

# SOUND DESIGN WITH HASKELL

# CSOUND-EXPRESSION

Haskell библиотека для музыки

- Генерирует Csound-код из Haskell-кода
- Сохраняет код в файле tmp.csd
- Создаёт звук вызовом Csound с файлом tmp.csd

# INSTALL

```
-- Установим Csound
```

```
> sudo apt-get install csound
```

```
-- Установим хаскель библиотеку
```

```
> cabal install csound-expression
```

# HELLO SOUND

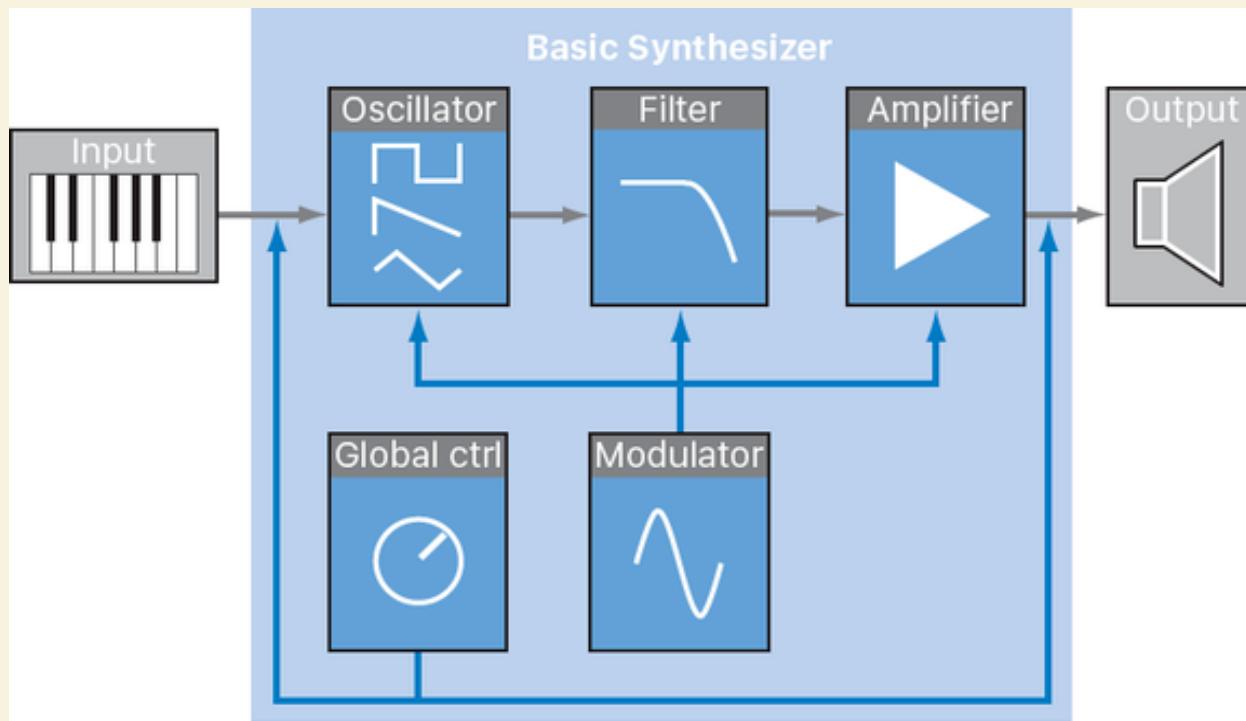
```
ghci
> import Csound.Base
> dac (0.5 * osc 440)
-- Ctrl+C - to stop audio
```

## ФУНКЦИИ

```
-- Послать поток аудио в колонки
-- (DAC - Digital to Analog Converter)
dac :: RenderCsd a => a -> IO ()
```

