Musical Metacreation: Papers from the 2012 AIIDE Workshop
AAAI Technical Report WS-12-16

Mezzo: An Adaptive, Real-Time Composition Program for Game Soundtracks

Daniel Brown
University of California at Santa Cruz
twitteringmachine2000@yahoo.com

Abstract

Mezzo is a computer program designed that procedurally writes Romantic-Era style music in real-time to accompany computer games. *Leitmotivs* are associated with game characters and elements, and mapped into various musical forms. These forms are distinguished by different amounts of harmonic tension and formal regularity, which lets them musically convey various states of markedness which correspond to states in the game story. Because the program is not currently attached to any game or game engine, "virtual" gameplays were been used to explore the capabilities of the program; that is, videos of various game traces were used as proxy examples. For each game trace, *Leitmotivs* were input to be associated with characters and game elements, and a set of 'cues' was written, consisting of a set of time points at which a new set of game data would be passed to *Mezzo* to reflect the action of the game trace. Examples of music composed for one such game trace, a scene from *Red Dead Redemption*, are given to illustrate the various ways the program maps *Leitmotivs* into different levels of musical markedness that correspond with the game state.

Introduction

Mezzo is a computer program designed by the author that procedurally writes Romantic-Era-style music in real time to accompany computer games. It was motivated by the desire for game music to be as rich and expressive as that written for traditional media such as opera, ballet, or film, while still being procedurally generated, and thus able to adapt to a variety of dramatic situations. To do this, it models deep theories of musical form and semiotics in Classical and Romantic music. Characters and other important game elements like props and environmental features are given *Leitmotivs*, which are constantly rearranged and developed throughout gameplay in ways that evoke the conditions and relationships of these elements. Story states that occur in a game are musically conveyed by employing or withholding normative musical features. This creates various states of markedness, a concept which is defined in semiotic terms as a valuation given to difference (Hatten 1994). An unmarked state or event is one that conveys normativity, while an unmarked one conveys deviation from or lack of normativity. A succession of musical sections that passes through varying states of markedness and unmarkedness, producing various trajectories of expectation and fulfillment, tension and release, correlates with the sequence of episodes that makes up a game story's structure.

Mezzo uses harmonic tension and formal regularity as its primary vehicles for musically conveying markedness; it is constantly adjusting the values of these features in order to express states of the game narrative. Motives are associated with characters, and markedness with game conditions. These two independent associations allow each coupling of a motive with a level of markedness to be interpreted as a pair of coordinates in a state space (a "semiotic square"), where various regions of the space correspond to different expressive musical qualities (Grabócz 2009).

Certain patterns of melodic repetition combined with harmonic function became conventionalized in the Classical Era as normative forms, labeled the sentence, period, and sequence (Caplin 1998, Schoenberg 1969). These forms exist in the middleground of a musical work, each comprising one or several phrase repetitions and one or a small number of harmonic cadences. Each musical form has a normative structure, and various ways in which it can be deformed by introducing irregular amounts of phrase repetition to make the form asymmetrical. *Mezzo*'s expressive capability comes from the idea that there are different perceptible levels of formal irregularity that can be quantitatively measured, and that these different levels convey different levels of markedness.

Overview of the Compositional Process

The actual amount of musical information that must be input by a user is very small. First, the *Leitmotivs* that will be associated with characters and game elements must be specified; these can be as long as the user desires, but produce the most interesting outcomes when they are only two to four measures long. Second, the harmonic vocabulary is given by inputting a small number of chord progressions (10 or so), although these progressions are not the ones that are actually used in *Mezzo'*s compositions. The program extracts information regarding voice-leading and levels of acoustic tension from the input progression, and then constructs its own progressions during gameplay (using a genetic algorithm) that are stylistically similar to those that were input. This allows the user to establish a general harmonic language—by taking progressions from a particular composer's work, say, such as Wagner or Chopin—without having to author any specific expressive details herself. The program takes care of that.

