We present work in progress on a new method for real-time music generation that can preserve traditional musical genres whilst being reactive in form to the activities of participants. The results of simulating participant trajectories and the effect this has on the music generation algorithms are presented, showing that the approach can successfully handle variable length forms whilst remaining substantially within the given musical style.

1. INTRODUCTION

A significant portion of current artistic and entertainment narrative-based experiences are structured episodically and the duration of these episodes (i.e. sections) may be only loosely determined due to semi-improvised content or participant-dependent advancement of their narrative. Such experiences include live theatre containing improvisation [1], theme parks [2, 3], extended theatrical interactive experiences [4], video games, interactive film [5] and virtual environments. We hereafter refer to all these as Extended Performance Experiences (EPE).

Musical accompaniment is well established in EPEs due to its potential for a more convincing and engaging experience on a "cultural, physical, social or historical level" [6]. In addition, music can play a narrative-defining role and, thus, significantly enhance the experience [7].

Computer generation of musical accompaniment for EPEs is necessary for a number of reasons: The physical presence of musicians may be counter-immersive or impractical e.g. where, owing to limited stamina of humans, rotation may be required during a long-running experience. This could lead to musical incoherence due to subjectivity of musical decisions if musical improvisation is involved. In addition, the resources required for rehearsals and performances may not be economically achievable for small-scale projects or independent companies.

It is common that EPE episodes are contrasting in content and aesthetic character and, thus, musical accompaniment needs to reflect this (for example, one might imagine four distinct zones within a single theatrical installation, each with a different cultural flavour and thus musical style). The transitions between episodes are an important aspect of the overall experience since, according to Benford *et al.* [8], transitions are benchmarks in continuous interactive experiences in space and time. Musical transitions (MT) must take place at appropriate narrative boundaries and without breaking musical continuity.

The indeterminate duration of EPE episodes, combined with the necessity of timely and well-placed musical transitions, raises the issue of musical coherence and continuity, i.e. we wish to avoid abrupt MTs or unexpected silences. In order to avoid such abruptness during an EPE transition, various techniques have been developed for real-time generation of music (see section 2).

We can summarise the requirements for a music generation system for EPE as: the need to provide continuous, non-repetitive music, with non-abrupt but distinct musical transitions, with large-scale form awareness and with note/chord level granularity. By large-scale form we mean musical forms that might be traditionally recognised e.g. blues, pop songs, binary, tertiary etc.

This paper addresses the problem of generating well-formed music incorporating timely transitions in the presence of indeterminate and dynamically changing compositional length in real time. We present the Form-Aware Transition Engine (FATE): an approach that combines probabilistic music generation with participant trajectory estimation to permit changes to be made to musical structure in real time within the constraints of a style (represented by a probabilistic grammar).

The rest of the paper presents background (Section 2), the FATE approach (Section 3), case studies (Section 4), results (Section 5) and conclusions (Section 6).

2. BACKGROUND

There has been a substantial amount of work on generative music (see [9] for a survey) and in particular real-time [e.g. 10, 11, 12, 13] and non-real-time [14, 15] generative music in EPE-like settings. The real-time work is of particular relevance here and has resulted in a number of common approaches (see summary in Table 1).

2.1 Pre-composed passages played once

This approach is traditionally taken in theatrical performances, in the principle of television 'stingers' [16], by manually triggering or conducting musical passages of shorter duration than the narrative episode lasts. As a result, this approach does not provide continuous musical accompaniment of the narrative.

2.2 Pre-composed consecutive or looped passages

This approach is common throughout the history of music for video games (Super Mario Series – 1985 onwards, Halo – 2001, Earth Eternal – 2009, among others). Sudden stoppages and starts of music passages in early video games were later replaced by volume crossfading [6]. Despite this, players have characterised such MTs as abrupt and loops as monotonous [6]. Müller and Drieger [10] developed a system for automated, real-time manipulation of pre-existing musical clips in response to narrative action in visual media. The system manages time-placement and concatenation of precomposed music. Hazzard [17] developed an adaptive music soundtrack for a musically augmented walking experience in Yorkshire Sculpture Park. The dynamically managed composition relies on triggering and looping short segments of music or the alignment of longer ones.

The lack of note-level flexibility in precomposed music may lead to abrupt changes in music and general inconsistency between music excerpts thus rendering it unsuitable for the problem being addressed here.

2.3 Dynamic management of pre-composed layers

In this approach, predefined melodies, rhythmic patterns, chord progressions or baselines that are musically compatible are vertically managed, i.e. triggered and stopped. Early instances of this approach include Langston's 'riffology' [18] for the Atari console game *Ballblazer* [19], an engine that linearly connects precomposed melodies based on interval connectivity as well as Whitmore's instrumental layer approach [20] for Microsoft's *Russian Squares* [21]. In the more recent music engine prototype [.talktome] [22] and in the recently released video game *NoMan's Sky* by Hello Games [11] music layers are also algorithmically managed, based on game state changes.