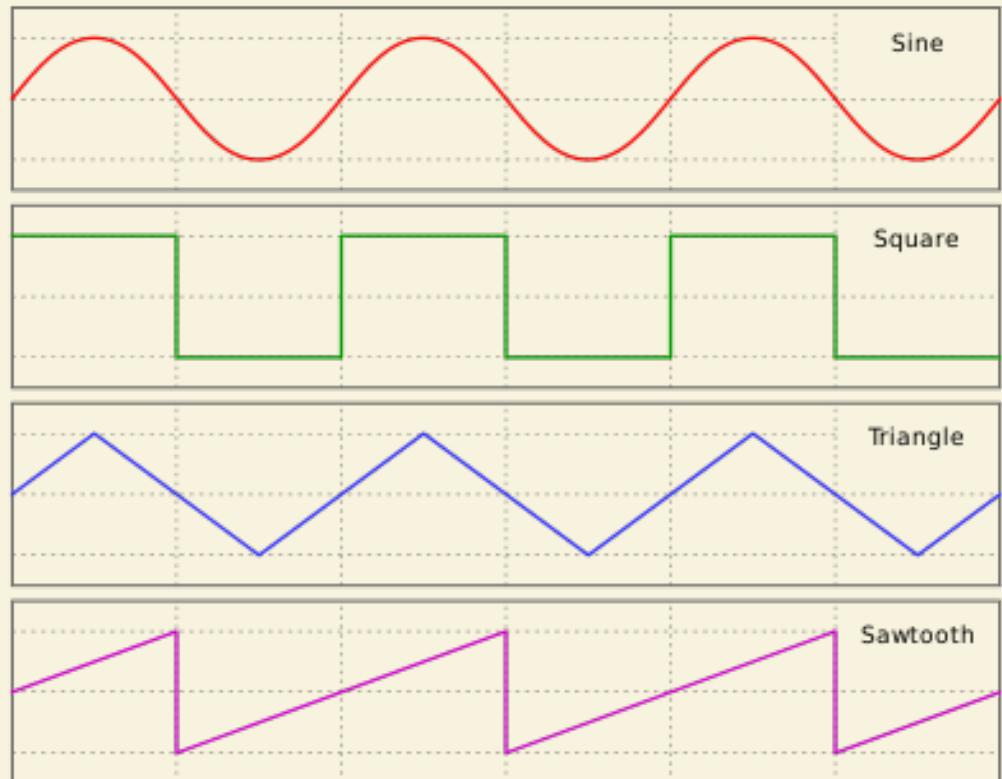
```
-- Чистый синус на данной частоте
-- osc - OSCillate
osc :: Sig -> Sig
```

# SYNTHESIZER

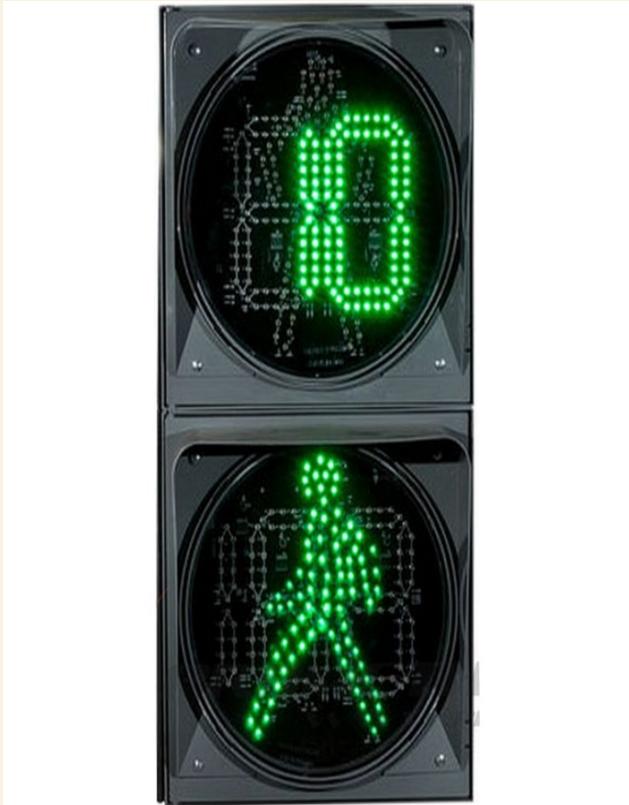


# OSCILLATOR

OSC  
SQR / PW  
TRI  
SAW



# CROSS ROAD



```
dac (osc 1000 * usqr 4)
```

Новые функции:

```
-- Unipolar sqr: [0, 1]
-- usqr x === 0.5 * (sqr x + 1)
usqr :: Sig -> Sig
```

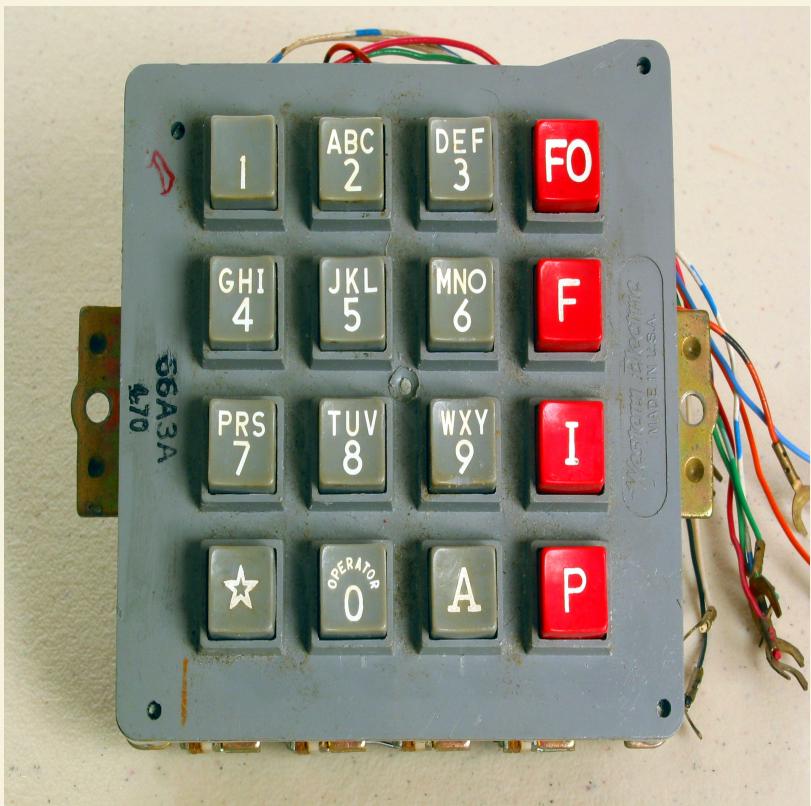
# RENDER TO WAV

```
crossRoad = osc 1000 * usqr 4  
  
writeHifi 35 "audio.wav" $ pure $ fromMono crossRoad
```

