

# SONIC BYTE



Anton Arhipov

# Table of Contents

Introduction

## Part I: Foundations

### 1. The Vision

    1.1. What is Dark Clubbing?

    1.2. Our Influences

    1.3. The Album Concept

### 2. Sonic Pi Essentials

    2.1. Playing Sounds

    2.2. Timing and Sleep

    2.3. Variables and Functions

    2.4. Threads and Concurrency

    2.5. Effects (FX)

## Part II: Sound Design

### 3. Building Sounds

    3.1. Choosing Synths

    3.2. Layering Techniques

    3.3. The ADSR Envelope

    3.4. Filters and Cutoff

### 4. Drum Programming

    4.1. Kick Drums That Hit Hard

    4.2. Snares and Claps

    4.3. Hi-Hats and Percussion

    4.4. Building Drum Patterns

### 5. Bass Design

    5.1. The TB303 Sound

    5.2. Layering Bass

    5.3. Writing Bass Patterns

### 6. Leads and Melodies

    6.1. Synth Selection for Leads

    6.2. Writing Dark Melodies

### **6.3. Arpeggios**

## **7. Generative Patterns**

### **Part III: Arrangement**

## **8. Song Structure**

### **8.1. Intro and Outro**

### **8.2. Builds and Drops**

### **8.3. Tension and Release**

### **8.4. DJ-Friendly Arrangements**

## **9. Dynamics and Energy**

### **9.1. Volume Automation**

### **9.2. Filter Sweeps**

### **9.3. Using Effects for Energy**

## **10. Going Live**

### **Part IV: The Album**

## **11. Track Walkthroughs**

### **11.1. 01: System Override**

### **11.2. 02: Nerve Damage**

### **11.3. 03: Chrome Cathedral**

### **11.4. 04: Skull Fracture**

### **11.5. 05: Midnight Protocol**

### **11.6. 06: Void Walker**

### **11.7. 07: Core Meltdown**

### **11.8. 08: Terminal Velocity**

### **Appendix**

## **12. Quick Reference**

## **13. Synth Cheat Sheet**

## **14. Scale Reference**

## **15. Troubleshooting**

## **16. Resources**

---

# A Sonic Byte

---

*A guide to programming dark electronic music with Sonic Pi*

---

---

## Listen First

Before reading a single line of code, [listen to the album on SoundCloud](#).

Eight tracks. Forty minutes. All made with code.

Hear what we're building. Then come back and learn how it's done.

---

## Welcome

You're holding a tutorial that will teach you how to create a complete 8-track dark electronic album using nothing but code. No DAW. No piano roll. No mouse clicking. Just pure, expressive code.

**Sonic Byte** — 8 tracks of dark clubbing, industrial electronic, and darksynth music, composed entirely in [Sonic Pi](#). This book will teach you how it was made, and give you the skills to create your own.

## Why Code?

In a traditional DAW, you click and drag. You draw notes on a grid. You twist virtual knobs with a mouse.

In Sonic Pi, you type this:

```
play scale(:d4, :minor).choose
```

One line. A random note from D minor. Every time.

Or this:

```
16.times do |i|
  play :d2, cutoff: 50 + i*4
  sleep 0.25
end
```

A bass note that gets brighter with each hit. The filter opens automatically. No automation lanes. No clicking. Just *describe what you want* and it happens.

Code lets you:

- **Generate patterns** that never repeat exactly the same way
- **Control everything** with variables and math
- **Build systems** that make creative decisions for you
- **Share your music** as text anyone can run

This isn't about replacing DAWs. It's about discovering what's possible when music becomes code.

## What You'll Learn

By the end of this tutorial, you'll understand:

- **Sound Design** — How to craft punishing kicks, grinding basslines, and atmospheric pads
- **Rhythm Programming** — Building drum patterns that drive the dancefloor
- **Melody Writing** — Creating dark, emotional melodies that cut through the mix

- **Arrangement** — Structuring tracks with builds, drops, and DJ-friendly sections
- **The Power of Code** — Using functions, threads, and generative patterns to create music that evolves

## Who This Book Is For

This tutorial assumes:

- **Basic programming knowledge** — You understand variables, functions, and loops
- **Some musical interest** — You don't need to read music, but curiosity helps
- **Sonic Pi installed** — Download it free at [sonic-pi.net](http://sonic-pi.net)

---

**New to Sonic Pi?** The [official tutorial](#) is excellent for learning the basics. We'll cover what you need here, but the built-in docs go deeper.

---

## The Album

Here's what we're building:

#	Track	BPM	Key	Character
01	System Override	100	D Minor	Aggressive opener
02	Nerve Damage	105	E Minor	Industrial crusher
03	Chrome Cathedral	98	A Minor	Atmospheric cyberpunk
04	Skull Fracture	108	F Minor	Maximum aggression

#	Track	BPM	Key	Character
05	Midnight Protocol	102	C Minor	Synthwave triumph
06	Void Walker	95	B Minor	Dark power
07	Core Meltdown	106	G Minor	Peak-time climax
08	Terminal Velocity	100	D Minor	Cinematic finale

Each track teaches different techniques. By the final chapter, you'll have built all eight.

## How to Use This Book

**Part I: Foundations** covers the vision and Sonic Pi basics. Start here if you're new.

**Part II: Sound Design** dives deep into creating individual elements — drums, bass, leads, and effects.

**Part III: Arrangement** teaches you how to structure complete tracks.

**Part IV: The Album** walks through each track, explaining the creative and technical decisions. Each chapter ends with **Hacker Challenges** — modifications for you to try.

**Appendix** provides quick reference materials you'll use constantly.

---

Let's make some noise.

```
use_bpm 100
sample :bd_tek, amp: 2.5, rate: 0.9
sleep 1
sample :bd_tek, amp: 2.5, rate: 0.9
```

*Press Run.*

# The Vision

Before we write a single line of code, let's understand what we're creating — and why.

## Why Code Music?

In a DAW, you drag. You click. You twist virtual knobs that look like real knobs but don't feel like anything.

In code, you *describe*. You say what you want, and it happens.

```
16.times do |i|
  play :d2, cutoff: 50 + i*4
  sleep 0.25
end
```

That's a bass note that gets brighter with each hit. Automatic filter sweep. No automation lanes. No clicking through menus. Just: *here's what I want*.

This isn't better or worse than a DAW. It's different. It's thinking about music as **systems** instead of **events**. And once you think that way, you can build things that would be tedious or impossible otherwise.

## The Aesthetic

**Sonic Byte** draws from several dark electronic genres:

- **Dark Clubbing** — Heavy, driving beats designed for the dancefloor
- **Darksynth** — 80s-inspired synths with a modern, aggressive edge
- **Industrial Electronic** — Mechanical textures, metallic percussion
- **Cyberpunk** — Futuristic, dystopian atmospheres

The common thread: **darkness with energy**. This isn't ambient music. It's music that moves bodies while unsettling minds.

## The Rules We Follow

Drawing from artists like Irving Force and Noisecream, we established production principles:

### 1. DJ-Functional Structure

Even if you never DJ, building tracks with clean structure makes them more impactful:

- 16 or 32-bar phrases
- Intro/outro with minimal melodic clutter
- Energy changes through layering, not constant new ideas

### 2. Aggression in the Midrange

The secret to heavy electronic music:

- Keep sub-bass (20-80 Hz) **clean and consistent**
- Put grit and aggression in **100 Hz - 4 kHz**
- This lets tracks hit hard without becoming muddy

### 3. The Hook Is a Sound

In pop music, the hook is a melody. In dark electronic:

- The hook is often a **sound** — a bass stab, an alarm, a texture
- Variation comes through **automation** — filter sweeps, distortion changes
- Repetition with subtle evolution creates hypnosis

## 4. Tension Through Subtraction

The most powerful drops happen after you **remove** elements:

- Strip the kick for 1-4 bars
- High-pass filter the bass
- Let reverb tails breathe
- Then slam everything back

## What Makes This Album Work

Each track in Sonic Byte serves a purpose:

1. **System Override** — Establishes the sound, aggressive opener
2. **Nerve Damage** — Pushes aggression further with industrial textures
3. **Chrome Cathedral** — Provides atmospheric contrast, a breath
4. **Skull Fracture** — Maximum energy, the first peak
5. **Midnight Protocol** — Melodic relief, synthwave influences
6. **Void Walker** — Slow power, building tension
7. **Core Meltdown** — The climax, everything combined
8. **Terminal Velocity** — Resolution, fading into darkness

This arc — tension, release, build, climax, resolution — is what makes an album feel like a journey, not just a collection of tracks.

---

Now let's learn the tools to build it.

# What is Dark Clubbing?

Dark clubbing is a subgenre of electronic music that merges the driving rhythms of club music with darker, more aggressive sonic textures. It sits at the intersection of several genres:

## The Sound

**Tempo:** Typically 95-115 BPM — slower than techno, faster than downtempo. This “midtempo” range creates a hypnotic, head-nodding groove rather than frantic energy.

**Bass:** Heavy, often distorted. The sub-bass provides physical weight while mid-bass frequencies carry grit and aggression.

**Drums:** Punishing kicks, industrial snares, metallic percussion. The beat is relentless but not necessarily fast.

**Atmosphere:** Dark pads, ominous textures, dystopian soundscapes. Think abandoned warehouses, not sunny beaches.

**Melody:** When present, melodies are often minor key, sometimes dissonant, but can be hauntingly beautiful.

## Related Genres

Dark clubbing draws from and overlaps with:

### Darksynth

80s-inspired synthesizers with modern production aggression. Artists like Perturbator and Carpenter Brut blend retro aesthetics with extreme intensity.

## Industrial

Mechanical, metallic, abrasive. Think Nine Inch Nails, Front 242. Industrial brings textures that sound like machines and factories.

## Techno

The four-on-the-floor foundation. Dark clubbing inherits techno's hypnotic repetition but slows it down and adds more harmonic content.

## Midtempo Bass

The heavier side of bass music at slower tempos. Artists like REZZ and 1788-L create hypnotic, dark grooves.

## The Aesthetic

Dark clubbing isn't just a sound — it's a visual and thematic aesthetic:

- **Cyberpunk:** Neon-lit dystopias, corporate oppression, technological anxiety
- **Horror:** Cinematic tension, dread, the supernatural
- **Industrial:** Machines, factories, dehumanization
- **Occult:** Esoteric symbolism, ritual, darkness

The music often sounds like a soundtrack to a film that doesn't exist — and that's intentional. Many dark clubbing producers cite film composers like John Carpenter as primary influences.

## Why Code Dark Clubbing?

This genre is particularly well-suited to Sonic Pi for several reasons:

1. **Repetition with variation** — The hypnotic nature rewards subtle changes over time, which code handles elegantly
2. **Sound design focus** — Dark clubbing is more about *how* sounds are sculpted than complex chord progressions
3. **Modular structure** — Tracks are built from layers that can be added/removed, perfect for functional programming
4. **Parameter automation** — Filter sweeps, amplitude changes, and effect automation are core to the genre

## Listening Recommendations

Before diving into production, immerse yourself in the genre:

- **Gesaffelstein** — “Pursuit”, “Hate or Glory”
- **REZZ** — “Edge”, “Relax”
- **1788-L** — “Pulsar”, “S Y N T H E T I K”
- **Irving Force** — “The Violence Suppressor”, “Godmode”
- **Noisecream** — “Dominance”, “The Dooms Party”
- **Carpenter Brut** — “Turbo Killer”, “Roller Mobster”

Listen for:

- How the kick and bass interact
- When elements enter and exit
- How tension is built and released
- The balance between aggression and groove

---

Now that you understand the genre, let's look at the specific artists who influenced this album.

# Our Influences

Every piece of music builds on what came before. Here are the artists who shaped the sound of *Sonic Byte*, and what we learned from each.

## Irving Force

**Style:** Darksynth, cyberpunk, film score-influenced

**Key Tracks:** "The Violence Suppressor", "Send Me Your Dreams", "Night Hunter"

### What We Learned:

- **Narrative structure** — Irving Force's tracks feel like movie scenes. There's a beginning, middle, and end, not just a loop.
- **Metal influences** — Despite being electronic, his music has the intensity of metal. Don't be afraid to push aggression.
- **Layered leads** — Multiple synth layers create rich, evolving melodies without complex chord progressions.

**Applied In:** Track 1 (System Override), Track 4 (Skull Fracture)

```
# Irving Force-inspired aggressive lead
define :lead do |n, v=1|
    use_synth :prophet
    play n, amp: 0.4*v, attack: 0.02, decay: 0.2,
           sustain: 0.1, release: 0.3, cutoff: 95
    use_synth :dark_ambience
    play n, amp: 0.15*v, attack: 0.05, release: 0.5
end
```

## Noisecream

**Style:** Aggressive electronic, game soundtracks, industrial

**Key Tracks:** "Dominance", "The Dooms Party", "Power Up"

### What We Learned:

- **Intensity without chaos** — Noisecream tracks are brutal but controlled. Every element has purpose.
- **Hook sounds** — The hook isn't always a melody; often it's a distinctive bass sound or rhythmic pattern.
- **Dynamic range** — Quiet moments make loud moments louder. Don't be afraid of space.

**Applied In:** Track 2 (Nerve Damage), Track 7 (Core Meltdown), Track 8 (Terminal Velocity)

```
# Noisecream-inspired grinding bass
define :grind do |n, v=1, c=80|
    use_synth :tb303
    play n, amp: 0.8*v, attack: 0.01, decay: 0.2,
        cutoff: c, res: 0.35, wave: 0
    use_synth :sine
    play n-12, amp: 1.1*v, sustain: 0.25, release: 0.15
end
```

## Gesaffelstein

**Style:** Dark clubbing, industrial techno, minimal

**Key Tracks:** "Pursuit", "Hate or Glory", "Viol"

### What We Learned:

- **Less is more** — Gesaffelstein tracks often use very few elements, but each one is perfectly crafted.
- **The power of silence** — Strategic silence creates impact when sound returns.
- **Bass as foundation** — The bass isn't just low end; it's the primary melodic element.

**Applied In:** Track 3 (Chrome Cathedral), Track 6 (Void Walker)

```
# Gesaffelstein-inspired minimal pattern
# Notice the space – not every beat has a sound
define :minimal_bass do |v=1|
    bass :d2, v; sleep 1.5
    sleep 0.5
    bass :d2, v*0.6; sleep 0.5
    sleep 1.5
end
```

## 1788-L

**Style:** Midtempo bass, cyberpunk, heavy

**Key Tracks:** "Pulsar", "SYNTHETIK", "Replica"

### What We Learned:

- **Sound design over songwriting** — The sounds themselves are the composition.
- **Slow builds** — Tension can build over 32+ bars before release.
- **Sub bass presence** — The physical weight of sub frequencies is essential.

**Applied In:** Track 4 (Skull Fracture), Track 6 (Void Walker)

## REZZ

**Style:** Hypnotic bass, dark electronic, minimal

**Key Tracks:** "Edge", "Relax", "Witching Hour"

### What We Learned:

- **Hypnotic repetition** — The same phrase repeated with subtle variations creates trance-like states.

- **Tension without release** — Not every build needs a massive drop.
- **Signature sounds** — Develop sounds that are instantly recognizable as “yours”.

**Applied In:** Track 3 (Chrome Cathedral), Track 5 (Midnight Protocol)

## Carpenter Brut

**Style:** Synthwave, darksynth, metal-influenced

**Key Tracks:** “Turbo Killer”, “Roller Mobster”, “Le Perv”

### What We Learned:

- **Triumphant melodies** — Even dark music can have uplifting moments.
- **Genre fusion** — Blend influences freely. Rules are guidelines.
- **Energy arcs** — Albums should take listeners on a journey.

**Applied In:** Track 5 (Midnight Protocol), Track 8 (Terminal Velocity)

```
# Carpenter Brut-inspired triumphant arp
define :arp1 do |v=1|
  use_synth :pulse
  [:d4,:f4,:a4,:d5,:a4,:f4,:d4,:a3].each do |n|
    play n, amp: 0.25*v, attack: 0.01, decay: 0.1,
      release: 0.1, cutoff: 105
    sleep 0.5
  end
end
```

## Synthesis: Our Sound

By studying these artists, we developed principles for *Sonic Byte*:

Principle	Source
Narrative structure	Irving Force

Principle	Source
Controlled aggression	Noisecream
Strategic silence	Gesaffelstein
Sound design focus	1788-L
Hypnotic repetition	REZZ
Triumphant moments	Carpenter Brut

The goal isn't to copy any single artist, but to synthesize their lessons into something new.

