Google

Android Haptics: API and Implementation on Pixel

A Tutorial by Ahmad Khalil, Nathan Kulczak & Hong Tan

World Haptics Conference 2025, Suwon, Korea
July 8, 2025

About Us

Ahmad Khalil

- Software engineer at Google
- Haptics framework
- OEM haptics integration

Nathan Kulczak

- Software engineer at Google
- Haptics System SW
- Pixel software team

Hong Tan

- Research scientist at Google
- Haptics research team
- Android & Pixel haptics
- User experience research
- Psychophysics

Structure of This Tutorial

Overview

Part 1	Basics of Human Haptic Perception
Part 2	Android Haptics APIs
Part 3	Pixel Implementation
Part 4	Demos!

Part 1: Basics of Human Haptic Perception

Multiple API Entry Points:
PWLE API Lets Designers & Developers Focus on Desired Sensations

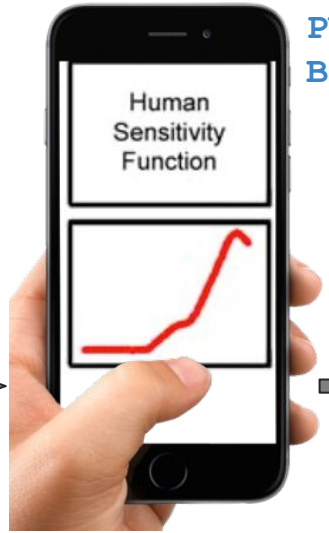
```
oneshot()
createWaveform()

PRIMITIVE:
QUICK_RISE
SLOW_RISE
QUICK_FALL
SPIN
THUD
```

```
PWLE API (Advanced):
WaveformEnvelopeBuilder()

PRIMITIVE:
CLICK
TICK
LOW_TICK
```

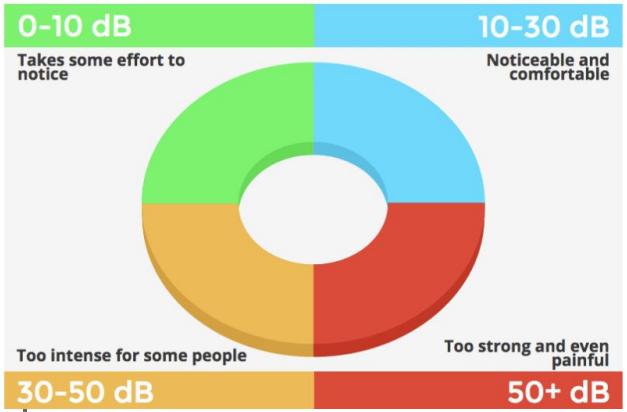
```
PWLE API (Basic):
BasicEnvelopeBuilder()
```

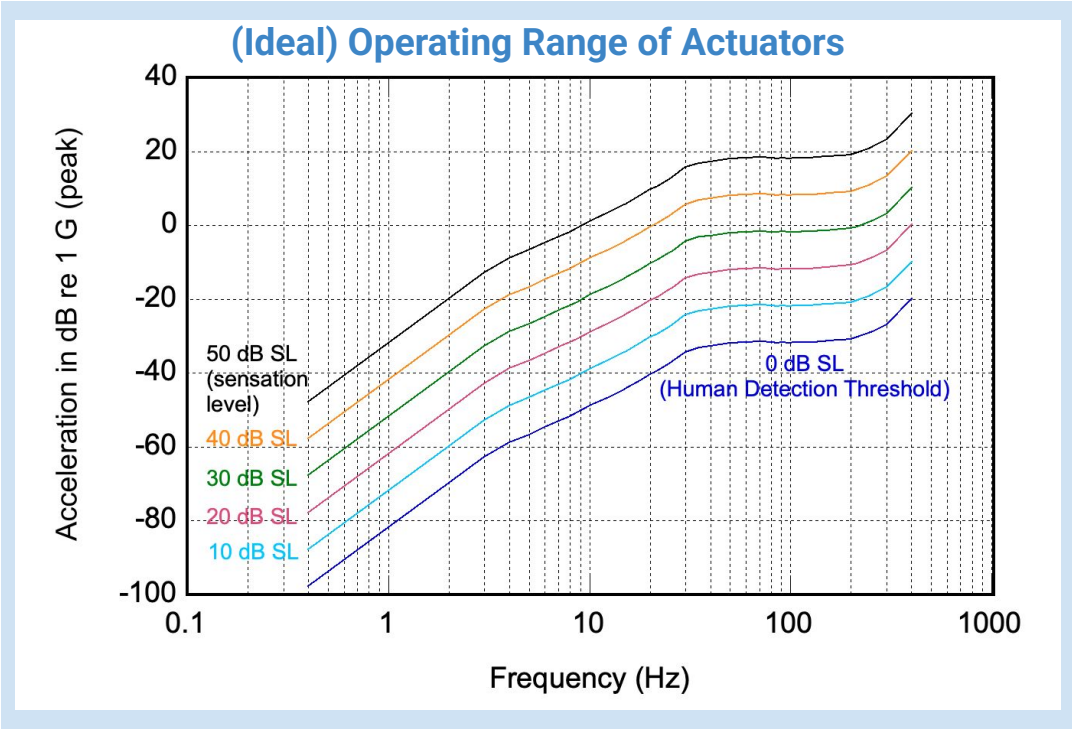


Voltage Waveform → Acceleration → Perception (Feels Like ...)