## Новинки

```
-- запись аудио в файл  
writeHifi :: RenderCsd a => D -> String -> SE Sig2 -> IO ()  
writeHifi duration file audio
```

# BEEPS



```
beep :: Sig -> Sig -> Sig  
beep x y =  
    0.4 * usqr 1 * (osc x + osc y)
```

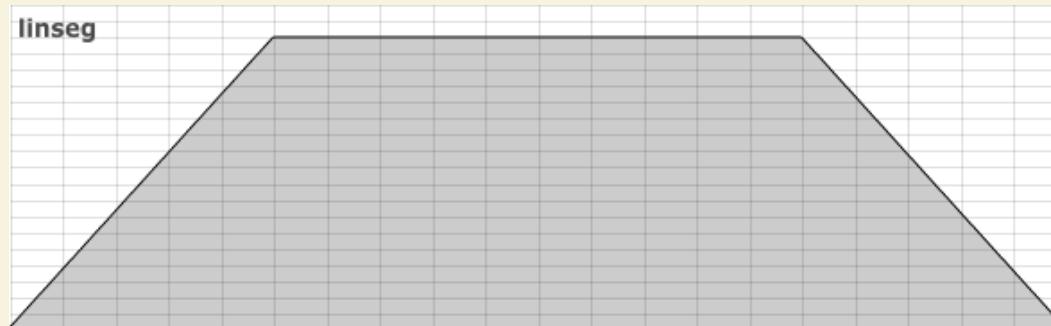
```
dac (beep 220 660)  
dac (beep 330 440)  
dac (beep 600 900)
```

# ENVELOPES

```
-- Кусочно-линейная функция
linseg :: [D] -> Sig

linseg [v1, time1, v2, time2, v3, time3, ..., vN]

-- Beep 3 раза
dac (linseg [1, 2.5, 1, 0.2, 0] * beep 200 300)
```



# SIREN

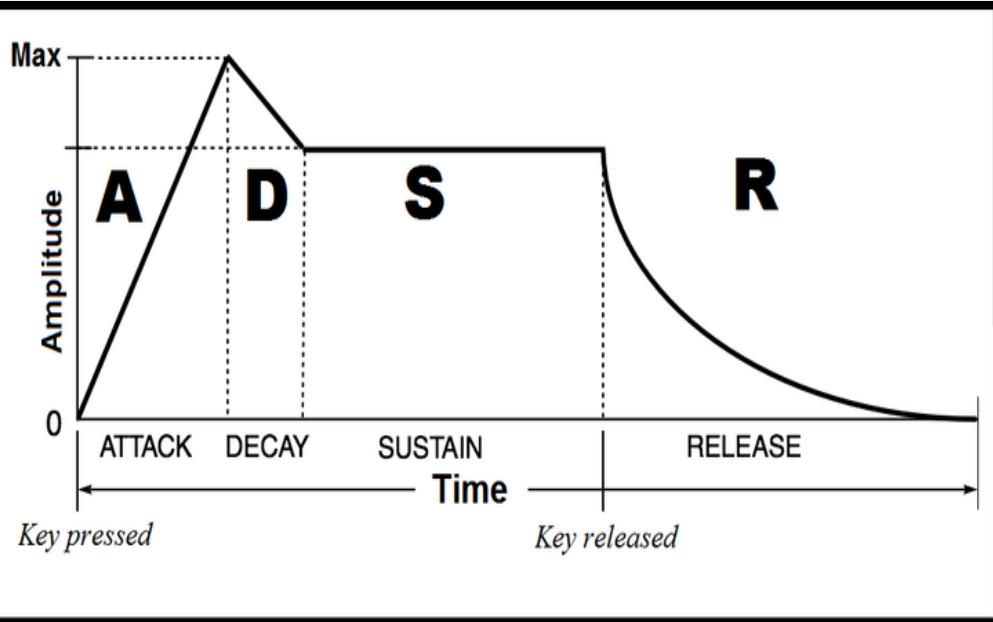


```
siren dt freq =  
    linseg [0, dt, 1]  
    * saw (expseg [10, dt, freq])  
  
dac $ 0.5 * siren 5 300
```

