CSOUND-EXPRESSION

БИБЛИОТЕКА ДЛЯ СОЗДАНИЯ ЭЛЕКТРОННОЙ МУЗЫКИ НА HASKELL

Автор Антон Холомьёв

ПРИВЕТ!

csound-expression это:

- EDSL для создания электронной музыки и синтезаторов
- генератор кода Csound

CSOUND

текстовый синтезатор (1985, MIT -> Maynooth)

- написан на C (запускается на всём подряд)
- работает в режиме реального времени
- много аудио юнитов
- очень хорошая документация/сообщество
- открытость (MIDI, OSC)
- язык очень древний, с древним синтаксисом

CSOUND = ТЕКСТ + МУЗЫКА

ФП?

Текст = Можно генерировать из Haskell

EDSL - ФУНКЦИИ И ЗНАЧЕНИЯ

Выразить концепции самыми простыми средствами Haskell

```
type Инструмент = Нота -> Сигнал

игратьПартию: Инструмент -> Партия -> Сигнал
```

ВСЁ - ЕСТЬ ВЫРАЖЕНИЕ

Everything is an expression

ВСЁ ЕСТЬ ВЫРАЖЕНИЕ: ВЫЗОВ ИНСТРУМЕНТА

Сигнал, полученный из инструмента, может стать частью другого инструмента

- Инструмент & Нота == Сигнал
- Инструмент & Поток нот == Сигнал
- UI == Визуальное представление & Сигнал

ВСЁ ЕСТЬ ВЫРАЖЕНИЕ: ИНТЕРФЕЙСЫ

```
UI -> (Картинка, Сигнал)

UI -> (Картинка, Поток)
```

ПРОЦЕСС

- Пользователь
- =>
  Бибилотека
- => AST
- => Текст .csd
- => Csound
- => Звук

СОЗДАНИЕ МУЗЫКИ В GHCI

```
$ ghci
> :m +Csound.Base
> let freq = triSeq [1, 0.5, 0.2, 0.9, 0.5, 0.2, 1, 0.5] 4
> dac $ mlp (1500 * freq) 0.1 $ saw 55
```

ОСНОВЫ СИНТЕЗА

- Формы волны (osc, saw, tri, sq, pw)
- Огибающие (linseg, expseg, ...)
- Фильтры (lp, hp, bp, br, mlp, ...)
- Эффекты (delay, reverb, distortion, ...)

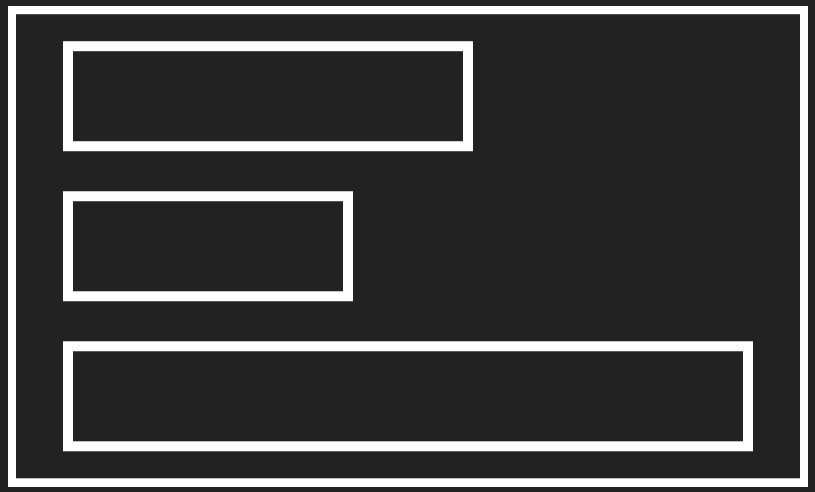
ПРОДВИНУТЫЕ ВАРИАНТЫ СИНТЕЗА

- Гранулярный синтез
- Спектральные эффекты (Фурье)
- Пошаговый секвенсор
- Преобразование сэмплов (независимое изменение темпа и частоты)
- ...