Human Vibrotactile Detection Threshold & Sensation Level (SL)

- **Detection Threshold**
  - Smallest displacement or lowest acceleration that can be reliably detected
  - Frequency dependency
- **Perceived Intensity**
  - Grows with Log of signal amplitude
  - Specified in Sensational Level (**SL**; dB above detection threshold)
- **Strength of Vibration**





From Bolanowski et al. (JASA 1988) & Verrillo et al. (P&P 1969)

Part 2:
Android Haptics APIs

Outline

01
Introduction to Android Haptics

Current Haptics Toolkit

- **Android Vibrator API**: Basic on/off controls with timings and predefined effects
- **Audio coupled haptics**: A method to create synchronized vibration via OGG files.
- **Primitives**: A predesigned set of haptic effects (`CLICK`, `TICK`, `LOW_TICK`, `SLOW_RISE`, `QUICK_RISE`, `QUICK_FALL`, `SPIN`, `THUD`) intended to provide a consistent experience for common actions

Current Haptics Toolkit - Drawbacks

- Vibration hardware (actuator + driver) varies across the wide range of Android devices
- This makes it difficult for developers to reliably design and ship consistent haptic feedback for all users
- This leads to inconsistent user experiences

Future Haptics in Android

- **Move towards wideband haptics:** Enable richer, more nuanced, and expressive haptic effects, beyond simple buzzing
- **Achieve a "Design-Once, Play-All" model:** Allow developers to design haptic effects once for a consistently high-quality experience across the entire Android ecosystem
- **Enable hardware-agnostic effects:** To achieve this, we need a new way for developers to define haptics that is not tied to the specific capabilities of a device's vibrator and driver

Introducing PWLE Vibrations in Android 16

- **PWLE** stands for **P**iece**W**ise **L**inear **E**nvelopes
- It's a new, expressive way to create haptic effects by defining vibration amplitude and frequency changes over time.
- This API allows for precise, hardware-agnostic control, ensuring the perceptual experience is consistent, even on different hardware.

New PWLE APIs in Android 16

- **BasicEnvelopeBuilder**: An accessible approach to creating hardware-agnostic haptic effects

- **WaveformEnvelopeBuilder**: A more advanced approach to creating haptic effects; requires familiarity with haptics hardware

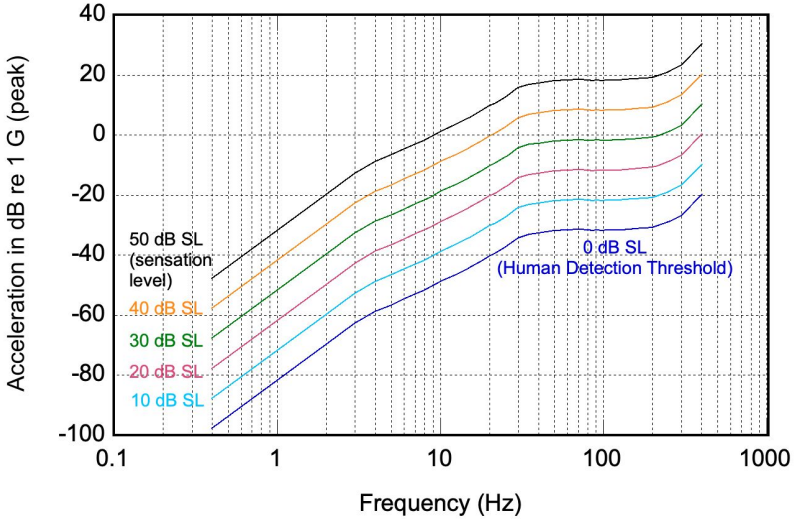
02
BasicEnvelopeBuilder

BasicEnvelopeBuilder

- An accessible approach to creating hardware agnostic haptic effects
- Requires minimal haptic hardware knowledge
- Vibrations defined using:
  - Intensity [0, 1]: Perceived strength
  - Sharpness [0, 1]: Crispness of the vibration

