ChucK, On-the-fly Programming, and the Audicle

Ge Wang, Perry R. Cook, and Ananya Misra
Princeton University*

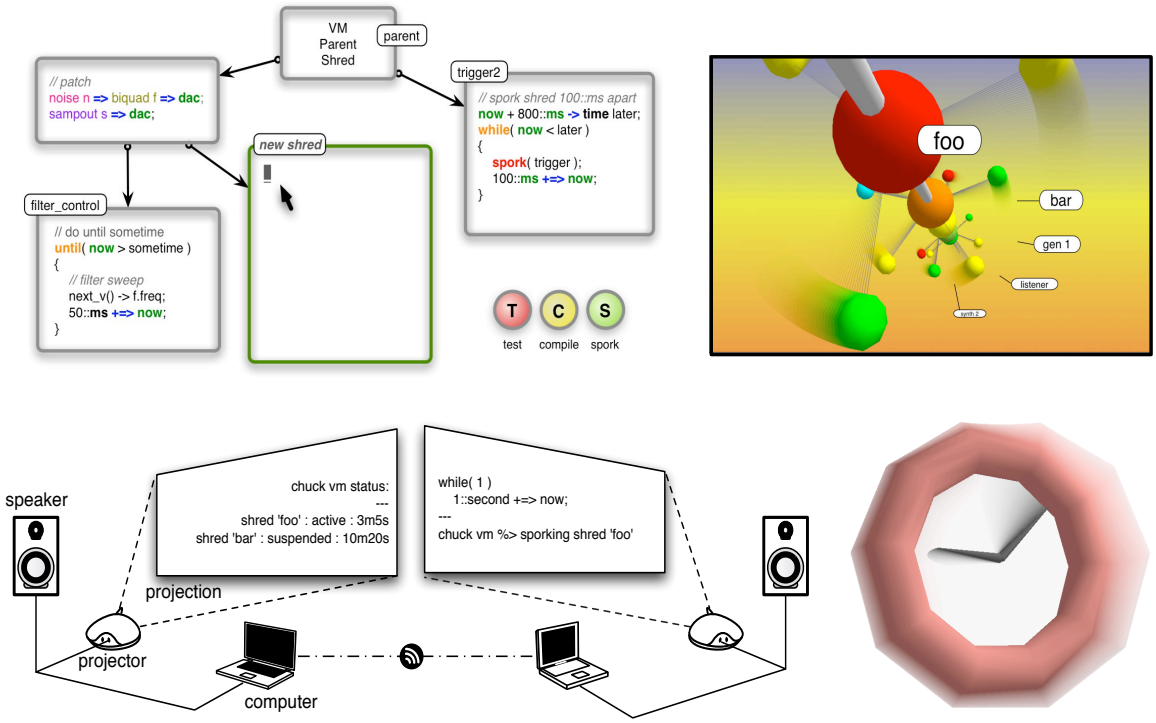

Figure 1: *Elements in the ChucK programming language and the Audicle programming environment.*

1 Introduction

ChucK, On-the-fly programming, and the Audicle are different parts of a new programming system and paradigm for real-time audio synthesis and multimedia. ChucK is the programming language; on-the-fly programming is the technique and aesthetic of writing/editing code during runtime; the Audicle is the context-sensitive graphical programming environment. Together, they form a theoretical and practical framework for writing and experimenting with complex audio/multimedia programs that (1) have powerful control over time and concurrency and (2) can be created on-the-fly. (see also: `http://chuck.cs.princeton.edu/`)

2 ChucK + Audicle

ChucK is an ongoing, open-source research experiment in designing a computer music language from the "ground up". A main focus of the design was the precise programmability of time and concurrency with high level of readability. System throughput remains an important consideration, especially for real-time audio, but was not our top priority. The language is designed to provide maximal control for the programmer. Design goals are as follows.

- **Flexibility**: allow the programmer to naturally specify both high and low level operations in time.
- **Concurrency**: allow the programmer to craft and precisely synchronize parallel modules that share both data and time.
- **Readability**: provide/maintain a strong correspondence between code structure and timing.
- **A do-it-yourself language**: combine the expressiveness of lower-level languages and the ease of high-level computer music languages; support high-level musical concepts, precise low-level timing, and the creation of "white-box" signal-processing elements—all directly in the language.
- **On-the-fly**: allow programs to be edited as they run.

ChucK enables time itself to be computable (as a first-class citizen of the language), and allows a program to be "self-aware" in the sense that it always knows where it is in time and can control its own progress over time. Furthermore, many processes can share a central notion of time, making it possible to naturally reason about parallel code based on time.

However, control over time alone is not enough—we also need concurrency to expressively capture parallelism. Fortunately, the timing mechanism lends itself directly to a concurrent programming model. Multiple processes (called *shreds*), each advancing time in its own manner, can be synchronized and serialized directly from the timing information. Thus arises our concept of a *strongly-timed language*, in which processes have precise control over their own timing and synchronization.

Using this timing/concurrency model, on-the-fly programming can be carried out by exchanging time-aware code segments. To further facilitate this, the Audicle provides a graphical environment in which to write ChucK programs on-the-fly, and to visualize the programs in terms of code, audio synthesis, concurrency, and timing—all in real-time. Together, ChucK, on-the-fly programming, and the Audicle form a system and workbench for experimenting with sound synthesis for composition and performance.

*e-mail: {gewang, prc, amisra}@cs.princeton.edu