Synthesizing Physical Contact Sound in VR with Modal Resonators

A real-time modal synthesizer that turns physical contact events into sound: collision / friction / material params → DSP → audio.

Principle

1 Resonance 101

Objects prefer certain vibration frequencies. Strike one, and those frequencies ring out.

These are its **modes** — shaped by **shape, density, material, and boundary conditions**. An object's sound is the sum of its modes.

2 Single Mode

A mode needs only three numbers:

- **frequency** — how high
- **amplitude** — starting level
- **decay** — fade time

Those three are enough to synthesize that mode.

3 16 Modes

Real objects ring through many modes at once — often hundreds.

The ear tracks dominant modes, so this engine uses **16 modes** per material to preserve its timbral fingerprint.

4 Inharmonicity

Strings ring at integer multiples — harmonic, pitched.

Bells, plates, and glass scatter modes — inharmonic, noise-like.

Inharmonicity slides the modal distribution between those extremes.

Value

5 Recording Scale

Wood on metal? One take. Metal on plastic? Another. Contact sounds quickly become X × Y × Z assets.

Each material needs only **7 parameters**:

1. `freq`
2. `decay`
3. `hf_damping`
4. `inharmonicity`
5. `tonality`
6. `hardness`
7. `roughness`

6 Live Morph

Recordings replay a fixed take. Synthesized sound can **change while it rings** — move a material slider and the tail morphs live.

The audio-thread DSP reads Unity C# parameter updates instantly: hardness, tonality, and more. No restart, no buffer swap. *Box 12 demos this live.*

Physics-Driven

7 Physics → Sound

Each collision gives **impulse, contact point, normal, and PhysicsMaterial** from Unity physics.

A **normal / tangential** split drives two paths: normal → **impact**, tangential → **friction**. One physical event, two independent exciters. *See back, S3 / S4.*

8 Coupled Materials

A metal ball on wood is **both objects sounding together**. One voice holds both materials so each side can damp the other.

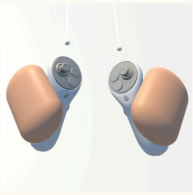
Each voice runs two mode banks (A / B). During contact, B shortens A's decay — steel on skin: 1.8s → 0.45s. This is **cross-damping**. *See back, S6 / S7.*

Demo — Try It Yourself

9 Physical Hands

Put on the Quest and raise your hands. The visible hands are **physics rigidbodies**, not just meshes.

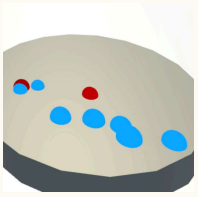
They push, hit, and grab with real collision response — the shared entry point for every demo.



10 Contact Markers

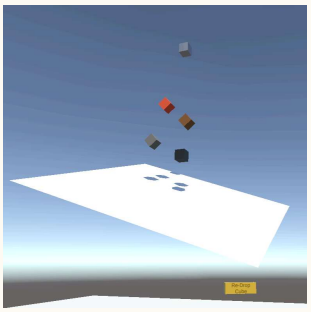
Impacts draw a **red dot** at contact; size = impulse. Friction draws **blue ticks** along the path; size = slide speed.

Audio, haptics, and visuals are locked to the same physics event.



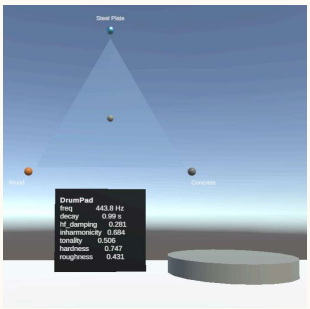
11 Ramp Drop

Press the button. The cube drops onto the ramp, then rolls and slides with generated contact sound.



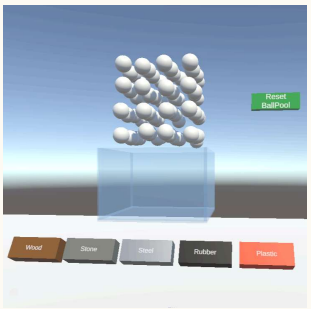
12 Tri-Mixer

Three corners, three materials. Drag the puck and the drumpad's timbre follows in real time.



13 Ball Pool

Reach in and stir. Dense granular contact sound emerges naturally; swap materials live.



Game Integration

14 Voice Pool

At startup, the DSP pre-allocates **64 voices** — each with its own mode bank, state, and output stream.

Each collision **borrows** an idle voice and **returns** it when silent. No runtime allocation: Quest 3's mobile CPU cannot afford it.

15 Physical Grabbing

Default XRI grabbing makes objects "float" in the hand — clipping through walls and making forward/back throws feel weak.