BasicEnvelopeBuilder

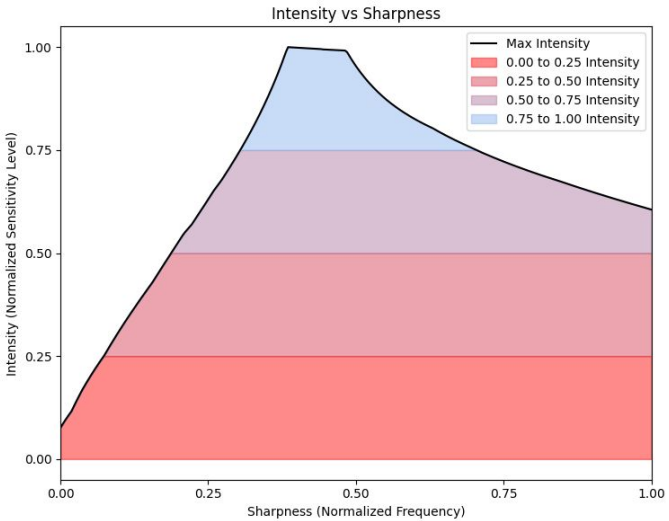
- SL = sensation level. It's defined as dB above Human detection threshold.
- Minimum sensitivity threshold of 10 dB
- Helps define the sharpness range
- The maximum acceleration at a given frequency needs to be at least 10 dB SL
- The lowest frequency that satisfy the 10 dB SL threshold is mapped to sharpness of 0.0
- A sharpness of 1.0 represents the highest frequency where the acceleration level drops to 10 dB SL, or the highest supported frequency if acceleration stays above that threshold

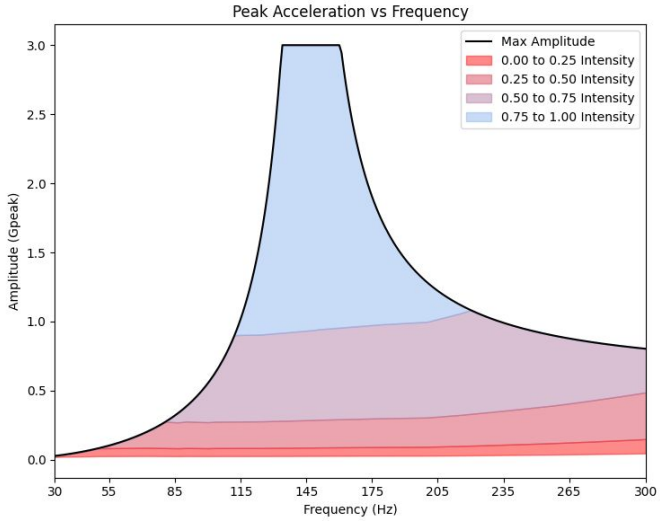


BasicEnvelopeBuilder

- The framework will handle mapping normalized parameters(intensity and sharpness) to physical ones (amplitude and frequency)

- Uses exponential function to map intensity to acceleration:
  - **`Map Intensity to SL`**
  - **`Map SL to dB`**
  - **`Acc = 10`$^{\mathbf{dB/20}}$**





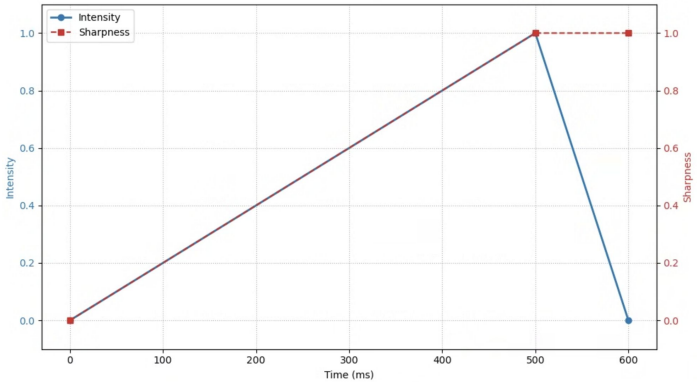
BasicEnvelopeBuilder

- Limitations:
  - Begin and end at 0 intensity
  - May not use the full range of frequencies
  - Restricted to a frequency range that can generate output of at least 10 db SL
- Android framework will automatically adjust the effect to fit within the allowed boundaries (as much as possible)

Code Samples

```
VibrationEffect basicEnvelopeEffect = new VibrationEffect.BasicEnvelopeBuilder()
            .setInitialSharpness(/*sharpness= */ 0.0f)
            .addControlPoint(/*intensity= */ 1.0f, /*sharpness= */ 1.0f, /*duration= */ 500)
            .addControlPoint(/*intensity= */ 0.0f, /*sharpness= */ 1.0f, /*duration= */ 100)
            .build();
```

- This example ramps up the intensity and sharpness from low to high over 500ms, then ramps down to 0 over 100ms.
- Setting `initialSharpness` is optional; defaults to the first control point if omitted.



