Shaping Sounds in Swift

Adam Bell - @b3ll

Hello | こんにちは

- @b3ll
- Software Engineer
- Canadian
- Spend a lot of my personal t ime reverse engineering iOS / macOS or working with animations / gestures / audio
- Currently working on Origami Studio at Facebook

Synthesizers

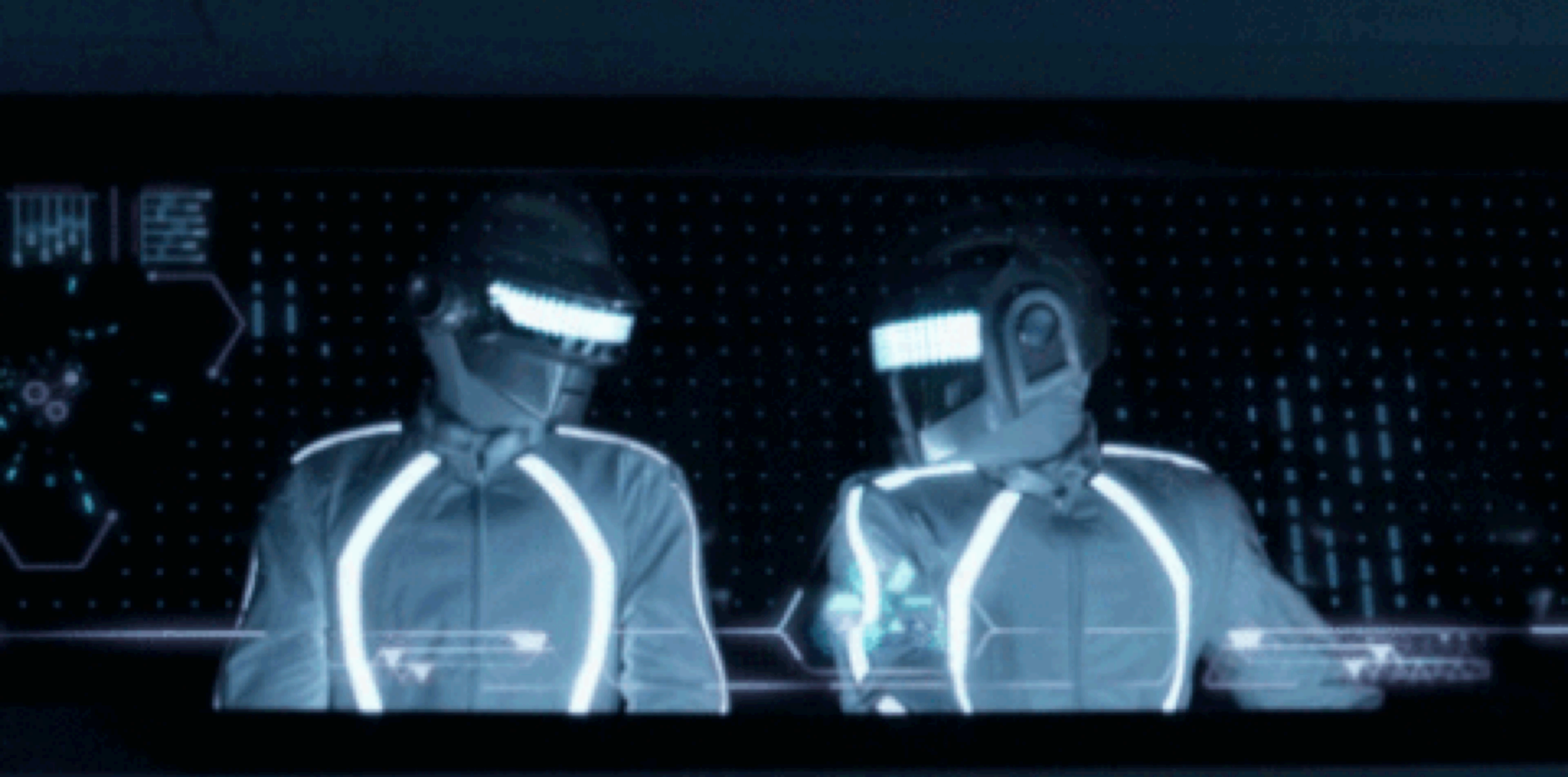
When I say the word synthesizer, what's the first thing that comes to mind?

- Daft Punk?
- Roland? Korg? Moog?
- Really futuristic / cool sounds?
- Really annoying sounds?


https://www.youtube.com/watch?v=vicpOlZAxsc

Roland Juno-60

https://en.audiofanzine.com/analog-synth/roland/JUNO-60/medias/videos/



https://giphy.com/gifs/michel-gondry-PtQBJX5Ae5uLK

Prophet-5

https://www.synthfind.com/sequential-circuits/sequential-circuits-prophet-5-rev-1-2/

https://www.youtube.com/watch?v=oAhvQoLpvsM

Who can tell me how that sound was made?

https://en.wikipedia.org/wiki/Roland_D-50

How to Make the Playstation 1 Startup Sound

1. Pick the Spacious Sweep preset on the Roland D-50.
2. Play some a low G note...
3. Done

Today I want to walk you through how to start with the basics of sound design, create some basic bland sounds, and turn them into something really cool.

"So how does a synthesizer work?"

Basically it's a big circuit board that generates manipulatable waves using:

- **Oscillators**
- **Envelopes**
- **Effects**
- ... and many more

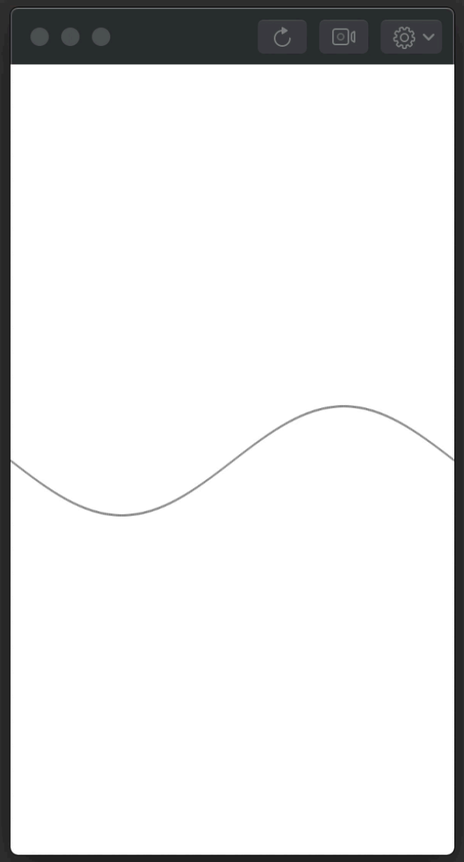
Oscillator ➡️ Envelopes ➡️ Effects ➡️ Speaker

Oscillator

An oscillator is a chip that creates a signal that moves back and forth (oscillates)

It generates a wave with different parameters:

- **Amplitude**: how big / loud is the wave?
- **Frequency**: how fast is the wave?
- **Shape**: what does it look like?