Here, the grabbed cube **stays a physics rigidbody**: collisions push back, and throws follow controller kinematics.

Lateral throws get a lever boost: wrist rotation becomes high velocity at the held object. **Forward/back throws** lack that boost.

Fix: on release, add forward velocity **proportional to grab distance**. Not strict physics, but it **matches player intuition**.

16 Audio-Driven Haptics

Controller vibration **tracks the sound's amplitude envelope in real time** — hand and ear stay on the same timeline.

Spatial Audio

17 HRTF Spatialization

Each voice position goes into **Steam Audio** and is rendered through a generic **HRTF (head-related transfer function)**.

HRTF: besides left/right level and timing, localization also depends on tiny EQ / delay cues from the pinna, head, and torso. HRTF folds those cues into stereo, so headphones can still convey front/back and up/down.

Audio System Architecture

18 Unity ↔ DSP

From Faust source to headphones: 5 single-purpose layers, one C# / C++ file each. *DSP signal flow on the back.*

1. **DSP** — `modal-resonator.dsp` → C++ `sound_engine.cpp` → `SE_*` plugin for Win / Mac / Android
2. **Bridge** — `NativeDSP.cs` exposes init, voice allocation, material pair, impact, contact, and compute calls
3. **Pool** — `VoicePool.cs` calls `SE_Init(sr, 64)` and creates 64 `SpatialVoice` objects
4. **Logic** — `CollisionSoundManager.cs` reads collision physics and updates the DSP
5. **Output** — `SpatialVoice` → spatialized `AudioSource` → `SE_Compute` → Steam Audio HRTF → headphones

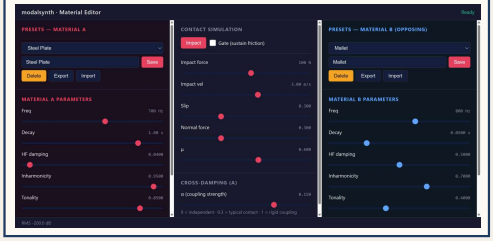
Material Editor — Now Live

19 WebUI Material Tuning

Drag 7 material sliders in the browser and hear the result live — no VR headset required.

Same DSP source as the VR demo: **Web, Unity, and Quest compile from this one file.**

`modalsynth.mofei.me`



Mofei Li

`https://mofei.me/` · `mofei_li@qq.com` · GitHub: `github.com/MofeiL1`

Faust DSP Signal Flow

8 Stages from Unity Collision to Steam Audio HRTF

https://mofei.me/

Normal / tangential split: the normal component drives the **impact path** (the strike moment), while the tangential component drives the **friction path** (continuous sliding). This is the physical basis for separating the two exciter paths in S3 / S4.

S1 Unity → DSP Input

For each collision, Unity physics outputs several values. `CollisionSoundManager` extracts them and sends them into DSP sliders:

One-shot (OnCollisionEnter): `impact_force` (impulse / dt = force), `impact_vel` (relative speed).

Per-tick (FixedUpdate): `gate` (0/1 contact), `tangent_vel` (tangential sliding speed), `normal_force` (normal force), `friction_coeff` (μ, from PhysicsMaterial).

Plus **14 material parameters** (7 × 2 sides), read from the two SoundMaterials.

```
tangentVel = relVel - (relVel · n) · n
```

These two combined values run through both the impact and friction exciters, defining the contact surface's **contact dynamics**: spectral brightness and asperity noise depth.

S2 Contact-Pair Properties

Hardness / roughness from the two materials are not simply averaged; each uses a physically motivated combination. The harmonic mean lets the **softer side dominate contact stiffness**; RSS (root-sum-square, adding independent variances) combines asperity height distributions from two independent rough surfaces.

```
combined_hardness  = 2·H_A·H_B / (H_A+H_B)        combined_roughness = √(R_A² + R_B²)
```

Impact behaves like a **Hertzian contact stress wave**: stiffer contact → steeper pulse → brighter spectrum. Steel-Steel ≈ 8.5 kHz click; Steel-Rubber ≈ 2.6 kHz thud.

