FARM 2017 Demo Summary

Jean Bresson
UMR STMS: IRCAM/CNRS/UPMC Sorbonne Universités
Paris, France
jean.bresson@ircam.fr

Michael Sperber
Active Group GmbH
Tübingen, Germany
sperber@deinprogramm.de

Abstract

This is a summary of the demonstrations presented at the 5th ACM SIGPLAN International Workshop on Functional Art, Music, Modelling and Design (FARM).

CCS Concepts • **Software and its engineering → Functional languages**; • **Applied computing → Arts and humanities**;

Keywords functional programming, art, music, graphics

ACM Reference Format:
Jean Bresson and Michael Sperber. 2017. FARM 2017 Demo Summary. In *Proceedings of 5th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling and Design, Oxford, UK, September 9, 2017 (FARM'17),* 1 pages.
https://doi.org/10.1145/3122938.3122945

Introduction

Six demonstrations were presented at the 5th ACM SIGPLAN International Workshop on Functional Art, Music, Modelling and Design (FARM). They range from applications of functional programming in music and sound, to more general purpose creative programming, and to 3D graphics and animation. All extended abstracts are available on the FARM 2017 web site at http://functional-art.org/2017/.

1 Representation of Musical Notation in Haskell

In this demonstration Edward Lilley (Institute of Astronomy, University of Cambridge, UK) introduces a formal representation of traditional musical notation suitable for computers. An algebra of musical intervals is found, which is closed with respect to interval addition. This makes it possible to consistently apply two families of tuning systems: syntonic temperaments and equal temperaments. A Haskell program implements this musical representation, with audio output in Csound. This program represents musical phrases internally using a RoseTree data structure, and taking advantage of this a parser for Lilypond files (exclusively using parser combinators from the Parsec library) is demonstrated.

2 The Arpeggigon: A Functional Reactive Musical Automaton

Henrik Nilsson from the University of Nottingham (UK) presents the *Arpeggigon*: an interactive cellular automaton for composing groove-based music. The *Arpeggigon* is implemented in Haskell using Functional Reactive Programming (FRP). It is based on a hexagonal grid laid out as a harmonic table where each of the six possible directions corresponds to a given musical interval. The automaton is configured by placing different types of tokens on the grid: when it runs, these tokens interact with one or more playheads moving around the grid, resulting in notes corresponding to the positions of the involved tokens to be played.

3 Vivid: Sound Synthesis with Haskell and SuperCollider

Thomas Murphy (USA) presents a Haskell library for music creation and sound synthesis, coupled with the SuperCollider synthesis engine. *Vivid* allows live-coding performance in Haskell, embedding advanced musical notions such as timing (at the musical and audio sample levels), parallelism, audio graphs, multichannel input and output, etc.

4 African Polyphony and Polyrhythm

Chris Ford (ThoughtWorks, UK) proposes a programmatic approach to non-Western music analysis and representation, using functional programming to revisit ethnomusicologist Simha Arom's work and insights on Central African polyphony. The demonstration takes examples notated in Arom's book *African Polyphony and Polyrhythm*, and renders them in Lisp using the Leipzig composition library for Clojure.

5 Ait: A Concatenative Language for Creative Programming

Stian Veum Møllersen (Bekk Consulting AS, Norway) presents the concatenative language *Ait*. In concatenative languages, composing functions is implemented as the simple concatenation of words. Møllersen demonstrates how this programming style fits the idea of creative programming. *Ait* inherits the main features of previous concatenative languages such as Forth and Joy. It is implemented in JavaScript with bindings to browser APIs, and mainly targets web programming and applications.

6 Octopus: A High-Level Fast 3D Animation Language

David Janin and Simon Archipoff (LaBRI, University of Bordeaux, Bordeaux INP, France) complement a paper presentation on a new algebraic approach to media and 3D graphics programming with this demonstration of the *Octopus* language. *Octopus* is an EDSL (embedded domain-specific language) extending Haskell with an API for the definition of procedural 3D animations. Conceptually the authors define their language as deriving from the classic LOGO with features extended to 3D graphics and animation (that is, including a time dimension). *Octopus* programs generate small abstract drawing specifications on the CPU that are sent to the OpenGL graphic pipeline for a fast expansion and rendering on the GPU.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
FARM'17, September 9, 2017, Oxford, UK
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5180-5/17/09…$15.00
https://doi.org/10.1145/3122938.3122945