МУЗКАЛЬНАЯ АЛГЕБРА

mel (мелодия):

har (гармония):



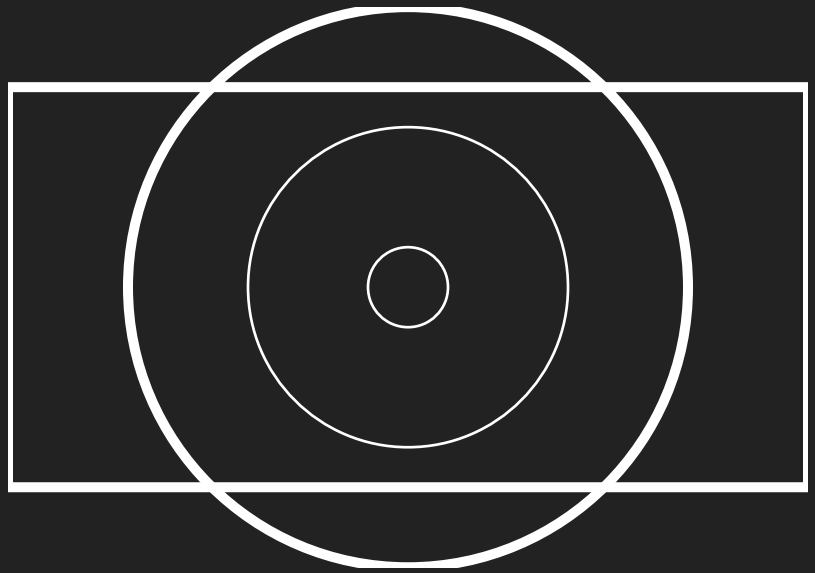
МУЗКАЛЬНАЯ АЛГЕБРА

del:

rest:

МУЗКАЛЬНАЯ АЛГЕБРА

loop:



СЭМПЛЫ

```
> let f1 = wav "file1.wav"
> let f2 = wav "file2.wav"
> let g  = mel [f1, f2]
```

НОТЫ

```
> let f1 = temp 440
> let f2 = temp 220
> let g  = mel [f1, f2]
```

ПОТОКИ

```
type Evt a = (a -> SE a) -> SE ()

instance Monoid (Evt a) where

instance Functor Evt

filterE, accumE
```

ПОТОКИ

```
cycleE [1, 2, 3]

oneOf [440, 220, 330]

appendE 0 (+) . fmap (const 1)
```

UI

Source

```
(Gui, SE a)
```

Sink

```
(Gui, a -> SE ())
```

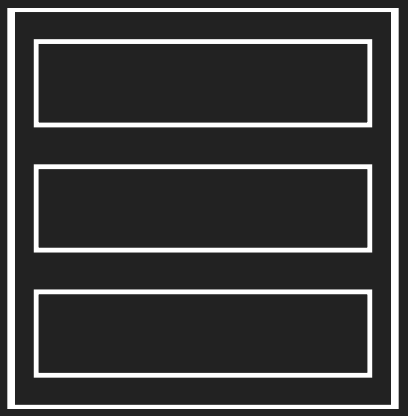
Display

```
(Gui, SE ())
```

АЛГЕБРА GUI

hor:

ver:



АЛГЕБРА GUI

space:

sca:

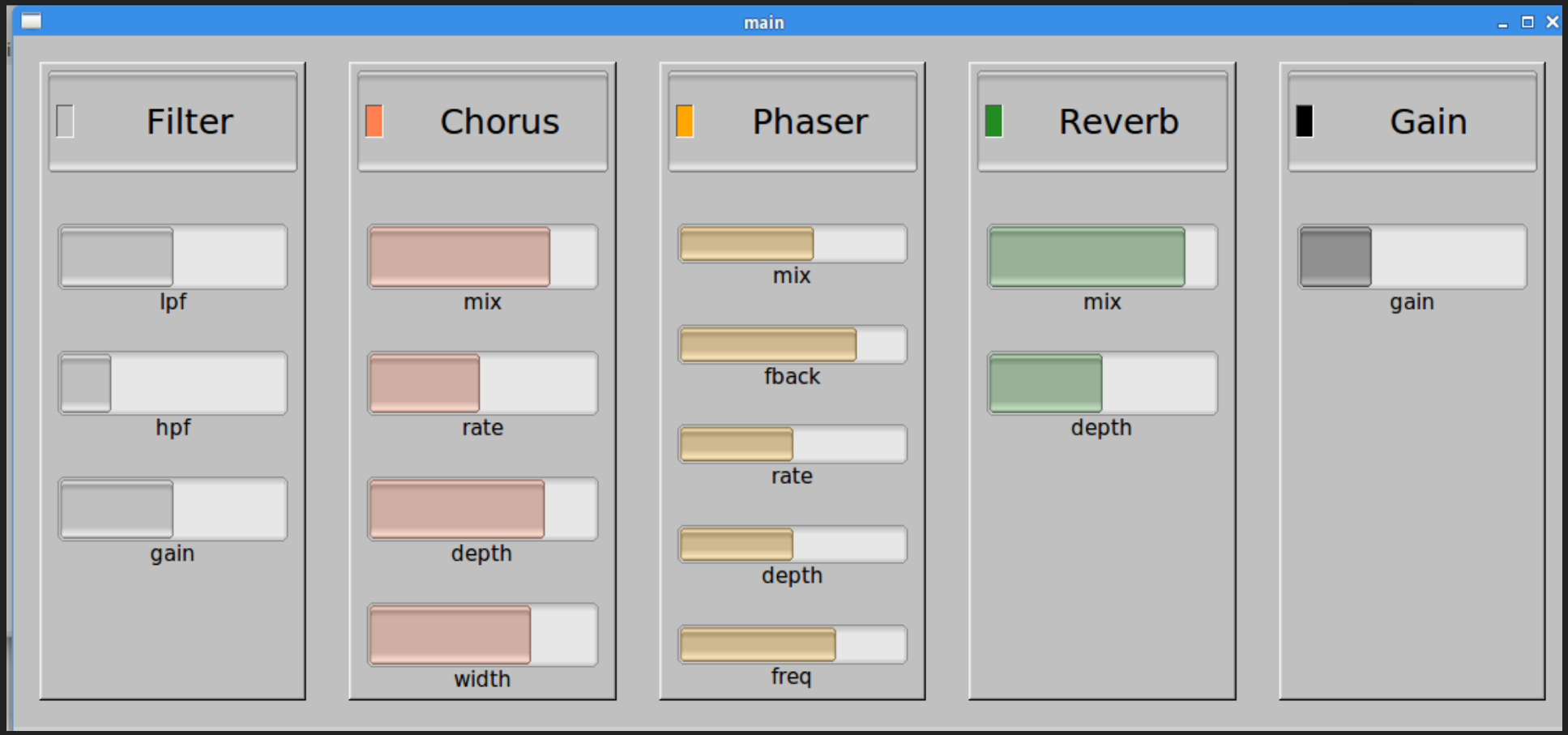
UI В АППЛИКАТИВНОМ СТИЛЕ

```
lift1 :: (a -> b) -> Source a -> Source b

hlift2, vlift2 :: (a -> b -> c) -> Source a -> Source b -> Source c

hlifts, vlifts :: ([a] -> b) -> [Source a] -> Source b
```

ПРИМОЧКИ


main
Filter
lpf
hpf
gain
Chorus
mix
rate
depth
width
Phaser
mix
fback
rate
depth
freq
Reverb
mix
depth
Gain
gain

ПРИМОЧКИ: КОД

```
module Fx where

import Csound.Base

main = dac $ lift1 (\fx -> fx $ fromMono $ saw 110) $ fxHor
               [ uiFilter False 0.5 0.5 0.5
               , uiChorus False 0.5 0.5 0.5 0.5
               , uiPhaser False 0.5 0.5 0.5 0.5 0.5
               , uiReverb True  0.5 0.5
               , uiGain   True  0.5 ]
```

ПОРТИРУЕМОСТЬ

пишем на Haskell, запускаем везде где работает Csound

(Linux, PC, OSX, Raspberry Pi, Android, iOS)

MIDI

```
midi :: Инструмент -> SE Сигнал
```

Пример

```
> vdac $ mul 0.5 $ at smallHall
       $ midi $ onMsg $ onCps (mlp 1500 0.1 . saw)
```

СПАСИБО ЗА ВНИМАНИЕ

Github | Hackage | soundcloud