An Introduction to the Synth-A-Modeler Compiler

Modular and Open-Source Sound Synthesis using Physical Models

Edgar Berdahl and Julius O. Smith III
INTRODUCTION

FAUST PROGRAMMING LANGUAGE

SYNTH-A-MODELER

MORE EXAMPLES

FINAL WORDS
Digital Sound Synthesis
Digital Sound Synthesis

- We can create any perceivable sound using digital sound synthesis.
Digital Sound Synthesis

- We can create any perceivable sound using digital sound synthesis.

- Almost all sounds created by computers are either not interesting, ugly, unpleasant, painful or dangerous.
Digital Sound Synthesis

• We can create any perceivable sound using digital sound synthesis.

• Almost all sounds created by computers are either not interesting, ugly, unpleasant, painful or dangerous.

• How do we obtain the specific sounds that we want?
Physical Modeling

```c
main{
...
...
}
```
Physical Modeling

- thinking of physical modeling as a creative activity or a design process
Requirements

• produce efficient and modern DSP code for real-time applications
• free and open-source
• combine digital waveguide, mass-interaction, and modal synthesis
• easy to quickly design a large number of models
• easy to extend and modify framework
• platform for pedagogical exploration of mechanics and dynamics
• accessible to artists who may have limited technical experience
• enable the development of MIDI-based synthesizers
• compatible with programming haptic force-feedback systems
• accessible from as many sound synthesis host environments as possible
INTRODUCTION

FAUST PROGRAMMING LANGUAGE

SYNTH-A-MODELER

MORE EXAMPLES

FINAL WORDS
Faust Dataflow Summary

Faust .DSP file

- Pd external
- Max/MSP external
- SuperCollider external
- VST plug-in...
- Custom host

Faust compiler
Functional AUDIO STREAM

\[
spring(k) = (_, _) : - : *(k) : _ <: (*(-1.0), _);
\]

process = spring(100.0);
Functional Audio Stream

\[ \text{spring}(k) = (_, _) : - : * (k) : _ <: (*(-1.0), _) ; \]

process = \text{spring}(100.0) ;

But FAUST signal flow is primarily left to right
INTRODUCTION

FAUST PROGRAMMING LANGUAGE

SYNTH-A-MODELER

MORE EXAMPLES

FINAL WORDS
Synthesizing A Model
Elements

Link-like elements

Mass-like elements

Wave variable elements
Example Model: Play A Resonator
Example Model: Play A Resonator

```plaintext
mass(0.001),m1,();
ground(0.0),g,();
port( ),dev1,();

link(4200.0,0.001),ll,m1,g,();
touch(1000.0,0.03,0.0),tt,m1,dev1,();

audioout,a1,m1,1000.0;
```
Example Model: Play A Resonator
Example Model: Play A Resonator

```plaintext
import("physicalmodeling.lib");
bigBlock(mlp,gp,dev1p) = (ml,g,dev1,a1) with {
    // Link-like objects:
    ll = (mlp - gp) : link(4200.0,0.001,0.0);
    tt = (mlp - dev1p) : touch(1000.0,0.03,0.0);
    // Mass-like objects:
    ml = (0.0-ll-tt) : mass(0.001);
    g = (0.0+ll) : ground(0.0);
    dev1 = (0.0+tt);
    // Additional audio output
    a1 = 0.0+ml*(1000.0);
}
process = (bigBlock)~(_,_):(!,!,_,_);
```
Example Model: Play A Resonator

```plaintext
import("physicalmodeling.lib");

bigBlock(m1p,gp,dev1p) = (m1,g,dev1,a1) with {
    // Link-like objects:
    ll = (m1p - gp) : link(4200.0,0.001,0.0);
    tt = (m1p - dev1p) : touch(1000.0,0.03,0.0);

    // Mass-like objects:
    m1 = (0.0-ll-tt) : mass(0.001);
    g = (0.0+ll) : ground(0.0);
    dev1 = (0.0+tt);

    // Additional audio output
    a1 = 0.0+m1*(1000.0);
}

process = (bigBlock)~(_,_):(!,!,_,_);
```

Example Model: Play A Resonator
INTRODUCTION

FAUST PROGRAMMING LANGUAGE

SYNTH-A-MODELER

MORE EXAMPLES

FINAL WORDS
Touching Interpolatable Resonators

Pluck A String
INTRODUCTION

FAUST PROGRAMMING LANGUAGE

SYNTH-A-MODELER

MORE EXAMPLES

FINAL WORDS
Thanks!

- To Alexandros Kontogeorgakopoulos, Yann Orlarey, and to other researchers in physical modeling who have inspired us very much.
Thanks!

- It’s not a synthesizer, it’s a **Synth-A-Modeler**