Dynamic management of precomposed layers provides arrangement flexibility and, specifically, the ability to punctuate MTs through the use of different instrument settings between music sections. However, the granularity of predefined music layers may not correspond to the granularity of the narrative and so abruptness is still a risk. This may also be musically restraining.

2.4 Note-level procedural generation of original material

Video games like *Journey* [12], *Spore* (music by Brian Eno) [23] and *Simcell* [24] generate an ambient music floor as well as melodic and rhythmic motifs in response to game states and events. However, the non-metric and ambient character of the music does not generate distinctively contrasting music sections. Even though musical accompaniment in these instances meets the needs of the specific video games, other EPEs may require musical contrast at the level of harmonic context, rhythm and large-scale musical form, which is something that these instances of note-level generation do not provide.

2.5 Note-level morphing

Beyond the aforementioned EPEs, Wooller [25] applied morphing programming techniques to music loops in order to generate MTs between mainstream pieces of popular dance music. This approach sets an initial loop as a starting point and a second as a goal point. Systems developed by Wooller [25] and Brown, Wooller and Thomas [13] can generate MTs in the form of music morphs between loops with music generation decisions at the note-level. The *Morph Table* [13] can control the progress of these morphs in real time using interfaces such as moving cubes on a table. Morphing was achieved by using parametric, probabilistic and evolutionary techniques. However, the system manipulates short music excerpts (i.e. loops) and does not take large-scale form into account. Instances of EPE episode transitions may require music to either develop formally or to simply stop by abiding to specific form rules. For instance, on-demand ending of a blues form should be managed in a way that preserves its character and does not sound imbalanced, as might be the case if it ends at an arbitrary point.

3. FORM-AWARE TRANSITION ENGINE

We present a Form-Aware Transition Engine (FATE) that aims to address the problem of MTs in EPEs.

3.1 Architecture

The FATE design consists of two main modules: a *prediction engine* provided with external stimuli and a *music generation engine*. The *prediction engine* receives data from an EPE environment, such as the location and movement of an EPE participant from a Microsoft Kinect™ sensor. Based on environment data, the prediction engine estimates the remaining time towards the next Narrative Transition Point (NTP): the point at which the current episode of music should finish. In the current implementation of FATE, the prediction engine receives its data from a mouse-controlled 2D surface of the computer screen (Figure 1). This simulates a simple version of an EPE episode, where a participant, represented by the dot (top left corner of Figure 1), is expected to arrive at the goal point (bottom right corner) after some time. Reaching the goal denotes the end of the episode. The prediction engine provides the music engine with data based on the three following elements:

(i) An exponentially weighted moving average (EWMA) of minimum remaining time towards an episode end. To calculate this, an EWMA of the absolute speed (irrespective of direction) of the participant is computed continuously. Based on the participant's distance from the goal

Musical elements / Generation approaches	Continuity	Non-abrupt development	Distinct Transitions	Repetition avoidance	Granular control (note/chord - level generation)	Large-scale form
Precomposed passages-once				X		
Precomposed passages-consecutive/looped	X		X			X
Precomposed layers	X		X	X		X
Note-level procedural generation	X			X	X	
Note-level morphing of loops	X		X		X	
Hypothesised algorithm	**X**	**X**	**X**	**X**	**X**	**X**

Table 1. Summary of real-time music generation approaches for EPE-like settings.

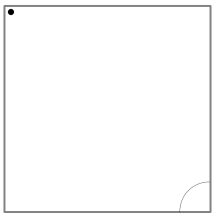

Figure 1. Mouse-controlled 2D surface.
Dot / participant (top-left), goal point (bottom-right).

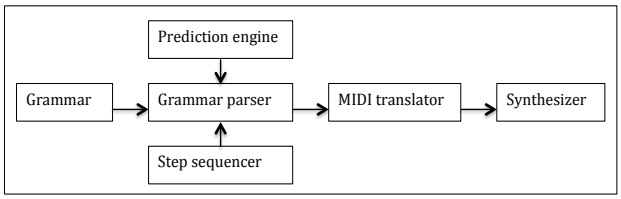

Figure 2. FATE design.

point at a given time, the remaining time is computed under the assumption that a direct linear path towards the goal is taken at that time.

(ii) Binary information on whether the participant is approaching or moving away from the goal point.

(iii) Binary information on whether or not the participant has reached the goal.

