Designing Interactions With Time-based Media

Eric Lee
Media Computing Group
RWTH Aachen University
eric@cs.rwth-aachen.de

Jan Borchers
Media Computing Group
RWTH Aachen University
borchers@cs.rwth-aachen.de

ABSTRACT
Designing interfaces for traditional media such as text and graphics has been at the forefront of human-computer interaction studies in recent years. In contrast, interfaces for time-based media such as video and audio remain comparatively primitive, often with no way to interact with the structure or semantics of the data. In this paper we will offer some discussion on the time design of interactive systems based on our previous work, including the challenges faced when examining interaction metaphors, physical interface design, and time manipulation algorithms. We will also discuss how these factors affect the user experience when interacting with time-based media.

Author Keywords
time design, design patterns, audio time-stretching, real-time systems, interactive music exhibits

INTRODUCTION
Much emphasis has been placed in recent years on improving user interfaces for static media such as text and images. Today, a multitude of interfaces exist for interacting with not only the syntax, but also the structure and semantics of static documents; some examples include Microsoft Word's grammar checker and the Google search engine[1].

In contrast, interfaces for multimedia that are in widespread use remain limited. Despite the promises brought by computers and technology, interaction with video and audio media today is still generally limited to playback in a sequential fashion with no understanding of the structure and semantics of the data itself. Even Google's image search [2], which allows one to search for images based on a keyword, treats images as a "black box" of data, and uses the context of its parent web page to determine the semantics.

For advanced interaction with time-based media, it is often necessary to parse and process the structure and semantics of the data. Music, for example, has multiple levels of structure. On one hand, it is composed of multiple instrument tracks (horizontal structure); however, a temporal structure also exists, defined by the bars, beats and notes of the piece. Given an audio recording, however, there is little support for accessing and processing this structure in today's toolkits. For example, Apple recently added support for real-time interactive text search in its user interface toolkit [3]. In contrast, a feature for pitch-preserving time stretching is often requested in the Apple QuickTime developer mailing lists, but remains unsupported; in fact, an Apple developer has been quoted as saying that the QuickTime architecture does not easily allow support of this type of functionality.

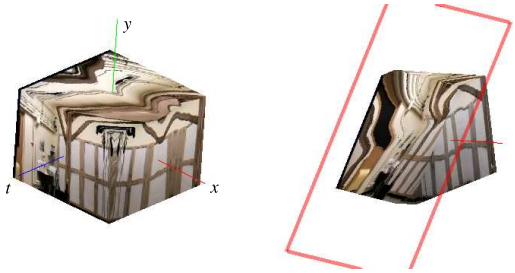

Figure 1: Video cubism. The image on the left shows the basic video cube, the image on the right shows a slice of the cube across space and time.

The aim of this paper is to provide some examples of our research on new interfaces for time-based multimedia, followed by some discussion on the design requirements and challenges associated with these types of interfaces. We will conclude with some thoughts on possibilities for future work.

PREVIOUS WORK
Video Cubism [9] proposes a novel way of interacting with multimedia streams by allowing the user to view and manipulate the stream's spatial and temporal dimensions simultaneously. Two-dimensional video data is shown as a cube with time forming the third dimension; one can manipulate the orientation of a plane cutting across the cube. The resulting image on the cut plane is a mixture of both space and time (see Fig. 1). This work is an example of a new interaction metaphor which interchanges space and time.

The WorldBeat interactive exhibit [5] examines various novel ways of interacting with music as an example of time-based media. One interaction mode allows a user to conduct a synthesized musical score; another mode allows the user

Figure 2. A user in front of the WorldBeat exhibit.

Figure 3. The Personal Orchestra system.

to play an accompaniment to a musical score using a virtual xylophone – the user input is adjusted to the harmonics of the accompaniment, so the user could never play any "wrong notes". The temporal characteristics of the user input, however, are not changed to ensure that the causal relationship between input and output remains unaffected. Finally, WorldBeat features a "query by humming" function that allows a user to search for a tune simply by humming it (see Fig. 2). This system explores various new interaction metaphors for music, and features infrared batons as a novel interface device.

The Personal Orchestra system [7] improves on the conducting interface of WorldBeat by allowing the user to conduct a recording of a live performance as opposed to synthesized music. The addition of video further enhances the user experience (see Fig. 3). Tempo, dynamics and instrument emphasis can all be controlled; the gesture recognition system detects a simple up-down motion, allowing users with no prior conducting experience to enjoy the exhibit. It has become one of the most successful permanent exhibits at the House of Music in Vienna. This work shows how the temporal characteristics of the medium (audio recording of orchestral music) affects the design of the system, since a naive approach of time stretching audio has the undesirable effect of changing the pitch of the audio.