Rectangle

Enable

Position X 0 Y 333.5 Z 0

Size W 0.70312 H 2

Color Fill

Voices vs. Oscillators

You'll probably hear me mention voices and oscillators interchangeably.

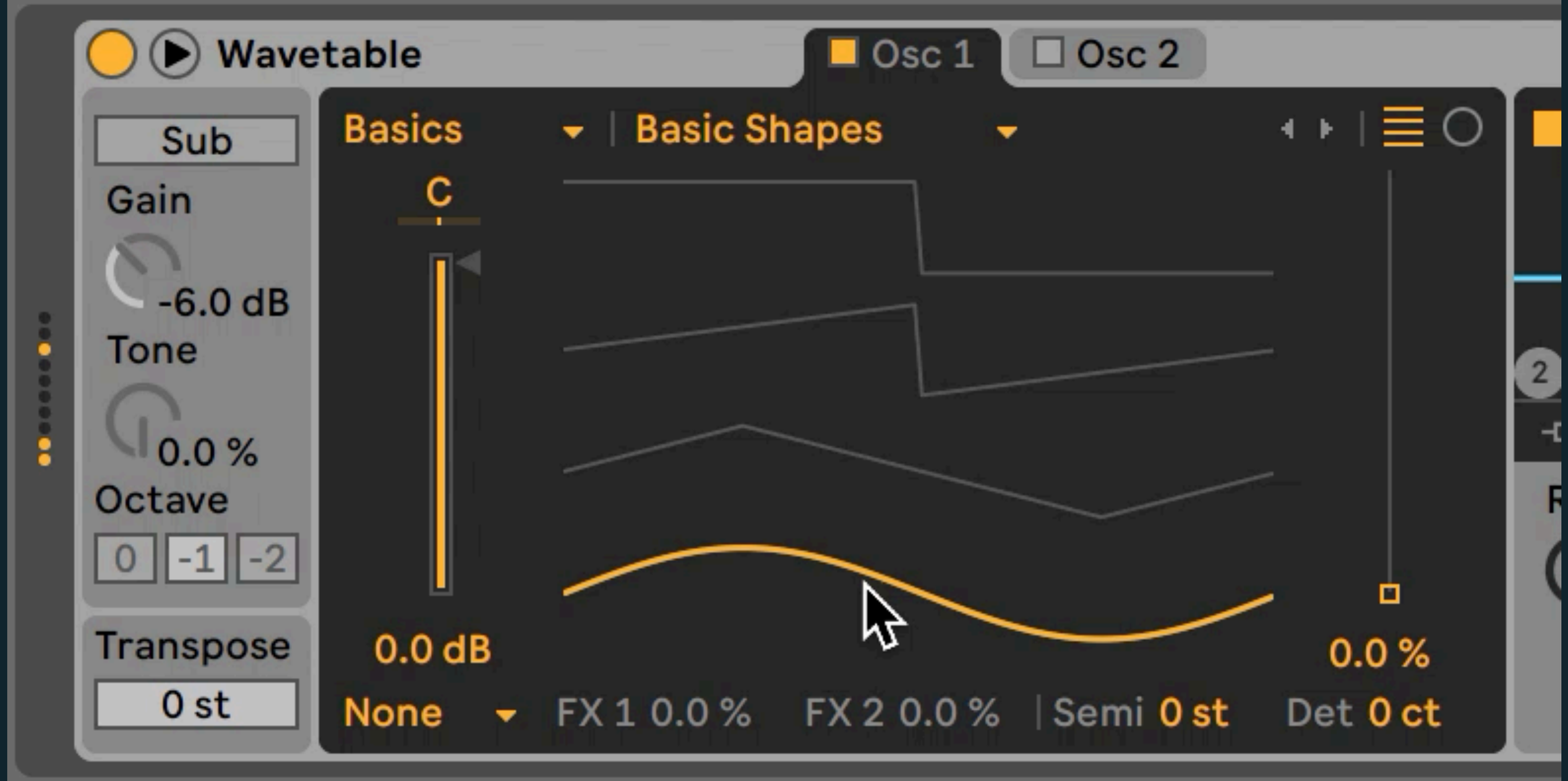
Oscillators generate waves.

Voices are the number of notes that can be played by the oscillators at the same time.

Waveforms

There’s a few types: sine, triangle, square, sawtooth...

Today we’re gonna focus on Square waves.


Wavetable
Osc 1
Osc 2
Sub
Gain
-6.0 dB
Tone
0.0 %
Octave
0
-1
-2
Transpose
0 st
Basics
Basic Shapes
C
0.0 dB
0.0 %
None
FX 1 0.0 %
FX 2 0.0 %
Semi 0 st
Det 0 ct

Frequency

The speed at which something oscillates, measured in cycles / second (aka Hertz, Hz)

Each of those waveform images were of one cycle.

To make a sound at a given pitch, you make lots of those waves in a single second.

Notes

Each octave in music has 7 major notes (C, D, E, F, G, A, B) and 5 other in-between notes (sharps / flats).

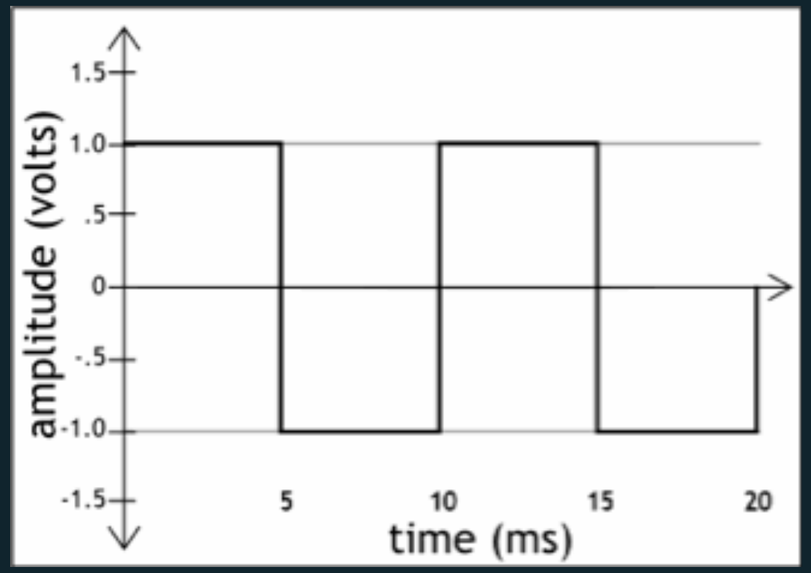
You can convert using the following equation (where x is the note number on a piano). Middle C is key number 40, so 261.62hz.

$$y = 440 \cdot 2^{\left(\frac{x-49}{12}\right)}$$

Square Waves

Very basic waveforms, they’re pretty much binary (either its -1 or 1)