---

Next, let's see how these influences shaped the album's overall concept.

# The Album Concept

*Sonic Byte* isn't just 8 random tracks — it's a designed journey. Understanding this structure will help you create cohesive albums yourself.

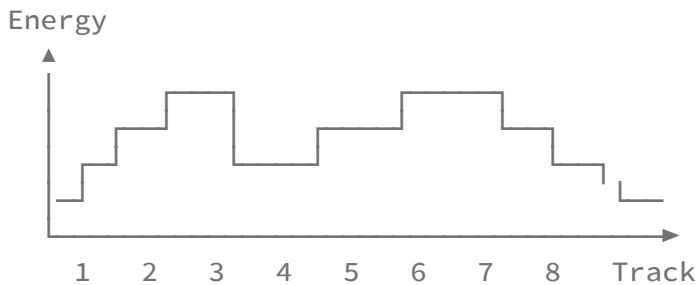
## The “Sonic Byte” Concept

8 tracks. 8 bits. 1 byte.

The name is a programming pun, but it's also practical: 8 tracks is enough to develop ideas without overstaying welcome. Each track can represent a “bit” of the overall message.

## The Energy Arc

Albums work best when they take listeners somewhere. Here's the energy map of *Sonic Byte*:



- **Track 1-2:** Establish aggression, set expectations
- **Track 3:** Breathing room, atmospheric contrast
- **Track 4:** First peak, maximum aggression
- **Track 5:** Melodic relief, emotional contrast
- **Track 6:** Slow build, gathering power
- **Track 7:** Album climax, everything combined

- **Track 8:** Resolution, fading to darkness

## Track-by-Track Breakdown

### 01: System Override (100 BPM, D Minor)

**Role:** Album opener, establish the sound

The first track sets expectations. It needs to be:

- Immediately engaging (no 2-minute intro)
- Representative of the album's sound
- Complete enough to stand alone

D Minor was chosen as the “home key” — we return to it in Track 8.

### 02: Nerve Damage (105 BPM, E Minor)

**Role:** Push aggression further

With expectations set, Track 2 can push harder. The tempo increase (100→105) and industrial textures signal: this album means business.

### 03: Chrome Cathedral (98 BPM, A Minor)

**Role:** Atmospheric contrast, breathing room

After two aggressive tracks, listeners need a breath. Chrome Cathedral is slower, more atmospheric, but still dark. It’s the calm before the next storm.

### 04: Skull Fracture (108 BPM, F Minor)

**Role:** First peak, maximum aggression

The fastest, most aggressive track. This is where the album hits hardest — positioned early enough that there's still journey ahead.

## **05: Midnight Protocol (102 BPM, C Minor)**

**Role:** Melodic relief, emotional contrast

After the violence of Track 4, listeners need emotional variety. This track introduces triumphant synthwave elements while maintaining the dark foundation.

## **06: Void Walker (95 BPM, B Minor)**

**Role:** Slow build, power accumulation

The slowest track, but not the quietest. Void Walker builds power through restraint — each element hits harder because there's space around it.

## **07: Core Meltdown (106 BPM, G Minor)**

**Role:** Album climax, everything combined

This is where everything comes together. Core Meltdown combines the aggression of Track 4, the melody of Track 5, and the power of Track 6. It's the emotional peak.

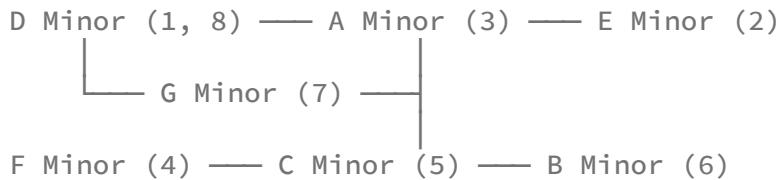
## **08: Terminal Velocity (100 BPM, D Minor)**

**Role:** Resolution, return home

We end where we began — 100 BPM, D Minor. But now the context has changed. The same key feels like resolution rather than beginning. The track fades into darkness, completing the journey.

# Key Relationships

The tracks are connected by key signatures:



Related keys share notes, making transitions feel natural even when tracks are played separately.

# BPM Flow

Track:	1	2	3	4	5	6	7	8
BPM:	100	105	98	108	102	95	106	100
—	↑↑	↓↓	↑↑	↓	↓↓	↑↑	↓	

The BPM rises and falls with energy:

- Aggressive tracks (2, 4, 7) are faster
- Atmospheric tracks (3, 6) are slower
- Bookends (1, 8) share the same tempo

# Planning Your Own Album

When planning an album, consider:

1. **What's the journey?** Beginning → middle → end
2. **Where's the peak?** Usually 2/3 through
3. **What's the home key?** Return to it at the end
4. **Where's the contrast?** You need breathing room

## **5. What connects the tracks?** Key relationships, tempo flow, thematic elements

An album is more than a collection — it's a composition at a larger scale.

---

Now that you understand what we're building and why, let's learn the tools to build it.

# Sonic Pi Essentials

This chapter covers the core Sonic Pi concepts used throughout the album. If you're familiar with Sonic Pi, skim for the patterns we use. If you're new, this is your foundation.

---

**Want to go deeper?** The [Sonic Pi built-in tutorial](#) is excellent and comprehensive. We'll cover what you need here, but the official docs go further.

---

## The Basics

Sonic Pi is a live coding synthesizer. You write code, press Run, and hear music immediately.

```
play 60 # Play middle C
sleep 1 # Wait 1 beat
play 64 # Play E
sleep 1
play 67 # Play G
```

That's a C major arpeggio. Simple, but it demonstrates the core loop: **play something, wait, repeat**.

## BPM: Setting the Tempo

Every track in our album starts with:

```
use_bpm 100
```

This sets beats per minute. When you `sleep 1`, you wait one beat — at 100 BPM, that's 0.6 seconds.

Our album ranges from 95-108 BPM:

Track	BPM
Void Walker	95
Chrome Cathedral	98
System Override	100
Terminal Velocity	100
Midnight Protocol	102
Nerve Damage	105
Core Meltdown	106
Skull Fracture	108

## Playing Samples

Samples are pre-recorded sounds. Sonic Pi includes many:

```
sample :bd_tek      # Techno kick drum
sample :sn_dub      # Dub snare
sample :drum_cymbal_closed  # Hi-hat
```

You can modify samples with parameters:

```
sample :bd_tek, amp: 2.5, rate: 0.9
```

- `amp` — Volume (1 is default, 2 is twice as loud)
- `rate` — Playback speed (0.5 is half speed/lower pitch, 2 is double speed/higher pitch)

## Key Sample Parameters

```
sample :bd_tek,  
amp: 2,          # Volume  
rate: 0.9,       # Speed/pitch  
attack: 0,        # Fade in time  
release: 0.3,     # Fade out time  
cutoff: 100       # Low-pass filter frequency
```

## Playing Synths

Synths generate sound mathematically. First, select a synth:

```
use_synth :tb303  
play :c2
```

Or specify inline:

```
synth :prophet, note: :c4
```

## Key Synth Parameters

```
use_synth :prophet  
play :c4,  
amp: 0.5,  
attack: 0.1,      # Time to reach full volume  
decay: 0.2,        # Time to drop to sustain level  
sustain: 0.3,      # How long to hold  
release: 0.4,      # Fade out time  
cutoff: 80         # Filter brightness (higher = brighter)
```

We'll explore synths deeply in Part II.

# Defining Functions

Functions let us reuse code. This is **essential** for our album:

```
define :kick do
  sample :bd_tek, amp: 2, rate: 0.9
end

# Now we can call it
kick
sleep 1
kick
```

## Functions with Parameters

```
define :kick do |volume|
  sample :bd_tek, amp: volume, rate: 0.9
end

kick 2      # Loud
sleep 1
kick 0.5    # Quiet
```

## Default Parameters

```
define :kick do |v=1|
  sample :bd_tek, amp: 2*v, rate: 0.9
end

kick      # Uses default v=1
kick 0.5  # Override with 0.5
```

This pattern appears in every track:

```
define :kick do |v=1|
  sample :bd_tek, amp: 2.2*v, rate: 0.9
  sample :bd_boom, amp: 0.5*v, rate: 1.2, cutoff: 70
end
```

# Threads: Playing Simultaneously

Music has multiple parts playing at once. Threads make this possible:

```
in_thread do
  4.times do
    kick
    sleep 1
  end
end

in_thread do
  4.times do
    sleep 0.5
    sample :drum_cymbal_closed
    sleep 0.5
  end
end
```

Both loops run **at the same time** — kicks on the beat, hi-hats on off-beats.

## Thread Pattern for Drums

This pattern appears in every track:

```

define :drums do |k=1, s=1, h=1|
  in_thread do
    4.times do
      kick k
      sleep 1
    end
  end

  in_thread do
    sleep 1
    snare s
    sleep 1
    snare s
    sleep 2
  end

  in_thread do
    8.times do
      hat h
      sleep 0.5
    end
  end

  sleep 4 # Wait for the pattern to complete
end

```

The `sleep 4` at the end is crucial — it makes the function take 4 beats total, so you can call it repeatedly:

```

8.times do
  drums 1, 0.9, 0.7
end

```

## Effects (FX)

Effects process sound. Wrap code in `with_fx`:

```
with_fx :reverb, room: 0.8, mix: 0.5 do
  play :c4
  sleep 0.5
  play :e4
end
```

## Key Effects We Use

**Reverb** — Adds space

```
with_fx :reverb, room: 0.8, mix: 0.5 do
  # room: size (0-1), mix: wet/dry balance
end
```

**Echo/Delay** — Repeating echoes

```
with_fx :echo, phase: 0.75, decay: 4, mix: 0.4 do
  # phase: time between echoes, decay: how long echoes last
end
```

**Low-pass Filter** — Removes high frequencies

```
with_fx :lpf, cutoff: 80 do
  # cutoff: frequency limit (higher = brighter)
end
```

**High-pass Filter** — Removes low frequencies

```
with_fx :hpf, cutoff: 80 do
  # cutoff: frequency limit (higher = more bass removed)
end
```

## Automating Effects

You can change effect parameters over time:

```

with_fx :lpf, cutoff: 60, cutoff_slide: 16 do |fx|
  control fx, cutoff: 120 # Slide to 120 over 16 beats
  16.times do
    kick
    sleep 1
  end
end

```

This creates filter sweeps — the sound “opens up” over time.

## Putting It Together

Here's a minimal example combining everything:

```

use_bpm 100

define :kick do |v=1|
  sample :bd_tek, amp: 2*v, rate: 0.9
end

define :snare do |v=1|
  sample :sn_dub, amp: v
end

define :drums do |k=1, s=1|
  in_thread do
    4.times { kick k; sleep 1 }
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s; sleep 2
  end
  sleep 4
end

with_fx :reverb, room: 0.5 do
  4.times do
    drums 1, 0.8
  end
end

```

This is the skeleton of every track in the album. The rest is sound design, pattern creation, and arrangement.

---

Next, we'll dive deeper into each element, starting with the sounds themselves.

# Playing Sounds

Sonic Pi has two fundamental ways to make sound: **samples** (pre-recorded audio) and **synths** (mathematically generated sound). Understanding both is essential.

## Samples

Samples are recordings. Sonic Pi includes hundreds built-in.

```
sample :bd_tek      # Techno kick drum
sample :sn_dub      # Dub snare
sample :drum_cymbal_closed  # Hi-hat
sample :bd_boom     # Boomy kick
```

## Sample Parameters

Every sample can be modified:

```
sample :bd_tek,
  amp: 2,           # Volume (1 = default, 2 = twice as loud)
  rate: 0.9,        # Playback speed (0.5 = half speed, 2 = double)
  attack: 0,        # Fade in time
  release: 0.5,    # Fade out time
  cutoff: 100       # Low-pass filter frequency
```

### amp (amplitude)

Controls volume. Values above 1 boost, below 1 reduce.

```
sample :bd_tek, amp: 0.5  # Quiet
sample :bd_tek, amp: 1    # Normal
sample :bd_tek, amp: 2.5  # Loud (used in our kicks)
```

## rate

Changes playback speed AND pitch:

```
sample :bd_tek, rate: 0.5 # Half speed, octave lower
sample :bd_tek, rate: 1   # Normal
sample :bd_tek, rate: 2   # Double speed, octave higher
sample :bd_tek, rate: -1  # Reversed!
```

We often use `rate: 0.9` on kicks to lower the pitch slightly for more weight.

## cutoff

Low-pass filter — removes frequencies above this value:

```
sample :bd_tek, cutoff: 60 # Dark, muffled
sample :bd_tek, cutoff: 100 # Balanced
sample :bd_tek, cutoff: 130 # Bright, all frequencies
```

# Finding Samples

Sonic Pi groups samples by type:

```
# Drums
sample :bd_tek, :bd_haus, :bd_boom, :bd_zum    # Kicks
sample :sn_dub, :sn_dolf, :sn_zome             # Snares
sample :drum_cymbal_closed, :drum_cymbal_open # Hi-hats

# Electronic
sample :elec_twang, :elec_blink, :elec_ping

# Ambient
sample :ambi_choir, :ambi_dark_woosh
```

In Sonic Pi, type `sample :` and press Ctrl+Space to see all options.

## Synths

Synths generate sound mathematically. They're more flexible than samples.

```
use_synth :prophet
play 60          # Play MIDI note 60 (middle C)
```

Or use note names:

```
play :c4          # Middle C
play :d2          # D, two octaves below middle C
play :fs4         # F sharp, fourth octave
```

## Choosing a Synth

```
use_synth :prophet    # Warm, fat analog sound
play :c3

use_synth :tb303      # Acid bass, squelchy
play :c2

use_synth :dsaw        # Aggressive, wide
play :c3

use_synth :sine        # Pure tone, no harmonics
play :c2
```

We'll explore synths deeply in Part II.

## Synth Parameters

```
use_synth :prophet
play :c4,
  amp: 0.5,          # Volume
  attack: 0.1,       # Time to reach full volume
  decay: 0.2,         # Time to drop to sustain level
  sustain: 0.3,       # Hold time
  release: 0.4,       # Fade out time
  cutoff: 80          # Filter brightness
```

The `attack`, `decay`, `sustain`, `release` parameters form the **ADSR envelope** — more on this in Part II.

## Playing Chords

Play multiple notes simultaneously:

```
play [:c4, :e4, :g4]           # C major chord
play chord(:c4, :minor)        # C minor chord
play chord(:d3, :m7)           # D minor 7th
```

Or play separately in quick succession:

```
play :c4
play :e4
play :g4
```

Without `sleep`, all three play at once.

## Samples vs Synths: When to Use Which

Use samples for:

- Drums and percussion (more realistic)
- Sound effects

- When you need a specific “real” sound

### Use synths for:

- Bass (precise pitch control)
- Leads and melodies (expressiveness)
- Pads and atmospheres (sustain control)

### Layer both:

```
# Our kick: sample for punch, synth for sub
define :kick do |v=1|
    sample :bd_tek, amp: 2*v, rate: 0.9
    sample :bd_boom, amp: 0.5*v, rate: 1.2
end
```

## Quick Reference

```
# Samples
sample :bd_tek, amp: 2, rate: 0.9, cutoff: 100

# Synths
use_synth :prophet
play :c4, amp: 0.5, attack: 0.1, release: 0.5, cutoff: 80

# Chords
play [:c4, :e4, :g4]
play chord(:c4, :minor)
```

---

Next: Timing and Sleep — how to control when sounds happen.

# Timing and Sleep

Music exists in time. In Sonic Pi, we control time with `sleep` and `use_bpm`.

## Setting the Tempo

Every track starts with:

```
use_bpm 100
```

BPM = Beats Per Minute. At 100 BPM:

- 1 beat = 0.6 seconds
- 4 beats (1 bar) = 2.4 seconds

Our album ranges from 95-108 BPM:

BPM	Feel	Used In
95	Slow, hypnotic	Void Walker
98	Atmospheric	Chrome Cathedral
100	Balanced	System Override, Terminal Velocity
102	Driving	Midnight Protocol
105	Energetic	Nerve Damage
106	Intense	Core Meltdown
108	Aggressive	Skull Fracture

## The Sleep Command

`sleep` pauses execution for a number of beats:

```
use_bpm 100

play :c4
sleep 1          # Wait 1 beat (0.6 seconds at 100 BPM)
play :e4
sleep 1
play :g4
```

## Common Sleep Values

```
sleep 4      # Whole note (1 bar in 4/4)
sleep 2      # Half note
sleep 1      # Quarter note
sleep 0.5    # Eighth note
sleep 0.25   # Sixteenth note
```

## Rhythmic Patterns

Different sleep values create different feels:

```
# Straight eighths
4.times do
  sample :drum_cymbal_closed
  sleep 0.5
end

# Syncopated
sample :bd_tek; sleep 0.75
sample :bd_tek; sleep 0.25
sample :bd_tek; sleep 1
sample :bd_tek; sleep 1
```

# Loops and Repetition

## Simple Loops

```
4.times do
  sample :bd_tek
  sleep 1
end
```

This plays 4 kick drums, one per beat.

