Computermusik und Multimedia

lOhannes m zmölnig

Institut für Elektronische Musik und Akustik

March 2, 2015

Languages of (Our) Time

Programs must be written for people to read, and only incidentally for machines to execute.
(Hal Abelson: Structure and Interpretation of Computer Programs)

Thema: Languages of (Our) Time

- Aufgabe: Entwurf/Implementierung einer Mini-Computermusiksprache
- Domänenspezifische Sprachen
  - Spezielle Probleme: z.B. Abbildung von *Loops*, *Polyrhythmik*,...
  - Echtzeitverarbeitung
  - Live-Programming
  - unübliche Interfaces
  - ...

Programmieren

- Programmieren = Problemlösen
- Kriterien
  - Korrektheit
  - Geschwindigkeit
  - Wartbarkeit
  - ...

Kriterien für Programmiersprachen I

„Korrektheit"

benötigt klare Problemstellung

- Harangehensweisen
  - bottom-up
    - spezifische Lösung für bestimmtes „Stück"
  - top-down
    - generische Lösung (z.B. *Kompositionsmaschine*)
    - praktisch unmöglich
    - → *Frameworks*

Kriterien für Programmiersprachen II

Wartbarkeit

- wichtig bei Frameworks
- weniger wichtig bei problemspezifischen ad-hoc Lösungen (bottom-up)
  - andererseits: Probleme der Archivierbarkeit

Geschwindigkeit

wichtig für Echtzeit-Sprachen

Kriterien für Computermusik-Sprachen

- Universelle Programmiersprache
  - Formales Kriterium: Turing-Vollständigkeit
- Domäne: Musik
  - Umgang mit Zeit
  - ...
- Interface

Turing-Completeness

The Turing machine has a tape, which consists of a infinite sequence of positions each of which contains a single symbol. The machine has a current state, which is initially q_0, and a current position, which is initially the leftmost position on the tape with the tape extending infinitely to the right. All tape positions initially contain σ_0.
The machine repeatedly consults the transition table δ. If δ is not defined for the current state and the symbol at the current position, then the machine halts. If it is defined as (σ', d', q'), then the current state is set to q', the symbol at the current position is set to σ', and the current position moves leftward one place (if d'=L) or rightward one place (if d'=R), and the machine continues.

Einfacher Scheduling-Algorithmus

1. inkrementiere logische Zeit
2. verarbeite alle angefallenen Events in Liste
   - wenn Event-Zeitpunkt vergangen, dann:
     - setze logische Zeit auf Event-Zeitpunkt
     - rufe Event-Handler auf
     - lösche Event aus Liste
3. verarbeite 1 Audio-Block (z.B. 64ms $\widehat{=}$ 1.45ms)
4. warte bis zum nächsten Audio-Block ($<$ 1.45ms)
   - füge anfallende Events in Liste ein
5. gehe zu 1

Interface I

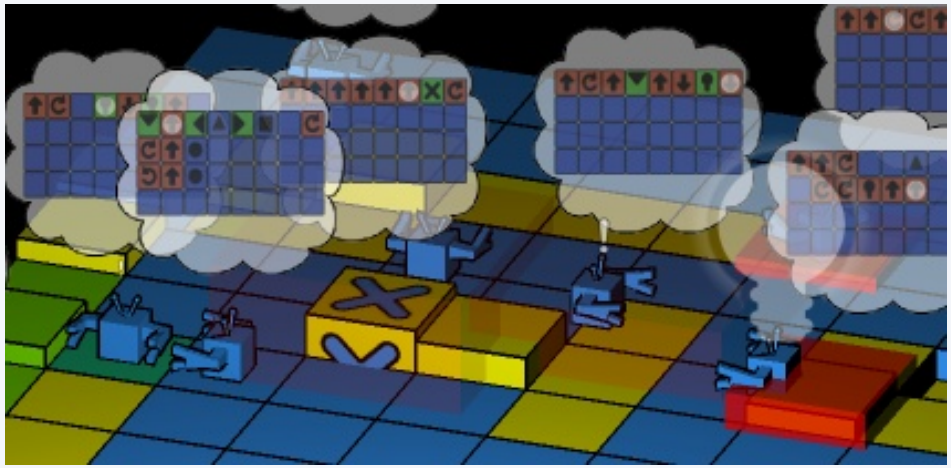
Wie interagiert man mit der Sprache?