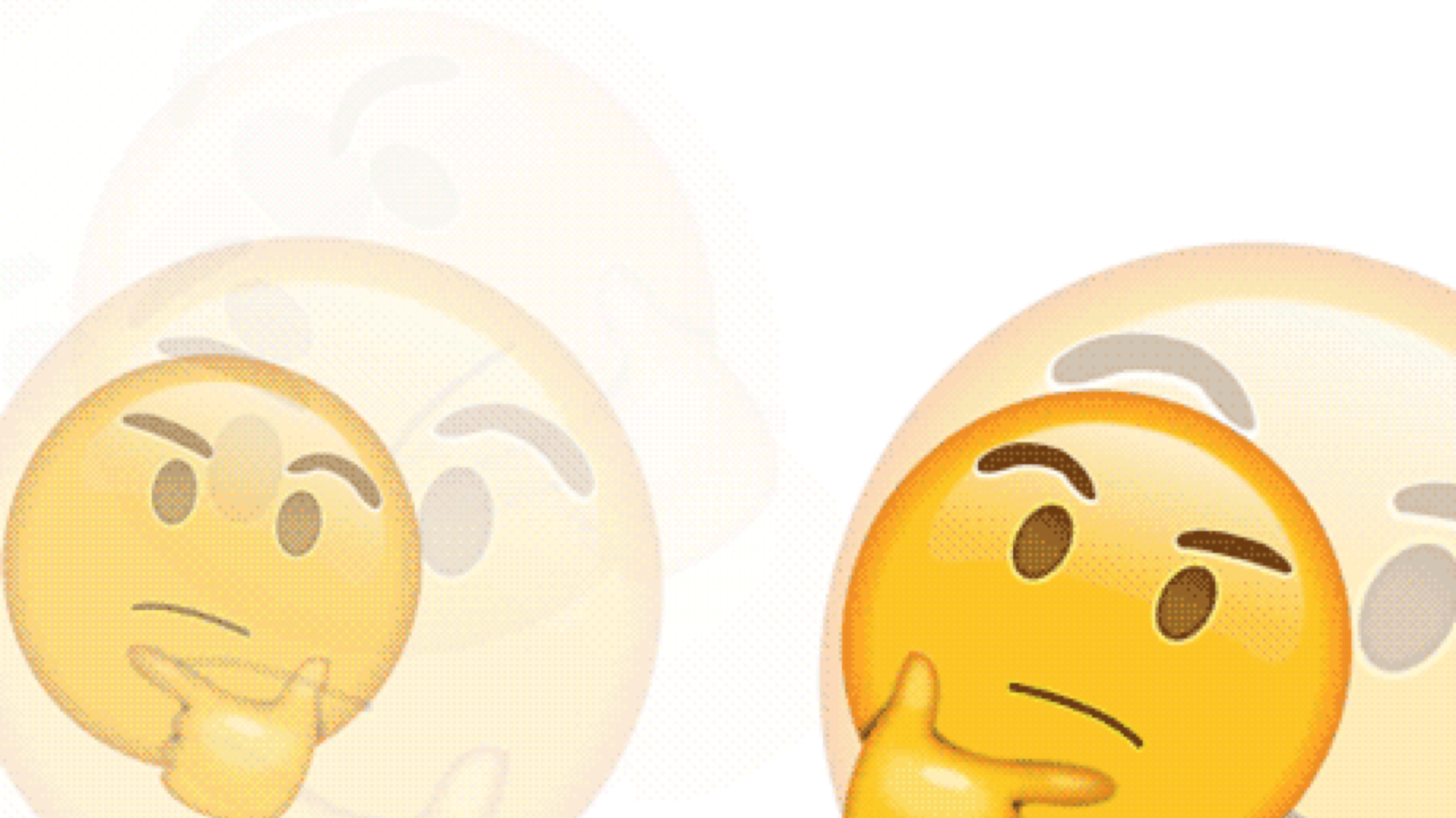
Either the signal is on, or off.



http://recordingology.com/in-the-studio/distortion/square-wave-calculations/

OK

So how would I make that on iOS?



iOS + Sound

Most of you, I'm assuming, have only really dealt with sounds on iOS by writing something like this:

```
let soundFileURL = URL(fileURLWithPath: "/System/Library/Audio/UISounds/payment_success.caf")

var soundID: SystemSoundID = 0
AudioServicesCreateSystemSoundID(soundFileURL as CFURL, &soundID)
AudioServicesPlaySystemSound(soundID)

// or

let soundEffect = AVAudioPlayer(contentsOf: soundFileURL)
soundEffect.prepareToPlay()
soundEffect.play()
```

CoreAudio

Core Audio is a very simple and very powerful framework on iOS.

iOS + Audio Primer

1. Create an `AudioStreamBasicDescription`
2. Set the sample rate
3. Set the format (`kAudioFormatLinearPCM`
4. Set the format flags (`kAudioFormatFlagIsBigEndian | kAudioFormatFlagIsSignedInteger | kAudioFormatFlagIsPacked`)
5. Set the number of bits per channel (16)
6. Set the number of channels per frame (1)

iOS + Audio Primer Continued...

1. Set the number of frames per packet (1)
2. Set the number of bytes per frame (2)
3. Set the number of bytes per packet (2)
4. Initialize some buffers using `malloc`
5. Fill the buffers
6. 🔊

SquareWave.mm

```
int main (int argc, const char * argv[]) {
@autoreleasepool {
  if (argc < 2) {
      return -1;
  }
  double hz = atof(argv[1]);
  assert (hz > 0);
  NSString *fileName = [NSString stringWithFormat: FILENAME_FORMAT, hz];
  NSString *filePath = [[[NSFileManager defaultManager] currentDirectoryPath]
                        stringByAppendingPathComponent: fileName]; NSURL *fileURL = [NSURL fileURLWithPath: filePath];
  // Prepare the format
  AudioStreamBasicDescription asbd;
  memset(&asbd, 0, sizeof(asbd));
  asbd.mSampleRate = SAMPLE_RATE;
  asbd.mFormatID = kAudioFormatLinearPCM;
  asbd.mFormatFlags = kAudioFormatFlagIsBigEndian | kAudioFormatFlagIsSignedInteger | kAudioFormatFlagIsPacked;
  asbd.mBitsPerChannel = 16;
  asbd.mChannelsPerFrame = 1;
  asbd.mFramesPerPacket = 1;
  asbd.mBytesPerFrame = 2; asbd.mBytesPerPacket = 2;
  // Set up the file
  AudioFileID audioFile;
  OSStatus audioErr = noErr;
  audioErr = AudioFileCreateWithURL((__bridge CFURLRef)fileURL,
                                    kAudioFileAIFFType,
                                    &asbd,
                                    kAudioFileFlags_EraseFile,
                                    &audioFile);
  assert (audioErr == noErr);
  // Start writing samples
  long maxSampleCount = SAMPLE_RATE;
  long sampleCount = 0;
  UInt32 bytesToWrite = 2;
  double wavelengthInSamples = SAMPLE_RATE / hz;
  while (sampleCount < maxSampleCount) {
      for (int i=0; i<wavelengthInSamples; i++) {
        // Square wave
        SInt16 sample;
        if (i < wavelengthInSamples/2) {
          sample = CFSwapInt16HostToBig (SHRT_MAX); } else {
          sample = CFSwapInt16HostToBig (SHRT_MIN); }
          audioErr = AudioFileWriteBytes(audioFile, false,
                                         sampleCount*2, &bytesToWrite, &sample);
          assert (audioErr == noErr); sampleCount++;
          audioErr = AudioFileClose(audioFile); assert (audioErr == noErr);
      }
    return 0;
  }
}
}
```

Easy, right?

nope.