Composition in *Mezzo* is a two-step process: first build forms, then deform them according to stochastic constraints. Both of these processes generate expressive features in the music being composed. The form-building stage composes chord progressions with the appropriate length, formal organization, cadence type, and amount of harmonic tension, and builds forms by mapping appropriate *Leitmotivs* and accompanying textures to them. In the second stage, each form that has been composed is mapped to a data structure that sets the stochastic constraints on its formal regularity, determining which, if any, formal sections will be repeated or omitted, and to what extent. This stochastic model is a generalization of the methods of deformation used by Classical and Romantic composers, and attempts to maintain continuity with their procedures while projecting them into a modern, interactive context. Furthermore, it extends this concept to function in the open-ended setting of a game. Each time a form is stated, it will, with some probability, be organized differently from previous times, and there will be no pattern to the way the organization changes from statement to statement. However, this randomness is controlled by the stochastic variables, so that a certain quality of irregular formal organization will always be met.

Composition Examples for Virtual Gameplay

Mezzo is currently not attached to any game or game engine, although development in this area is underway.[1] To explore the capabilities of the program, "virtual" gameplays have been used; that is, videos of various game traces were used as proxy examples, each ranging from about six to 13 minutes long. Videos of scenes from three games have so far been used: a level of *Red Dead Redemption*, in which a cowboy must herd cattle and stop a train robbery; a short episode from *Star Wars: The Old Republic*, in which a Jedi Knight must decide whether or not to give in to his romantic attraction to an alien; and a walkthrough of the entire last level of *Super Mario 3D,* where the protagonist Mario must rescue a princess held captive by a monster atop a castle turret.[2]

For each game trace, a set of *Leitmotivs* was written for each character and game element I considered significant, which were then loaded into the program as MIDI files. Each game trace was also given a set of harmonic progressions taken from a work by a Romantic-Era composer which seemed appropriate for the game: Liszt for *Red Dead Redemption* (because of its pastoral character), Chopin for *Star Wars* scene (because it's a love scene), and Wagner for *Super Mario* (because of its overblown, mythological nature). Then a short set of 'cues' was written, consisting of a set of time points at which a new set of game data would be passed to *Mezzo*.

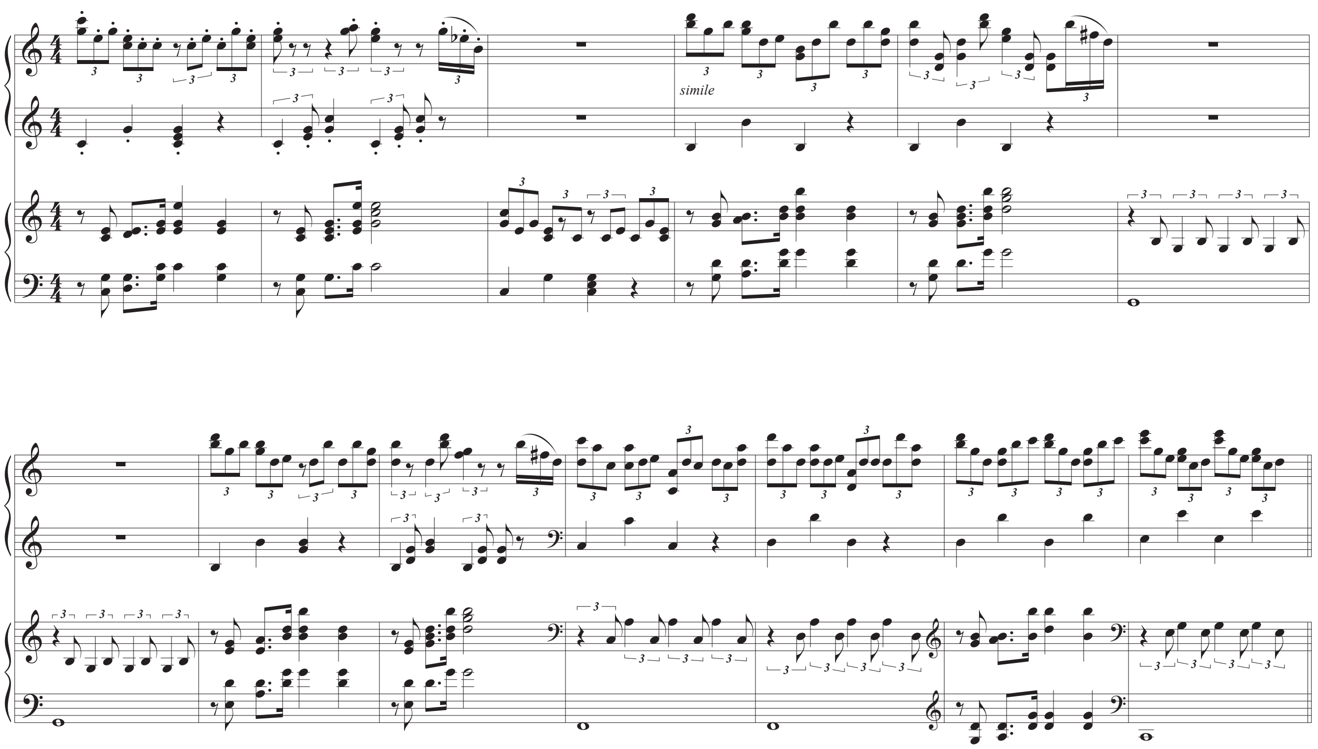
To compose music for a certain state of a game, *Mezzo* uses a number of different settings. Any characters and elements associated with *Leitmotivs* which are involved in the current scene are passed to the program, as well as the characters' states. *Mezzo* is designed to work with games in which some normative state can be defined for a character, be it measured by health or goal achievement or some other interpretation, and the distance from this normativity can be measured.

