CHIME One Day Workshop 2023

The Open University, Milton Keynes
4 December 2023

TimeLines: Time, Paradigms, Forgotten Histories & Imagined Futures of Live Coding

Dimitris Kyriakoudis[1]

[1] Experimental Music Technologies Lab,
University of Sussex, UK,
d.kyriakoudis@sussex.ac.uk

***Abstract*—We look at the current state of affairs regarding the paradigms of computational systems we design and how we interact with them. We then look back in recent history for inspiration, exploring fundamentally different and often forgotten system architectures and paradigms of interaction, examining their potential benefits over those that passed the filter of widespread popular adoption. Lastly, we imagine alternate presents and futures of real time computation-based systems that are meant to be interacted with while exhibiting a time-varying behaviour, with a focus on Live Coding Musical Instruments (LCMIs). TimeLines, an on-going research project in LCMI design & implementation, is presented, utilising techniques such as linguistic extensibility via homoiconic metaprogramming, an explicitly symbolic and static encoding of time and time-varying behaviour, projectional and structural editing, and logic programming.**

I. Introduction

A Live Coding Musical Instrument (LCMI) is fundamentally a computational instrument. The real-time and autotelic natures of LCMIs present their own unique sets of requirements and challenges when it comes to system design and technical implementation, and they are often some of the first technologies to be informed by, apply, and build upon theoretical advances and explorations in the wider field of Computational Informatics (CI). This talk presents TimeLines, an ongoing research project on LCMI design, implementation, and practice, which looks back into the less-remembered recent past of CI to draw inspiration and imagine alternate presents and possible futures of designs and interactions with LCMIs, both on a software and hardware level; while arguing that the former greatly influences and informs the latter.

II. The Programmable Universal Computer

Modern programmable digital computers are, much like Alan Turing's vision of a Programmable Universal Computer (PUC), blank-slate machines whose sole function is to receive an encoded description of any information system and shapehift into assuming its behaviour and functionalities. As such, it becomes clear that any and all interactions with such a computer are inevitably mediated through the abstract design and concrete representations of the systems and behaviours we project onto them.

III. The Affordance Stack

The ways in which we interact with modern digital computers have been shaped by decades of research, industry, and commercial practices in hardware and software. At every step in their practice, the live coder interacts with a towering stack of assumptions, design choices, limitations, and affordances that propagate (or get obscured) through the stack's various levels: The hardware architectures of the chips that we're ultimately programming, the design choices of the programming languages we program them in, the programs we build to write and interact with code in those languages, the frameworks we build with that code, the programs we build around those frameworks, the interfaces we design to interact with those programs etc.

At any moment, the range and characteristics of all possible interactions with such an instrument, its ergonomics (the "how it works with & against the human body and mind") and ergodynamics (the "how it feels to interact with") are inevitably shaped by its *ergologics*, the concrete specifics of its operations throughout the stack and their in-between interactions. TimeLines explores the tensions between those three aspects of a LCMI, examining the stack and questioning as many of its parts as is realistically possible.

IV. Old New Paradigms

TimeLines explores the use and potential application of a number of computational paradigms that appeared throughout the 20th century (but have not enjoyed widespread use and popular adoption), including:

- Homoiconic Metaprogramming.
- Symbolic Computation.
- Explicit and reactive modeling of time and time-varying behaviours.
- Graphical, projectional, and structural editing.
- Declarative, data-oriented approaches towards extensible application development.
- Logic programming and knowledge-based ("expert") systems.

[1]