The signals sent from the prediction engine to the music engine are configured thus: If (i) is less than a specified duration appropriate to the musical form being generated (e.g. the duration of four bars in the following case studies – see Section 4) and (ii) the participant is approaching the goal, then a signal ("ENDING") is sent to the music engine to request that a process to end the music should begin (*cadencing* - Section 3.2.3). If "ENDING" is true and either i or ii cease to apply, then the "RECOVERY" signal is sent. Finally the "GOAL_REACHED" state becomes true at the end of the episode and is irreversible. The *music engine* generates chords in real time in order to populate (currently) a blues twelve-bar form (the particular musical style required is captured by a probabilistic grammar). Data received from the prediction engine enables the music engine to manipulate the generation of chords so that the musical end matches the episode end in a timely fashion, i.e. reaching the goal.

MTs must happen in a way that makes musical sense in the context of both the finishing and upcoming music sections. At this stage FATE has been developed to tackle the first part of this issue i.e. ending and recovering (more details in Sections 3.2.3 to 3.2.6) a finishing music section in such a way that musical form is retained or at least stopped in as least an abrupt manner as possible. At this stage we use the twelve-bar blues because it is a well-defined form of medium length.

Figure 2 presents the FATE design. Namely, the grammar parser loads a set of structural attributes and grammar rules to the system. A step sequencer drives and continuously updates the system on musical time. Grammar productions happen according to musical time as well as incoming data from the prediction engine. At each step of harmonic progression – in this case defined at one chord per bar – a chord symbol is sent to the MIDI translator which finally outputs the MIDI notes (chords) to a virtual synthesizer.

3.2 Grammar

The generation of chords relies on a hand-built, stochastic, context-sensitive generative grammar. A grammatical approach was chosen as grammars are good for supporting the management of the "macrostructure" of form [26].

Music generation is driven by two main types of decisions: a) musically-driven decisions (3.2.1), defined by the norms of the style (here blues), and b) event driven decisions (3.2.3 to 3.2.6), defined by incoming data from the prediction engine.

3.2.1 Temporal Resolution of Hierarchical Decision-Making (Form-Dependent)

The top-level of chord decision-making is made at specific points of the blues structure. Specifically, *top-level*

dec_1	|	|	|	|
dec_5	|	|	dec_7	|
dec_9	|	|	dec_11	|

Figure 3: Top-level decision points.

```
rule: v7  I_1  I → 0.3  v7  i  I
                 → 0.3  v7  i6  I
                 → 0.4  v7  i7  I
                 :end_rule
```

Figure 4: Context-aware time-specific rule for tonic in bar 1.

```
rule: dec_1 → 0.2  I  I  I  I
            → 0.1  I  I  V  I
            → 0.25 I  IV I  I
            → 0.06 I  IV V  I
            → 0.08 I  IV I  V
            → 0.1  I  I  I  V
            → 0.1  I  V  I  I
            → 0.06 I  V  I  V
            → 0.05 I  V  IV I
              :end_rule
```

Figure 5: LHS is a time-specific (bar 1) top-level decision rule. On the RHS each production probability is defined along with the type-level non-terminals pro-

decisions are made in bars 1, 5, 7, 9 and 11 of the form as shown in Figure 3 (denoted as 'dec_*') for each blues cycle. In bar one a set of four non-terminal, *type-level* chord symbols (see 3.2.2) is probabilistically chosen. It follows that the rest of the top-level 'dec_*' decisions return two non-terminal chord symbols each. In other words, at certain points in the structure the grammar returns a number of productions scheduled as future rewrites or musical events. This approach is similar to the way that Keller and Morrison [27] used probabilistic grammars for the generation of style-abiding jazz melodies. In [27] the grammar controls both rhythmic and melodic structure of melodies and the latter are extended by controlling the length of terminal strings produced (at each point). The points at which top-level choices should be made are determined by the form.

It can be argued that the blues form can be divided into three four-bar sections (for clarity, note that here we take "blues form" to indicate the basic blues form along with its harmonic language (or variations known as rock 'n' roll blues, rock blues or others) and not what is known as jazz blues, *Blues for Alice* form etc.).

For the sake of clarity of harmonic form and form flexibility, in our example further division has been applied (bars 5 to 12). Namely, dec_1, dec_7 and dec_11 can be seen as extended tonics (I), dec_5 as extended subdominant (IV) and dec_9 extended dominant.

Once 'dec_*' non-terminals are rewritten as non-terminal chord symbols, a type-level rule is applied to produce a terminal chord symbol. Rewrites occur in a time-controlled manner. For instance, for the rule shown in Figure 5 the non-terminal 'dec_1' may rewrite as "I IV I I" with probability 0.25 (Figure 5). Next, if the context matches the rule in Figure 4 the tonic (I) in bar 1 will be rewritten as a terminal 'i7' with probability 0.4. Once a terminal has been reached, rewriting stops until the next musical time arrives (the next bar in this case).

