Demonstrating MMorph:
A System for Morphing Music in Real-Time

Daniel V. Oppenheim
Computer Music Center
IBM T. J. Watson Research Center
P. O. Box 218, NY 10598
music@watson.ibm.com

ABSTRACT: MMorph is an interactive system for morphing music in real-time. Morphing music sources of di\text{-}verse styles and different keys produces a smooth and gradual transition from one to the other. During the transition the listener perceives musical characteristics from one source gradually becoming less prominent as those of the other sources become dominant. We will demonstrate MMorph with diverse musical styles, including Baroque, Classical, Jazz, Tango, and Latin Percussion. The morphing engine will be explained, and its potential for music composition and performance will be discussed.

Morphing music
Utilizing a musical motif that listeners can perceive easily and through which they can follow and comprehend the compositional process began in the late Renaissance and became a characteristic of Baroque music. Classical composers experimented with the new Sonata form that incorporated two contrasting themes. Faced with a new compositional challenge of unifying a work containing contrasting elements, they began to emphasize the bridge and development sections. One could argue that Beethoven was the first composer to consciously attempt musical morphing—often, his bridge sections gradually introduced the musical elements of the target theme as characteristics from the previous were placed in the background. In modern music the idea of morphing expanded to include sonic environments, as in Penderecki's second string quartet. Computer music offers an ideal workspace for such exploration. Typically composers write dedicated programs for realizing morphing within a specific composition, such as in Paul Lansky's Quackerbridge or Table's Clear (see http://www.music.princeton.edu/~paul).

An interactive tool that will enable composers to morph between different sonic environments seems highly desirable for computer music. Observing the broader domain of multimedia where visual morphing is now common practice, suggests that such a tool might also enable a better integration of music into the virtual worlds of cyberspace.

What is morphing and the musical elements involved
Morphing deals directly with the musical elements of a source and target musical work. For morphing to 'work' the listener must be able to perceive these elements easily and to associate them readily with their origin. Any combination of the following elements could be used in a morph: melody (pitch or interval), rhythm, harmony, dynamics, texture, timbre, gestures (articulations), and so forth. For each such element a new value is calculated. This value is derived from the values of that element in all participating sources and is a function of an intermediate state. This state is a weighting factor that determines the relative prominence of each source in relation to others; it varies as a function of time as a transition is made from one musical work to another. Each state will produce a different musical result. Morphing can be said to work well when at any given state the listener can still perceive the relation to both the source and target music.

Several issues should be considered for morphing. First, different themes are characterized by different musical elements. A listener can follow how one source morphs into another only if the appropriate musical elements are taking part in the morphing process. Pitch and rhythm may be a good default for music that is instrumental in nature but not necessarily for acoustic music with rich sonic textures. A morphing system must enable its user to select the musical elements he wants to work with; each combination of elements may produce a very different musical result. Second, one must determine the morphing algorithm. For example, interpolation is reasonable for rhythms, but with pitch it may produce frequencies that not only do not belong to any source but are also not part of any source's scale or even tuning system. We found that a weighted selection algorithm often worked better for pitch. A third issue relates to handling time; how do time-onsets of musical events occurring in an intermediate state relate to their onset within their origin? With a linear time scheme time-onsets are retained; with a time-warping scheme they are not. Again, each approach produces a different result.
Implementation

MMorph is implemented in ParcPlace Smalltalk/VisualWorks 2.0 within DMIX (Oppenheim, D., 1993a). Due to the limited scope of this paper only the user-interface will be described. A detailed report describing the morphing engine and algorithms should be available by the time of publication and may be requested via email to reports@watson.ibm.com.

MMorph is designed to handle most common score representations. It is currently interfaced only to MIDI, but in theory should handle equally well more complex musical events as found in Csound, Music Kit, or Common Music (assuming real time synthesis is available). For each parameter (p-field) the user may determine whether it will participate in the morphing process and select one of several morphing algorithms.

Figure 1 shows the current front end to MMorph. Up to four musical works can be used as sources for a morphing process (Music i through Music iv), they can be Slapped (in Oppenheim, D. 1993b) or loaded from disk (v, numbers refer to the figure). Each music source can be individually turned on or off (1), and the user can determine which of its parameters will participate in the process (2). For each parameter the user can choose between an interpolation or weighted selection algorithm (3). A separate mechanism deals with MIDI track allocation to ensure the best choice of instruments/time for each voice within a music (4)—a problem that is MIDI specific. Finally, while morphing, the user can interactively drag a control that determines the relative weight of each of the four musical works (3).

Limitations and some future extensions

Having MIDI as the sole synthesis engine produces several serious limitations that make it hard, if not impossible, to deal in any serious fashion with timbre or musical gestures. MMorph currently deals only with MIDI note entries and ignores all others, a problem we may address in the future. Interfacing to a real time sound synthesis engine should eliminate many of these shortcomings. We also hope to implement the morphing algorithms using a DMIX CodeDictionary so that users may easily plug in their own algorithms.

References