- Maschinensprache
- menschliche Sprache
- visuelle Codes
- ...

Interface II

Piet (David Morgan-Mar)

Al-Jazari (Dave Griffiths)



Mini-Sprachen

- Konzentration of ein kleines Problemfeld, z.B.
  - Strukturerzeugung für Loops
  - Ausgabe von Events (per OSC um Synth zu steuern)
  - ...
- auf vorhandene (nicht-CM) Sprache aufsetzen
- oder sehr einfache Syntax („esoterische Sprache")

turingmächtig!

- Implementierung als Proof-of-Concept

LV-Anforderungen

- Abstract + Hintergrund-Recherche
- Realisierung einer *Computermusik Sprache*
  - Mini-Sprache (Proof-of-Concept)
- Präsentation
  - 30 Min. (inkl. Demonstration)
- schriftliche Arbeit
  - Motivation/Hintergrund/Referenzsprachen
  - Beschreibung der Sprache
    - Turing-Vollständigkeit
    -
  - Beispielprogramm
- *Bachelorarbeit (∼40 S.), Seminararbeit (∼20 S.)*

Termine

Datum	Uhrzeit	Inhalt	Anmerkung
2.3	14:00	Einführung	
...	...	Themenfindung	*Heimarbeit*
13.4	14:00-	Hackshop	
27.4	14:00-	Hackshop	
11.5	14:00-	Hackshop	
...	...	Ausarbeitung	*Heimarbeit*
15.6&22.6	14:00-15:30	Präsentationen	

Themenfindung

- Recherche
  - musikalische Problemstellung
    - Realtime vs Non-realtime
    - Domain (Klangsynthese, Kontrollstrukturen, . . .)
    - Interface
  - ähnliche Sprachen
  - Texte, Videos
- Kurzpräsentation
- Abstract (∼120 W.)
- Quellenverzeichnis
  - Referenzprojekte, Videos, Texte (als URLs)

Hausaufgabe

- `https://iaem.at/kurse/ss15/cmmm/workspace/`
- login mit `s+<matrikelnummer>` und KUG-passwort
- *Eigenen* Ordner anlegen: `<nachname>-<vorname>`
- vor dem nächsten Termin abstract+quellenverzeichnis hochladen

Abstract

Dieses Seminar beschäftigt sich mit "Languages of (Our) Time", also dem Entwurf von Programmiersprachen die mit Musikalischen Strukturen (wie Zeit) arbeiten, d.h. Computermusiksprachen. Als Computermusiksprachen werden domänenspezifische Programmiersprachen bezeichnet, mit deren Hilfe Strukturgeneratoren und/oder Klangsynthese-Algorithmen möglichst einfach implementiert werden können. Indem nur ein kleiner Problembereich abgedeckt wird, lassen sich minimale Programmiersprachen entwerfen. Dabei können entweder bestehende Programmiersprachen um Computermusik-Funktionalität erweitert werden, oder alternative Sprachkonzepte ausprobiert werden (z.B. graphische Programmierung oder Anleihen aus dem Gebiet der *Esoterischen Programmiersprachen*). Die Sprachen sollten dabei neben der domänen-spezifischen Komponente auch Turing-Vollständigkeit besitzen um (zumindest theoretisch) universell einsetzbar zu sein.
Im Rahmen des Seminars konzipieren die Studierenden eine eigene Sprache und implementieren sie in einem Proof-of-Concept.

Literatur

- Esoterische Programmiersprachen, `http://en.wikipedia.org/wiki/Esoteric_programming_language`
- Pure Data: another integrated computer music environment (Miller Puckette, 1996)
- SuperCollider: a new real time synthesis language (James McCartney, 1996)