## Новые функции

```
-- Экспоненциальная функция  
expseg :: [D] -> Sig
```

# ADSR



-- линейный ADSR

```
leg :: D -> D -> D -> D -> Sig  
leg attack decay sustain release =
```

-- экспоненциальный ADSR

```
xeg :: D -> D -> D -> D -> Sig
```

-- плавное нарастание

```
dac $ leg 1.5 1 0.75 1 * osc 330
```

-- ударная нота

```
dac $ leg 0.001 0.3 0 0.5 * osc 330
```

-- Послушать релиз на виртуальном синтезаторе

```
instr freq = leg 1 1 0.5 1 * osc freq
```

```
vdac $ mul 0.5 $ tryMidi instr
```

# LFO, NOT UFO

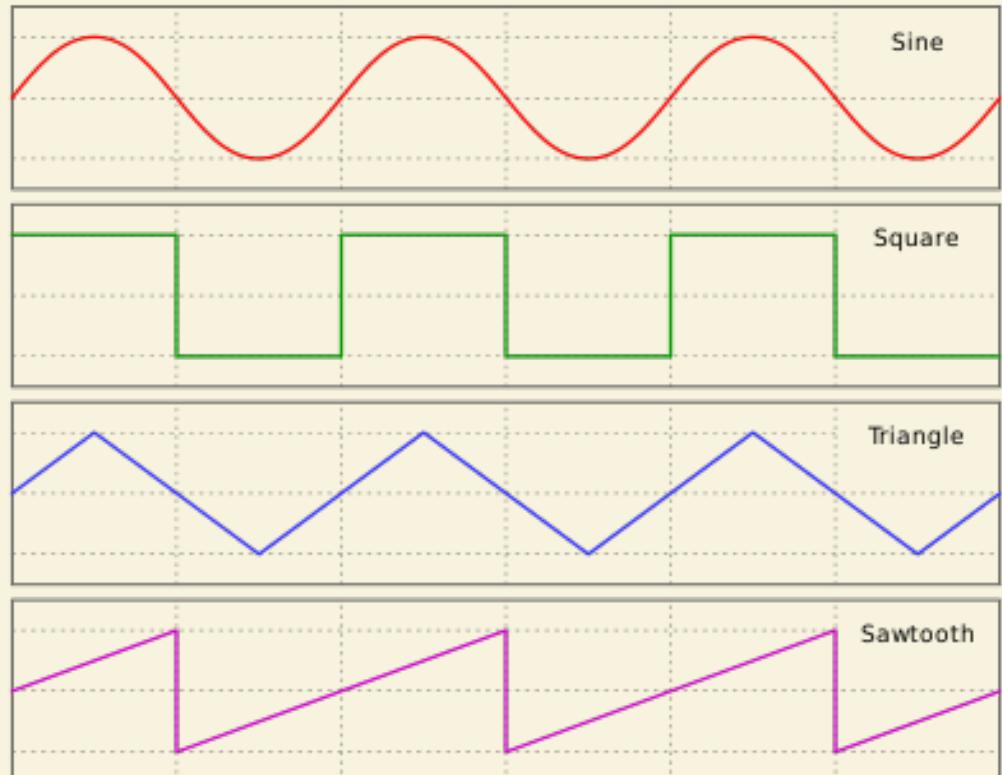
Low Frequency Oscillator

Осцилятор низких частот:

[0, 20] Hz

```
-- Биполярные [-1, 1]
osc, tri, sqr, saw
:: Sig -> Sig

-- Униполярные [0, 1]
uosc, utri, usqr, usaw
:: Sig -> Sig
```



# LFO: EXAMPLES

-- Светофор:

```
dac $ usqr 1 * osc 1000
```

-- Вибратор:

```
dac $ usqr 1 * osc (1000 * (1 + 0.05 * osc 6))
```

-- Двухнотный арпеджиатор:

```
dac $ usqr 1 * osc (220 + 110 * usqr 0.5)
```

-- Четыре ноты:

```
???
```

# LFO: EXAMPLES

-- Четыре ноты:

```
dac $ usqr 1 * osc (220 + 110 * usqr 0.5 + 110 * sqr 0.25)
```