Bouncing Spring

```
// Create a <<bo-ing>> envelope vibration effect that fades out.
vibrator.vibrate(
    VibrationEffect.BasicEnvelopeBuilder()
    // Starting from zero sharpness here, will simulate a smoother
    // <<bo-ing>> effect.
    .setInitialSharpness(/*sharpness= */ 0.0f)
    // Add a control point to reach the desired intensity and
    // sharpness very quickly
    .addControlPoint(intensity, sharpness, 20L)
    // Add a control point to fade out the vibration intensity while
    // maintaining sharpness.
    .addControlPoint(0.0f, sharpness, fadeOutDuration)
    .build()
)
```

(Visit the Google booth to experience it first hand)

03
WaveformEnvelopeBuilder

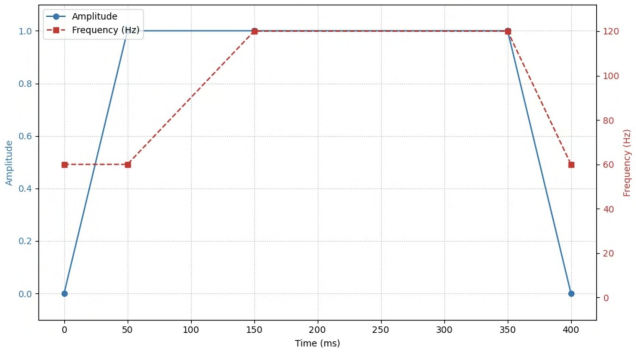
WaveformEnvelopeBuilder

- An advanced approach for detailed control of haptic effects
- Requires advanced haptics hardware knowledge
- Vibrations defined using:
  - Amplitude [0, 1]: 1 representing the max achievable strength at that frequency
  - Frequency: Specified directly in Hz
- No enforced limits
  - Developers must ensure frequencies are within the device's capabilities

Code Samples

```
VibrationEffect waveformEnvelopeEffect = new VibrationEffect.WaveformEnvelopeBuilder()
            .addControlPoint(/*amplitude= */ 1.0f, /*frequency= */ 60f, /*duration= */ 50)
            .addControlPoint(/*amplitude= */ 1.0f, /*frequency= */ 120f, /*duration= */ 100)
            .addControlPoint(/*amplitude= */ 1.0f, /*frequency= */ 120f, /*duration= */ 200)
            .addControlPoint(/*amplitude= */ 0.0f, /*frequency= */ 60f, /*duration= */ 50)
            .build();
```

- This example defines a 400ms vibration effect. It begins at 60Hz for 50ms, ramps up to 120Hz over 100ms, stays at 120Hz for 200ms, then returns to 60Hz over 50ms while amplitude drops to 0.
- Initial frequency can be omitted; it defaults to the first control point.



Rocket Launch

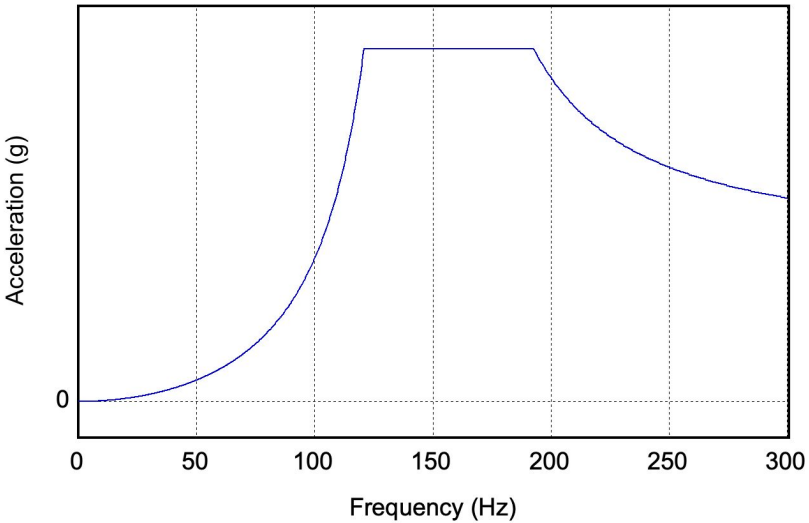
```
val startFrequency = vibrator.frequencyProfile?
                          .getFrequencyRange(minOutputAccelerationGs)?.lower
val minDurationMs = vibrator.envelopeEffectInfo
.minControlPointDurationMillis
val rampUpDurationMs = (riseBias * totalDurationMs).toLong() - minDurationMs
val rampDownDurationMs = totalDurationMs - rampUpDuration - minDurationMs
vibrator.vibrate(
  VibrationEffect.WaveformEnvelopeBuilder()
    .addControlPoint(0.1f, startFrequency, minDurationMs)
    .addControlPoint(0.1f, vibrator.resonantFrequency, rampUpDurationMs)
    .addControlPoint(0.1f, startFrequency, rampDownDurationMs)
    // Controlled ramp down to zero to avoid ringing after the vibration.
    .addControlPoint(0.0f, startFrequency, minDurationMs)
    .build()
    )
```