CoreAudio

Core Audio is **not** a very simple and very powerful framework on iOS.

It's very powerful, but also very low level.

AudioKit

AudioKit is an audio synthesis, processing, and analysis platform for iOS, macOS, and tvOS.

Makes audio development far more approachable and easier to learn.

You spend less time worrying about buffer overflows, and more on tonal qualities.

AudioKit

It automates a bunch of the boilerplate code for audio development on iOS (no more Core Audio, or AVFoundation!), and has lots of effects and plugins for doing really cool stuff.

It’s open source, and even has some great playgrounds to explore audio / sound design.

SquareWave.mm

```
int main (int argc, const char * argv[]) {
@autoreleasepool {
  if (argc < 2) {
      return -1;
  }
  double hz = atof(argv[1]);
  assert (hz > 0);
  NSString *fileName = [NSString stringWithFormat: FILENAME_FORMAT, hz];
  NSString *filePath = [[[NSFileManager defaultManager] currentDirectoryPath]
                        stringByAppendingPathComponent: fileName]; NSURL *fileURL = [NSURL fileURLWithPath: filePath];
  // Prepare the format
  AudioStreamBasicDescription asbd;
  memset(&asbd, 0, sizeof(asbd));
  asbd.mSampleRate = SAMPLE_RATE;
  asbd.mFormatID = kAudioFormatLinearPCM;
  asbd.mFormatFlags = kAudioFormatFlagIsBigEndian | kAudioFormatFlagIsSignedInteger | kAudioFormatFlagIsPacked;
  asbd.mBitsPerChannel = 16;
  asbd.mChannelsPerFrame = 1;
  asbd.mFramesPerPacket = 1;
  asbd.mBytesPerFrame = 2; asbd.mBytesPerPacket = 2;
  // Set up the file
  AudioFileID audioFile;
  OSStatus audioErr = noErr;
  audioErr = AudioFileCreateWithURL((__bridge CFURLRef)fileURL,
                                    kAudioFileAIFFType,
                                    &asbd,
                                    kAudioFileFlags_EraseFile,
                                    &audioFile);
  assert (audioErr == noErr);
  // Start writing samples
  long maxSampleCount = SAMPLE_RATE;
  long sampleCount = 0;
  UInt32 bytesToWrite = 2;
  double wavelengthInSamples = SAMPLE_RATE / hz;
  while (sampleCount < maxSampleCount) {
      for (int i=0; i<wavelengthInSamples; i++) {
        // Square wave
        SInt16 sample;
        if (i < wavelengthInSamples/2) {
          sample = CFSwapInt16HostToBig (SHRT_MAX); } else {
          sample = CFSwapInt16HostToBig (SHRT_MIN); }
          audioErr = AudioFileWriteBytes(audioFile, false,
                                         sampleCount*2, &bytesToWrite, &sample);
          assert (audioErr == noErr); sampleCount++;
          audioErr = AudioFileClose(audioFile); assert (audioErr == noErr);
      }
    return 0;
  }
}
}
```

Creating an Oscillator

```
let oscillator = AKOscillator()
oscillator.frequency = 440
```

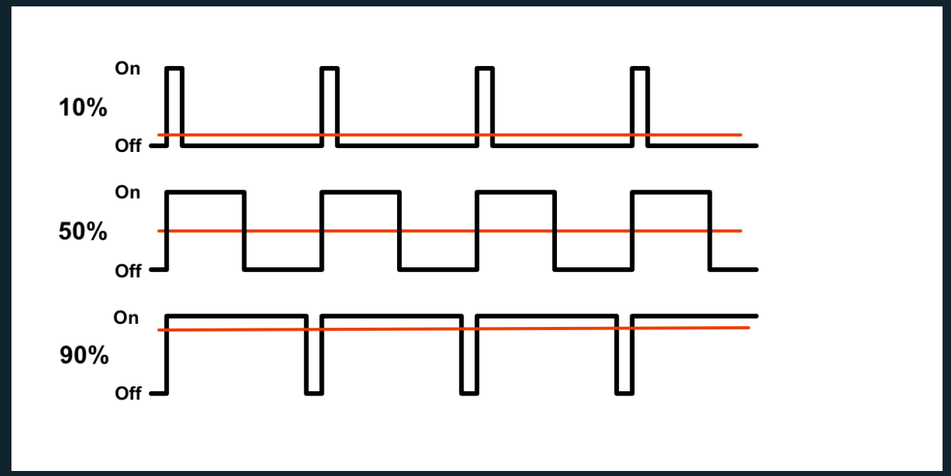
Creating a Square Wave Oscillator

```
let oscillator = AKPWMOscillator()
oscillator.frequency = 440
```

Pulse Width Modulation - PWM

PWM is the ability to modulate / change a square wave over time, by changing the width of each of the crests / troughs it produces.

This width is called a pulse width.



https://www.siriusmicro.com/chrp3/pwm-c.html


OFF
TEMPO SYNC
MONO
UNISON
POLY
PART ON/OFF
LOWER
UPPER
PANEL SELECT
LOWER
UPPER
SHIFT
MENU
PATCH PLG2 *A-3
SYSTEM-8
PLUG-OUT 1
PLUG-OUT 2
PLUG-OUT 3
WRITE
EXIT
ENTER
EDIT/DISP

Pulse Width Modulation - PWM

```
let oscillator = AKPWMOscillator()
oscillator.frequency = 440
oscillator.pulseWidth = 0.5
```

OK, so now what?