# LFO: TRUE SIREN



```
env = 1 - usaw 0.25  
  
siren freq =  
    env * saw (freq * env)  
  
dac (siren 400)
```

# LFO: PURE TONE KICK



```
kick dt =  
let env = usaw dt  
in env * osc (150 * env)  
  
dac $ kick 2
```

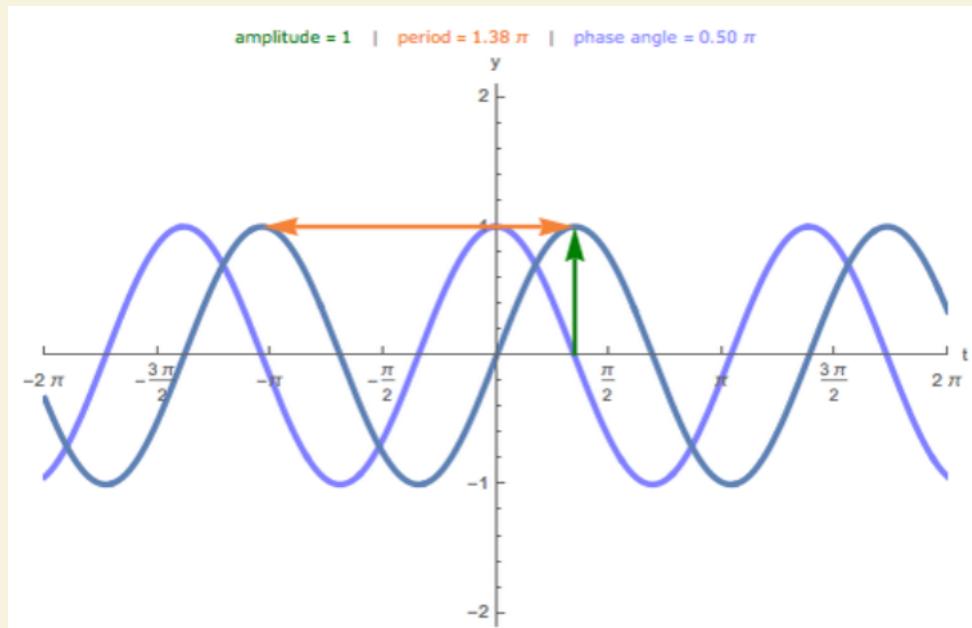
# PHASE

## Фаза - сдвиг начала волны осцилятора

```
osc', tri', saw', sqr' :: D -> Sig -> Sig
```

```
-- phase: [0, 1]
```

```
osc' phase frequency
```



# SUPER KICK



```
kick dt =  
let env = usaw dt  
in env * osc' 0.25 (150*env)  
  
dac $ kick 2
```

# MAKE SOME NOISE

```
-- Белый шум
white :: SE Sig

-- Розовый шум
pink, pinker :: SE Sig

-- Броуновский шум
brown :: SE Sig

-- пример
dac $ fmap (0.5 *) pink
```

# LFO + NOISE: HI-HATS



```
hihat :: Sig -> SE Sig  
hihat dt = mul (saw dt) pinker  
  
dac $ hihat 4 + pure (kick 2)
```

## Новинка:

```
-- УМНОЖИТЬ ЧТО-ТО на СИГНАЛ  
mul :: SigSpace a =>  
      Sig -> a -> a
```

# RANDOM LFO

```
-- Случайный сплайн: амплитуда, минимальная и максимальная частота изменений
jspline :: Sig -> Sig -> SE Sig
jspline amp minCps maxCps

-- 10 случайных медленных сигналов
ks = sequence $ replicate 10 (jspline 1 0.2 0.6)

-- спектр с этими сигналами
f x = fmap (
  (0.3 *)
  . sum
  . zipWith (\n k -> k * osc ((sig $ int n) * x)) [1 .. ]
) ks

dac $ mul 0.5 $ f 55 + f 110 + 0.3 * f 220 + 0.1 * f 330
```

# FILTERS



## Фильтр - душа синтезатора

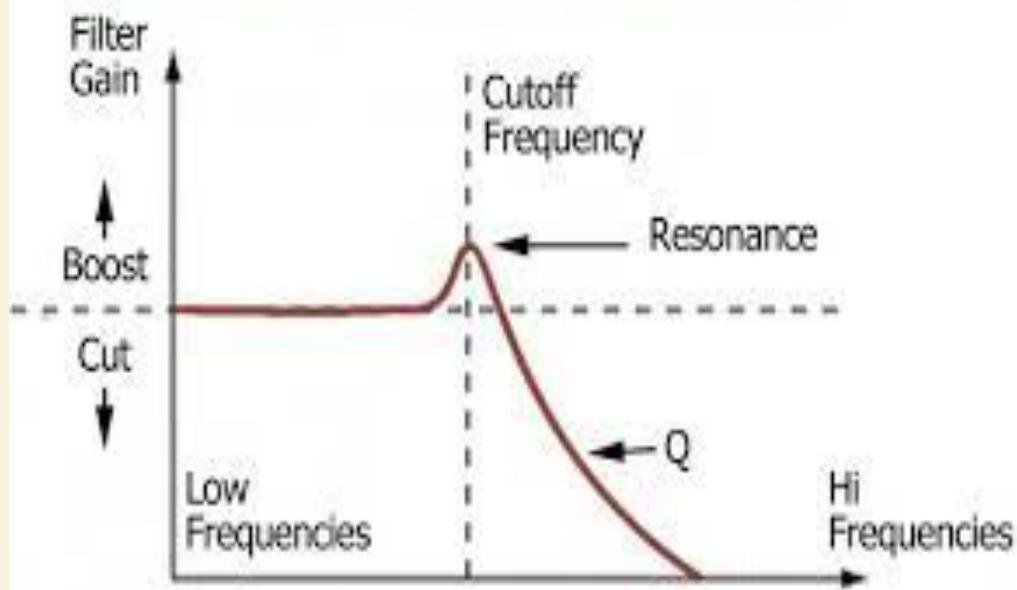
-- Moog low-pass filter

mlp :: Sig -> Sig -> Sig -> Sig

mlp cutoff resonance input = ..