Another small set of states associated with the player's character is also passed to the system. These states are related to the measure of normativity, but must also be interpreted according to the character's relation to other characters and the game story. These states are defined as *stability* (when the player is not currently facing a problem to overcome), *battling* (when the player is in direct conflict with another character who is present), *questing* (when the player is trying to achieve some goal, but is not directly faced with an opposing character), *failing* (when the character has made some irreversible mistake, like getting killed), *succeeding* (when the character has completely overcome some problem, like destroying an opponent), and *introduction* (when a character first appears and is not in any of the other states).

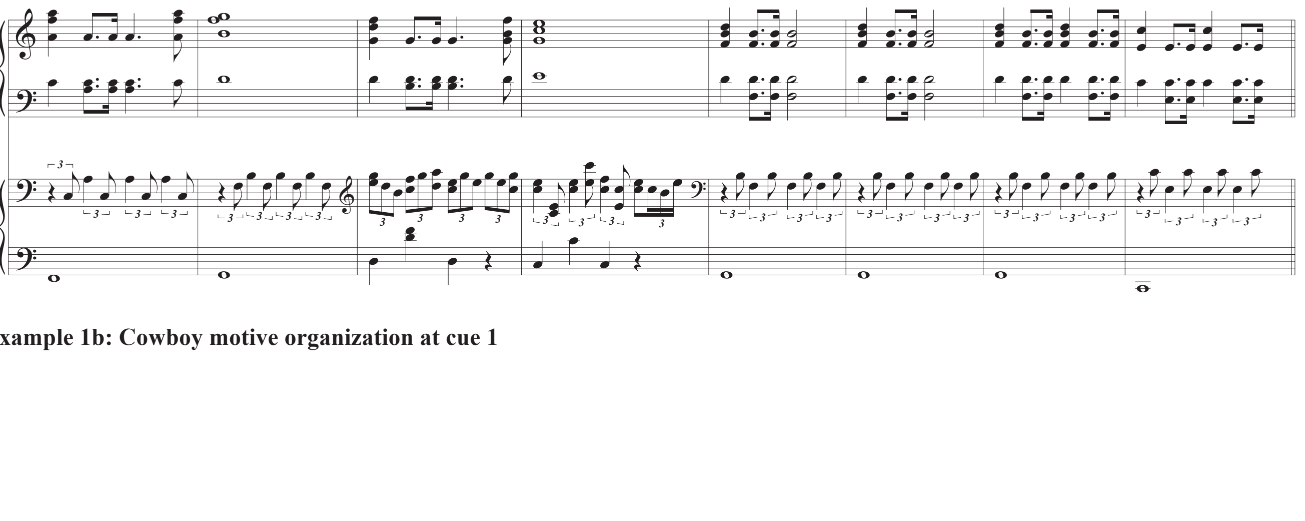
Finally, features relating to the level of activity are used by *Mezzo* to determine aspects of the musical texture, like how energetic it will be, how much space will occur