DESIGN REQUIREMENTS AND CHALLENGES

We now discuss some of the challenges associated with the design of the systems in our previous work.

Real-Time Processing of Polyphonic Audio

Non-trivial processing of time-based media often requires working with uncompressed data. Popular compression algorithms, such as MPEG, work with multiple frames of data over time to remove redundancy; to uncompress a single frame of video then, requires the context in which it was compressed. This restriction can be a problem since processing bandwidth may already be dedicated to performing the actual image or audio processing. Thus, it is unsurprising that many audio and video processing applications choose to work with and store uncompressed data, which then creates unique challenges for maintaining real-time performance due to the sheer amount of data involved.

As an example, our previous work has required time stretching audio whilst preserving the pitch. In the past, this problem has been solved in one of the following ways:

1. *Synthesized music:* Music synthesizers such as MIDI (Musical Instrument Digital Interface) have inherent knowledge of the musical semantics, so it is trivial to change the tempo by changing the time at which the notes are dispatched. Unfortunately, synthesized music is often unsatisfying for the user because of its lack of realism.

2. *Pre-stretched audio:* The Personal Orchestra system had prior knowledge of the audio data, so it made use of pre-stretched audio tracks. Unfortunately, this technique has limited applicability – while it produces high quality output, the system cannot respond to real-time input from the user. Furthermore, the system is restricted to speed changes at discrete intervals, reducing the perceived responsiveness of the system. Finally, storing many pre-stretched audio tracks increases the data storage requirements immensely.

3. *Low quality time-stretch:* This technique is often found in digital answering machines to allow users to scan through their messages. However, for applications such as the Personal Orchestra which utilize polyphonic audio, a low-quality time stretch would be unsatisfying for the end user, as hearing high fidelity orchestral music is an essential part of the user experience.

It was not until recently that it became possible to perform a pitch-preserving, high-quality time-stretch of polyphonic audio in real-time using inexpensive hardware. When one compares the processing bandwidth required for audio (kilobytes per second) and video (megabytes per second), real-time processing becomes a challenge that must be accounted for early in the design process.

The question remains: is it possible to correctly time-stretch video? Certainly, one can simply change the playback speed, which yields satisfactory results in many cases, such as with a video of an orchestral performance. However, if one were to apply this change to raindrops falling from the sky, the semantics are no longer correct. What we would like is for the *amount* of raindrops falling from the sky to decrease, not the *speed* at which they fall.

Support for Multiple Devices

One of the ideas that video cubism explored was how to use a spatial dimension on a display to unravel the temporal characteristics of time-based media. This interchangeability of time and space is not new – image-processing techniques such as super-resolution imagery often make use of the time dimension to enhance the spatial dimension by combining a sequence of low-resolution images into one, high resolution image. We are currently examining interfaces for large displays as part of our research; in theory, we could continue along this idea of interchanging time and space, especially with interfaces that span multiple large displays. However, the design of such a system imposes unique design challenges because it requires a framework for multiple devices to interact with each other, since each display must be accompanied by its own processor.

At Stanford, we were part of a group that created the iROS (Interactive Room Operating System), a TCP-based middleware that allows multiple devices to exchange information [10]. It offers a layer of abstraction to better support dynamic, heterogeneous environments by decoupling the devices from specific machines and applications. Unfortunately, such a loosely-coupled system introduces additional latencies in the overall system as events from a particular source must propagate across multiple devices before reaching its destination. We are currently studying how this latency affects interaction in an interactive-room type of environment.

Latency

Our experience has showed us that latency is a significantly larger problem in multimedia interfaces. Jeff Johnson [11] proposes three latency thresholds for maintaining responsiveness: 0.1, 1 and 10 seconds, depending on the nature of the feedback. The time-based nature of multimedia accentuates the need for the user to perceive a cause and effect relationship during the interaction; according to Johnson, this requirement falls in the 0.1 second threshold. Moreover, time-based interactions often involve a tightly closed feedback loop where the system produces output based on user input and the user, in turn, feeds in new input by reacting to this system output. Thus, consistently and continuously meeting this low response time becomes critical to convey to the user the feeling of causality and overall system responsiveness.