## Infinite Loops

```
live_loop :drums do
  sample :bd_tek
  sleep 1
end
```

`live_loop` runs forever until you stop it. Great for live coding, but we avoid it in composed tracks.

## Nested Loops

```
4.times do                      # 4 bars
  4.times do                    # 4 beats per bar
    sample :drum_cymbal_closed
    sleep 0.5                   # Eighth notes
  end
end
```

# Pattern Timing

A typical 4-beat drum pattern:

```
define :drums do
  # Beat 1
  sample :bd_tek
  sleep 0.5
  sample :drum_cymbal_closed
  sleep 0.5

  # Beat 2
  sample :bd_tek
  sample :sn_dub
  sleep 0.5
  sample :drum_cymbal_closed
  sleep 0.5

  # Beat 3
  sample :bd_tek
  sleep 0.5
  sample :drum_cymbal_closed
  sleep 0.5

  # Beat 4
  sample :bd_tek
  sample :sn_dub
  sleep 0.5
  sample :drum_cymbal_closed
  sleep 0.5
end
```

Total sleep: 4 beats = 1 bar.

## Timing Math

Always ensure your patterns add up to complete bars:

```

# Good: adds to 4 beats
define :pattern do
  bass :d2; sleep 0.75
  bass :d2; sleep 0.25
  bass :f2; sleep 1
  bass :d2; sleep 2
  # Total: 0.75 + 0.25 + 1 + 2 = 4 ✓
end

# Bad: adds to 3.5 beats (will drift!)
define :broken_pattern do
  bass :d2; sleep 0.5
  bass :d2; sleep 1
  bass :f2; sleep 1
  bass :d2; sleep 1
  # Total: 3.5 ✗
end

```

## Synchronization

When combining multiple patterns, ensure they align:

```

# 8 repetitions of 4-beat drums = 32 beats
8.times do
  drums
end

# Must match: 32 beats of bass
8.times do
  bassline # Also 4 beats each
end

```

If patterns have different lengths, use least common multiples:

```

# 3-beat pattern + 4-beat pattern
# LCM(3,4) = 12 beats to sync
4.times { three_beat_pattern } # 12 beats
3.times { four_beat_pattern } # 12 beats

```

# Quick Reference

```
use_bpm 100          # Set tempo

sleep 1              # Wait 1 beat
sleep 0.5            # Wait half beat
sleep 4              # Wait 1 bar

4.times do           # Repeat 4 times
  sample :bd_tek
  sleep 1
end

# Beat values at any BPM:
# 4 = whole note (1 bar)
# 2 = half note
# 1 = quarter note
# 0.5 = eighth note
# 0.25 = sixteenth note
```

---

Next: Variables and Functions — how to organize and reuse code.

# Variables and Functions

Clean code makes complex music manageable. Functions are the building blocks of every track in this album.

## Why Functions?

Without functions, a drum pattern looks like this:

```
sample :bd_tek, amp: 2, rate: 0.9
sample :bd_boom, amp: 0.5, rate: 1.2
sleep 1
sample :bd_tek, amp: 2, rate: 0.9
sample :bd_boom, amp: 0.5, rate: 1.2
sleep 1
# ... repeated dozens of times
```

With functions:

```
define :kick do
  sample :bd_tek, amp: 2, rate: 0.9
  sample :bd_boom, amp: 0.5, rate: 1.2
end

kick; sleep 1
kick; sleep 1
# Much cleaner!
```

## Defining Functions

Basic syntax:

```
define :function_name do
  # Code here
end
```

Then call it:

```
function_name
```

## Example: Kick Drum

```
define :kick do
  sample :bd_tek, amp: 2, rate: 0.9
end

# Use it
4.times do
  kick
  sleep 1
end
```

## Functions with Parameters

Parameters make functions flexible:

```
define :kick do |volume|
  sample :bd_tek, amp: volume, rate: 0.9
end

kick 2      # Loud kick
kick 0.5    # Quiet kick
kick 1      # Normal kick
```

## Multiple Parameters

```
define :kick do |volume, pitch|
    sample :bd_tek, amp: volume, rate: pitch
end

kick 2, 0.9      # Loud, slightly lower
kick 1, 1.2      # Normal volume, higher pitch
```

## Default Parameters

This is the pattern used throughout the album:

```
define :kick do |v=1|
    sample :bd_tek, amp: 2*v, rate: 0.9
end

kick          # Uses default v=1, so amp = 2*1 = 2
kick 0.5      # v=0.5, so amp = 2*0.5 = 1
kick 1.2      # v=1.2, so amp = 2*1.2 = 2.4
```

The `v=1` means “if no value provided, use 1”.

## Multiple Defaults

```
define :bass do |n, v=1, c=80|
    use_synth :tb303
    play n, amp: v, cutoff: c
end

bass :d2          # note=D2, v=1, cutoff=80
bass :d2, 0.5     # note=D2, v=0.5, cutoff=80
bass :d2, 0.5, 60 # note=D2, v=0.5, cutoff=60
```

# The Sound Functions Pattern

Every track defines sound functions first:

```
# SOUND DEFINITIONS
define :kick do |v=1|
    sample :bd_tek, amp: 2*v, rate: 0.9
    sample :bd_boom, amp: 0.5*v, rate: 1.2
end

define :snare do |v=1|
    sample :sn_dub, amp: v, rate: 0.85
end

define :hat do |v=1|
    sample :drum_cymbal_closed, amp: 0.3*v, rate: 2.2
end

define :bass do |n, v=1, c=80|
    use_synth :tb303
    play n, amp: 0.8*v, cutoff: c, res: 0.3
    use_synth :sine
    play n-12, amp: v
end
```

Then pattern functions:

```

# PATTERN DEFINITIONS
define :drums do |k=1, s=1, h=1|
  # Uses the sound functions defined above
  in_thread do
    4.times { kick k; sleep 1 }
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s; sleep 2
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  sleep 4
end

define :bassline do |v=1, c=80|
  bass :d2, v, c; sleep 1
  bass :d2, v*0.7, c-10; sleep 1
  bass :f2, v*0.9, c; sleep 1
  bass :d2, v, c; sleep 1
end

```

## Variables

Variables store values for reuse:

```

my_note = :c4
my_volume = 0.8

play my_note, amp: my_volume

```

## Variables in Patterns

```
define :melody do |v=1|
  notes = [:d4, :f4, :a4, :g4, :f4, :d4]

  notes.each do |n|
    play n, amp: v
    sleep 0.5
  end
end
```

## Arrays for Patterns

```
define :arp do |v=1|
  pattern = [:d4, :f4, :a4, :d5, :a4, :f4, :d4, :a3]

  pattern.each do |n|
    use_synth :pulse
    play n, amp: 0.25*v, release: 0.1
    sleep 0.5
  end
end
```

## Local vs Global

Variables inside `define` are local:

```
define :example do
  x = 10 # Only exists inside this function
end

# x doesn't exist here
```

Variables outside are global:

```

$global_volume = 0.8 # Available everywhere

define :example do
  play :c4, amp: $global_volume
end

```

We rarely use global variables — parameters are cleaner.

## The Complete Pattern

Here's the structure used in every album track:

```

use_bpm 100

# 1. SOUND DEFINITIONS
define :kick do |v=1|
  # ...
end

define :snare do |v=1|
  # ...
end

define :bass do |n, v=1, c=80|
  # ...
end

# 2. PATTERN DEFINITIONS
define :drums do |k=1, s=1, h=1|
  # Uses sound functions
end

define :bassline do |v=1, c=80|
  # Uses bass function
end

# 3. ARRANGEMENT
# Intro
8.times { drums 0.7, 0, 0.5 }

# Main
16.times { drums 1, 0.9, 0.8 }

```

This separation makes code:

- **Readable** — Clear what each section does
  - **Reusable** — Same sounds, different patterns
  - **Tweakable** — Change one function, affect everywhere
- 

Next: Threads — how to play multiple parts simultaneously.

# Threads and Concurrency

Music has multiple parts playing simultaneously — drums, bass, melody, effects. Threads make this possible.

## The Problem

Without threads, code runs sequentially:

```
# This plays ONE thing at a time
4.times do
  sample :bd_tek
  sleep 1
end

4.times do
  sample :sn_dub
  sleep 1
end
```

The snare only starts AFTER all kicks finish. That's not music — that's a sequence.

## The Solution: `in_thread`

`in_thread` runs code in the background:

```
in_thread do
  4.times do
    sample :bd_tek
    sleep 1
  end
end

in_thread do
  4.times do
    sleep 0.5
    sample :sn_dub
    sleep 0.5
  end
end
```

Now both run simultaneously! Kicks on the beat, snares on the off-beat.

## Thread Timing

Threads start at the same moment:

```
in_thread do
  puts "Thread 1 starts"
  sleep 2
  puts "Thread 1 ends"
end

in_thread do
  puts "Thread 2 starts"
  sleep 1
  puts "Thread 2 ends"
end

puts "Main continues immediately"
```

Output:

```
Main continues immediately
Thread 1 starts
Thread 2 starts
Thread 2 ends      # After 1 beat
Thread 1 ends      # After 2 beats
```

## The Drum Pattern

Here's how we build drums with threads:

```
define :drums do |k=1, s=1, h=1|
  # Kick thread
  in_thread do
    4.times do
      kick k
      sleep 1
    end
  end

  # Snare thread
  in_thread do
    sleep 1          # Wait for beat 2
    snare s
    sleep 1          # Wait for beat 4
    snare s
    sleep 2          # Fill remaining time
  end

  # Hi-hat thread
  in_thread do
    8.times do
      hat h
      sleep 0.5
    end
  end

  # IMPORTANT: Wait for pattern to complete
  sleep 4
end
```

## The Final Sleep

The `sleep 4` at the end is crucial:

```
# WITHOUT final sleep
8.times { drums } # All 8 start immediately!

# WITH final sleep
8.times { drums } # Each waits 4 beats before next
```

The main function must “take time” equal to the pattern length.

## Visualizing Threads

Time:	1	2	3	4
Kick:	X	X	X	X
Snare:		X		X
Hat:	x	x	x	x

All three threads run in parallel, each following its own timing.

## Complex Drum Patterns

The album uses more complex patterns:

```

define :drums_power do |k=1, s=1, h=1|
  in_thread do
    kick k; sleep 0.75
    kick k*0.6; sleep 0.25
    kick k*0.8; sleep 1
    kick k; sleep 0.75
    kick k*0.7; sleep 0.25
    kick k*0.9; sleep 1
  end

  in_thread do
    sleep 1.5
    snare s*0.7
    sleep 0.5
    snare s
    sleep 2
  end

  in_thread do
    8.times { hat h; sleep 0.5 }
  end

  sleep 4
end

```

This creates syncopated kicks with ghost notes.

## Layering with Threads

Use threads to layer different elements:

```
# ARRANGEMENT SECTION
in_thread do
  12.times { drums 1, 0.9, 0.7 }
end

in_thread do
  12.times { bassline 1, 80 }
end

in_thread do
  sleep 16 # Wait 4 bars before melody enters
  8.times { melody 0.8 }
end

sleep 48 # Total section length: 12 bars
```

## Thread Naming

For debugging, name your threads:

```
in_thread name: :drums do
  # ...
end

in_thread name: :bass do
  # ...
end
```

# Common Mistakes

## 1. Forgetting the Final Sleep

```
# BAD - drums function returns immediately
define :drums do
  in_thread { 4.times { kick; sleep 1 } }
  in_thread { 4.times { snare; sleep 1 } }
  # No sleep 4!
end

8.times { drums } # All start at once!
```

## 2. Mismatched Timing

```
# BAD - threads have different lengths
in_thread do
  4.times { kick; sleep 1 } # 4 beats
end

in_thread do
  3.times { snare; sleep 1 } # 3 beats - will drift!
end
```

## 3. Too Many Nested Threads

```
# AVOID - hard to track timing
in_thread do
  in_thread do
    in_thread do
      # ???
    end
  end
end
```

Keep thread structure flat when possible.

# The Arrangement Pattern

Full tracks use threads for sections:

```
# INTRO
in_thread do
  8.times { drums 0.7, 0, 0.5 }
end
in_thread do
  with_fx :lpf, cutoff: 60 do
    8.times { bassline 0.6 }
  end
end
sleep 32

# MAIN
in_thread do
  16.times { drums 1, 0.9, 0.8 }
end
in_thread do
  16.times { bassline 1, 85 }
end
in_thread do
  sleep 16
  8.times { melody 0.8 }
end
sleep 64
```

# Quick Reference

```
# Basic thread
in_thread do
  # Code runs in background
end

# Named thread
in_thread name: :my_thread do
  # ...
end

# Pattern with threads
define :drums do |k=1, s=1, h=1|
  in_thread do
    4.times { kick k; sleep 1 }
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s; sleep 2
  end
  sleep 4  # IMPORTANT!
end
```

---

Next: Effects — adding space, texture, and movement to sounds.

# Effects (FX)

Effects transform sounds. They add space, texture, movement, and character. In dark electronic music, effects are essential — they create atmosphere and energy.

## Basic Syntax

Wrap code in `with_fx`:

```
with_fx :reverb do
  play :c4
  sleep 1
  play :e4
end
```

Everything inside the block is processed through the effect.

## Key Effects for Dark Electronic

### Reverb

Reverb simulates acoustic spaces — from small rooms to vast halls.

```
with_fx :reverb, room: 0.8, mix: 0.5 do
  play :c4
end
```

#### Parameters:

- `room` (0-1): Size of space. Higher = larger, longer tail.
- `mix` (0-1): Wet/dry balance. 0 = dry only, 1 = wet only.

## Usage by section:

Section	Room	Mix	Purpose
Intro	0.9-1.0	0.5-0.7	Atmospheric, spacious
Main	0.5-0.7	0.3-0.4	Present but not washy
Break	0.85-1.0	0.5-0.6	Tension, space
Outro	0.95-1.0	0.6-0.7	Fade into space

```
# Ethereal melody
with_fx :reverb, room: 0.9, mix: 0.55 do
  4.times { melody 0.7 }
end

# Tight drums (less reverb)
with_fx :reverb, room: 0.4, mix: 0.2 do
  4.times { drums 1 }
end
```

## Echo / Delay

Echo creates rhythmic repeats.

```
with_fx :echo, phase: 0.75, decay: 4, mix: 0.4 do
  play :c4
end
```

### Parameters:

- `phase` : Time between echoes (in beats)
- `decay` : How long echoes continue (seconds)
- `mix` : Wet/dry balance

### Common phase values:

- `0.25` — Sixteenth note delays (fast, rhythmic)
- `0.5` — Eighth note delays (driving)
- `0.75` — Dotted eighth (classic dub feel)

- 1.0 — Quarter note (spacious)

```
# Driving delay
with_fx :echo, phase: 0.5, decay: 3, mix: 0.3 do
  4.times { melody 0.8 }
end

# Ethereal long delay
with_fx :echo, phase: 1.0, decay: 8, mix: 0.5 do
  play :c5
end
```

## Low-Pass Filter (LPF)

Removes high frequencies. Essential for bass and filtered builds.

```
with_fx :lpf, cutoff: 80 do
  play :c4
end
```

### Parameters:

- `cutoff` (0-130): Frequency limit. Lower = darker, more muffled.

```
# Dark, filtered bass
with_fx :lpf, cutoff: 60 do
  8.times { bassline 0.7 }
end

# Opening filter over time
with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
  control fx, cutoff: 110 # Slides from 50 to 110 over 32 beats
  8.times { bassline 1 }
end
```

## High-Pass Filter (HPF)

Removes low frequencies. Creates tension by removing bass.

```
with_fx :hpf, cutoff: 80 do
  play :c2 # Bass frequencies removed!
end
```

**Common use:** Breakdowns

```
# Tension before drop
with_fx :hpf, cutoff: 100 do
  with_fx :reverb, room: 0.9 do
    4.times { melody 0.6 }
  end
end
```

## Combining Effects

Effects can be nested:

```
with_fx :reverb, room: 0.8, mix: 0.5 do
  with_fx :echo, phase: 0.75, decay: 4, mix: 0.35 do
    4.times { melody 0.7 }
  end
end
```

The sound passes through echo first, then reverb.