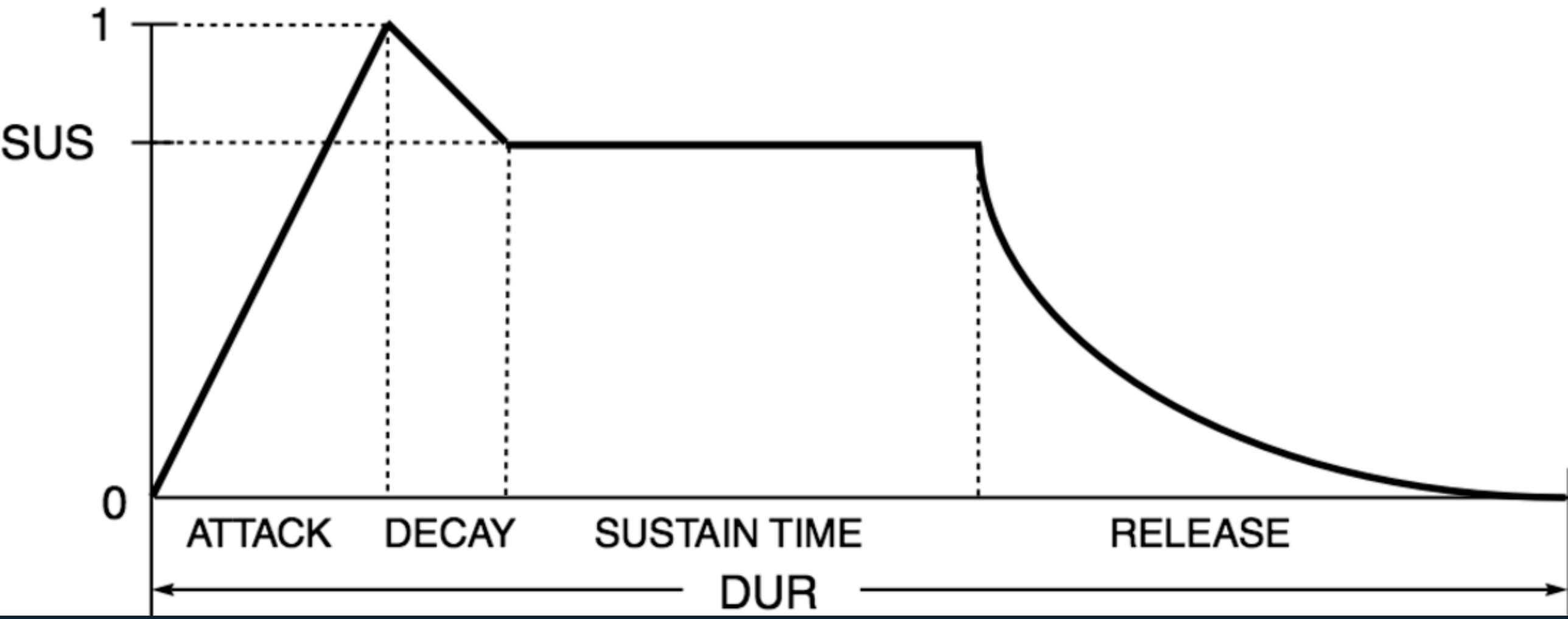
Envelopes

Sound envelopes are one of the most important things in synthesizing sounds, as it allows one to shape various parts of the sound.

One of the most common is called ADSR (Attack, Decay, Sustain, Release).

ADSR - Amplitude

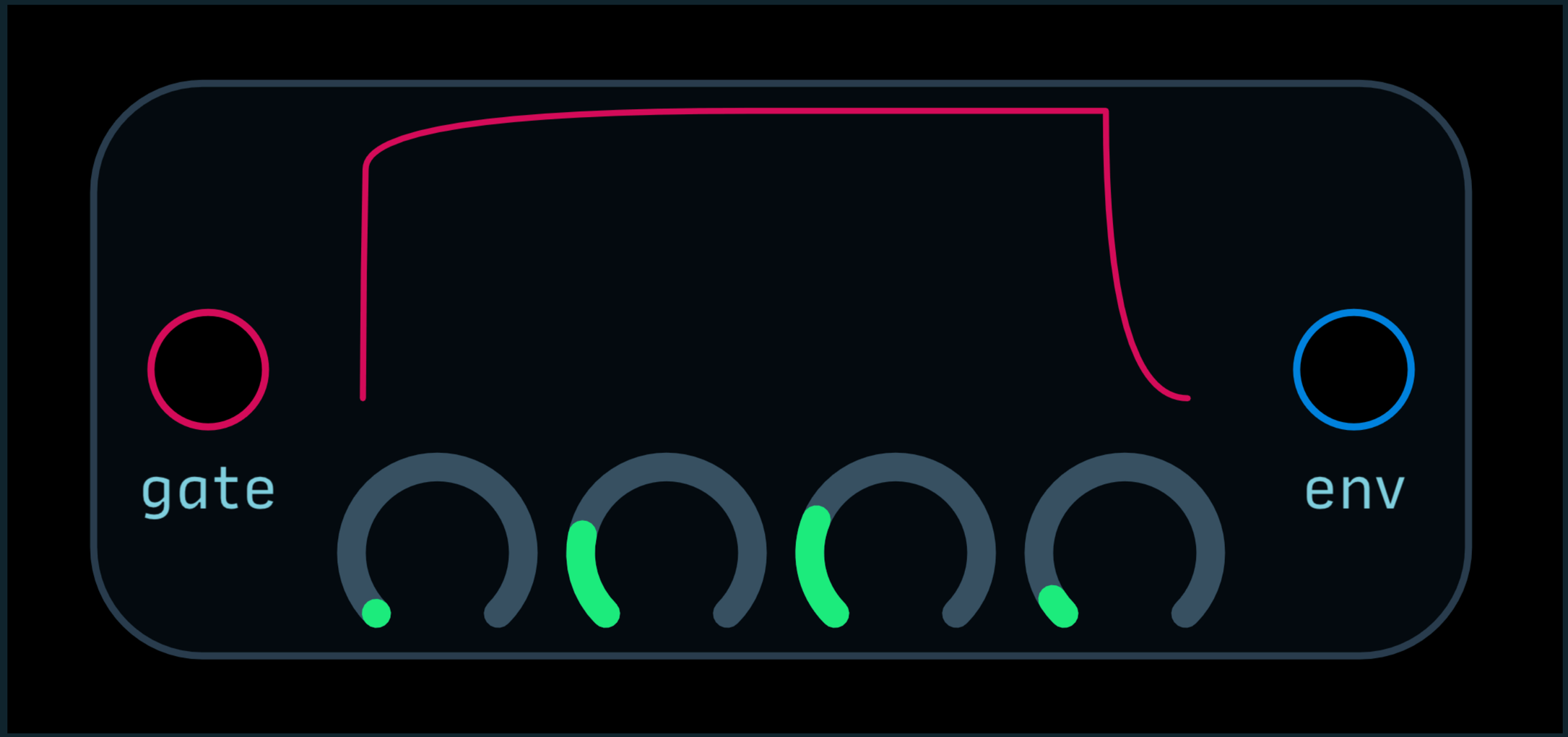
- **Attack**: How long it takes for the sound to be fully heard.
- **Decay**: How long it takes for the sound to decay to a sustained level.
- **Sustain**: How loud that sustained level of the sound is.
- **Release**: How long it takes for the sound to become quiet.