(Visit the Google booth to experience it first hand)

04
Frequency To Acceleration Map

Frequency to Output Acceleration Map (FOAM)

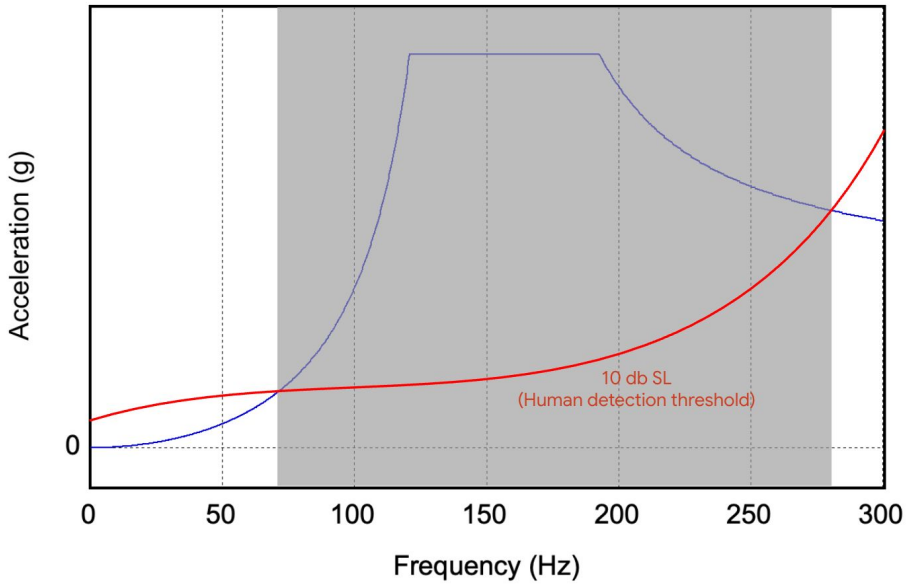
- FOAM defines the frequency to maximum achievable output acceleration curve
- Crucial for understanding vibration output across the frequency range
  - Essential for hardware aware haptic feedback
- Helps ensure vibration effects use frequencies within the device's capabilities
- FOAM can be retrieved via `VibratorFrequencyProfile` API



Frequency to Output Acceleration Map (FOAM)

- The shaded area defines the Basic API's sharpness range
- The range is the first segment where output acceleration remains above 10 dB SL
  - Starting from the first point it exceeds this level
  - Ending immediately before it falls below

(Just an illustration)



05
Vibrator Vendor APIs

Vibrator Vendor APIs

- Introducing @SystemApi to coordinate access to vibrator hardware
- Allow vendors to prototype and experiment
- Vendor privileged apps will send opaque VibrationEffects to vibrator HAL
    - Vendor effects

Vendor Effect

- Vendor extension @SystemApi for VibrationEffect
- Hardware-specific opaque effects sent directly to the vibrator
- Framework metadata sent separately
    - Like user settings intensity values

Vendor Vibration Session

- Vendor extension @SystemApi for vibration playback control
- Vibration session is initiated with fixed VibrationAttributes (usage, flags)
  - Async operation that will return the session once successfully started
  - Vibration usage prioritizes session vibrations against system vibrations (notifications, ringtones)
  - Once started, the framework grants the session sole control over the vibrator.

Vendor Vibration Session

- Session callbacks for state change
  - Session can be immediately interrupted by the vendor client
  - Session can be aborted by vibrator hardware or Android framework
  - Session closure is also asynchronous, with a final callback once the vibrator hardware has stopped

Vendor Vibration Session

- Session vibration effects are forwarded to the vibrator
  - Vibration effects can be sent through the session while it’s active
  - Effects are forwarded to the hardware immediately
  - Give flexibility for vendors to decide how to manage multiple effects:
    - Queueing: Sequential playback
    - Real-time updates: Modify or replace ongoing effects
    - Mixing: Combine multiple effects simultaneously