**Order matters:**

```

# Echo -> Reverb (echo tails get reverb)
with_fx :reverb do
  with_fx :echo do
    play :c4
  end
end

# Reverb -> Echo (reverb gets echoed)
with_fx :echo do
  with_fx :reverb do
    play :c4
  end
end

```

Generally: inner effects process first, outer effects last.

## Effect Automation

Change effect parameters over time using `control`:

```

with_fx :lpf, cutoff: 60, cutoff_slide: 16 do |fx|
  control fx, cutoff: 120 # Slides over 16 beats

  16.times do
    bass :d2
    sleep 1
  end
end

```

The `|fx|` captures the effect reference, `cutoff_slide` sets transition time.

## Riser Effect

```
# Build tension with rising filter
with_fx :lpf, cutoff: 40, cutoff_slide: 32 do |fx|
    control fx, cutoff: 130

    in_thread do
        32.times do
            sample :drum_cymbal_closed, amp: 0.3
            sleep 0.5
        end
    end

    sleep 32
end
```

## Effect Patterns in the Album

### Intro Pattern

```
# Spacious, atmospheric
with_fx :reverb, room: 0.95, mix: 0.6 do
    with_fx :echo, phase: 1, decay: 6, mix: 0.45 do
        4.times { arp 0.4 }
    end
end
```

### Main Section Pattern

```
# Tighter, more present
with_fx :reverb, room: 0.6, mix: 0.35 do
    with_fx :echo, phase: 0.5, decay: 3, mix: 0.25 do
        8.times { melody 0.8 }
    end
end
```

## Break Pattern

```
# Maximum space before drop
with_fx :reverb, room: 1.0, mix: 0.7 do
  with_fx :echo, phase: 1.5, decay: 8, mix: 0.55 do
    melody 0.5
  end
end
```

## Outro Pattern

```
# Fading into infinity
with_fx :reverb, room: 1.0, mix: 0.7 do
  with_fx :echo, phase: 1.25, decay: 10, mix: 0.6 do
    arp 0.3
    sleep 8
    melody 0.2
  end
end
```

## Other Useful Effects

### Distortion

```
with_fx :distortion, distort: 0.7 do
  play :c2
end
```

### Bitcrusher

Lo-fi digital degradation:

```
with_fx :bitcrusher, bits: 8, sample_rate: 8000 do
  play :c4
end
```

## Wobble

Automatic filter modulation:

```
with_fx :wobble, phase: 0.5, mix: 0.7 do
  play :c2, sustain: 4
end
```

## Quick Reference

```
# Reverb
with_fx :reverb, room: 0.8, mix: 0.5 do ... end

# Echo
with_fx :echo, phase: 0.75, decay: 4, mix: 0.4 do ... end

# Low-pass filter
with_fx :lpf, cutoff: 80 do ... end

# High-pass filter
with_fx :hpf, cutoff: 80 do ... end

# Filter automation
with_fx :lpf, cutoff: 60, cutoff_slide: 16 do |fx|
  control fx, cutoff: 120
  #
end

# Combined effects
with_fx :reverb, room: 0.8 do
  with_fx :echo, phase: 0.5, decay: 3 do
    # Sound is echoed, then reverbed
  end
end
```

---

This completes Part I: Foundations. You now understand:

- The vision and aesthetic
- Sonic Pi basics: samples, synths, timing
- Functions and code organization
- Threads for simultaneous playback
- Effects for space and movement

In Part II, we dive deep into **Sound Design** — building the drums, bass, and leads that define dark electronic music.

# Building Sounds

Part II focuses on crafting the individual elements of dark electronic music. We'll cover drums, bass, leads, and how they work together.

## The Sound Palette

Every track in *Sonic Byte* uses a consistent palette:

### Drums

- **Kick:** Layered samples (`bd_tek` + `bd_boom` or `bd_zum`)
- **Snare:** `sn_dub` with optional layering
- **Hi-hat:** `drum_cymbal_closed`, high rate for metallic sound

### Bass

- **Character layer:** `tb303`, `prophet`, or `dsaw`
- **Sub layer:** sine wave, one octave below

### Leads

- **Primary:** `prophet` for warm, musical tones
- **Texture:** `dark_ambience` or `hollow` for atmosphere
- **Arps:** `pulse` for rhythmic patterns

### Effects

- **Reverb:** Space and atmosphere

- **Echo:** Rhythmic interest
- **Filters:** Movement and energy

## The Layering Principle

Single sounds are rarely enough. Layering creates:

1. **Frequency coverage** — Different sounds fill different ranges
2. **Character** — Combine punchy attack with sustained body
3. **Width** — Some layers can be wider than others

Example — A layered kick:

```
define :kick do |v=1|
  # Layer 1: Attack and punch
  sample :bd_tek, amp: 2*v, rate: 0.9

  # Layer 2: Sub weight
  sample :bd_boom, amp: 0.5*v, rate: 1.2, cutoff: 70
end
```

## The Definition Pattern

Every track follows this structure:

```

use_bpm 100

# === SOUND DEFINITIONS ===
# Individual sounds with volume control

define :kick do |v=1|
    # ...
end

define :snare do |v=1|
    # ...
end

define :bass do |n, v=1, c=80|
    # ...
end

define :lead do |n, dur=0.5, v=1|
    # ...
end

# === PATTERN DEFINITIONS ===
# Combinations of sounds into musical phrases

define :drums do |k=1, s=1, h=1|
    # Uses kick, snare, hat
end

define :bassline do |v=1, c=80|
    # Uses bass
end

define :melody do |v=1|
    # Uses lead
end

# === ARRANGEMENT ===
# Structure using patterns

# Intro
# Build
# Main
# Break
# Peak
# Outro

```

This separation is key to maintainable music code.

## Sound Quality Principles

### 1. Leave Headroom

Don't max out volumes:

```
# BAD
sample :bd_tek, amp: 5 # Likely to distort

# GOOD
sample :bd_tek, amp: 2 # Loud but clean
```

### 2. Cut Before Boost

Use filters to remove unwanted frequencies rather than boosting what you want:

```
# Remove rumble from hi-hats
sample :drum_cymbal_closed, cutoff: 130, hpf: 70

# Remove harshness from bass
use_synth :tb303
play :d2, cutoff: 75 # Not too bright
```

### 3. Contrast Creates Impact

Quiet moments make loud moments louder:

```

# Quiet verse
8.times { drums 0.6, 0.5, 0.4 }

# LOUD chorus (feels even louder after quiet)
8.times { drums 1.1, 1.0, 0.8 }

```

## 4. Frequency Separation

Each element should have its own space:

Element	Frequency Range	Role
Sub bass	20-60 Hz	Physical weight
Bass body	60-200 Hz	Warmth, power
Kick punch	60-100 Hz	Attack, drive
Snare body	200-400 Hz	Body
Snare crack	2-5 kHz	Presence
Hi-hats	5-15 kHz	Air, rhythm
Leads	200 Hz - 5 kHz	Melody, hooks

When elements compete, use filters:

```

# Bass with sub emphasis
define :bass do |n, v=1|
  use_synth :tb303
  play n, cutoff: 70 # Roll off highs

  use_synth :sine
  play n-12 # Pure sub
end

# Lead with bass rolled off
define :lead do |n, v=1|
  use_synth :prophet
  play n, cutoff: 90 # Brighter than bass
end

```

# Sound Design Workflow

When creating sounds:

1. **Start simple** — One layer, basic parameters
  2. **Add layers** — One at a time, listen to each
  3. **Adjust balance** — Volume relationships between layers
  4. **Shape with filters** — Remove unwanted frequencies
  5. **Add effects** — Reverb, delay as needed
  6. **Test in context** — Does it work with other elements?
- 

The following chapters dive deep into each element: drums, bass, and leads.

# Choosing Synths

Sonic Pi includes dozens of synths. For dark electronic music, you only need to master a handful.

## The Core Palette

These six synths cover 90% of *Sonic Byte*:

Synth	Character	Primary Use
:tb303	Acid, squelchy	Aggressive bass
:prophet	Warm, analog	Leads, warm bass
:dsaw	Wide, aggressive	Bass layers, stabs
:sine	Pure, clean	Sub bass layer
:pulse	Hollow, rhythmic	Arpeggios
:dark_ambience	Eerie, evolving	Pads, stabs

## Synth Selection by Role

### For Bass

**Aggressive bass:** :tb303

```
use_synth :tb303
play :d2, cutoff: 80, res: 0.3
```

The resonant filter creates the classic acid sound.

**Warm bass:** :prophet

```
use_synth :prophet
play :d2, cutoff: 75, res: 0.2
```

Smoother, less aggressive, more musical.

**Wide bass:** :dsaw

```
use_synth :dsaw
play :d2, cutoff: 82, detune: 0.2
```

The detuned oscillators create stereo width.

**Sub layer:** :sine (always)

```
use_synth :sine
play :d1 # One octave below main bass
```

Pure low frequency, no harmonics. Mix it high.

## For Leads

**Musical lead:** :prophet

```
use_synth :prophet
play :d4, attack: 0.05, cutoff: 90
```

Warm and expressive, good for melodic lines.

**Aggressive lead:** :mod\_saw

```
use_synth :mod_saw
play :d4, mod_phase: 0.1, mod_range: 6, cutoff: 110
```

The modulation creates movement and urgency.

**Shimmering layer:** :saw

```
use_synth :saw
play :d5, amp: 0.1, cutoff: 75 # Octave up, quiet
```

Add sparkle to prophet leads.

## For Arpeggios

**Classic arp:** :pulse

```
use_synth :pulse
play :d4, pulse_width: 0.35, cutoff: 100
```

Hollow, rhythmic, synthwave-appropriate.

**Aggressive arp:** :dsaw

```
use_synth :dsaw
play :d4, detune: 0.1, cutoff: 95
```

Wider, more present.

## For Pads/Atmosphere

**Dark pad:** :dark\_ambience

```
use_synth :dark_ambience
play [:d3, :a3, :d4], attack: 2, sustain: 3, release: 4
```

Evolving, eerie texture.

**Ethereal pad:** :hollow

```
use_synth :hollow
play :d4, attack: 0.5, release: 2, cutoff: 85
```

Breathy, ghostly presence.

## For Special Effects

**Alarm/hook:** :mod\_saw

```
use_synth :mod_saw
play :f4, mod_phase: 0.08, mod_range: 8, cutoff: 115
```

Fast modulation = alarm-like urgency.

**Stab:** :dsaw or :dark\_ambience

```
use_synth :dsaw
play [:d2, :a2, :d3], attack: 0, decay: 0.15, release: 0.1
```

Short, punchy chord hits.

## Synth Comparison

### Bass Synths

```
# Same note, different character
use_bpm 100

use_synth :tb303
play :d2, cutoff: 80, res: 0.3
sleep 2

use_synth :prophet
play :d2, cutoff: 80
sleep 2

use_synth :dsaw
play :d2, cutoff: 80, detune: 0.2
sleep 2
```

Listen for:

- TB303: Brighter, more “squelchy”
- Prophet: Warmer, rounder
- Dsaw: Wider, more aggressive

## Lead Synths

```
use_synth :prophet
play :d4, cutoff: 90
sleep 2

use_synth :saw
play :d4, cutoff: 90
sleep 2

use_synth :mod_saw
play :d4, cutoff: 100, mod_phase: 0.15, mod_range: 4
sleep 2
```

Listen for:

- Prophet: Musical, singable
- Saw: Raw, brighter
- Mod\_saw: Movement, urgency

# Decision Framework

Need bass?

- └─ Aggressive → :tb303
- └─ Warm → :prophet
- └─ Wide → :dsaw
- └─ Always add → :sine (sub layer)

Need lead?

- └─ Melodic → :prophet
- └─ Aggressive → :mod\_saw
- └─ Add shimmer? → :saw (octave up)

Need arp?

- └─ Classic → :pulse
- └─ Aggressive → :dsaw

Need atmosphere?

- └─ Dark → :dark\_ambience
- └─ Ethereal → :hollow

Need hook/alarm?

- └─ → :mod\_saw (fast mod\_phase)

## Exploring More Synths

Sonic Pi has many more synths. Some worth exploring:

- :fm — FM synthesis, metallic/bell tones
- :mod\_tri — Modulated triangle, softer than mod\_saw
- :tech\_saws — Multiple detuned saws, huge sound
- :blade — Aggressive, cutting

To see all available synths:

```
puts synth_names
```

But for *Sonic Byte*, the core six are all you need.

# Quick Reference

```
# Aggressive bass
use_synth :tb303
play :d2, cutoff: 80, res: 0.3, wave: 0

# Warm bass
use_synth :prophet
play :d2, cutoff: 75, res: 0.2

# Sub layer
use_synth :sine
play :d1, amp: 1.1

# Lead
use_synth :prophet
play :d4, attack: 0.05, cutoff: 90

# Arp
use_synth :pulse
play :d4, pulse_width: 0.35, cutoff: 100

# Pad
use_synth :dark_ambience
play [:d3, :a3], attack: 2, release: 4
```

# Layering Techniques

Single sounds are rarely enough. Layering combines multiple sounds to create fuller, more powerful, and more interesting textures.

## Why Layer?

A single synth playing a bass note covers a limited frequency range. Layering fills the gaps:

```
# Single synth - thin
use_synth :tb303
play :d2

# Layered - full
use_synth :tb303
play :d2, cutoff: 80      # Mid frequencies
use_synth :sine
play :d1, amp: 1.1        # Sub frequencies
```

## The Three-Layer Model

Most sounds in *Sonic Byte* use up to three layers:

### 1. Character Layer

The main sound that defines the timbre.

```
use_synth :tb303
play :d2, cutoff: 80, res: 0.3
```

## 2. Sub Layer

Pure low frequency for physical weight.

```
use_synth :sine
play :d1 # One octave below
```

## 3. Texture Layer (Optional)

Adds width, grit, or atmosphere.

```
use_synth :dsaw
play :d2, cutoff: 70, detune: 0.15, amp: 0.4
```

# Bass Layering

## Two-Layer (Standard)

```
define :bass do |n, v=1, c=80|
  use_synth :tb303
  play n, amp: 0.8*v, cutoff: c, res: 0.3

  use_synth :sine
  play n-12, amp: 1.1*v # Octave below
end
```

## Three-Layer (Aggressive)

```
define :bass do |n, v=1, c=80|
  use_synth :tb303
  play n, amp: 0.75*v, cutoff: c, res: 0.3

  use_synth :dsaw
  play n, amp: 0.4*v, cutoff: c-10, detune: 0.18

  use_synth :sine
  play n-12, amp: 1.15*v
end
```

## Lead Layering

### Prophet + Shimmer

```
define :lead do |n, dur=0.5, v=1|
  use_synth :prophet
  play n, amp: 0.4*v, attack: 0.05, cutoff: 90

  use_synth :saw
  play n+12, amp: 0.1*v, cutoff: 75  # Octave up, quiet
end
```

### Prophet + Atmosphere

```
define :lead do |n, dur=0.5, v=1|
  use_synth :prophet
  play n, amp: 0.4*v, attack: 0.05, cutoff: 88

  use_synth :dark_ambience
  play n, amp: 0.12*v, attack: 0.1, release: dur*0.8
end
```

# Drum Layering

## Kick

```
define :kick do |v=1|
  sample :bd_tek, amp: 2.2*v, rate: 0.9      # Attack/punch
  sample :bd_boom, amp: 0.5*v, rate: 1.2     # Sub/weight
end
```

## Snare

```
define :snare do |v=1|
  sample :sn_dub, amp: v, rate: 0.8          # Body
  sample :drum_snare_hard, amp: 0.4*v        # Crack
end
```

## Volume Balance

Each layer needs proper level:

Layer	Role	Typical Amp
Character	Primary	0.6-0.8
Texture	Supporting	0.3-0.5
Sub	Foundation	1.0-1.2
Shimmer	Sparkle	0.08-0.12

**Rule:** Sub can be loudest (low frequencies need more energy to be perceived).

# Frequency Separation

Filter layers to avoid mud:

```
define :bass do |n, v=1, c=80|
    use_synth :tb303
    play n, cutoff: c          # Main cutoff

    use_synth :dsaw
    play n, cutoff: c-10       # Darker (less overlap)

    use_synth :sine
    play n-12                  # No filter needed
end
```

# ADSR Alignment

Layers should have similar timing:

```
# All layers: quick attack, similar decay
use_synth :tb303
play n, attack: 0.01, decay: 0.2, release: 0.15

use_synth :dsaw
play n, attack: 0.01, decay: 0.15, release: 0.12

use_synth :sine
play n-12, attack: 0.01, sustain: 0.25, release: 0.2
```

# When to Layer

**Use more layers for:**

- Peak/drop sections
- Main hooks
- When you need to fill the mix

## Use fewer layers for:

- Intros/outros
- Atmospheric sections
- When other elements need space

## Quick Reference

```
# Two-layer bass
define :bass do |n, v=1, c=80|
  use_synth :tb303
  play n, amp: 0.8*v, cutoff: c
  use_synth :sine
  play n-12, amp: 1.1*v
end

# Three-layer bass
define :bass do |n, v=1, c=80|
  use_synth :tb303
  play n, amp: 0.75*v, cutoff: c
  use_synth :dsaw
  play n, amp: 0.4*v, cutoff: c-10, detune: 0.18
  use_synth :sine
  play n-12, amp: 1.15*v
end

# Layered lead
define :lead do |n, dur=0.5, v=1|
  use_synth :prophet
  play n, amp: 0.4*v, cutoff: 90
  use_synth :saw
  play n+12, amp: 0.1*v, cutoff: 75
end
```

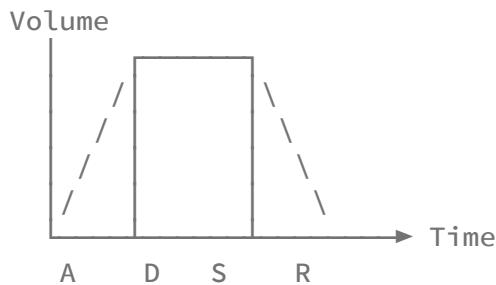
# The ADSR Envelope

Every sound has a shape over time. The ADSR envelope controls that shape.