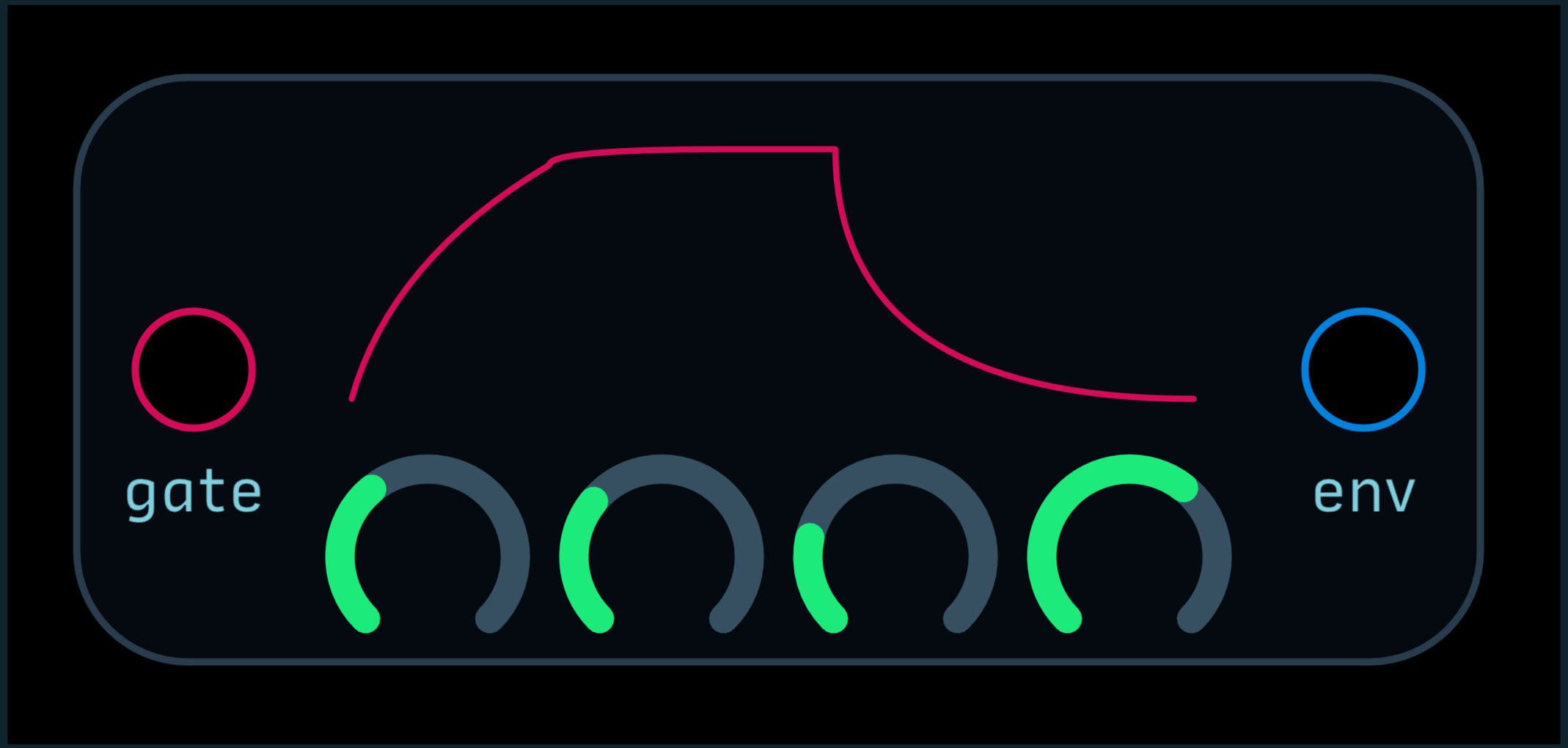
https://audiokitpro.com/lets-build-a-basic-synth-with-audiokit/

ADSR - Off

ADSR - Pluck Sound



ADSR - Pad Sound




OSC 1
OSC 2
OSC 3/SUB OSC
MIXER
PITCH
FILTER
AMP
EFFECT
NOISE TYPE
WHITE
PINK
RING
SYNC
TEMPO SYNC
MONO
UNISON
POLY
SYSTEM-8
PLUG-OUT 1
PLUG-OUT 2
PLUG-OUT 3
WRITE
EXIT
ENTER
SHIFT
MENU
PATCH PLG2 *A-3
KY Poly Chorus
STEP SEQUENCER
EDIT/DISP
REST
TIE
ERASE
LENGTH
STEP REC
REAL TIME REC
START
SCALE
PLAY MODE
A
B
C
D
E
F
G
H

Typically after an Amplitude Envelope (ADSR that controls the volume of the sound), you'll want to add on some effects that distort or further shape the sound.

The cool thing about envelopes... you can use them on effects too! (we'll get into that a bit later).

AudioKit Effects

Most effects in AudioKit follow a similar pattern, where the first argument is what the input is:

```
let oscillator = AKPWMOscillator()

// This applies an effect to the oscillator
let effect = AKSomeEffect(oscillator)
```

Filter

A filter, filters out unwanted sounds.

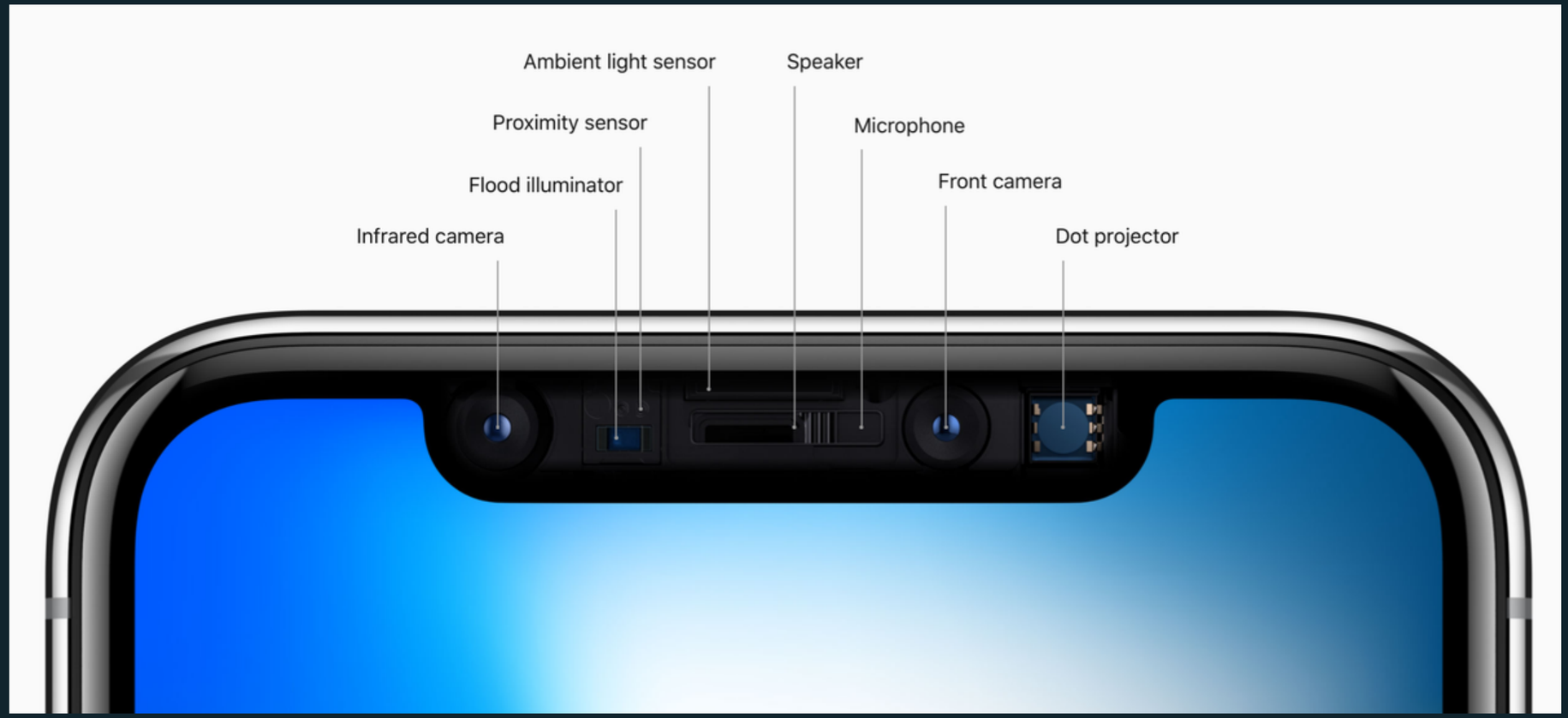
Low Pass Filter: Let everything that's lower than a frequency pass through.