[1] *Mezzo* is written in Python, and uses Max as an interface for sending game signals to the Python code and handling playback. Communication between Python and Max is implemented using Open Sound Control.

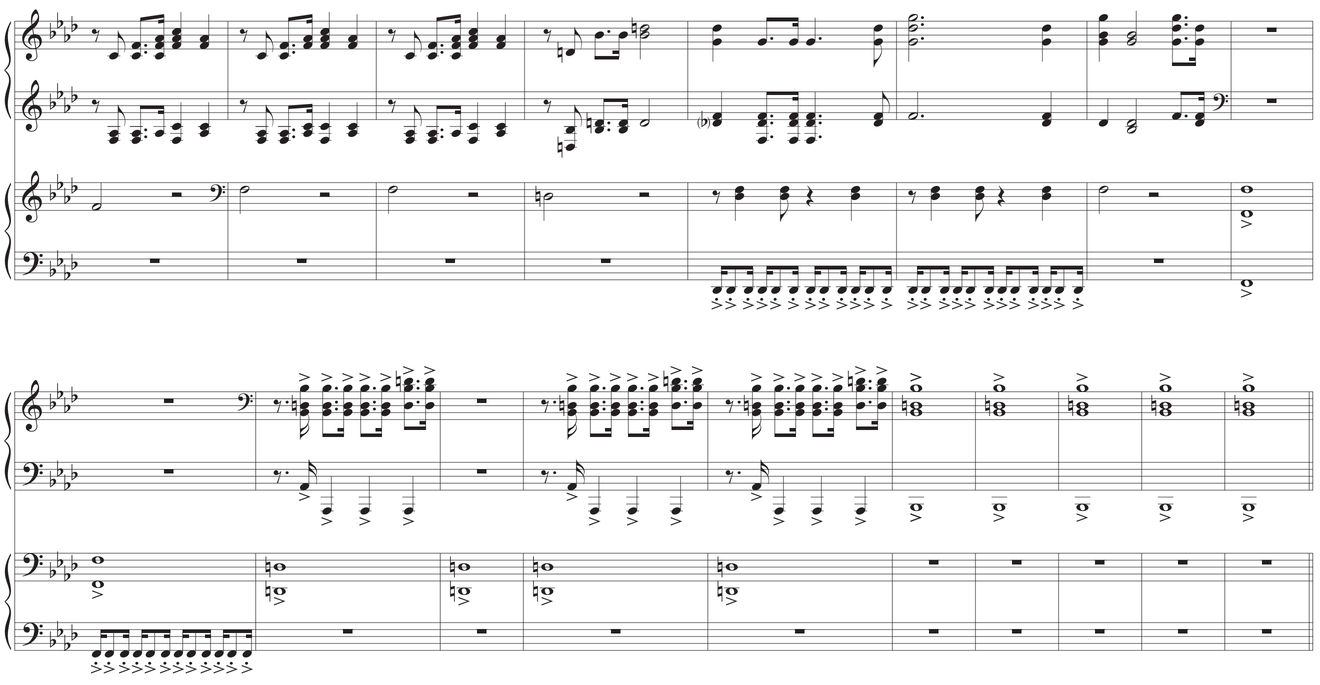
[2] At the time this paper was written, these videos were available at the following URLs:
http://www.youtube.com/watch?v=LSIdhm_EOec
http://www.youtube.com/watch?v=X1ZWJdbdsCM
http://www.youtube.com/watch?v=-9-WfJLlhXE