3.2.2 Grammar Design

As Section 3.2.1 partly conveys, the grammar can be abstracted as G = (M, T, R, S, P) where we have:

- Start point S → dec_* dec_* dec_* … (form-dependent time-stamped decision points at the beginning of every cycle)
- Non-terminals (non-t) M divided in:
  - Top-level non-t = { dec_*, cad, fin, rec } (see also Sections 3.2.3 to 3.2.5)
  - Type-level non-t = { I, II, III, IV, V }
- Terminals T = { i, i6, i7, iim7, iiim, iiim7, iv, iv6, iv7, v7 }
- Set of rules R
- Set of rewrite probabilities P

Type-level non-t chord symbols function as an intermediate step after top-level non-terminals in order to control the final configuration of each chord according to its position and context (e.g. IV rewrites in either of 'iv', 'iv6' or 'iv7').

For our blues example, the probabilities of production rules are hand-coded based on musical experience, however, we intend that in the future, probabilities will be learnt from style-appropriate corpora. Beyond the above elements, the system accepts a number of form-defining data, i.e. form length, harmonic rhythm (chords per bar), time-placements of top-level decisions in the form ('dec', 'cad'), time signature and an optimal harmonic form (see Section 3.2.5).

Rules R are divided in *timed* (denoted with '_*' in the grammar, Figures 3, 4 & 5) and *general* as well as context-aware or not. Two main divisions of rules R apply to non-terminals:

- *Timed rules* (TR - denoted as '_*' in the grammar) vs. *general rules* (GR).
- and *context-aware* rules vs. *context-free* rules.

As Roads and Wieneke state [26], a grammar described only by rewrite rules is weak for music description and generation of musical structure, unless somehow enhanced. In our case this enhancement relies on the time-specification of certain rules as well as time-based resolution of hierarchical rewrites.

The following sections (3.2.3 to 3.2.6) describe how the music engine responds to the signals received from the prediction engine, i.e. ENDING, RECOVERY and GOAL_REACHED (see Section 3.1).

3.2.3 Cadencing

When 'ENDING' (Section 3.1) becomes true, the music is scheduled to cadence within a number of bars, according to the chosen generated form. In the blues form that has been chosen for the case studies in this paper (see Section 4) two-bar cadences can be placed at every four bars, i.e. bars 3, 7, 11 with the prospect for each cadence to end at bars 5, 9 and 13 (i.e. 1 of the form) respectively (Table 2). This choice was made based on consistency with the 12-bar blues form. In other words, finishing at every two bars or on the even bars of the form would have a syncopated feel, compared to finishing on either of bars 5, 9 or 13.

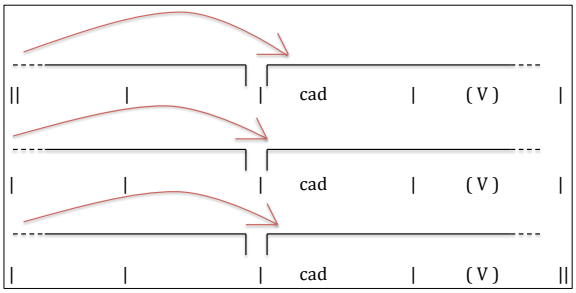

Figure 6. Correspondence of 'ENDING' occurrence with cadence placement.

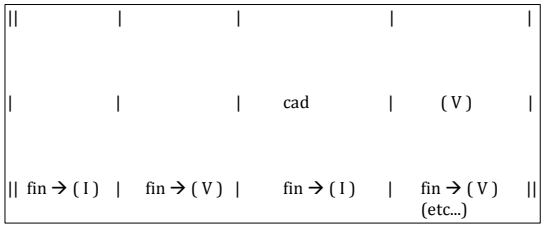

Figure 7. Post-cadencing: 'fin' placed in remaining form.

ENDING occurrence bars	Cadence placement bar	Expected end bar
11 - 2	3	5
3 - 6	7	9
7 - 10	11	13 (1)

Table 2. Potential cadence ('cad') and ending bars.

	Traj. 1	*Traj. 2*	*Traj. 3*
ENDING	21	24, 32	13, 21, 29, 35
RECOVERY	-	31	18, 25, 33
GOAL_REACHED	31	38	41

Table 3: Bars when prediction engine events occur in each trajectory. Bars are numbered linearly irrespective of form.

Depending on when the ENDING signal is received, the cadence is scheduled for the equivalent point in the form as shown in Figure 6 and Table 2. E.g. if the ENDING signal is received within bars three to six, the cadence will be scheduled for bar seven (second row of Table 2). The top-level non-terminal 'cad' probabilistically produces two type-level chords (pre-dominant, dominant) based on the context that precedes the cadence placement bar. It follows that the cadence at bar 11 is placed there despite it being the end of the cycle in order to highlight the ending of the form.