High Pass Filter: Let everything that's higher than a given frequency pass through.

```
let lowpassFilter = AKMoogLadder(input)
lowPassFilter.cutoffFrequency = 6000.0
lowPassFilter.resonance = 0.8
```

Notch Filter

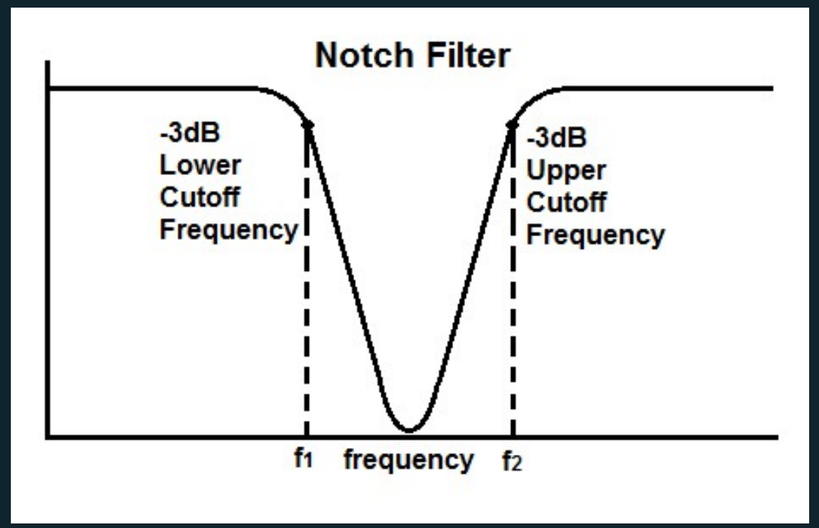
- As you expect, it's for filtering notches.



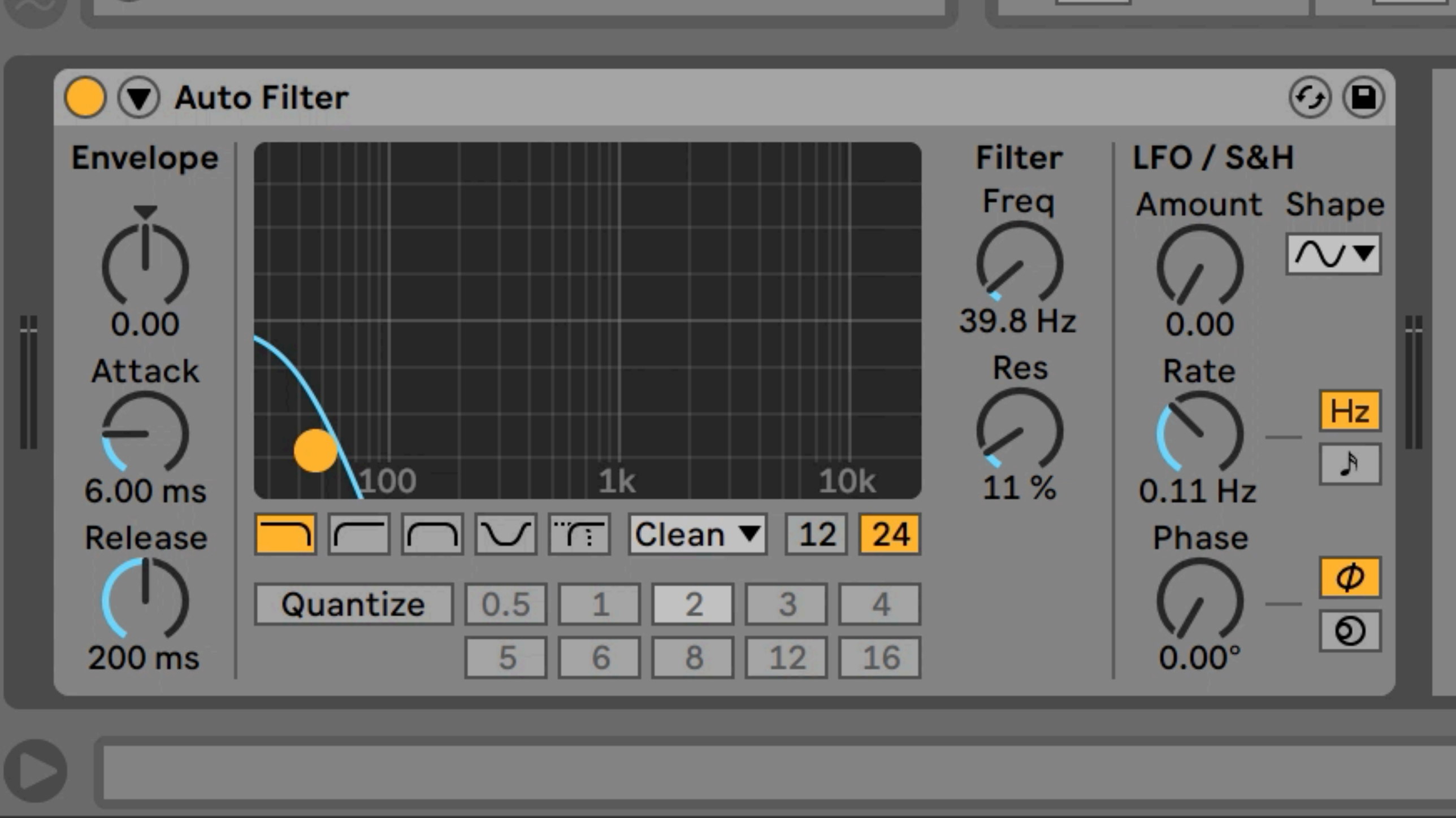

https://www.youtube.com/watch?v=O41PY6Rspq4

Notch Filter

- As you expect, it's for filtering notches of sound.
- It is basically the same as a high pass and a low pass working together.



https://oscarliang.com/filter-betaflight/


Auto Filter
Envelope
0.00
Attack
6.00 ms
Release
200 ms
100
1k
10k
Clean
12
24
Quantize
0.5
1
2
3
4
5
6
8
12
16
Filter
Freq
39.8 Hz
Res
11 %
LFO / S&H
Amount
0.00
Shape
Rate
0.11 Hz
Hz
Phase
0.00°

