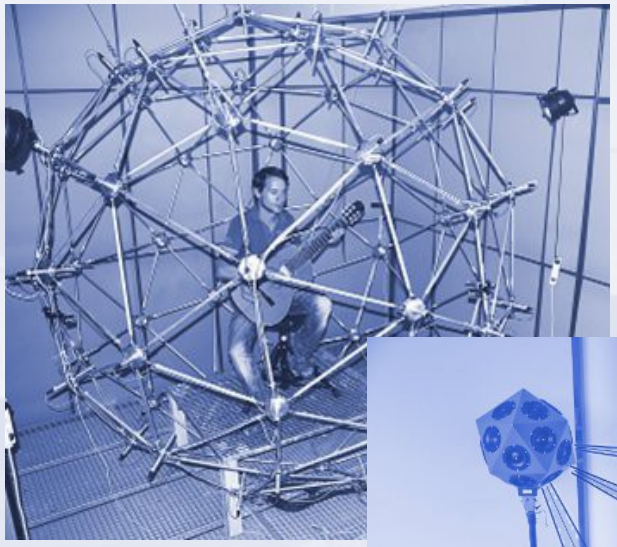
Implementierung von Algorithmen
Pure Data: Messages



IOhannes m zmölnig

Pd Objekte

- Funktionaler Ansatz
  - Objekte manipulieren Daten: „Funktionen“
  - Eingangsdaten → Ausgangsdaten

  - Dataflow
    - Objekte(Funktionen) werden nicht nacheinander abgerufen

      sondern
    - Daten „fließen“ durch Programm und werden modifiziert

Pd Messages

- Daten
- **flüchtig**: existieren nur zum Zeit*PUNKT* des Auftretens
- asynchron/on demand
  - user/patch bestimmt Zeitpunkt des Auftretens
- MessageBox
  - „eingefrorene Message“
  - wird „aufgetaut“ durch
    - Klicken
    - Message, die an MessageBox geschickt wird

Pd Objekte

- Objektorientierter Ansatz
  - Objekte haben **inneren Zustand**
    - von Außen nicht sichtbar
  - Methoden
    - interagieren mit innerem Zustand
- „Klasse“: Abstrakte Idee
  - „Zahlenspeicher“
- „Objekt“: Instanz einer Klasse

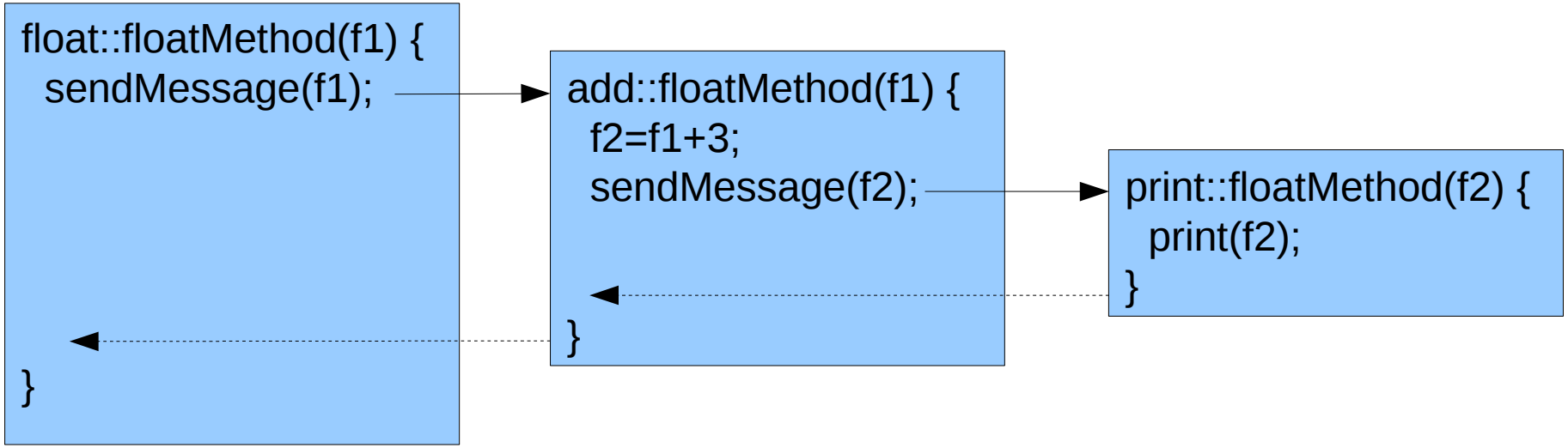
Pd Objekte: Methoden

- Messages rufen Methoden auf
- Objekt bestimmt, was mit Message (Daten) passiert
  - z.B. ob Ausgangsdaten produziert werden

  - MyObject::fooMethod(input) {
    output=function(input);
    sendMessage(output);
    }

Pd Objekte: Methoden

- rekursives Abarbeiten



- Pd-Dispatcher „ ⟶ “
  - nimmt Message auf Kanal entgegen
  - schickt Message an alle Objekte, die an diesem Kanal „hören“
- Messages bestimmen, welche Methode aufgerufen wird

Messages: Aufbau

- <selector> {<atom>}

- <atom>
  - Zahl (floating point)
  - Symbol (String in Hashtable)
  - Zeiger
- <selector>
  - Symbol

  - **jede** Message hat einen Selector
    - falls nicht explizit angegeben, wird automatisch „list“ (bzw. „float“) hinzugefügt

Atome

- float
  - alle Zahlen in Pd:
    - Single Precision (IEEE 32bit float)
    - Integers nur bis ±16777216
- symbol
  - Hashtable
    - jeder „string“ wird in einer Tabelle gespeichert, und fortan nur noch indiziert
    - Tabelle wächst!

Messages

- flüchtig!
  - Messages werden vom System on-demand erzeugt und wieder zerstört
  - malloc()/free()

Spezielle Message-Selektoren

- **bang**
  - Trigger!
  - immer nur <selector>, keine daten
- **float**
  - immer genau ein <atom> vom typ **number**
- **symbol**
  - immer genau ein <atom> vom typ **symbol**
- **list**
  - allgemeine Daten

Dispatcher

- `<selector>` einer Message wählt Methode eines Objektes aus
  - Objekte melden Methoden für bestimmte `<selector>`en an
  - Evtl. catch-all Methode
- Message an Objekt
  - gibt’s Methode für `<selector>` → aufrufen
  - sonst „catch-all“
- Implizite Konvertierung
  - „float <f>“ ↔ „list <f>“
  - „bang“ ↔ „list“
  - „symbol <s>“ ↔ „list <s>“

Pd-Objectmaker

- auch „Objekte“ werden als Message erzeugt:
  - <selector>: Objektname
  - {<atom>}: Übergabeargument
- der *objectmaker* hat eine Methode für jede bekannte Klasse
- catch-all (unbekannte Klassen)
  - versucht von Festplatte Klassen nachzuladen