## What is ADSR?

ADSR stands for:

- **Attack** — How quickly the sound reaches full volume
- **Decay** — How quickly it drops to the sustain level
- **Sustain** — The level held while the note plays
- **Release** — How quickly it fades after the note ends



## In Sonic Pi

```
use_synth :prophet
play :c4,
  attack: 0.1,      # 0.1 beats to reach full volume
  decay: 0.2,       # 0.2 beats to drop to sustain
  sustain: 0.5,    # Hold for 0.5 beats
  release: 0.3     # 0.3 beats to fade out
```

Total note duration = attack + decay + sustain + release = 1.1 beats

# ADSR for Different Sounds

## Punchy Bass (short, percussive)

```
attack: 0.01,      # Instant attack
decay: 0.2,        # Quick drop
sustain: 0.1,      # Short sustain
release: 0.15     # Quick release
```

**Character:** Tight, rhythmic, punchy

## Pad (slow, evolving)

```
attack: 1.5,       # Slow fade in
decay: 0.5,        # Gentle drop
sustain: 2.0,      # Long sustain
release: 2.0       # Long fade out
```

**Character:** Atmospheric, washy, ambient

## Lead (expressive)

```
attack: 0.05,      # Quick but not instant
decay: 0.25,        # Moderate drop
sustain: 0.4,       # Medium sustain
release: 0.5        # Smooth fade
```

**Character:** Musical, expressive, singable

## Stab (aggressive hit)

```
attack: 0,      # Instant
decay: 0.1,     # Very quick
sustain: 0.05,   # Almost none
release: 0.1    # Quick fade
```

**Character:** Aggressive, punchy, rhythmic

## Relative vs Absolute Duration

You can make ADSR relative to a note duration parameter:

```
define :lead do |n, dur=0.5, v=1|
  use_synth :prophet
  play n,
    amp: v,
    attack: dur * 0.1,      # 10% of duration
    decay: dur * 0.25,     # 25% of duration
    sustain: dur * 0.4,    # 40% of duration
    release: dur * 0.5     # 50% of duration (overlaps next note
slightly)
end

lead :c4, 0.5    # Short note
lead :c4, 2.0    # Long note (same proportions)
```

This keeps the same “feel” regardless of note length.

## Attack Shapes

### Zero Attack (instant)

```
attack: 0
```

- Used for: Drums, stabs, bass
- Creates: Punchy, percussive sounds

## Short Attack (0.01-0.05)

attack: 0.03

- Used for: Most bass, leads
- Creates: Quick but slightly softer entry

## Medium Attack (0.05-0.2)

attack: 0.1

- Used for: Expressive leads, pads
- Creates: More musical, less aggressive

## Long Attack (0.5+)

attack: 1.5

- Used for: Pads, atmospheric sounds
- Creates: Swelling, evolving textures

## Release and Legato

Release affects how notes connect:

## **Short Release (staccato)**

```
play :c4, release: 0.1
sleep 1
play :d4, release: 0.1
```

Notes are clearly separated.

## **Long Release (legato)**

```
play :c4, release: 0.8
sleep 0.5
play :d4, release: 0.8
```

Notes overlap, creating smoother transitions.

# **ADSR in the Album**

## **Kick Drum Pattern**

Kicks don't use ADSR directly (samples), but the concept applies:

- Instant attack (the transient)
- Quick decay (the punch)
- Minimal sustain
- Short release (tight low end)

## TB303 Bass

```
use_synth :tb303
play :d2,
  attack: 0.01,
  decay: 0.2,
  sustain: 0.1,
  release: 0.15
```

Quick attack for punch, short overall for rhythmic bass lines.

## Prophet Lead

```
use_synth :prophet
play :d4,
  attack: 0.05,
  decay: 0.25,
  sustain: 0.4,
  release: 0.5
```

Slightly softer attack for musical feel, longer release for expression.

## Dark Ambience Pad

```
use_synth :dark_ambience
play :d3,
  attack: 1.5,
  sustain: 2.5,
  release: 3
```

Slow everything for atmospheric, evolving texture.

# Quick Reference

Sound Type	Attack	Decay	Sustain	Release
Punchy Bass	0.01	0.2	0.1	0.15
Lead	0.05	0.25	0.4	0.5
Stab	0	0.1	0.05	0.1
Pad	1.5	0.5	2.0	2.0
Arp Note	0.01	0.1	0.05	0.1

## Tips

1. **Start with presets** — Use the values above as starting points
2. **Match the rhythm** — Faster BPM often needs shorter ADSR
3. **Leave space** — Not everything needs long sustain
4. **Release > Sleep** — Notes can ring past the next sleep for legato
5. **Attack affects feel** — Even 0.01 vs 0.05 changes the vibe significantly

# Filters and Cutoff

Filters shape the frequency content of sounds. In dark electronic music, they're essential for creating movement, building energy, and separating elements.

## What Filters Do

Filters remove frequencies from a sound:

- **Low-pass filter (LPF):** Removes high frequencies (makes sound darker/muffled)
- **High-pass filter (HPF):** Removes low frequencies (makes sound thinner)
- **Band-pass filter (BPF):** Keeps only a range of frequencies

## The Cutoff Parameter

Most Sonic Pi synths have a `cutoff` parameter — this controls the low-pass filter:

```
use_synth :tb303

play :d2, cutoff: 60    # Dark, muffled
sleep 1
play :d2, cutoff: 80    # Balanced
sleep 1
play :d2, cutoff: 100   # Bright
sleep 1
play :d2, cutoff: 120   # Very bright, almost harsh
```

## Cutoff Values Guide

Cutoff	Sound	Use Case
40-60	Very dark, subby	Filtered builds, intros
60-75	Dark, warm	Verses, atmospheric sections
75-90	Balanced	Main sections
90-110	Bright, present	Drops, peaks
110+	Very bright	Leads, aggressive moments

## Filter as Effect

You can also apply filters as effects to any sound:

### Low-Pass Filter (LPF)

```
with_fx :lpf, cutoff: 70 do
  sample :loop_amen
end
```

### High-Pass Filter (HPF)

```
with_fx :hpf, cutoff: 80 do
  sample :loop_amen # Bass removed
end
```

## Combining Both (Band-Pass)

```
with_fx :hpf, cutoff: 60 do
  with_fx :lpf, cutoff: 100 do
    sample :loop_amen # Only mid frequencies
  end
end
```

## Filter Movement

Static filters are useful, but moving filters create energy.

### Cutoff Slide (On Synths)

```
use_synth :tb303
play :d2, cutoff: 60, cutoff_slide: 4
sleep 0.1
control cutoff: 100 # Slides from 60 to 100 over 4 beats
```

### Filter Automation (On Effects)

```
with_fx :lpf, cutoff: 50, cutoff_slide: 16 do |fx|
  control fx, cutoff: 110 # Opens over 16 beats

16.times do
  bass :d2
  sleep 1
end
end
```

This is the classic “filter opening” build.

# Resonance

The `res` parameter adds emphasis at the cutoff frequency:

```
use_synth :tb303
play :d2, cutoff: 75, res: 0.1 # Subtle
sleep 1
play :d2, cutoff: 75, res: 0.4 # Classic acid
sleep 1
play :d2, cutoff: 75, res: 0.7 # Screaming (careful!)
```

## Resonance Values

Res	Sound	Character
0.1-0.2	Subtle color	Clean, professional
0.3-0.4	Pronounced	Classic analog
0.5-0.6	Strong	Acid, aggressive
0.7+	Extreme	Screaming, use sparingly

# Filter Patterns in the Album

## Intro Pattern

Start filtered, gradually open:

```
# INTRO
with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
  control fx, cutoff: 80 # Slow open
  8.times { bassline 0.6 }
end
```

## Build Pattern

Open filter as energy builds:

```
# BUILD
with_fx :lpf, cutoff: 60, cutoff_slide: 24 do |fx|
  control fx, cutoff: 100 # Opens during build
  6.times { |i| bassline (0.7 + i*0.05) }
end
```

## Main Section

Filter at “home” position:

```
# MAIN
bass :d2, v, 80 # Cutoff 80 = balanced default
```

## Peak Section

Filter fully open for brightness:

```
# PEAK
bass :d2, v, 90 # Brighter!
bassline 1.1, 92 # Even brighter at maximum energy
```

## Break Pattern

High-pass to remove bass, create tension:

```
# BREAK
with_fx :hpf, cutoff: 90 do
  with_fx :reverb, room: 0.9 do
    melody 0.5 # No bass frequencies, floating
  end
end
```

## Outro Pattern

Filter closing as energy fades:

```
# OUTRO
with_fx :lpf, cutoff: 80, cutoff_slide: 24 do |fx|
  control fx, cutoff: 50 # Closing
  6.times { |i| bassline (0.8 - i*0.1) }
end
```

## Cutoff Variation in Patterns

Don't use static cutoff — vary it within patterns:

```
define :bassline do |v=1, c=80|
  bass :d2, v, c      # Root cutoff
  sleep 0.5
  bass :d2, v*0.7, c-10 # Darker ghost note
  sleep 0.5
  bass :f2, v*0.9, c    # Back to root cutoff
  sleep 0.5
  bass :d2, v*0.8, c-5  # Slightly darker
  sleep 0.5
  bass :a2, v, c+5      # Brighter for emphasis
  sleep 0.5
  bass :d2, v, c        # Return to root
  sleep 1.5
end
```

The cutoff variations ( `c-10` , `c-5` , `c+5` ) add subtle movement.

## Filter Per Section

Typical cutoff progression through a track:

Section	Bass Cutoff	Character
Intro	50→70	Dark, opening
Build	70→90	Building energy
Main A	80	Balanced
Break	HPF instead	Tension, no bass
Main B/Pink	85-92	Bright, energetic
Outro	80→55	Fading, closing

## Quick Reference

```

# Synth cutoff
use_synth :tb303
play :d2, cutoff: 80, res: 0.3

# LPF effect (removes highs)
with_fx :lpf, cutoff: 70 do
    # dark sound
end

# HPF effect (removes lows)
with_fx :hpf, cutoff: 80 do
    # thin sound
end

# Filter automation
with_fx :lpf, cutoff: 50, cutoff_slide: 16 do |fx|
    control fx, cutoff: 110 # Opens over 16 beats
end

# Cutoff in patterns
bass :d2, v, c      # Main note
bass :d2, v, c-10   # Ghost note (darker)
bass :a2, v, c+5    # Accent (brighter)

```

## Tips

1. **Start dark, open up** — Classic build technique
2. **Close for outros** — Signals ending
3. **Vary within patterns** — Static cutoff is boring
4. **Higher cutoff = more energy** — Use for peaks
5. **HPF for tension** — Remove bass before drops
6. **Resonance with caution** — A little goes a long way

# Drum Programming

Drums are the backbone of dark electronic music. They provide the driving force that moves bodies and creates hypnotic grooves.

## The Core Elements

### Kick Drum

The kick is the heartbeat. In dark electronic, it needs to hit hard.

```
define :kick do |v=1|
    sample :bd_tek, amp: 2*v, rate: 0.9
    sample :bd_boom, amp: 0.5*v, rate: 1.2, cutoff: 70
end
```

### Why two layers?

- `bd_tek` — Provides the punchy attack
- `bd_boom` — Adds low-end weight

### Key parameters:

- `rate: 0.9` — Slightly lowers pitch for more weight
- `cutoff: 70` — Removes high frequencies from the boom layer

### Snare

The snare provides accent and groove:

```
define :snare do |v=1|
    sample :sn_dub, amp: v, rate: 0.85
end
```

For more aggressive tracks:

```
define :snare do |v=1|
    sample :sn_dub, amp: v, rate: 0.8
    sample :drum_snare_hard, amp: 0.5*v, rate: 0.88
end
```

## Hi-Hats

Hi-hats provide rhythmic texture:

```
define :hat do |v=1|
    sample :drum_cymbal_closed, amp: 0.25*v, rate: 2.2, release: 0.05
end
```

## Key parameters:

- `rate: 2.2` — Higher pitch, more metallic
- `release: 0.05` — Short, tight decay

## Basic Patterns

### Four-on-the-Floor

The classic electronic drum pattern:

```

define :drums do |k=1, s=1, h=1|
  in_thread do
    4.times { kick k; sleep 1 }
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s; sleep 2
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  sleep 4
end

```

### Pattern visualization:

Beat:	1	2	3	4
Kick:	X	X	X	X
Snare:		X		X
Hat:	X	X	X	X

## Syncopated Kicks

More interesting than straight four-on-the-floor:

```

define :drums_sync do |k=1, s=1, h=1|
  in_thread do
    kick k; sleep 0.75
    kick k*0.6; sleep 0.25
    kick k*0.85; sleep 1
    kick k; sleep 0.75
    kick k*0.55; sleep 0.25
    kick k*0.9; sleep 1
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s; sleep 2
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  sleep 4
end

```

## **Pattern:**

Beat:	1	.	2	3	.	4
Kick:	X	x	X	X	x	X
	1	.75		1	.75	

The ghost kicks at 0.75 create groove.

## **Double-Time Kicks**

For maximum aggression (used in Nerve Damage):

```
define :drums_x do |k=1, s=1, h=1|
  in_thread do
    8.times do
      kick k * (0.7 + rand(0.3)) # slight velocity variation
      sleep 0.5
    end
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s; sleep 2
  end
  in_thread do
    16.times { hat h * 0.8; sleep 0.25 }
  end
  sleep 4
end
```

8 kicks per bar = relentless energy.

## **Half-Time Feel**

Slower, heavier (used in Void Walker):

```

define :drums_half do |k=1, s=1, h=1|
  in_thread do
    kick k; sleep 2
    kick k*0.7; sleep 0.5
    kick k*0.8; sleep 1.5
  end
  in_thread do
    sleep 2; snare s; sleep 2
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  sleep 4
end

```

## Velocity Variation

Static velocity = robotic and boring. Add variation:

```

define :drums do |k=1, s=1, h=1|
  in_thread do
    kick k          # Beat 1: Full
    sleep 1
    kick k*0.8      # Beat 2: Slightly softer
    sleep 1
    kick k*0.9      # Beat 3: Almost full
    sleep 1
    kick k*0.85     # Beat 4: Medium
    sleep 1
  end
  #
  # ...
end

```

This creates groove even with straight timing.

# Building Energy

## Progressive Intensity

Drums should build through a track:

```
# Intro: Reduced elements
8.times { drums 0.7, 0, 0.4 }      # Kicks + hats only

# Build: Add snares
8.times { drums 0.85, 0.6, 0.55 }  # All elements, building

# Main: Full power
12.times { drums 1, 0.9, 0.7 }     # Strong

# Peak: Maximum
12.times { drums 1.15, 1, 0.8 }    # Pushed
```

## Fading Out

For outros:

```
6.times do |i|
  drums (1-i*0.12), (0.85-i*0.1), (0.7-i*0.08)
end
```

Each iteration gets quieter.