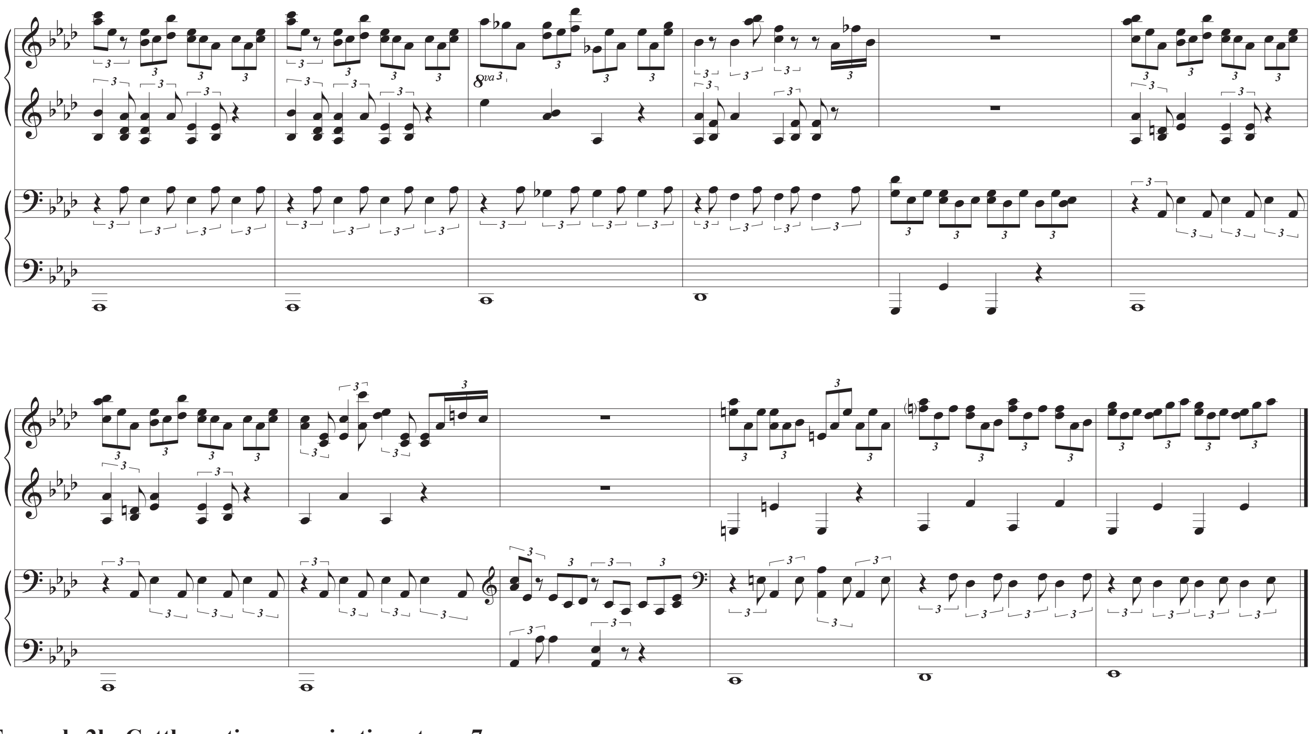
Example 1a: Cattle motive organization at cue 1



Example 1b: Cowboy motive organization at cue 1



Example 2a: Cowboy motive organization at cue 5



Example 2b: Cattle motive organization at cue 7

between statements of *Leitmotiv*s, what registers they will occur in, etc.

Each time point in a script was associated with a set of values for each of the states listed above. This process, of course, assumes an amount of interpretive ability on the part of an actual game, which here is being done by a human author. While this is currently a drawback of the design of the demonstration, the interpretive assumptions being made are reasonable; this is because the states a game must pass *Mezzo*, as described above, are well defined in terms of explicit game elements.