An example to illustrate this problem is video conferencing. In video conferencing, audio can be received much sooner than video because of an additional delay incurred by collecting a series of video frames and compressing them before transmission. This situation creates an unfortunate dilemma: if the audio is played as soon as it is received, the system behaves responsively but synchronization with video is sacrificed. Conversely, if audio playback is delayed until the video is received, system responsiveness suffers. In both cases, the user experience is unsatisfying. One of our former colleagues at Stanford developed an algorithm for addressing this latency issue by manipulating the time scale of the audio: audio is played back as soon at is received, and using a real-time time-stretching algorithm, the audio is slowed down until the video has time to catch up [8]. His test users have responded very favorably to this solution.

Post-Desktop Interfaces

It is often challenging to create interfaces for multimedia systems because the familiar keyboard and mouse were designed for text and graphics and are often inadequate for interaction with multimedia. Many interactive media applications today go to great lengths to re-create metaphors for multimedia manipulation, but the devices themselves remain virtual. An on-screen "wheel" for scrubbing through video controlled by a mouse in some video editing applications is one such example. We argue that this physical aspect of interaction is equally as important as the software aspects described above. To this end, our group has contributed significant efforts into creating iStuff, a prototyping framework for physical interfaces [4]. iStuff provides toolkit support for hardware interfaces akin to what GUI toolkits, such as Java Swing, provide for software interfaces.

Our book on design patterns [6] discusses multiple patterns which support the use of infrared batons in a conducting exhibit as an example: batons are a natural input device for conducting (*domain appropriate devices*); an infrared baton looks intriguing and is thus likely to attract visitors (*innovative appearance*); a baton is simple, and shield the user away from the complex hardware in the underlying implementation (*invisible hardware*); and a single baton to control all modes of interaction reduces confusion for the user (*one input device*).

FUTURE WORK

There are many possibilities for new interfaces which require time manipulation of multimedia. Our work on conducting exhibits is just one area we have been working with. Other opportunities exist for interfaces based on the notion of time stretching – one could imagine an "instrumental karaoke" where the user plays a melody and the system adapts the accompaniment to your tempo and dynamics.

We are continuing to explore new areas of interaction with time-based media. At the Media Computing Group at RWTH Aachen, founded only recently in October 2003, our current flagship project is to build the MediaSpace, a next generation interactive room with large mobile displays and computers. Our focus will be on interactivity with time-based multimedia and the frameworks required to support it, such as extending the iROS system, next generation musical exhibits, interactions with large displays across multiple heterogeneous hardware, and physical user interfaces.

We hope to share our experiences with designing interactive systems for time-based media at this upcoming workshop on time design, and participate in discussion at multiple levels, ranging from high level issues such as interaction metaphors and system architectures to low level issues such as real-time audio time-stretching and system latency. We believe discussion with our peers sharing similar interests will assist us in our ambitious project to build the MediaSpace and initiate future projects in the area of time-based interaction.

ACKNOWLEDGMENTS

The authors would like to thank Rafael Ballagas for providing valuable feedback on this paper.

REFERENCES

1. Google. http://www.google.com.
2. Google image search. http://images.google.com.
3. Apple Computer. *Search Kit Reference*, November 2003. http://developer.apple.com/documentation/UserExperience/Reference/SearchKit/.
4. R. Ballagas, M. Ringel, M. Stone, and J. Borchers. istuff: a physical user interface toolkit for ubiquitous computing environments. In *Proceedings of the conference on Human factors in computing systems*, pages 537–544. ACM Press, 2003.
5. J. Borchers. Worldbeat: Designing a baton-based interface for an interactive music exhibit. CHI, pages 131–138, Atlanta, 1997. ACM.
6. J. Borchers. *A pattern approach to interaction design*. John Wiley & Sons, New York, 2001. http://hcipatterns.org.
7. J. Borchers, E. Lee, W. Samminger, and M. Mühlhäuser. A real-time audio/video system for interactive conducting, 2003.
8. M. Chen. A low-latency lip-synchronized videoconferencing system. In *Proceedings of the conference on Human factors in computing systems*, pages 465–471. ACM Press, 2003.
9. S. Fels, E. Lee, and K. Mase. Techniques for interactive cubism. In *Proceeedings of the ACM Conference on Multimedia*, pages 368–370, October 2000.
10. B. Johanson, A. Fox, and T. Winograd. The interactive workspaces project: Experiences with ubiquitous computing rooms. *IEEE Pervasive Computing*, 1(2):67–74, 2002.
11. J. Johnson. *GUI Bloopers: Don'ts and Do's for Software Developers and Web Designers*. Morgan Kaufmann, 1st edition edition, March 2000.