Part 3:
Pixel Implementation

Supporting Vibration Envelopes

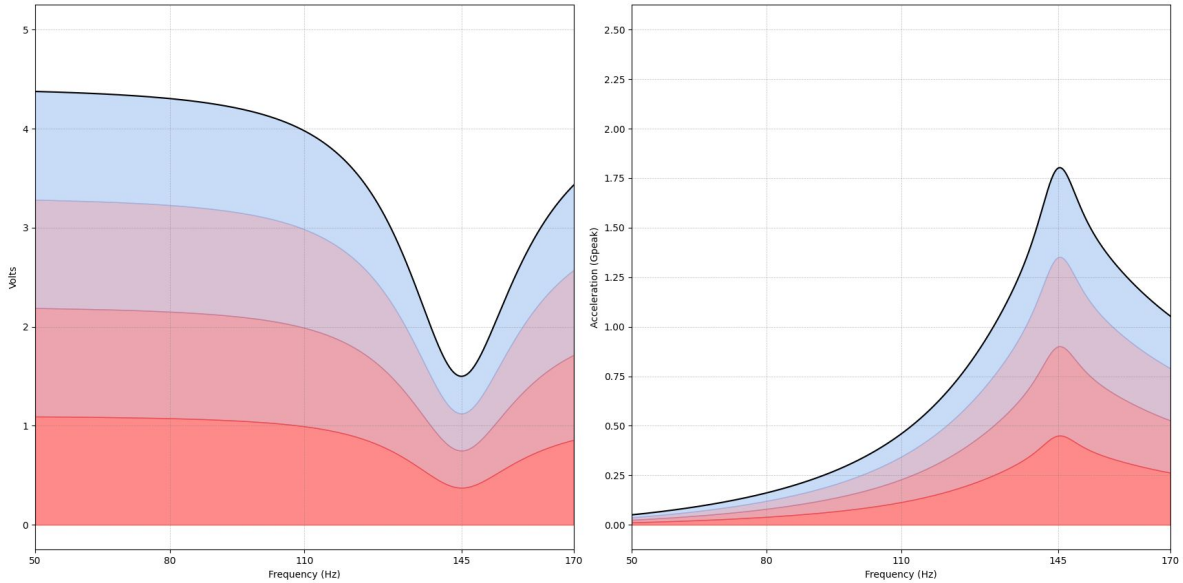
- OEM implements new AIDL methods for IVibrator service
  - IVibrator::composePwleV2
- Requires
  - Hardware support for dynamic amplitude and frequency
  - Actuator response profile

Pixel's System Software

- Pixel implements IVibrator::composePwleV2 using Cirrus' OWT Feature
- Additional mechanism is used for HW level device safety
- Device calibration data is used to calculate FOAM
    - Improve response range + scaling accuracy

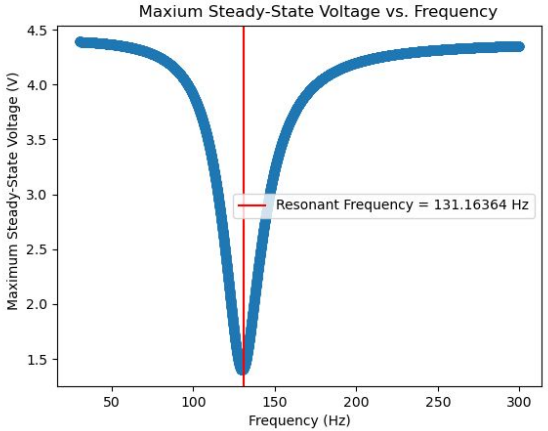
Pixel's FOAM

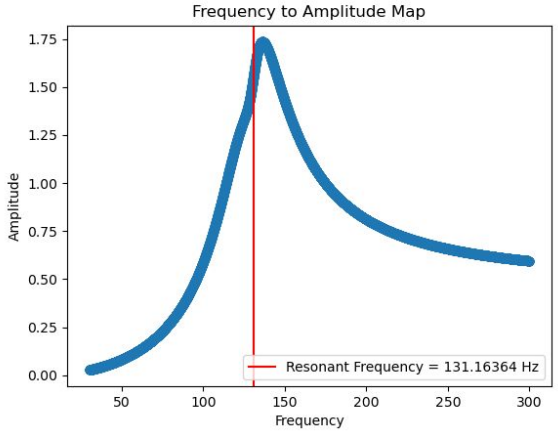
- Pixel uses open loop drive mode
  - Voltage x frequency -> acceleration model
- [0.0, 1.0] -> [0.0 V, max safe V]



Pixel Calibration

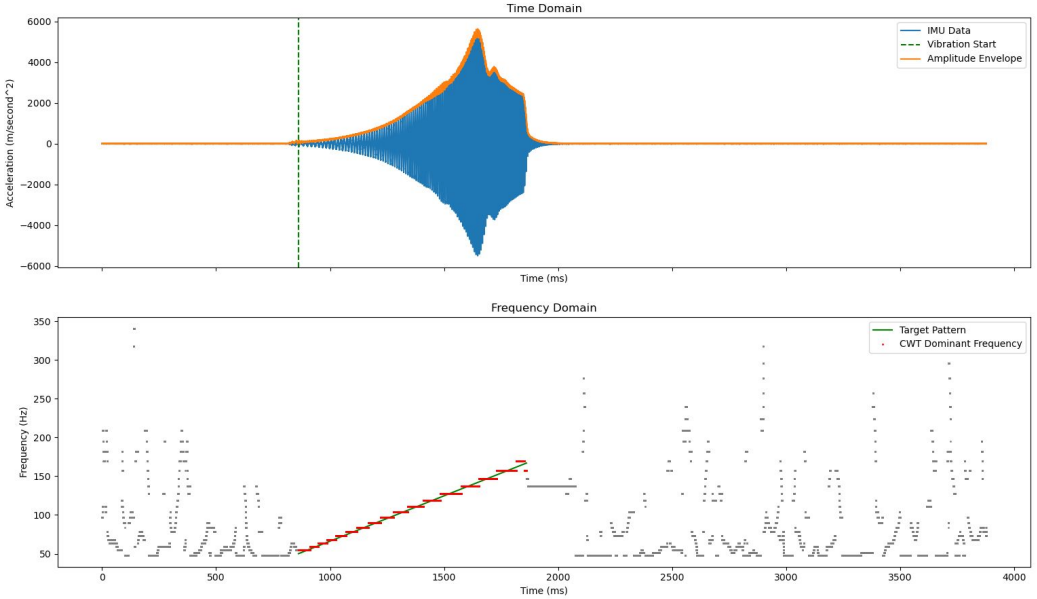
- Actuator properties are are calibrated in factory
- F0 calibration is particularly important because small variations can throw off actual scaling significantly near F0
- Q calibration is also very important because small variations will throw off bandwidth and can cause significant amplitude distortion