A Max patch was used to trigger each set of values to *Mezzo* at each associated time point, and this process and a video were begun at the same time. The music was composed for four MIDI channels, all set to a piano sound. Each time the program is run alongside a video, the music is similar, although it is never the same.

For the *Red Dead Redemption* scene, ten cues were determined for which the scene changes:

1: cattle introduced; cowboy rides into herd
2: explosion on horizon; cowboy rides toward it
3: cowboy encounters train robbery in progress
4: gunfight between cowboy and train robbers
5: cowboy's horse shot; cowboy looks for another horse; train robbers flee
6: cowboy mounts new horse and bids farewell to train passengers he rescued from robbers
7: cowboy attempts to marshal cattle that have scattered during commotion of train robbery
8: cowboy tries to corral one errant cow back to herd; gives up, rides off with remaining cattle
9: the herd, minus one cow, reaches the pasture
10: cowboy rides home, mostly successful

Two types of *Leitmotivs* were input, corresponding to the player's character (a cowboy), and the herd of cattle. The player's character had four similar *Leitmotivs*, which I composed, and the herd only one, which was taken from a Liszt scherzo. The musical examples given below show some of the resulting music that was written for these cues. The cowboy's motives and the cattle's can be distinguished in the following way: the cowboy's are made up of block chords in dotted-eighth- and dotted-sixteenth-note rhythms; the cattle's motive is made up of scherzo-like arpeggiated triplets.

Example 1 shows their organization at cue 1; here, the formal organization is very regular—both the cowboy's and the cattle's motives are organized as normative sentences, and the harmonic tension is very low. This evokes the relatively normative state that both the player and the cattle are in: the player is faced with a problem (herding the cattle to pasture), but the game has just begun, and the problem does not seem dire.

Example 2 shows music written in later stages of the game, in which the same *Leitmotivs* are used, but are now mapped to non-normative—that is, highly marked—musical settings. First shown is a section using the cowboy's motives during cue 5, when his horse has been shot and he is under great duress. Second shown is the cattle's motive at cue 7, in which the herd is highly disorganized, presenting the cowboy with the problem of coercing them back together. In both of these sections, the motives are organized in a highly non-normative way, and the harmonic tension is high, in order to express the highly marked states the characters are experiencing.[3]

Further Avenues for Research

While many important expressive elements are not handled by Mezzo, it still offers a framework for procedurally composing cohesive music that adapts to dramatic elements in a game in real time. Elements such as orchestration, motivic development, and expressive playback (rubato, dynamics, etc.) are certainly necessary to make this an effective music engine for game development. Furthermore, a standardized communication protocol for mapping game states to musical settings needs to be developed and implemented. However, the program as it now stands is a proof of concept of a method of procedurally composing music that is expressive and interesting. It also offers a foundation to which more implementations of musical expressiveness can, and hopefully will be, added in the future.

References

BioWare Austin and BioWare Edmonton. *Star Wars: The Old Republic*. BioWare, 2008.

Brown, Daniel. 2012. Expressing Narrative Function in Adaptive, Computer-Composed Music. D.M.A. diss, Department of Music, University of California at Santa Cruz, Santa Cruz, CA.

Caplin, William. 1998. *Classical Form: A Theory of Formal Functions for the Instrumental Music of Haydn, Mozart, and Beethoven*. New York: Oxford U Press.

Hatten, Robert S. 1994. *Musical Meaning in Beethoven: Markedness, Correlation, and Interpretation*. Bloomington: Indiana U Press.

Grabócz, Márta. 2009. *Musique, narrativité, signification*. Paris: L'Harmattan.

Nintendo EAD Tokyo. *Super Mario 3D Land*. Kyoto: Nintendo, 2011.

Rockstar San Diego. *Red Dead Redemption*. New York: Rockstar Games, 2010.

Schoenberg, Arnold. 1969. *Structural Functions of Harmony*. New York: W.W. Norton & Co.

[3] A full analysis of the music composed for this game trace is in Chapter 7 of (Brown 2012).