# Pattern Variations

Keep things interesting with multiple patterns:

```

define :drums_a do |k=1, s=1, h=1|
  # Standard pattern
end

define :drums_b do |k=1, s=1, h=1|
  # Fill/variation pattern
end

# Usage
3.times { drums_a 1, 0.9, 0.7 }  # 3 bars normal
drums_b 1, 0.9, 0.7             # 1 bar fill

```

## The Hit Function

Impact moments need special treatment:

```

define :hit do |v=1|
  sample :bd_boom, amp: 2*v, rate: 0.4
  sample :drum_splash_hard, amp: 0.7*v, rate: 0.6
end

```

Used at section transitions:

```

8.times { drums 0.9, 0.7, 0.6 }
hit 1.2 # Impact before new section
8.times { drums 1, 0.9, 0.8 }

```

# Quick Reference

```
# Basic kick
define :kick do |v=1|
    sample :bd_tek, amp: 2*v, rate: 0.9
    sample :bd_boom, amp: 0.5*v, rate: 1.2
end

# Basic snare
define :snare do |v=1|
    sample :sn_dub, amp: v, rate: 0.85
end

# Basic hat
define :hat do |v=1|
    sample :drum_cymbal_closed, amp: 0.25*v, rate: 2.2
end

# Standard pattern
define :drums do |k=1, s=1, h=1|
    in_thread do
        4.times { kick k; sleep 1 }
    end
    in_thread do
        sleep 1; snare s; sleep 1; snare s; sleep 2
    end
    in_thread do
        8.times { hat h; sleep 0.5 }
    end
    sleep 4
end
```

---

Next chapters cover the individual drum elements in more detail: kick drums, snares, hi-hats, and pattern construction.

# Kick Drums That Hit Hard

The kick is the heartbeat. When it hits right, you feel it in your chest before you hear it. When it's wrong, nothing else matters — the track limps no matter how good everything else is.

In dark electronic music, the kick needs to do two things: punch through the mix and move air. That's why we layer.

## The Basic Kick

Every track in *Sonic Byte* uses a layered kick:

```
define :kick do |v=1|
  sample :bd_tek, amp: 2.2*v, rate: 0.9
  sample :bd_boom, amp: 0.5*v, rate: 1.2, cutoff: 70
end
```

## Why Two Layers?

- **:bd\_tek** — Provides the punchy attack (the “click”)
- **:bd\_boom** — Adds low-end weight (the “boom”)

Together, they create a kick that has both presence and physical impact.

## Key Parameters

### amp (Amplitude)

Controls volume. For kicks, we use higher values:

```
sample :bd_tek, amp: 2.2    # Loud, punchy
sample :bd_boom, amp: 0.5   # Supporting, not dominant
```

## rate (Playback Speed/Pitch)

Lower rate = lower pitch = more weight:

```
sample :bd_tek, rate: 1.0  # Original pitch
sample :bd_tek, rate: 0.9  # Lower, heavier
sample :bd_tek, rate: 0.85 # Even lower (Void Walker)
```

## cutoff (Filter)

Removes high frequencies for a cleaner low end:

```
sample :bd_boom, cutoff: 70  # Keeps only the sub
```

# Kick Variations by Track

## Standard (Tracks 1, 5, 8)

```
define :kick do |v=1|
  sample :bd_tek, amp: 2.2*v, rate: 0.9
  sample :bd_boom, amp: 0.5*v, rate: 1.2, cutoff: 70
end
```

## Heavy (Tracks 2, 4)

```
define :kick do |v=1|
  sample :bd_tek, amp: 2.5*v, rate: 0.85
  sample :bd_zum, amp: 0.7*v, rate: 1.1
end
```

Uses :bd\_zum for more attack, lower rate for more weight.

## Deep (Track 6 - Void Walker)

```
define :kick do |v=1|
  sample :bd_tek, amp: 2.4*v, rate: 0.85
  sample :bd_zum, amp: 0.6*v, rate: 1.0, cutoff: 65
end
```

Lowest rate (0.85) for the slowest, heaviest track.

## Sample Selection

Sonic Pi includes many kick samples. For dark electronic:

### Punchy/Attack Samples

- :bd\_tek — Tight, techno-style (our main choice)
- :bd\_haus — Punchy house kick
- :bd\_klub — Club-ready punch

### Weight/Body Samples

- :bd\_boom — Deep, boomy (good for layering)
- :bd\_zum — Punchy with sub weight
- :bd\_fat — Wide, heavy

## Testing Combinations

```
# Try different combinations
sample :bd_tek, amp: 2; sample :bd_boom, amp: 0.5
sleep 1
sample :bd_tek, amp: 2; sample :bd_zum, amp: 0.5
sleep 1
sample :bd_haus, amp: 2; sample :bd_fat, amp: 0.5
```

## The Volume Parameter

Using a volume parameter allows dynamic control:

```
define :kick do |v=1|
    sample :bd_tek, amp: 2.2*v
    sample :bd_boom, amp: 0.5*v
end

kick 1      # Full volume
kick 0.7    # 70% volume (intro)
kick 1.2    # 120% volume (peak)
```

## Kick Timing Patterns

### Four-on-the-Floor

```
4.times { kick; sleep 1 }
```

Kick on every beat. Classic, driving.

## Syncopated

```
kick; sleep 0.75
kick; sleep 0.25
kick; sleep 1
kick; sleep 1
kick; sleep 1
```

Ghost kick at 0.75 creates groove.

## Double-Time

```
8.times { kick; sleep 0.5 }
```

Kick every half beat. Aggressive, relentless.

## Half-Time

```
kick; sleep 2
kick; sleep 0.5
kick; sleep 1.5
```

Sparse, heavy. Each kick has more impact.

## Velocity Variation

Static velocity sounds robotic. Add variation:

```
in_thread do
  kick 1; sleep 1      # Beat 1: Full
  kick 0.8; sleep 1    # Beat 2: Slightly softer
  kick 0.9; sleep 1    # Beat 3: Almost full
  kick 0.85; sleep 1   # Beat 4: Medium
end
```

# Quick Reference

```
# Basic layered kick
define :kick do |v=1|
    sample :bd_tek, amp: 2.2*v, rate: 0.9
    sample :bd_boom, amp: 0.5*v, rate: 1.2, cutoff: 70
end

# Heavier kick
define :kick do |v=1|
    sample :bd_tek, amp: 2.5*v, rate: 0.85
    sample :bd_zum, amp: 0.7*v, rate: 1.1
end

# Rate guide:
# 1.0 = original pitch
# 0.9 = slightly lower (standard)
# 0.85 = noticeably lower (heavy)
```

# Snares and Claps

The snare provides the backbeat — the accent that defines the groove. In dark electronic music, snares should crack without being harsh.

## The Basic Snare

```
define :snare do |v=1|
  sample :sn_dub, amp: v, rate: 0.85
end
```

### Why :sn\_dub?

:sn\_dub has a deep, weighty character that fits dark electronic. The `rate: 0.85` lowers the pitch further, making it less “snappy” and more “thuddy.”

## Key Parameters

### rate (Pitch)

Lower rate = deeper, darker snare:

```
sample :sn_dub, rate: 1.0  # Original pitch
sample :sn_dub, rate: 0.85 # Deeper, darker (our standard)
sample :sn_dub, rate: 0.75 # Very deep (Skull Fracture)
```

## amp (Volume)

Snares are typically quieter than kicks:

```
kick 1      # Kick at 100%
snare 0.9   # Snare slightly under
```

## Snare Variations

### Standard (Most tracks)

```
define :snare do |v=1|
    sample :sn_dub, amp: v, rate: 0.85
end
```

### Industrial (Track 2 - Nerve Damage)

```
define :snare do |v=1|
    sample :sn_dub, amp: v, rate: 0.8
    sample :drum_snare_hard, amp: 0.4*v, rate: 0.9
end
```

Two layers: :sn\_dub for body, :drum\_snare\_hard for crack.

### Heavy (Track 4 - Skull Fracture)

```
define :snare do |v=1|
    sample :sn_dub, amp: v*1.1, rate: 0.75
    sample :drum_snare_hard, amp: 0.5*v, rate: 0.85
end
```

Lower rates across both layers for maximum weight.

## Tight (Track 3 - Chrome Cathedral)

```
define :snare do |v=1|
    sample :sn_dub, amp: v*0.9, rate: 0.88
end
```

Slightly higher rate for a more controlled sound in the atmospheric track.

## Snare Samples

### Body Samples

- :sn\_dub — Deep, weighty (primary choice)
- :sn\_dolf — Snappy, electronic
- :sn\_zome — Punchy, tight

### Crack/Layer Samples

- :drum\_snare\_hard — Sharp attack
- :drum\_snare\_soft — Gentler layer
- :elec\_snare — Electronic character

## Claps

Claps can replace or layer with snares:

```
define :clap do |v=1|
    sample :drum_snare_hard, amp: v*0.6, rate: 1.1
    sample :sn_dub, amp: v*0.4, rate: 0.9
end
```

Or use actual clap samples:

```
sample :perc_snap, amp: 0.8, rate: 0.9
```

## Snare Placement

### Standard (Beats 2 and 4)

```
in_thread do
  sleep 1; snare; sleep 1; snare; sleep 2
end
```

### Single (Beat 3 only - Chrome Cathedral)

```
in_thread do
  sleep 2; snare; sleep 2
end
```

Half as many snares = more space, more atmospheric.

## Syncopated

```
in_thread do
  sleep 1.5; snare 0.7; sleep 0.5 # Ghost snare
  snare; sleep 2
end
```

# Velocity Variation

```
in_thread do
  sleep 1
  snare 1      # Beat 2: Full
  sleep 1
  snare 0.95   # Beat 4: Slightly softer
  sleep 2
end
```

Subtle variation creates groove.

## Quick Reference

```
# Basic snare
define :snare do |v=1|
  sample :sn_dub, amp: v, rate: 0.85
end

# Layered snare (industrial)
define :snare do |v=1|
  sample :sn_dub, amp: v, rate: 0.8
  sample :drum_snare_hard, amp: 0.4*v, rate: 0.9
end

# Placement (beats 2 and 4)
in_thread do
  sleep 1; snare; sleep 1; snare; sleep 2
end

# Rate guide:
# 0.9+ = snappy, bright
# 0.85 = balanced (standard)
# 0.75-0.8 = deep, heavy
```

# Hi-Hats and Percussion

Hi-hats provide rhythmic texture and forward momentum. In dark electronic music, they should be crisp and metallic without being harsh.

## The Basic Hi-Hat

```
define :hat do |v=1|
  sample :drum_cymbal_closed, amp: 0.25*v, rate: 2.2, release: 0.05
end
```

### Key Parameters

**rate: 2.2** — Higher rate = higher pitch = more metallic  
**release: 0.05** — Short release = tight, crisp  
**amp: 0.25** — Quiet relative to kick/snare

## Hi-Hat Samples

### Closed Hi-Hats

- :drum\_cymbal\_closed — Our primary choice
- :drum\_cymbal\_pedal — Softer attack
- :elec\_tick — Electronic, clicky

### Open Hi-Hats (Accents)

- :drum\_cymbal\_open — Sustained ring

- `:drum_cymbal_soft` — Gentle accent

## Rate Variations

```
sample :drum_cymbal_closed, rate: 1.5 # Lower, softer
sample :drum_cymbal_closed, rate: 2.0 # Balanced
sample :drum_cymbal_closed, rate: 2.2 # Bright, metallic (standard)
sample :drum_cymbal_closed, rate: 2.5 # Very bright, aggressive
```

Higher rate = brighter, more cutting.

## Hi-Hat Patterns

### Eighth Notes (Standard)

```
in_thread do
  8.times { hat; sleep 0.5 }
end
```

Hi-hat on every eighth note. Classic, driving.

### Sixteenth Notes (Intense)

```
in_thread do
  16.times { hat; sleep 0.25 }
end
```

Double the density for aggressive tracks (Nerve Damage, Skull Fracture).

## Syncopated

```
in_thread do
  hat; sleep 0.5
  hat; sleep 0.25
  hat; sleep 0.25
  hat; sleep 0.5
  hat; sleep 0.5
  hat; sleep 0.5
  hat; sleep 0.25
  hat; sleep 0.25
  hat; sleep 0.5
end
```

## Velocity Variation

Static hi-hats sound mechanical. Add subtle variation:

```
in_thread do
  8.times do |i|
    v = [1, 0.7, 0.85, 0.75, 0.95, 0.7, 0.8, 0.9][i]
    hat v
    sleep 0.5
  end
end
```

Or randomize slightly:

```
in_thread do
  8.times do
    hat (0.7 + rand(0.3)) # Random between 0.7 and 1.0
    sleep 0.5
  end
end
```

# Open Hi-Hat Accents

Add open hats for emphasis:

```
define :hat_open do |v=1|
    sample :drum_cymbal_open, amp: 0.3*v, rate: 1.8, release: 0.2
end

# Pattern with open hat accent
in_thread do
    3.times { hat; sleep 0.5 }
    hat_open # Accent before beat 3
    sleep 0.5
    4.times { hat; sleep 0.5 }
end
```

## Hi-Hats by Track Type

### Standard (Most tracks)

```
define :hat do |v=1|
    sample :drum_cymbal_closed, amp: 0.25*v, rate: 2.2, release: 0.05
end
```

### Aggressive (Tracks 2, 4)

```
define :hat do |v=1|
    sample :drum_cymbal_closed, amp: 0.22*v, rate: 2.4, release: 0.04
end
```

Lower amp (more hats = less per hat), higher rate (brighter).

## Atmospheric (Track 3)

```
define :hat do |v=1|
  sample :drum_cymbal_closed, amp: 0.28*v, rate: 2.0, release: 0.06
end
```

Lower rate for a softer, less aggressive sound.

## Hi-Hat Dynamics Through Sections

Section	Hi-Hat amp	Character
Intro	0.4-0.5	Quiet, atmospheric
Build	0.55-0.7	Growing presence
Main	0.7-0.8	Full presence
Break	0.35-0.45	Stripped back
Peak	0.8-0.85	Maximum energy
Outro	0.7→0.3	Fading

## Percussion Beyond Hi-Hats

### Shakers

```
sample :drum_cymbal_closed, amp: 0.15, rate: 3.0, release: 0.03
```

Very high rate, very short = shaker-like.

### Rides

```
sample :drum_cymbal_open, amp: 0.2, rate: 1.5, release: 0.3
```

Lower rate, longer release = ride cymbal.

## Metallic Textures

```
sample :elec_tick, amp: 0.3, rate: 1.2
sample :elec_ping, amp: 0.2, rate: 0.8
```

## Quick Reference

```
# Basic hi-hat
define :hat do |v=1|
    sample :drum_cymbal_closed, amp: 0.25*v, rate: 2.2, release: 0.05
end

# Open hi-hat
define :hat_open do |v=1|
    sample :drum_cymbal_open, amp: 0.3*v, rate: 1.8, release: 0.2
end

# Eighth note pattern
in_thread do
    8.times { hat; sleep 0.5 }
end

# Sixteenth note pattern
in_thread do
    16.times { hat; sleep 0.25 }
end
```

# Building Drum Patterns

Individual drum sounds combine into patterns. The pattern is where groove lives.

## The Standard Pattern

```
define :drums do |k=1, s=1, h=1|
  in_thread do
    4.times { kick k; sleep 1 }
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s; sleep 2
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  sleep 4
end
```

## Visualized

Beat:	1	2	3	4
Kick:	X	X	X	X
Snare:		X		X
Hat:	X	X	X	X

## The Final Sleep

The `sleep 4` at the end is crucial — it makes the function take 4 beats total, allowing:

```
8.times { drums 1, 0.9, 0.7 } # Plays 8 bars correctly
```

# Pattern Variations

## Syncopated Kicks

```
define :drums_synco do |k=1, s=1, h=1|
  in_thread do
    kick k; sleep 0.75
    kick k*0.6; sleep 0.25 # Ghost kick
    kick k*0.85; sleep 1
    kick k; sleep 0.75
    kick k*0.55; sleep 0.25 # Ghost kick
    kick k*0.9; sleep 1
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s; sleep 2
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  sleep 4
end
```

The ghost kicks at 0.75 create swing and groove.