3.2.4 Post-Cadencing

Under the same circumstances, along with scheduling 'cad', the top-level non-terminal 'fin' is placed in every bar after the cadence is finished (Table 2 - column 3, Fig. 7) until the music ends. A micro-grammar of non-terminals 'fin' guarantees that a sequence of V – I chords will be played after the music has cadenced. The purpose of this feature is to prolong the end of music until the goal is reached, by providing some basic harmonic movement that is flexible enough to end within one or two bars after the goal. I.e. once the goal is reached and a cadence has completed, the music stops at the first tonic chord generated.

3.2.5 Recovering

A significant feature of the music engine is its ability to recover from a 'false alarm' from the prediction engine. By that it is meant that the music can be led back into the previously ending form. Unless the goal has been reached, the prediction engine may, theoretically, change from ENDING to RECOVERY (Section 3.1) countless times. Music generation can support this in the following way: Regardless of whether the previously ENDING music has reached its scheduled cadence or not, or even if it is already in a post-cadencing phase, the music engine takes a two-step recovery approach. When a new bar is reached and ENDING has just turned to RECOVERY, an optimum harmonic sequence of the form is applied from one harmonic-rhythm step (i.e. 1 bar) ahead onwards. This aims to re-establish the feel of the form by placing the most "classic" chords at each bar. The optimum harmonic sequence is placed one bar later so that the one gap-bar that connects it with the previous harmonic phase (i.e. pre-cadencing, cadencing or post-cadencing) works as a reconciling chord between the falsely ending harmony and the optimum harmony. In order to preserve non-abrupt harmonic progress, an additional number of timed and general (i.e. non-timed), context-aware rules are defined by the grammar. These rules can support all cases of recovery regardless of the preceding harmonic phase.

The top-level non-terminal for the reconciling gap-bar is 'rec'. As with rule production probabilities, the optimum chord progression has been hand-coded here, but will be statistically modeled in the future.

3.2.6 Stopping

When the goal point is reached and the post-cadencing phase has been entered (even by 1 bar), the grammar productions and music in general stop once a tonic chord is reached, i.e. until a top-level non-terminal 'fin' is rewritten as one of terminals 'i', 'i6' or 'i7'. As mentioned earlier (Section 3.1) reaching the goal point is irreversible, so the 'episode' finishes and recovery is no longer an option.

4. CASE STUDIES

In order to test the effectiveness of the system in generating musical cadences, form recoveries and musical endings on demand, we provided the music engine with data generated by the prediction engine. Computer mouse movement, which served as a proxy for the movement of a virtual participant for our case studies, produced the test trajectories offline. For these case studies of grammar productions, the sequencer was not used but instead a test harness replayed the offline-recorded trajectory data. All three trajectories were performed in a distinctly different way in order to challenge the music engine under different series of events. Trajectory 1 is the smoothest of the three, where the virtual participant approaches the goal point rather directly (Fig. 8) with relatively consistent speed, trajectory 2 significantly diverges from what would be an ideal trajectory towards the goal point with moderate speed variation and, finally, trajectory 3 approaches the goal point but spins in circles before it reaches it, this time with significant speed variation.

Regarding the data received from the prediction engine, the 'ENDING' signal is sent based on a logical conjunction of data levels (i) and (ii) (Section 3.1). Specifically 'ENDING' is sent if (i) the EWMA of minimum remaining time towards an episode end is less than 4 bars' AND if (ii) the participant is approaching the goal point. As explained in Section 3.2.3, 4 bars duration for (i) has been chosen because the blues form ends more naturally in this configuration. Finally, the 'RECOVERY' signal (3.2.5) is sent when 'ENDING' ceases to apply and 'GOAL_REACHED' (3.2.6) occurs as the episode ends.

The recorded prediction engine data were provided to the parser in order to produce the three chord sequences in response to trajectories' events. In order to render audio examples, these chord sequences were translated into MIDI and rendered using Ableton Live[1]. Drums and bass parts were added as pre-programmed MIDI loops, triggered alongside the harmonic parts and – in the case of the bass part – selected based on the grammar-generated harmony. The tempo of music was set at 80 bpm and harmonic rhythm at 1, i.e. one chord per bar.

Decisions of the music engine are made at the beginning of each bar, so Tables 3, 4, 5 and 6 represent the sequence of prediction engine data as the music engine examines them, i.e. at every bar.

5. RESULTS AND DISCUSSION

All music generated from the above input data was successful in cadencing, form recovering and ending the music on demand, in a form-complying manner.