# FILTERS: EXAMPLES

## LOW-PASS FILTER WITH RESONANCE



Увеличиваем частоту среза  
(cutoff)

```
cutOff = linseg [1, 8, 2500]  
dac $ mlp cutOff 0.1 (saw 220)
```

LFO на частоте среза

```
cutOff = 50 + 2500 * uosc 1  
dac $ mlp cutOff 0.1 (saw 220)
```

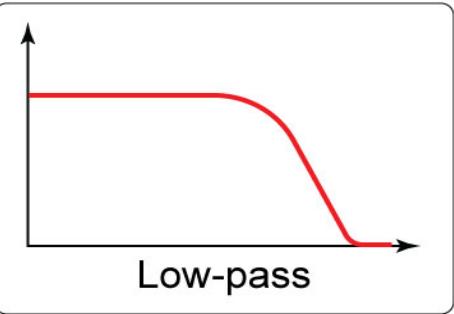
# DUBSTEP BASS

```
cutoff = 50 + 2000 * uosc (2 + 2 * uosc 0.5)
res    = 0.1 + 0.8 * uosc 0.1

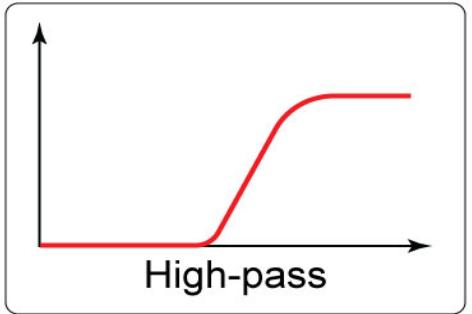
bas    = mlp cutoff res (saw 110)

dac bas
```

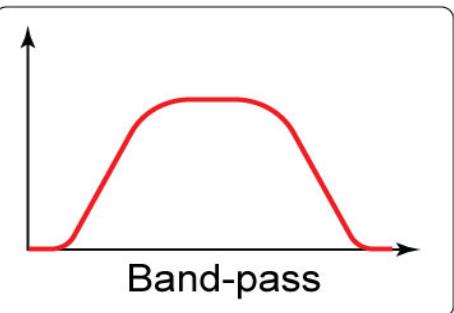
# FILTERS TYPES



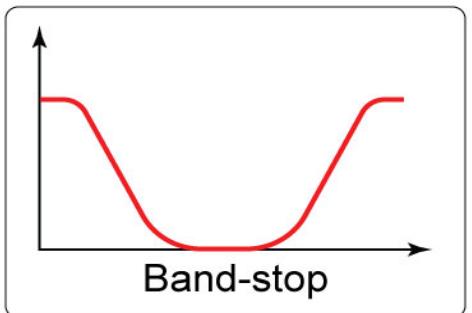
Low-pass



High-pass



Band-pass



Band-stop

-- *Low-pass*

lp :: Sig -> Sig -> Sig  
lp cutOff resonance input =

-- *High-pass*

hp :: Sig -> Sig -> Sig

-- *Band-pass*

bp :: Sig -> Sig -> Sig

-- *Band-reject*

br :: Sig -> Sig -> Sig

# BIG FAMILY

```
mlp, mlp2, mlp3      -- Moog filters
tbf                  -- TB 303 filter
blp                  -- Butterworth filters
lp                   -- state-variable filter
alp, alp2, alp3, alp4 -- "analog" filters
korg_lp              -- Korg filters
zlp                  -- zero-delay ladder filter
...
and many more: Csound.Air.Filter
```