## Double-Time (Aggressive)

```
define :drums_x do |k=1, s=1, h=1|
  in_thread do
    8.times { kick k; sleep 0.5 } # 8 kicks per bar
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s; sleep 2
  end
  in_thread do
    16.times { hat h; sleep 0.25 } # 16th note hats
  end
  sleep 4
end
```

Used in: Nerve Damage, Skull Fracture

## Half-Time (Heavy)

```
define :drums_half do |k=1, s=1, h=1|
  in_thread do
    kick k; sleep 2
    kick k*0.7; sleep 0.5
    kick k*0.8; sleep 1.5
  end
  in_thread do
    sleep 2; snare s; sleep 2 # Snare on beat 3 only
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  sleep 4
end
```

Used in: Void Walker — fewer hits, more impact.

## Stripped (Intro/Outro)

```
define :drums_intro do |k=1, h=1|
  in_thread do
    4.times { kick k; sleep 1 }
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  sleep 4
end
```

No snare — creates anticipation.

## Combining Patterns

Use multiple patterns for variation:

```

define :drums_a do |k=1, s=1, h=1|
  # Main pattern
end

define :drums_b do |k=1, s=1, h=1|
  # Fill/variation pattern
end

# Usage: 3 bars normal, 1 bar fill
3.times { drums_a 1, 0.9, 0.7 }
drums_b 1, 0.9, 0.7

```

## Velocity Curves

### Building Energy

```

8.times do |i|
  drums (0.6 + i*0.05), (0.5 + i*0.05), (0.4 + i*0.04)
end

```

Volume rises from 0.6 to 0.95 over 8 bars.

### Fading Out

```

6.times do |i|
  drums (1 - i*0.12), (0.9 - i*0.1), (0.7 - i*0.08)
end

```

Volume falls from 1.0 to 0.28 over 6 bars.

## Pattern Density by Section

Section	Kicks/bar	Hats/bar	Feel
Intro	4	8	Driving, clean
Build	4→8	8→16	Increasing
Main	4-6	8	Groovy
Break	0-2	8	Floating
Peak	6-8	8-16	Intense
Outro	4→0	8→0	Fading

## The Hit Function

For transitions:

```
define :hit do |v=1|
  sample :bd_boom, amp: 2*v, rate: 0.4
  sample :drum_splash_hard, amp: 0.7*v, rate: 0.6
end

# Usage
8.times { drums 0.9, 0.8, 0.7 }
hit 1.2 # Impact!
8.times { drums 1, 0.9, 0.8 }
```

# Quick Reference

```
# Standard 4/4 pattern
define :drums do |k=1, s=1, h=1|
  in_thread do
    4.times { kick k; sleep 1 }
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s; sleep 2
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  sleep 4 # Important!
end

# Syncopated pattern
define :drums_synco do |k=1, s=1, h=1|
  in_thread do
    kick k; sleep 0.75
    kick k*0.6; sleep 0.25
    kick k*0.85; sleep 1
    kick k; sleep 0.75
    kick k*0.55; sleep 0.25
    kick k*0.9; sleep 1
  end
  # ... snare and hat threads
  sleep 4
end

# Building energy
8.times { |i| drums (0.6+i*0.05), (0.5+i*0.05), (0.4+i*0.04) }

# Fading out
6.times { |i| drums (1-i*0.12), (0.9-i*0.1), (0.7-i*0.08) }
```

# Bass Design

Bass is weight. Bass is darkness. Bass is the thing that makes people move without thinking about it.

In a club, you don't just hear bass — you feel it in your ribs, in your teeth, in your bones. That physical impact is what we're building here.

## The Layering Philosophy

A powerful bass isn't one sound — it's several sounds working together:

1. **Sub layer** — Pure low frequency (sine wave), felt more than heard
2. **Mid layer** — Character and grit (saw, square, TB303)
3. **Top layer** (optional) — Attack and presence

```
define :bass do |n, v=1, c=80|
  # Mid layer - character
  use_synth :tb303
  play n, amp: 0.8*v, attack: 0.01, decay: 0.2,
        sustain: 0.1, release: 0.15, cutoff: c, res: 0.3

  # Sub layer - weight
  use_synth :sine
  play n-12, amp: 1.1*v, attack: 0.01, sustain: 0.25, release: 0.2
end
```

The sub plays one octave lower ( `n-12` ) and uses a pure sine wave. The TB303 provides the gritty character.

## The TB303 Sound

The Roland TB-303 bass synthesizer defined acid house and remains central to electronic music. Sonic Pi's `:tb303` synth emulates it:

```

use_synth :tb303
play :e2,
  amp: 0.8,
  attack: 0.01,
  decay: 0.2,
  sustain: 0.1,
  release: 0.15,
  cutoff: 80,      # Filter frequency - higher = brighter
  res: 0.3,        # Resonance - higher = more "squelchy"
  wave: 0          # 0 = saw, 1 = square

```

## Key Parameters

**cutoff** (30-130): Controls brightness

- 60-70: Dark, subby
- 75-85: Balanced, present
- 90+: Aggressive, cutting

**res** (0-1): Resonance adds character

- 0.2-0.3: Subtle color
- 0.4-0.5: Classic acid sound
- 0.6+: Screaming, use carefully

**wave**: Waveform shape

- 0 (saw): Brighter, more harmonics
- 1 (square): Hollow, more fundamental

# Alternative Bass Synths

## Prophet — Warm and Musical

```
use_synth :prophet
play :d2,
  amp: 0.6,
  attack: 0.03,
  decay: 0.25,
  sustain: 0.15,
  release: 0.2,
  cutoff: 75,
  res: 0.2
```

Used in: Chrome Cathedral, Terminal Velocity

## Dsaw — Aggressive and Wide

```
use_synth :dsaw
play :d2,
  amp: 0.5,
  attack: 0.01,
  decay: 0.2,
  release: 0.15,
  cutoff: 80,
  detune: 0.2    # Spread between oscillators
```

The `detune` parameter creates width. Higher values = wider, more aggressive.

# Bass Layering Examples

## Track 1: System Override

```
define :bass do |n, v=1, c=80|
    use_synth :dsaw
    play n, amp: 0.6*v, attack: 0.01, decay: 0.15,
        sustain: 0.1, release: 0.15, cutoff: c, detune: 0.15

    use_synth :tb303
    play n, amp: 0.5*v, attack: 0, decay: 0.2,
        sustain: 0, release: 0.1, cutoff: c-10, res: 0.4, wave: 1

    use_synth :sine
    play n-12, amp: 1.2*v, attack: 0.02, sustain: 0.3, release: 0.2
end
```

Three layers:

1. **Dsaw** — Width and aggression
2. **TB303** (square wave) — Grit and character
3. **Sine** — Sub foundation

## Track 2: Nerve Damage

```
define :grind do |n, v=1, c=75|
    use_synth :tb303
    play n, amp: 0.8*v, attack: 0.01, decay: 0.2,
        sustain: 0.1, release: 0.15, cutoff: c, res: 0.3, wave: 0

    use_synth :sine
    play n-12, amp: 1.1*v, attack: 0.01, sustain: 0.25, release: 0.15
end
```

Simpler layering, but with higher resonance for that “grinding” character.

# Writing Bass Patterns

## The One-Bar Pattern

Most tracks use 4-beat bass patterns:

```
define :bassline do |v=1, c=80|
    bass :d2, v, c; sleep 0.75
    bass :d2, v*0.7, c-10; sleep 0.25
    bass :f2, v*0.9, c; sleep 0.5
    bass :d2, v*0.8, c; sleep 0.5
    bass :a2, v, c+5; sleep 0.5
    bass :d2, v, c; sleep 1.5
end
```

Notice:

- **Velocity variation** (`v*0.7`, `v*0.8`) — Creates groove
- **Cutoff variation** (`c-10`, `c+5`) — Adds movement
- **Rhythmic variety** — Mix of 0.5, 0.75, 1.5 beat notes

## Creating Movement

Use multiple bassline variations:

```

define :bass1 do |v=1, c=75|
  bass :g1,v,c; sleep 0.5
  bass :g1,v*0.5,c-5; sleep 0.5
  bass :bb1,v*0.8,c; sleep 0.5
  bass :g1,v*0.9,c; sleep 0.5
  bass :f1,v*0.7,c; sleep 0.5
  bass :g1,v,c+5; sleep 0.5
  bass :d2,v*0.8,c; sleep 0.5
  bass :g1,v,c; sleep 0.5
end

define :bass2 do |v=1, c=75|
  bass :g1,v,c; sleep 0.75
  bass :bb1,v*0.85,c; sleep 0.25
  bass :c2,v*0.9,c; sleep 0.5
  bass :d2,v,c+8; sleep 0.5
  bass :eb2,v*0.95,c+5; sleep 0.5
  bass :d2,v*0.8,c; sleep 0.5
  bass :bb1,v*0.7,c; sleep 0.5
  bass :g1,v,c; sleep 0.5
end

```

Then alternate them:

```

4.times do
  bass1 1, 80
  bass2 1, 80
end

```

## Bass Notes by Key

Each track uses a specific key. Here are the root notes:

Key	Root	Common Notes
D Minor	D	D, E, F, G, A, Bb, C
E Minor	E	E, F#, G, A, B, C, D
A Minor	A	A, B, C, D, E, F, G
F Minor	F	F, G, Ab, Bb, C, Db, Eb
C Minor	C	C, D, Eb, F, G, Ab, Bb

Key	Root	Common Notes
B Minor	B	B, C#, D, E, F#, G, A
G Minor	G	G, A, Bb, C, D, Eb, F

## Filter Automation

Make bass evolve over time:

```
in_thread do
  with_fx :lpf, cutoff: 60, cutoff_slide: 32 do |fx|
    control fx, cutoff: 90 # Slide from 60 to 90 over 32 beats
    8.times do
      bassline 1, 70
    end
  end
end
```

This creates a “filter opening” effect — the bass gets brighter over time, building energy.

## Tips for Heavy Bass

1. **Don't overcomplicate** — Two layers often beat four
2. **Keep sub clean** — No effects on the sine layer
3. **Use cutoff variation** — Even ±5 adds life
4. **Leave space** — Not every beat needs bass
5. **Match the kick** — Bass and kick should complement, not fight

---

Next: Leads and Melodies — how to write the hooks that define each track.

# The TB303 Sound

The Roland TB-303 defined acid house and remains central to electronic music. Sonic Pi's `:tb303` synth captures its essential character.

## Basic Usage

```
use_synth :tb303
play :d2, cutoff: 80, res: 0.3
```

## Key Parameters

### cutoff (Filter Frequency)

The most important parameter. Controls brightness:

```
play :d2, cutoff: 60    # Dark, subby
play :d2, cutoff: 75    # Warm, balanced
play :d2, cutoff: 90    # Bright, present
play :d2, cutoff: 110   # Aggressive, cutting
```

### res (Resonance)

Adds emphasis at the cutoff frequency — the classic “squench”:

```
play :d2, res: 0.1    # Subtle, clean
play :d2, res: 0.3    # Classic, present
play :d2, res: 0.5    # Pronounced, acid
play :d2, res: 0.7    # Extreme, screaming
```

**Warning:** High resonance can be harsh. 0.2-0.4 is usually sufficient.

## wave (Waveform)

```
play :d2, wave: 0    # Sawtooth - brighter, more harmonics  
play :d2, wave: 1    # Square - hollow, more fundamental
```

Sawtooth (wave: 0) is more common for bass. Square (wave: 1) has a darker, hollow character.

## ADSR for TB303

For punchy bass:

```
use_synth :tb303  
play :d2,  
    attack: 0.01,      # Near-instant attack  
    decay: 0.2,        # Quick drop  
    sustain: 0.1,      # Short sustain  
    release: 0.15,     # Quick release  
    cutoff: 80,  
    res: 0.3
```

This creates tight, rhythmic bass lines.

## The Album's TB303 Sound

### Standard Bass (Track 1)

```
use_synth :tb303  
play n, amp: 0.8*v, attack: 0.01, decay: 0.2,  
    sustain: 0.1, release: 0.15, cutoff: 80, res: 0.3, wave: 0
```

## Grinding Bass (Track 2)

```
use_synth :tb303
play n, amp: 0.8*v, attack: 0.01, decay: 0.2,
        sustain: 0.1, release: 0.15, cutoff: 75, res: 0.35, wave: 0
```

Higher resonance for more grit.

## Aggressive Bass (Track 4)

```
use_synth :tb303
play n, amp: 0.75*v, attack: 0, decay: 0.18,
        sustain: 0.05, release: 0.1, cutoff: 82, res: 0.35, wave: 1
```

Square wave for darker character.

## Filter Movement

The TB303's magic is in filter movement:

## Cutoff Variation in Patterns

```
define :bassline do |v=1, c=80|
  bass :d2, v, c          # Main note
  sleep 0.5
  bass :d2, v*0.7, c-10 # Ghost note, darker
  sleep 0.5
  bass :f2, v*0.9, c+5  # Accent, brighter
  sleep 1
end
```

## Cutoff Slide

```
use_synth :tb303
play :d2, cutoff: 60, cutoff_slide: 2
sleep 0.1
control cutoff: 100 # Slides from 60 to 100
```

## As Effect Automation

```
with_fx :lpf, cutoff: 50, cutoff_slide: 16 do |fx|
  control fx, cutoff: 100
  16.times do
    bass :d2
    sleep 1
  end
end
```

## TB303 vs Other Bass Synths

Synth	Character	Best For
:tb303	Squelchy, acid	Aggressive bass
:prophet	Warm, smooth	Melodic bass
:dsaw	Wide, aggressive	Layering, stabs

## When to Use TB303

- Aggressive, driving tracks
- Acid-influenced sounds
- When you need filter resonance
- Bass lines that need to cut through

## When to Use Alternatives

- Warmer, smoother bass: :prophet
- Wider, more aggressive: :dsaw
- Pure sub: :sine (always layer this)

## Layering with TB303

TB303 handles mid frequencies well. Layer with sine for sub:

```
define :bass do |n, v=1, c=80|
  use_synth :tb303
  play n, amp: 0.8*v, attack: 0.01, decay: 0.2,
    sustain: 0.1, release: 0.15, cutoff: c, res: 0.3

  use_synth :sine
  play n-12, amp: 1.1*v, attack: 0.01, sustain: 0.25, release: 0.2
end
```

The TB303 provides character; the sine provides weight.

# Quick Reference

```
# Basic TB303 bass
use_synth :tb303
play :d2,
    amp: 0.8,
    attack: 0.01,
    decay: 0.2,
    sustain: 0.1,
    release: 0.15,
    cutoff: 80,          # 60-100 typical
    res: 0.3,            # 0.2-0.4 typical
    wave: 0              # 0=saw, 1=square

# Cutoff guide:
# 60-70: Dark, subby
# 75-85: Balanced
# 90+: Aggressive

# Resonance guide:
# 0.2: Subtle
# 0.3: Classic
# 0.4+: Pronounced acid
```

# Layering Bass

A single bass synth rarely provides the full frequency spectrum needed for powerful bass. Layering fills the gaps.

## The Two-Layer Foundation

At minimum, every bass in *Sonic Byte* has two layers:

```
define :bass do |n, v=1, c=80|
    # Character layer - defines the sound
    use_synth :tb303
    play n, amp: 0.8*v, cutoff: c, res: 0.3

    # Sub layer - provides weight
    use_synth :sine
    play n-12, amp: 1.1*v  # One octave below
end
```

## Why This Works

- **TB303** covers ~80-800 Hz (audible bass character)
- **Sine** covers ~30-80 Hz (felt sub bass)
- Together: full 30-800 Hz coverage

## The Three-Layer Stack

For maximum power:

```

define :bass do |n, v=1, c=80|
  # Layer 1: Character (TB303)
  use_synth :tb303
  play n, amp: 0.75*v, attack: 0.01, decay: 0.2,
        sustain: 0.1, release: 0.15, cutoff: c, res: 0.3

  # Layer 2: Width/Grit (dsaw)
  use_synth :dsaw
  play n, amp: 0.4*v, attack: 0.01, decay: 0.15,
        release: 0.12, cutoff: c-10, detune: 0.18

  # Layer 3: Sub (sine)
  use_synth :sine
  play n-12, amp: 1.15*v, attack: 0.01, sustain: 0.25, release: 0.2
end

```

## Layer Roles

Layer	Synth	Role	Frequency
Character	:tb303	Defines sound	80-500 Hz
Width	:dsaw	Adds stereo width, grit	60-400 Hz
Sub	:sine	Physical weight	30-80 Hz

## Layer Combinations

### Aggressive (Tracks 1, 2, 4)

```

:tb303 + :sine          # Track 1
:tb303 + :sine          # Track 2
:dsaw + :tb303 + :sine   # Track 4 (most aggressive)

```

## Warm (Tracks 3, 5)

```
:prophet + :sine          # Smoother character
```

## Powerful (Tracks 6, 7)

```
:prophet + :dsaw + :sine # Warm + wide + sub
```

# Volume Balancing

Each layer needs proper level:

```
# Character: Primary presence
use_synth :tb303
play n, amp: 0.75*v  # Main volume

# Width: Supporting, not dominant
use_synth :dsaw
play n, amp: 0.4*v  # About half of character

# Sub: Felt, not heard
use_synth :sine
play n-12, amp: 1.1*v # Can be louder (low frequencies need more
energy)
```

**Rule:** If you can clearly hear the sub as a separate note, it's too loud.

# Cutoff Relationships

Filter each layer differently:

```

# Character: Main cutoff
use_synth :tb303
play n, cutoff: c      # e.g., 80

# Width: Slightly darker
use_synth :dsaw
play n, cutoff: c-10   # e.g., 70

# Sub: No cutoff needed (sine has no harmonics)
use_synth :sine
play n-12

```

The width layer is darker so it doesn't compete with the character layer.

