Critical Point, A Composition for Cello and Computer

Roger Dannenberg
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15217 USA
rbd@cs.cmu.edu

Tomas Laurenzo
Facultad de Ingeniería
Universidad de la República
Herrera y Resissig 565, piso 5
Montevideo, Uruguay 11300
laurenzo@fing.edu.uy

CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.
ACM 978-1-60558-930-5/10/04.

Abstract
Critical Point is written for solo cello and interactive computer music system with two to four channel sound system and computer animation. The cellist plays from a score, and the computer records and transforms the cello sounds in various ways. Graphics and video are also projected. The computer-generated graphics are affected by audio from the live cellist. Critical Point is written in memory of the artist Rob Fisher.

Keywords
Computer Music, Interactive, Performance, Animation, Cello, Multimedia

ACM Classification Keywords
H5.5 [Information interfaces and presentation (e.g., HCI)] Sound and Music Computing; H5.1 [Information interfaces and presentation (e.g., HCI)] Multimedia Information Systems.

General Terms
Design

Introduction

Computer music encompasses many approaches to the composition and performance of music. The work "Critical Point" explores the use of computing to extend the capabilities and sound palette of a traditional acoustic instrument, the cello. The traditional performance setting is also extended by real-time animation and video.

One of the motivations for computing in music is to impose fresh perspectives and constraints on composition and performance. This in turn stimulates creative thinking, leads to new artistic and scientific discoveries, and offers new ways to connect audiences to contemporary art.

This discussion of Critical Point includes some thoughts by the composer and animator on their intentions and processes. We also include some technical details on the realization of Critical Point.

The Music

Critical Point was originally written in 2006 for the U3 festival of university composers in Pittsburgh. Hampton Mallory, who was at the time teaching the composer's son, agreed to perform the piece. The musical material for the cello was sketched, played, and discussed almost weekly over a couple of months. The cello has an extremely wide pitch range, and it seems that cellists can play almost anything, but there are physical limits on what chords are possible. It was very helpful to try different possibilities before actually writing the piece.

The main musical goal is to give the cellist an expanded instrument through computation, capable of new sounds and sonic textures, but at the same time giving as much expressive control as possible to the cellist. To accomplish this, almost all sounds from the computer originate from the live cello. In spite of extreme audio effects, the sounds still carry the imprint of the performer and give quite a range of control, including the overall tempo and pacing, the quality of the sound, the dynamics (loudness) and the overall "shape" of the piece. Of course, the acoustic cello sound is important too, and the performer is called upon to play with a variety of sounds ranging from harsh almost scraping sounds to intensely pure melody.

There are also sections where the cellist is allowed to improvise. For the most part, pitches are provided, but rhythm is optional. This allows the cellist to engage in a kind of call-and-response dialog with the electronics, which echoes processed sounds from the cello. The cellist can wait for just the right moment to continue.

The digital audio effects include a variety of techniques. The most often used processing is a combination of pitch shifting with delay and feedback. This is used to create complex polyphonic textures of cello sound. Sometimes, random algorithms are used to modify the delay and pitch shift amounts rapidly, resulting in even more complexity and variation. Another effect is a vocoder that filters the cello to mimic vowel sounds spoken or sung live by the performer. Some of these vowels are also recorded and applied to the cello when the performer is no longer vocalizing. Finally, there is a prominent section where rising and falling glissandi are recorded and overlapped to create the effect of continuously rising or falling sheets of sound.

The Animation and Video

In 2009, the second author created the animation component of Critical Point for a performance by the Pittsburgh New Music Ensemble (see figure 1). The inspiration for the images came directly from Rob Fisher's work [1].

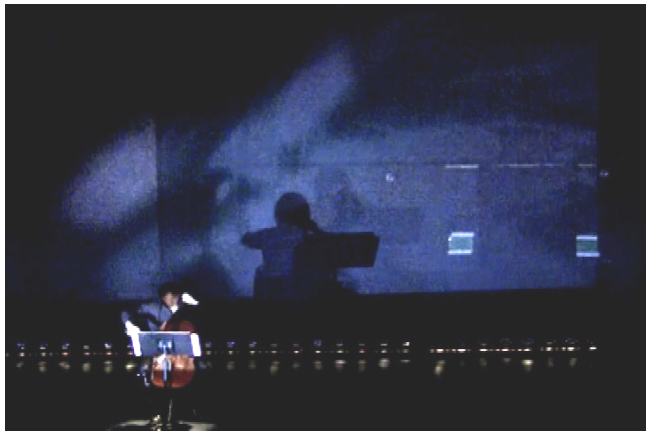

figure 1. A still image from "Critical Point."

The animation mixes reactive graphics generated in real time with short segments of recorded video. The images are intended for projection onto the cellist who performs near the projection screen or surface. The idea is to play with the canvas/performer relationship, and the performer's shadow is a key element of the visual composition.

The animation process is roughly synchronized to the performance so that the animation can coordinate with both the audio effects and the cello music. In addition, the animation receives an audio signal from the cello so that some graphical details are controlled directly by the intensity of sound. As with the audio processing, however, there are random elements applied to the image generation. Thus, the images are partly controlled by the performer and partly independent.

The Implementation

Critical Point is implemented using Aura [2], a software framework for building real-time, interactive music compositions and applications. Aura, in turn, is written in C++ to obtain efficient computation, straightforward extensibility, and to allow debugging and development with standard, mature programming tools. However, the actual programming of Critical Point is divided among (1) low-level audio processing in C++, (2) a graphical programming language for software digital audio patching (part of the Aura system), and (3) Serpent, a real-time scripting language (also a part of the Aura system).

The animation runs on a second computer dedicated to this task. The animation software is implemented in openFrameworks (openframeworks.cc), which is a software framework written in C++.

During the performance, a human (usually the composer) uses a graphical interface to cue various changes in audio and animation processing while following the cellist's progress in the written musical score. On the audio side, these cues change parameters and alter connections between software signal processing modules. On the animation side, the cues control parameters and flags that in turn determine which routines are used to generate each frame of the projected video.

Conclusion

We hope that Critical Point offers an interesting experience to the listener and viewer. We find it interesting that Critical Point takes its form and direction from multiple sources. The composer and animator establish the design of the main musical and visual elements. The computer generates reactive forms algorithmically and stochastically. Finally, the performer is constantly making creative decisions and responding to sound, light, and perhaps even the audience. We believe that accepting and integrating creative input from all of these sources leads to an artistic result with many interconnected layers and structures.

Acknowledgements

We would like to thank previous performers of Critical Point, including Jason Duckles, Norbert Lewandowski, Richard Dannenberg, and especially Hampton Mallory for their dedication to this project. Thanks also to the Universidad de la República, Uruguay, for financing travel that made our collaboration possible.

References

[1] Rob Fisher Sculpture, LLC. http://www.sculpture.org/portfolio/sculptorPage.php?sculptor_id=1000080

[2] Dannenberg, R.B. A Language for Interactive Audio Applications. In *Proc. of the 2002 Intl. Comp. Music. Conf.* Intl. Comp. Music Assn. (2002).