S3 Impact Exciter

One-shot collision trigger:

- **env(AR)**: 0.5ms attack; 1–21ms release from combined_hardness
- **amp = √(F • v)**: power-equivalent law shared with friction
- **cutoff**: hardness + velocity brighten the hit; impedance mismatch darkens it

```
exciter_impact = noise · env(AR)
  · √(F·v) → lowpass(cutoff_imp)
```

S4 Friction Exciter

Continuous while contact holds:

- `slip` = tangent_vel: asperity hit rate
- `√(N·μ)` = peak asperity force
- `η` = 0.03 + roughness • 0.2: radiation eff
- `am_mod` adds random asperity timing — **granular low-speed → hissy high-speed**

```
cont_amp = slip · √(N·μ) · η · gate
am_mod   = 1 - depth · noise
         · lp(40+slip·3000)
exciter_fric = noise · cont_amp
   · am_mod → lp(cont_cutoff)
```

Rough-surface asperities collide and create broadband noise. Roughness shapes spectrum; normal force sets loudness; tangential speed sets granularity / hiss.

The whole DSP uses only one global `master_gain`. Impact and friction share the same coefficient, so their relative loudness must come from physical parameters — no perceptual fudge like "make impact louder than friction."

S5 Unified Exciter • Fan-Out

Add impact and friction together, then multiply by `master_gain`.

The same exciter signal then fans out to the A-side and B-side mode banks.

```
exciter = (exciter_impact
        + exciter_friction) · 0.012

exciter ─┬─→ Side A
         └─→ Side B
```

Cross-damping **changes only decay**, not damping_exp (the high-frequency rolloff shape). The latter belongs to material A itself and should not change just because it is touching B.

S6 Cross-Damping

While contact is held, side A's effective damping receives damping from side B, and vice versa. Energy leaks into B and is dissipated by B's viscoelastic damping, shortening A's decay. `cd_alpha` controls coupling strength; **the demo uses 0.15** (DSP default: 0.3).

Steel + Steel	1.8s → 1.57s (matched, weak coupling)
Steel + Skin	1.8s → **0.45s** (strong coupling)
Rubber + Steel	almost independent (extreme impedance mismatch)

```
hf_damp_eff_A = hf_damp_A
  + s_gate · cd_alpha · hf_damp_B
decay_eff_A = decay_A
  · (hf_damp_A / hf_damp_eff_A)
```

Mode-bank controls:

- `inharmonicity` → mode frequency spread (harmonic → bar/plate-like)
- `tonality` → resonance sharpness (clear / pitched → noisy / smeared)
- `hf_damping` → high-frequency decay (metallic ring → dull thud)

S7 Mode Bank A

Each side runs 16 parallel `resonbp` filters (resonant bandpass), turning the exciter signal into modal vibration:

```
ratio_exp   = 1 + 2.3·B - 2.6·B²
damping_exp = hf_damping · 3
F_i    = freq · (i+1)^ratio_exp
T60_i  = decay_eff
       · (freq/F_i)^damping_exp
G_i    = 1/√(i+1)
```

S7 Mode Bank B

Side B uses material B's parameters (`freq_B`, `decay_eff_B`, `hf_damping_B`, `inharmonicity_B`, `tonality_B`). The same formulas act on a different set of partials.

16 modes × 2 sides = 32 modes ringing together, all sharing the same exciter signal.

The tradeoff: dual-bank inside one voice, but only one spatializer. On Quest, Steam Audio HRTF is the real bottleneck (each extra voice adds another HRTF kernel, roughly 100× the cost), while extra modal banks inside the DSP are almost free (~16 IIR states).

Putting impact + friction + A + B into one spatializer saves **50% spatializer cost** compared with a dual-source design.

Diffuse texture is a material-specific high-frequency rasp that fills in randomness not covered by the modal bank. Lower tonality increases texture weight — this layer is why skin friction sounds sandier than steel.

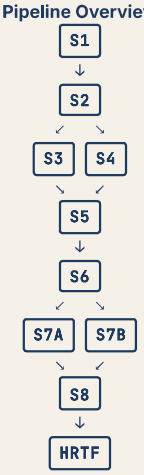
S8 Diffuse Texture + Output

Each side adds 4 inharmonic high-frequency texture modes on top of its 16 modal modes (frequency ratios **1.7 / 3.3 / 5.7 / 9.1**), weighted by `(1-tonality)·0.5`.

The whole voice finally outputs **one mono signal** → Unity Steam Audio → HRTF spatialization into 3D stereo (*see front-side Box 17*).

```
side_A = modal_bank_A + diffuse_bank_A
side_B = modal_bank_B + diffuse_bank_B
output = (side_A + side_B)
       → tanh(soft clip) → duplicate to stereo
```

Pipeline Overview



Symbols

`F` : impact force
`v` : relative speed
`N` : normal force
`μ` : friction coef
`η` : radiation efficiency
`gate` : contact on/off
`T_AB` : impedance match
`B` : inharmonicity
`cd_α` : cross-damping coupling
`master_gain` : global gain
`F_i/T60_i/G_i` : mode i freq/decay/gain