No Two Filters are the Same

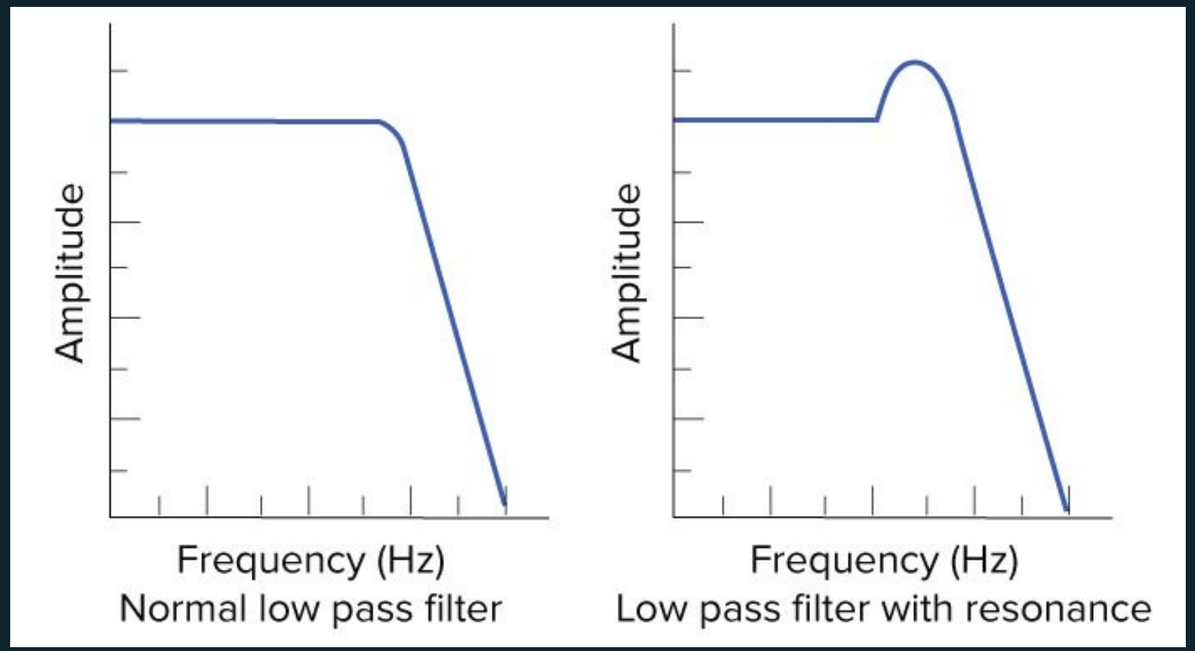
All filters, regardless if they're low or high, vary in how they filter sound.

This is usually what gives synthesizers such distinctive character and feel. Some are warmer, some are more harsh, etc.

A Moog's ladder low pass filter, for example, will usually sound totally different than a Korg or Roland.

Filter - Resonance

Resonance takes all the sound just before the cutoff point, and raises the volume of it.



http://www.roland.co.uk/blog/guide-to-subtractive-synthesis/

A pretty common thing is to change the cutoff and resonance at the same time.

How can you?

You don't have 3 hands 🤔

Low Frequency Oscillator - LFO

LFOs are a way to generate waves that aren't heard, but are used to change parameters (turn knobs) for you.

They're very slow waves (less than 20hz, usually).

Everytime there's a crest in a wave, that means the parameter is being turned up, and when there's a trough, it's being turned down.

One really useful example of this is taking an LFO and hooking it up to a Square Wave’s pulse width.

LFO vs ADSR

Both change parameters over time, however an LFO uses a repeating wave to change the parameter and an ADSR uses a graph with a defined beginning an end.

Filter - LFO / ADSR

LFOs and ADSRs can also be used to control filters. This allows you to not control how loud the sound is, but how much sound you hear (i.e. the frequencies that you can hear overtime).

You can have LFOs applied to the resonance, the cutoff point, or both!


OSC 2
OSC 3/SUB OSC
ARPEGGIO
ARP TYPE
ARP STEP
KEY HOLD
CHORD MEMORY
VELOCITY
TRANSPOSE
OCTAVE
DOWN
UP
PATCH
MODE
PERFORMANCE
MANUAL
LEGATO
PART ON/OFF
LOWER
UPPER
PANEL SELECT
TEMPO SYNC
MONO
UNISON
POLY
SHIFT
MENU
RING
SYNC
KEY TRIG
TRIG ENV
BEND SENS
MOD SENS
PITCH
FILTER
SYSTEM-8
PLUG-OUT 1
PATCH PLG2 +0-3
KY Poly Chorus

Delay

Delay is an effect used to take an audio signal and repeat it.

... and repeat it... and repeat it... until you can't hear it anymore.

It’s really great for making echos or layering sounds.

```
let delay = AKDelay(input)
delay.time = 1.5 // seconds
```

Chorus

Chorus is an effect that makes the audio signal sound much wider.

It's created using two alternating delays on the same signal (which creates an echo along the Left and Right channel in stereo)

```
let chorus = AKChorus(input)
```

Reverb

Reverb is an effect used to take the audio signal and make it sound like it's being played in a different room:

- a concert hall
- a small studio
- a stadium, etc.

```
let reverb = AKReverb(input)
```

Simple Square Synth

```
let squareOscillator = AKPWMOscillatorBank()
squareOscillator.pulseWidth = 0.65

squareOscillator.attackDuration = 5.0
squareOscillator.decayDuration = 2.5
squareOscillator.sustainLevel = 1.0
squareOscillator.releaseDuration = 1.5

let lowPassFilter = AKMoogLadder(squareOscillator)
lowPassFilter.cutoffFrequency = 16000.0

let chorus = AKChorus(lowPassFilter)
chorus.dryWetMix = 0.5

let reverb = AKReverb(chorus)

AudioKit.output = reverb
try? AudioKit.start()
```

Demo

Sound Design

With sound design, you're constantly iterating and developing new ideas. You'll eventually end up making mistakes that lead to completely new ideas and inspiration.

It's really hard to do sound design wrong, because it's all about exploration.

You'll end up taking something super basic, and transforming it into something completely new and artistic.


MARIO
019100
×00
WORLD
1-1
TIME
062
5000

App Design

This iterative process of manipulating and refining to see what feels / sounds right can be applied to a lot of aspects of writing apps.

You'll find ways to make things feel great, that'll feed into other parts to make them better as well.

Push yourself to refine every little interaction, focus on every nuanced detail, and try to turn "that feels good enough" into "that feels great".

As someone who makes apps, you're an artist, and by pushing yourself to make the best thing you can, you'll make room for innovating and developing an identity associated with something great.

Next time you're working on an app, try to treat it as if you were modulating knobs on a synthesizer to shape "your" sound that resonates with people.

Thanks | ありがとうございました 🎹

If you have any questions or just want to talk about synths, come say hi or message me! @b3ll

If you want to try some sound design checkout AudioKit:
audiokit.io