# SUPER HI-HATS



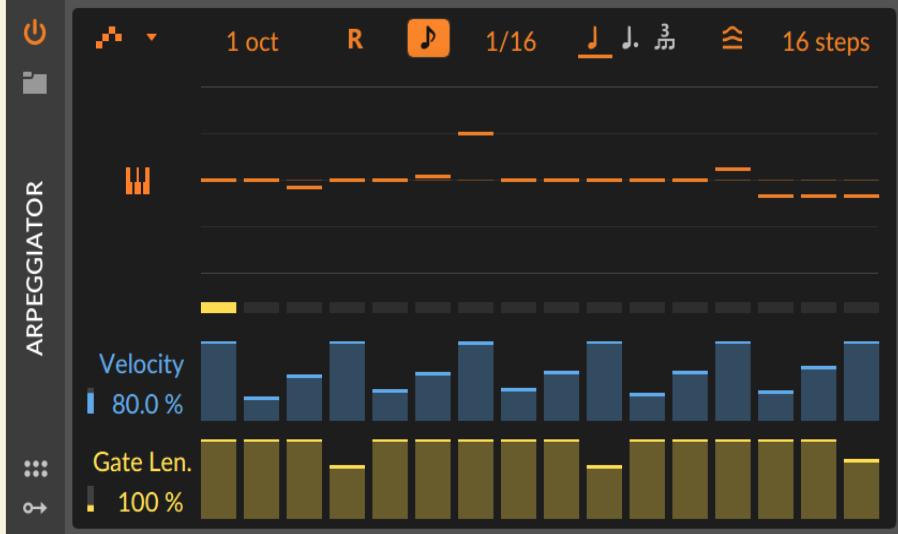
```
hihat :: Sig -> SE Sig
hihat dt =
  at (hp 400 0.1)
  (mul (saw dt) pinker)

dac $ hihat 4 + pure (kick 2)
```

## Новинка:

```
-- Преобразовать что-то
at :: At a b c
=> (a -> b)
-> c -> AtOut a b c
```

# ARPEGGIATORS



```
-- Цикличная кусочно-постоянная функция  
constSeq :: [Sig] -> Sig -> Sig  
constSeq vals frequency = ...
```

## Пример

```
-- ноты в Герцах  
notes = [220, 330, 660, 110]  
  
dac $ osc $ constSeq notes 4
```

# ARPEGGIATOR SHAPE

Нам доступны арпеджиаторы с разной формой ступенек:

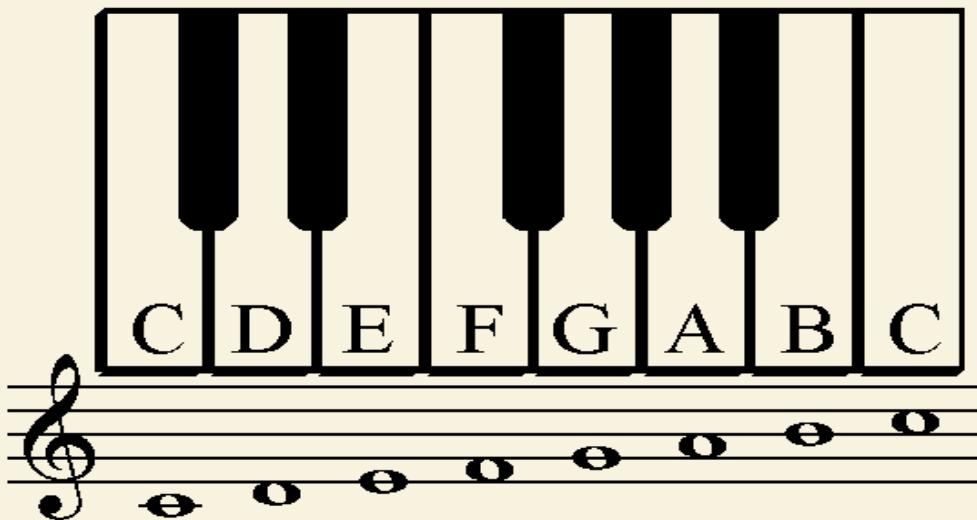
```
sawSeq    -- пилообразные ступеньки
sqrSeq    -- ступеньки из квадратных волн
triSeq    -- ступеньки треугольники
isawSeq   -- перевёрнутые пилообразные ступеньки
```

Пример:

```
freqs = constSeq [220, 330, 660, 110] 4
amps  = sawSeq   [1,    0.5, 0.2, 0.3] 4

dac $ mul amps $ osc freqs
```

# PITCH TO FREQUENCY



-- Переводит ноты в частоты  
cspch :: SigOrD a => a -> a

cspch 6.00	-- до	6 - октава
cspch 6.01	-- до диез	
cspch 6.02	-- ре	
cspch 6.03	-- ре диез	
cspch 6.04	-- ми	
...		
cspch 6.11	-- си	
cspch 7.00	-- до	7 - октава
...		
cspch 8.09	-- ля 440 Hz	

# PITCH ARPEGGIO

```
--          C      D     Eb      G
freqs = constSeq (fmap cpspch [8.00, 8.02, 8.03, 8.07]) 4
amps  = sawSeq    [1,   0.5, 0.2, 0.3] 4

dac $ mul (smooth 0.02 amps) $ osc freqs
```

## Новинка

```
-- сглаживает "острые" углы функций
-- (особый фильтр низких частот)
--

-- аргумент - длина окна сглаживания примерно в секундах
smooth :: Sig -> Sig -> Sig
```

# DETUNE: FAT ANALOG SOUND

```
-- Переводит центы в множители частот
-- 
-- 100 центов = расстояние между
-- соседними нотами
cent :: SigOrD a => a -> a
```

```
lead x = mul 0.4 $ sum
  [ saw x
  , saw (x * cent 4)
  , sqr (2 * x * cent 11)
  ]

freqs = constSeq (fmap cpspch [6.00, 6.01, 6.03, 5.10]) 4

dac $ mlp 4500 0.1 $ lead freqs
```