Tables 4, 5 and 6 demonstrate the harmonic states of the blues cycle. Specifically, the column *Grammar states of cycle* in each figure presents bars from 1 through to 12 at the time of corresponding events. '||' denotes the end of music generation. Bars are shown in terms of linear ('*L*') and cycle ('*C*') numbering and each row demonstrates the post-rewrite state of its corresponding bar, i.e. a terminal has been produced for that bar. Arrows from each table row show the trajectory points that trigger each corresponding event, which, in turn, impacts music generation (Tables 4, 5, 6 and Figures 8, 9, 10).

Music for *trajectory 1* is generated according to form rules until bar 21, i.e. bar nine of the form, second time round, when a cadence is introduced. The cadence is placed in bar 11 of the form (second time round) since 'ENDING==true' occurred in bar 9 of the form, i.e. within bars seven to ten (Figure 6). In bar 25, i.e. the first bar of the form in the third cycle, the post-cadence phase is reached and the top-level non-terminal 'fin' populates the form. 'GOAL_REACHED' occurs in bar 31 (or seven in the third cycle) where a tonic has been produced, so grammar productions and music stop there.

In *trajectory 2* we have a similar development until bar 30. As shown in Table 5 at bar 29 a post-cadencing phase has been reached ('fin') while previously the cadence at bar three of the form has been scheduled since bar 24. When bar 31 is reached, a recovery bar is scheduled for the next bar, i.e. bar eight of the form. Along with it, optimum chords are placed in the remaining bars. Subsequently, at bar 32 a cadence is scheduled again for bar 11 of the form and post-cadencing begins at bar 37. The goal is reached at bar 38 and music generation continues only for one more bar (39) when a tonic is produced.

In a similar way *trajectory 3* schedules a cadence at bar 24 (for the third bar of the form), post-cadences at bar 17 and schedules recovery at bar 18 for the next bar (seventh of the form). At bar 25 it is interesting to note that post-cadencing is cancelled by a recovery at that point. So instead of a 'fin' population (for bars two to twelve of the form), the optimum chords populate bars three to twelve with a 'reconciling' chord ('rec') at bar two of the form. The same happens at bar 33 when post-cadencing would be expected at bar ten of the form. However, post-cadencing coincides with recovery again and recovery overrules as expected. In addition, it is worth mentioning that at bar 35, when a cadence is scheduled for bar three of the form, 'rec' is still present at bar ten of the cycle. This is a leftover symbol from the preceding recovery (bar 33) and is cancelled by 'fin' once the post-cadence phase is reached (bar 41). Finally, post-cadencing (bar 41) coincides with GOAL_REACHED and since the 'current' chord is a tonic (form bar five) the music stops.

Generally, as Tables 4, 5 and 6 show, the positioning of top-level non-terminals (*dec*, *cad*, *fin, rec*) is, in most cases, scheduled in advance. Thus, the non-terminal is not rewritten until its musical time (i.e. bar) has arrived.

6. CONCLUSIONS AND FUTURE WORK

In summary, FATE is a real-time system design that generates music in a form-aware manner and can respond to environment data to cadence, recover and end musical form on demand. It aims to generate music in a way that retains the flexibility of the short-form approaches like

[1] At this stage translation into MIDI was made by hand although it will be automatic in the future.

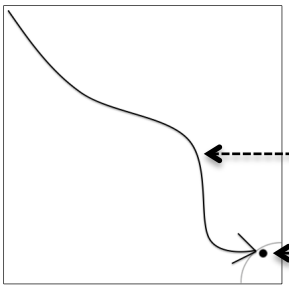

Figure 8. Trajectory 1.

Trajectory 1			
Events	*Bars*		*Grammar states of cycle*
	L	*C*	
No significant input	20	8	i i6 v7 i7 iv6 iv6 i7 i6 dec v7 dec v7
ENDING	21	9	i i6 v7 i7 iv6 iv6 i7 i6 v7 IV **cad** v7
Post-cadencing	25	1	i6 fin fin fin fin fin fin fin fin fin fin fin
GOAL_REACHED	31	7	i6 v7 i6 v7 i6 v7 i7 fin fin fin fin fin
Stopping	31	7	i6 v7 i6 v7 i6 v7 i7 **||** fin fin fin fin fin

Table 4. Events, bars of occurrence and current cycle state at each bar for trajectory 1.

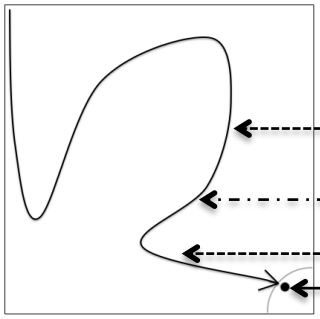

Figure 9. Trajectory 2.