Pixel Validation

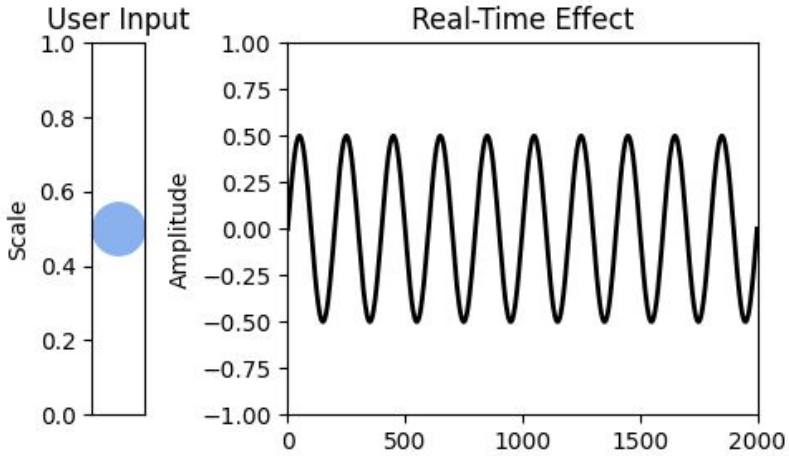
- Pixel has automatic waveform analysis tests that execute regularly and quantify FOAM accuracy and precision
  - Detect regressions in quality
  - Ensure minimum quality bar for support
- Integration tests check for regressions in stability and latency

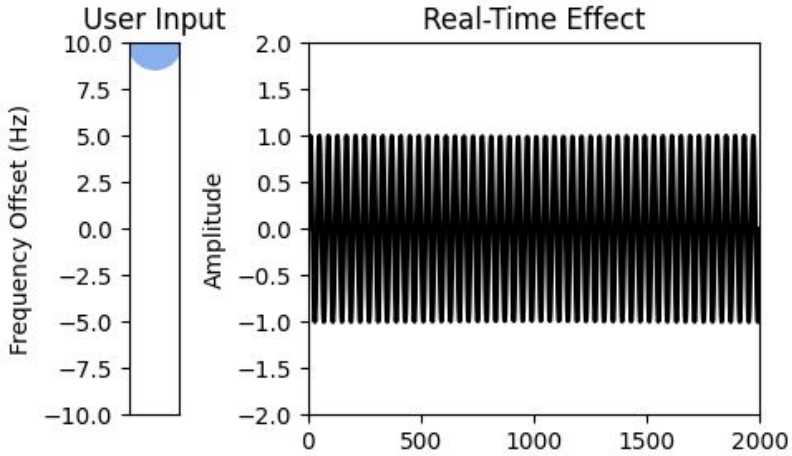


Real-Time PWLE on Pixel Devices

Real-Time Expansion

- Real-Time amplitude and frequency modulation for PWLE effects
- Pixel exploring several options for interface and implementation