# SPICY LFO + PORTAMENTO

```
lead x = mul 0.4 $ sum
[ saw x
, saw (x * cent 4)
, sqr (2 * x * cent 11)
]

-- добавим "портаменто" (слайд между нотами)
freqs = smooth 0.015 $ constSeq (fmap cpspch [6.00, 6.01, 6.03, 5.10]) 4

-- добавим LFO-шек
cutoff = 700 + 4500 * utri 0.2 + 500 * osc 8
res    = 0.2 + 0.4 * uosc 0.2

dac $ mlp cutoff res $ lead freqs
```

# REVERB

```
-- Комната
-- dry/wet amount - соотношение между чистым и окрашенным сигналом
room :: Sig -> a -> a
room dryWetAmount input = ...

-- Камерный Зал
chamber :: Sig -> a -> a

-- Большой Зал
hall :: Sig -> a -> a

-- Пещера
cave :: Sig -> a -> a
```

## Добавим пространства

```
dac $ hall 0.25 $ mlp cutOff res $ lead freqs
```

# DELAY

```
-- Простой дилэй
echo :: DelayTime -> Feedback -> Sig -> Sig
echo delayTime feedback input

-- Аналоговый дилэй (с затуханием отражений)
adele :: Balance -> DelayTime -> Feedback -> ToneSig -> a -> a
```

## Простой пример

```
dac $ echo 0.5 0.7 $ leg 0.01 0.2 0 0 * osc 440
```

# DELAY FUN

```
dac $ echo 0.333 0.3 $ echo 0.5 0.7 $ leg 0.01 0.2 0 0 * osc 440
```

# DELAY + HI-HATS

```
-- Скучный хай-хэт
```

```
--
```

```
-- pw - Pulse-Width обобщение квадратной волны, прямоугольная волна  
hihat = mul (upw 0.1 1) white
```

```
dac hihat
```

```
-- Хай-хэт и дилэй
```

```
dac $ mixAt 0.3 (echo 0.25 0.5) hihat
```

```
-- И немногого ревера
```

```
dac $ chamber 0.3 $ mixAt 0.3 (echo 0.25 0.5) hihat
```

# KICK IN

```
hihat = chamber 0.3 $ mixAt 0.3 (echo 0.25 0.5) $  
    at (hp 400 0.2) $ mul (upw 0.1 1) white  
  
kick dt = let env = usaw dt  
           in env * osc' 0.25 (150*env)  
  
dac $ sum [ pure $ fromMono $ kick 2, hihat ]
```

# BASS

```
lead x = mul 0.4 $ sum [ saw x, saw (x * cent 4), sqr (2 * x * cent 11) ]  
  
freqs = constSeq (fmap cpspch [6.00, 6.01, 6.03, 5.10]) 4  
cutoff = 700 + 4500 * utri 0.2 + 500 * osc 8  
res    = 0.2 + 0.4 * uosc 0.2  
  
bass = hall 0.2 $ mlp cutoff res $ lead freqs
```

# KICK + HI-HAT + BASS

```
hihat = chamber 0.3 $ mixAt 0.3 (echo 0.25 0.5) $  
    at (hp 400 0.2) $ mul (upw 0.1 1) white  
  
kick dt = let env = usaw dt  
           in env * osc' 0.25 (150*env)  
  
lead x = mul 0.4 $ sum [ saw x, saw (x * cent 4), sqr (2 * x * cent 11) ]  
  
bass = hall 0.2 $ mlp cutOff res $ lead freqs  
where  
  freqs = constSeq (fmap cpspch [6.00, 6.01, 6.03, 5.10]) 4  
  cutOff = 700 + 4500 * utri 0.2 + 500 * osc 8  
  res    = 0.2 + 0.4 * uosc 0.2  
  
main = dac $ sum [ pure $ fromMono $ kick 2, hihat, pure bass ]
```

# ACID BASS

```
-- фильтр
-- попробуем: tbf -- имитация фильтра TB303
filt = mlp (900 + 300 * osc 0.25) (0.1 + 0.4 * uosc 0.2)

-- detune инструмент
lead x = mul 0.9 $ saw x + saw (x * cent 5) + sqr (x * 2 * cent 11)

-- частоты с портаменто
freqs = smooth 0.01 $
  stepSeq (fmap (* 0.5) [220, 440, 0, 330, 220, 0, 880 + 440 * usqr 0.5, 0]) 1

-- Bass
bas = filt $ lead freqs
```

# THANKS

- **ackage:** csound-expression, csound-catalog
- **real-world examples:** csound-bits (github repo)
- **github:** anton-k, spell-music
- **soundcloud:** anton-kho, tag #csound
- **bandcamp, itunes, spotify, boom, etc.:** Anton Kholomiov