Trajectory 2			
Events	*Bars*		*Grammar states of cycle*
	L	*C*	
No significant input	23	11	i7 v7 i7 i7 iv7 iv7 iiim7 i6 v7 iv7 i7 V
ENDING	24	12	i7 v7 **cad** i7 iv7 iv7 iiim7 i6 v7 iv7 i7 V
Post-cadencing	29	5	i7 v7 iv7 v7 i7 fin fin fin fin fin fin fin
RECOVERY	31	7	i7 v7 iv7 v7 i7 v7 i7 **rec** V IV I I
ENDING	32	8	i7 v7 iv7 v7 i7 v7 i7 i7 V IV **cad** I
Post-cadencing	37	1	i7 fin fin fin fin fin fin fin fin fin fin fin
GOAL_REACHED	38	2	i7 v7 fin fin fin fin fin fin fin fin fin fin
Stopping	39	3	i7 v7 i7 **||** fin fin fin fin fin fin fin fin fin

Table 5. Events, bars of occurrence and current cycle state at each bar for trajectory 2.

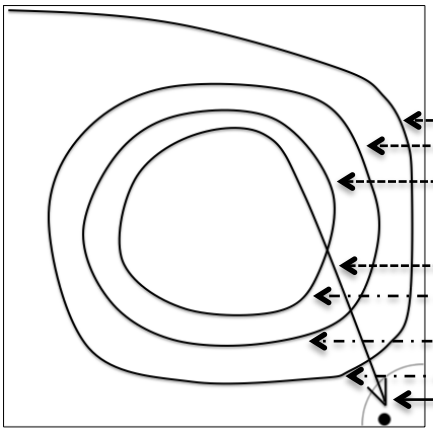

Figure 10. Trajectory 3

Trajectory 3			
Events	*Bars*		*Grammar states of cycle*
	L	*C*	
No significant input	12	12	i7 iv7 i7 v7 iv7 iv6 i7 i6 v7 iv7 i7 v7
ENDING	13	1	i7 IV **cad** V dec iv6 dec i6 dec iv7 dec v7
Post-cadencing	17	5	i7 iv7 iim7 v7 i7 fin fin fin fin fin fin fin
RECOVERY	18	6	i7 iv7 iim7 v7 i7 v7 **rec** I V IV I I
ENDING	21	9	i7 iv7 iim7 v7 i7 v7 i7 i6 v7 IV **cad** I
Post-cadencing	25	1	(cancelled by RECOVERY–see next row)
RECOVERY	25	1	i6 **rec** I I IV IV I I V IV I I
ENDING	29	5	i6 iv7 i7 i7 iv7 IV **cad** I V IV I I
Post-cadencing	33	9	(cancelled by RECOVERY–see next row)
RECOVERY	33	9	i6 iv7 i7 i7 iv7 iv6 iim7 v7 i7 **rec** I I
ENDING	35	11	i6 iv7 **cad** i7 iv7 iv6 iim7 v7 i7 (rec) I I
Post-cadencing	41	5	i6 iv7 iim7 v7 i7 fin fin fin fin fin fin fin
GOAL_REACHED	41	5	i6 iv7 iim7 v7 i7 fin fin fin fin fin fin fin
Stopping	41	5	i6 iv7 iim7 v7 i7 **||** fin fin fin fin fin fin

Table 6. Events, bars of occurrence and current cycle state at each bar for trajectory 3.

procedural note/chord-level generation and dynamic layers, while managing larger-scale form.

The design is divided in two: a prediction engine that captures EPE episode data and estimates the remaining time of that episode and a music engine that generates music in real-time. The prediction engine uses an EWMA of a participant's speed (among other calculations) to estimate the remaining time. The music engine uses a stochastic, context-sensitive grammar that allows for hierarchical grammar rewrites at specific times of the musical form. Currently we address the non-abrupt musical termination of episodes.

The parser enables the programmer/computer composer (length, top-level time-specific decision points, rewrite rules) in order to generate the harmony of a medium-(length, top-level time-specific decision points, rewrite rules) in order to generate the harmony of a medium-length chorus-based form such as 12-bar blues and jazz standards.

Future work will address a number of open challenges:
1. Grammatical production probabilities as well as the optimum harmonic sequence will be learnt from a corpus.
2. Longer 'reconciling' musical periods will be applied to test their impact on musical smoothness of harmonic recovery and how effectively the form is re-established. Also, the location of the reconciling period will be varied.
3. The algorithm will be further developed to support generation and non-abrupt ending of musical forms that are longer and less repetitive than the medium-length chorus-based forms that are currently supported.
4. The algorithm will be further developed in order to generate musical transitions between two musical sections rather than simply ending and recovering one.