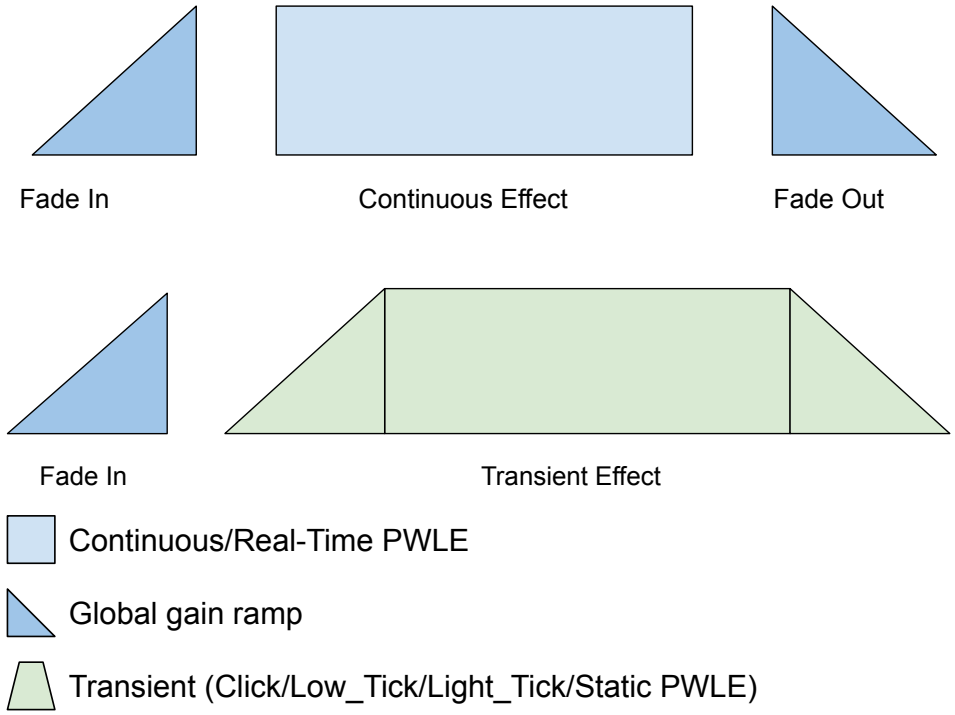


Continuous and Transient Effects

- Current demo adds new effect categories:

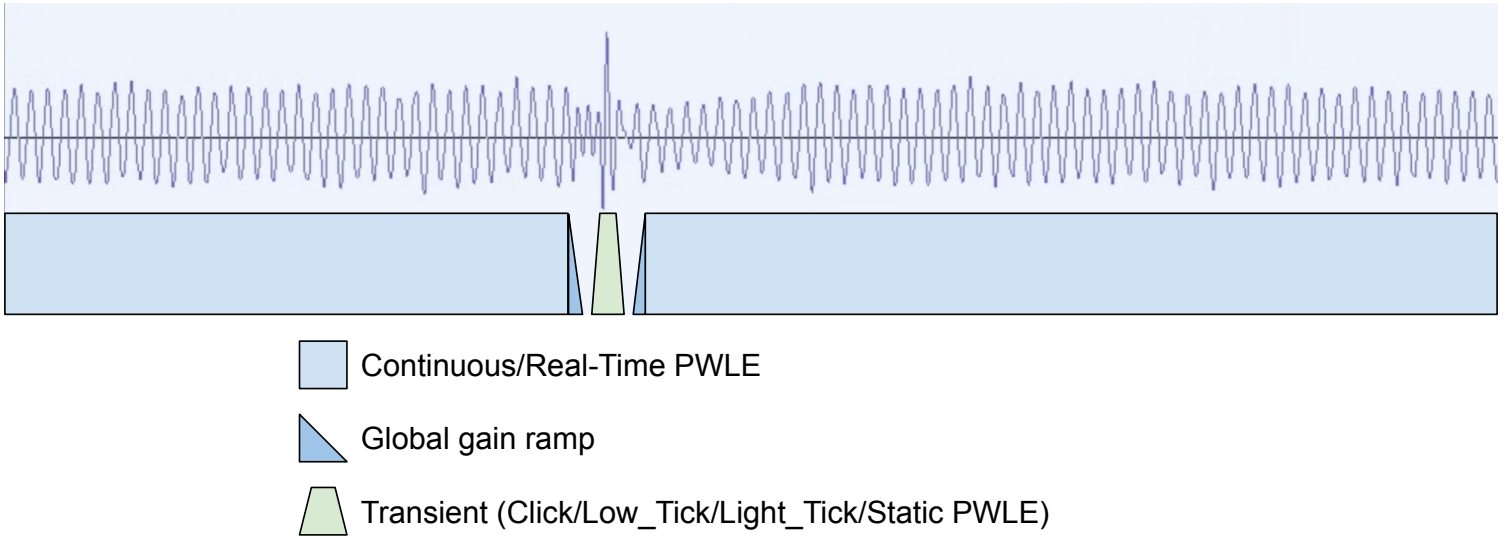
- Continuous Effects
  - Periodic definition
  - Indefinitely long

- Transient Effects
  - Definite duration
  - Fully defined like typical effects



Effect Interactions

- Transient Effects can interrupt:
  - Transient effects
  - Continuous Effects
- Continuous Effects can replace:
  - Continuous Effects



Thank you!

Contact Information

Ahmad Khalil
khalilahmad@google.com

Nathan Kulczak
nathankulczak@google.com

Hong Tan
hongtan@google.com

To Learn More

Android Documentation:
https://developer.android.com/reference/android/os/VibrationEffect.WaveformEnvelopeBuilder

Visit us at Google booth (#1)

Part 4:
It’s time for demos 😊