5. The system will be applied for musical accompaniment of a real-world interactive theatrical installation.

Acknowledgments

This work is funded by the UK Engineering and Physical Sciences Research Council through the VEIV Doctoral Training Centre at UCL [grant number: EP/G037159/1] and the Alexander S. Onassis Foundation. We are grateful for the partnership of Penny Dreadful Productions in this work. Data in support of this paper is available at DOI: 10.14324/000.ds.1503636

7. REFERENCES

[1] B. Magerko, W. Manzoul, M. Riedl, A. Baumer, D. Fuller, K. Luther, C. Pearce, An Empirical Study of Cognition and Theatrical Improvisation – creativity '09, 2009, Berkeley, California, USA.

[2] Cirque du Soleil Theme Park, https://www.cirquedusoleil.com/en/press/news/2014/grupo-vidanta-nueva-vallarta-entertainment-park-12[2], 2014 (announcement).

[3] DisneyQuest®, https://disneyworld.disney.go.com/entertainment/disney-springs/disney-quest-indoor-interactive-theme-park/, 1998 – present.

[4] Blast Theory, Desert Rain, www.blasttheory.co.uk/, 2000.

[5] M. Niemann, Five Minutes, www.fiveminutes.gs, 2014.

[6] K. Collins, Game sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design, MIT Press, 2008.

[7] S. Williams, Music in the Theatre, The Oxford Companion to Theatre and Performance, ed. by D. Kennedy, Oxford University Press, 2010.

[8] S. Benford, G. Giannachi, B. Koleva, T. Rodden, "From Interaction to Trajectories: Designing Coherent Journeys Through User Experiences", CHI 2009, Boston, USA.

[9] J.D. Fernandez, F. Vico,"AI Methods in Algorithmic Composition: A Comprehensive Survey", Journal of Artificial Intelligence Research 48 513-582, 2013.

[10] M. Muller, J. Driedger, "Data-driven soundtrack generation, Multimodal Music Processing", Dagstuhl Follow-Ups - vol. 3, Multimodal Music Processing, ed. by M. Muller, M. Goto, M. Schedl, ISBN 978-3-939897-37-8, 2012.

[11] Hello Games, No Man's Sky, http://www.no-mans-sky.com/, 2016.

[12] Thatgamecompany, Journey, http://thatgamecompany.com/games/journey/, 2012.

[13] A. R. Brown, R. Wooller, K. Thomas, "The Morph Table: A collaborative interface for musical interaction", Proc. Australasian Computer Music Conference 2007, Lefkada, Greece.

[14] M. O. Jewell, Motivated Music: Automatic Soundtrack Generation for Film, PhD Thesis, University of Southampton, 2007.

[15] E. Vane, W. Cowan, "A computer-aided soundtrack composition system designed for humans", International Computer Music Conference, 2007.

[16] D. Goldmark, Pixar and the Animated Soundtrack, The Oxford Handbook of New Audiovisual Aesthetics, ed. by J. Richardson, C. Gorbman, C. Vernallis, Oxford University Press, 2010.

[17] A. Hazzard, Guidelines for Composing Locative Soundtracks, PhD Thesis, University of Nottingham, 2014.

[18] S. P. Langston, (201) 644-2332 or Eedie & Eddie on the wire: An experiment in music generation, Bell Communications Research Morristown, NJ, 1986.

[19] Lucasfilm Games, Ballblazer, http://www.mobygames.com/game/ballblazer, 1984.

[20] G. Whitmore, Adaptive Audio Now! A Spy's Score: A Case Study for No One Lives Forever, DirectX 9 Audio Exposed: Interactive Audio Development, ed. by T. M. Fay, S. Selfon, T. J. Fay, Plano, Texas: Wordware Publishing, 2004.

[21] Microsoft, Russian Squares, https://www.youtube.com/watch?v=a-ZyozBeUDg, 2002.

[22] Y. Ioannides, [.talktome]: adaptive/dynamic audio prototyping for video games, https://cycling74.com/project/talktome-adaptivedynamic-audio-prototyping-for-video-games/#.V4pq-TaqTIw, 2012.

[23] B. Eno, Spore, http://www.spore.com, Electronic Arts, 2009.

[24] Strange Loop Games, Simcell, http://www.strangeloopgames.com/education/, 2012.

[25] R. Wooller, Techniques for automated and interactive note sequence morphing of mainstream electronic music, Queensland University of Technology, PhD Thesis, 2007.

[26] C. Roads, P. Wieneke, "Grammars as representations for music", Computer Music Journal, vol.3, no.1, 1979.

[27] R. M. Keller, D. R. Morrison, "A grammatical approach to automatic improvisation", Proc. Sound and Music Computing Conference, 2007, Lefkada, Greece.

[2]All URLs shown were last accessed on 15/07/2016.