## ADSR Alignment

Layers should have similar envelopes:

```

define :bass do |n, v=1, c=80|
  use_synth :tb303
  play n, amp: 0.75*v,
    attack: 0.01, decay: 0.2, sustain: 0.1, release: 0.15,
    cutoff: c

  use_synth :dsaw
  play n, amp: 0.4*v,
    attack: 0.01, decay: 0.15, sustain: 0.08, release: 0.12,
    cutoff: c-10

  use_synth :sine
  play n-12, amp: 1.1*v,
    attack: 0.01, sustain: 0.25, release: 0.2
end

```

**Note:** The sine sub has slightly longer sustain/release — this gives weight without muddying attacks.

# **When to Use Each Configuration**

## **Two Layers (:character + :sine)**

- Standard bass lines
- Cleaner mixes
- Atmospheric tracks

## **Three Layers (:character + :width + :sine)**

- Maximum aggression
- Peak sections
- When you need to fill the stereo field

# Quick Reference

```
# Two-layer bass
define :bass do |n, v=1, c=80|
  use_synth :tb303
  play n, amp: 0.8*v, cutoff: c, res: 0.3
  use_synth :sine
  play n-12, amp: 1.1*v
end

# Three-layer bass
define :bass do |n, v=1, c=80|
  use_synth :tb303
  play n, amp: 0.75*v, cutoff: c, res: 0.3
  use_synth :dsaw
  play n, amp: 0.4*v, cutoff: c-10, detune: 0.18
  use_synth :sine
  play n-12, amp: 1.15*v
end

# Volume guidelines:
# Character: 0.6-0.8
# Width: 0.3-0.5 (about half)
# Sub: 1.0-1.2 (can be loudest)
```

# Writing Bass Patterns

Bass patterns drive the groove. They need to lock with the kick drum while providing harmonic interest.

## The Basic Pattern

A simple 4-beat bass pattern:

```
define :bassline do |v=1, c=80|
  bass :d2, v, c; sleep 1
  bass :d2, v*0.7, c-10; sleep 1
  bass :f2, v*0.9, c; sleep 1
  bass :d2, v, c; sleep 1
end
```

## Pattern Elements

1. **Root note** (:d2) — Anchors the key
2. **Ghost note** (v\*0.7, c-10) — Adds groove
3. **Movement** (:f2) — Harmonic interest
4. **Return** (:d2) — Resolution

## Velocity Variation

Static velocity = boring. Vary it:

```
bass :d2, v, c      # Full velocity
bass :d2, v*0.7, c  # Softer (ghost)
bass :f2, v*0.9, c  # Slightly reduced
bass :d2, v*0.8, c  # Medium
bass :a2, v, c      # Full (accent)
```

**Rule:** Strong beats get full velocity, weak beats get reduced.

## Cutoff Variation

Add movement with filter changes:

```
bass :d2, v, c      # Main cutoff  
bass :d2, v*0.7, c-10 # Darker (muted)  
bass :f2, v*0.9, c+5 # Brighter (accent)
```

Even ±5-10 on cutoff adds subtle life.

## Rhythmic Patterns

### Quarter Notes (Simple)

```
bass :d2, v, c; sleep 1  
bass :d2, v, c; sleep 1  
bass :d2, v, c; sleep 1  
bass :d2, v, c; sleep 1
```

### Eighth Notes (Driving)

```
bass :d2, v, c; sleep 0.5  
bass :d2, v*0.6, c-8; sleep 0.5  
bass :d2, v*0.8, c; sleep 0.5  
bass :d2, v*0.5, c-10; sleep 0.5  
# ... continue
```

## Syncopated

```
bass :d2, v, c; sleep 0.75
bass :d2, v*0.7, c-10; sleep 0.25
bass :f2, v*0.9, c; sleep 0.5
bass :d2, v*0.8, c; sleep 0.5
# ...
```

The  $0.75 + 0.25$  creates swing.

## Harmonic Movement

### In Key Movement (D Minor)

```
# D minor scale: D E F G A Bb C
bass :d2, v, c; sleep 1      # Root
bass :f2, v, c; sleep 1      # Minor 3rd
bass :g2, v, c; sleep 1      # 4th
bass :a2, v, c; sleep 1      # 5th
```

### Common Intervals

Interval	From D	Character
Root	D	Stable, home
Minor 3rd	F	Dark color
4th	G	Movement
5th	A	Power
Minor 6th	Bb	Very dark
Minor 7th	C	Tension

## **Pattern by Key**

### **D Minor (Tracks 1, 8):**

```
bass :d2; bass :f2; bass :a2; bass :d2
```

### **E Minor (Track 2):**

```
bass :e2; bass :g2; bass :b2; bass :e2
```

### **G Minor (Track 7):**

```
bass :g2; bass :bb2; bass :d2; bass :g2
```

## **Multiple Pattern Variations**

Create variety with A/B patterns:

```

define :bass1 do |v=1, c=80|
  bass :d2, v, c; sleep 0.5
  bass :d2, v*0.6, c-8; sleep 0.5
  bass :f2, v*0.85, c; sleep 0.5
  bass :d2, v*0.75, c; sleep 0.5
  bass :a2, v*0.9, c+5; sleep 0.5
  bass :d2, v, c; sleep 1.5
end

define :bass2 do |v=1, c=80|
  bass :d2, v, c; sleep 0.75
  bass :f2, v*0.8, c; sleep 0.25
  bass :g2, v*0.9, c; sleep 0.5
  bass :a2, v, c+5; sleep 0.5
  bass :bb2, v*0.85, c; sleep 0.5 # Dark note!
  bass :a2, v*0.7, c; sleep 0.5
  bass :d2, v, c; sleep 1
end

# Usage
4.times do
  bass1 1, 80
  bass2 1, 80
end

```

## Locking with Kick

Bass and kick should complement, not fight:

```

# Kick on beats 1, 2, 3, 4
# Bass leaves space or aligns

# Good: Bass and kick together
kick; bass :d2; sleep 1

# Good: Bass between kicks
kick; sleep 0.5; bass :d2; sleep 0.5

# Careful: Bass just before kick (can muddy)
bass :d2; sleep 0.125; kick; sleep 0.875

```

# Space in Patterns

Don't fill every beat:

```
# Too busy
bass :d2; sleep 0.5
bass :d2; sleep 0.5
bass :d2; sleep 0.5
bass :d2; sleep 0.5
# ... no rest

# Better: Leave space
bass :d2, v, c; sleep 1
bass :d2, v*0.7, c; sleep 1
bass :f2, v, c; sleep 1
sleep 1 # Rest - space to breathe
```

# Quick Reference

```
# Basic pattern (4 beats)
define :bassline do |v=1, c=80|
    bass :d2, v, c; sleep 1
    bass :d2, v*0.7, c-10; sleep 1
    bass :f2, v*0.9, c; sleep 1
    bass :d2, v, c; sleep 1
end

# Velocity variation
v, v*0.7, v*0.9, v*0.8, v # Full, ghost, medium, medium, full

# Cutoff variation
c, c-10, c+5, c # Main, dark, bright, main

# Common movements (D minor)
:d2 → :f2 → :a2 → :d2 # Root, 3rd, 5th, root
:d2 → :g2 → :a2 → :d2 # Root, 4th, 5th, root
:d2 → :f2 → :bb2 → :a2 → :d2 # With dark Bb
```

# Leads and Melodies

Melodies give tracks emotional identity. In dark electronic music, leads walk the line between menacing and beautiful — they should feel dark but remain musically satisfying.

## Lead Synth Design

### The Prophet Sound

Our primary lead synth is :prophet — warm, analog-style:

```
define :lead do |n, dur=0.5, v=1|
  use_synth :prophet
  play n,
    amp: 0.4*v,
    attack: 0.05,
    decay: dur*0.25,
    sustain: dur*0.45,
    release: dur*0.5,
    cutoff: 90
end
```

### Parameters explained:

- attack: 0.05 — Quick but not instant (softer entry)
- cutoff: 90 — Bright enough to cut through, not harsh

## Layering for Richness

Single synths can sound thin. Layer for fullness:

```

define :lead do |n, dur=0.5, v=1|
  # Main layer
  use_synth :prophet
  play n, amp: 0.4*v, attack: 0.05,
        decay: dur*0.25, sustain: dur*0.45,
        release: dur*0.5, cutoff: 90

  # Shimmer layer (octave up, quiet)
  use_synth :saw
  play n+12, amp: 0.1*v, attack: 0.1,
        sustain: dur*0.3, release: dur*0.5, cutoff: 75
end

```

The high octave adds presence without overpowering.

## Dark Atmosphere Layer

For darker tracks:

```

define :lead do |n, dur=0.5, v=1|
  use_synth :prophet
  play n, amp: 0.4*v, attack: 0.08,
        decay: dur*0.3, sustain: dur*0.4,
        release: dur*0.6, cutoff: 88

  use_synth :dark_ambience
  play n, amp: 0.12*v, attack: 0.1, release: dur*0.8
end

```

## Writing Dark Melodies

### Minor Keys

All album tracks use minor keys. Common notes in minor scales:

Key	Notes
D Minor	D, E, F, G, A, Bb, C
E Minor	E, F#, G, A, B, C, D
A Minor	A, B, C, D, E, F, G
G Minor	G, A, Bb, C, D, Eb, F

## The “Dark but Pleasant” Balance

**Too dark** (dissonant, uncomfortable):

```
# Harsh intervals
lead :d4; sleep 0.5
lead :eb4; sleep 0.5 # Minor 2nd - tense
lead :ab4; sleep 0.5 # Tritone - very dark
```

**Too bright** (sounds like pop):

```
# Major-sounding
lead :d4; sleep 0.5
lead :f4; sleep 0.5 # Major 3rd if you're not careful
lead :a4; sleep 0.5 # Sounds happy
```

**Dark but musical:**

```
# Use minor intervals, resolve nicely
lead :d4; sleep 0.5
lead :f4; sleep 0.5 # Minor 3rd
lead :a4; sleep 0.5 # Fifth - strong
lead :g4; sleep 0.5 # Step down - tension
lead :f4; sleep 0.5 # Continue descent
lead :d4; sleep 1 # Resolve to root
```

## Melodic Patterns

**Ascending (builds energy):**

```

define :mel_rise do |v=1|
  lead :d4, 0.5, v; sleep 0.5
  lead :f4, 0.5, v*0.95; sleep 0.5
  lead :a4, 0.5, v; sleep 0.5
  lead :d5, 1, v; sleep 1
end

```

### Descending (releases tension):

```

define :mel_fall do |v=1|
  lead :d5, 0.5, v; sleep 0.5
  lead :c5, 0.5, v*0.9; sleep 0.5
  lead :a4, 0.5, v*0.95; sleep 0.5
  lead :g4, 0.5, v*0.85; sleep 0.5
  lead :d4, 1, v; sleep 1
end

```

### Call and response:

```

define :mel1 do |v=1| # The "question"
  lead :d4, 0.5, v; sleep 0.5
  lead :f4, 0.5, v; sleep 0.5
  lead :a4, 0.75, v; sleep 0.75
  sleep 0.25
end

define :mel2 do |v=1| # The "answer"
  lead :a4, 0.5, v; sleep 0.5
  lead :g4, 0.5, v*0.9; sleep 0.5
  lead :f4, 0.5, v*0.95; sleep 0.5
  lead :d4, 1, v; sleep 1
end

# Usage
4.times do
  mel1 0.8; mel2 0.75
end

```

## Velocity Variation

Static velocity is boring. Create expression:

```

define :melody do |v=1|
  lead :d4, 0.5, v; sleep 0.5
  lead :f4, 0.5, v*0.85; sleep 0.5      # Softer
  lead :a4, 0.75, v; sleep 0.75        # Strong
  lead :g4, 0.5, v*0.9; sleep 0.5      # Medium
  lead :f4, 0.5, v*0.8; sleep 0.5      # Softer
  lead :d4, 1, v; sleep 1              # Resolve strong
end

```

## Rhythmic Considerations

### Locking to the Grid

Melodies should feel rhythmic. Use consistent beat divisions:

```

# Good: Clear rhythmic feel
lead :d4, 0.5, v; sleep 0.5    # Half beat
lead :f4, 0.5, v; sleep 0.5    # Half beat
lead :a4, 1, v; sleep 1        # Full beat

# Awkward: Doesn't groove
lead :d4, 0.7, v; sleep 0.7    # Odd timing
lead :f4, 0.4, v; sleep 0.4    # Feels random

```

### Leaving Space

Don't fill every beat:

```

define :melody do |v=1|
  lead :d4, 0.5, v; sleep 0.5
  lead :f4, 0.5, v; sleep 0.5
  lead :a4, 1, v; sleep 1
  sleep 1                      # Rest - space to breathe
  lead :g4, 0.5, v; sleep 0.5
  lead :d4, 0.5, v; sleep 0.5
end

```

# Effects on Leads

## Standard Processing

```
with_fx :reverb, room: 0.6, mix: 0.4 do
  with_fx :echo, phase: 0.5, decay: 3, mix: 0.3 do
    4.times { melody 0.8 }
  end
end
```

## Ethereal (for breaks/intros)

```
with_fx :reverb, room: 0.95, mix: 0.6 do
  with_fx :echo, phase: 1, decay: 6, mix: 0.5 do
    melody 0.5
  end
end
```

## Tight (for peak sections)

```
with_fx :reverb, room: 0.5, mix: 0.25 do
  with_fx :echo, phase: 0.5, decay: 2, mix: 0.2 do
    4.times { melody 0.9 }
  end
end
```

# Arpeggios

Arpeggios (broken chords) add movement without complex melodies:

```

define :arp do |v=1|
  use_synth :pulse
  pattern = [:d4, :f4, :a4, :d5, :a4, :f4, :d4, :a3]

  pattern.each do |n|
    play n, amp: 0.25*v, attack: 0.01,
      decay: 0.1, release: 0.1, cutoff: 100
    sleep 0.5
  end
end

```

### Arp variations:

```

# Rising
[:d4, :f4, :a4, :d5, :f5, :a5]

# Falling
[:a5, :f5, :d5, :a4, :f4, :d4]

# Wave
[:d4, :f4, :a4, :d5, :a4, :f4, :d4, :a3]

# Pulsing
[:d4, :d4, :f4, :f4, :a4, :a4, :d5, :d5]

```

## The Melody Arrangement

### Entry Strategy

Don't start with melody. Let it enter after groove is established:

```

# MAIN section
in_thread do
  12.times { drums 1, 0.9, 0.7 }
end

in_thread do
  12.times { bassline 1, 80 }
end

in_thread do
  sleep 16                      # Wait 4 bars
  8.times { melody 0.8 }          # Then enter
end

sleep 48

```

## Building Through the Track

```

# Build: Melody teases
with_fx :reverb, room: 0.8, mix: 0.5 do
  sleep 16
  melody 0.5; melody 0.45 # Quiet, distant
end

# Main: Melody present
with_fx :reverb, room: 0.6, mix: 0.35 do
  4.times { melody 0.8 }
end

# Peak: Melody strong
with_fx :reverb, room: 0.5, mix: 0.25 do
  4.times { melody 0.95 }
end

```

# Quick Reference

```
# Basic lead
define :lead do |n, dur=0.5, v=1|
  use_synth :prophet
  play n, amp: 0.4*v, attack: 0.05,
    decay: dur*0.25, sustain: dur*0.45,
    release: dur*0.5, cutoff: 90
end

# Basic arp
define :arp do |v=1|
  use_synth :pulse
  [:d4,:f4,:a4,:d5,:a4,:f4,:d4,:a3].each do |n|
    play n, amp: 0.25*v, release: 0.1, cutoff: 100
    sleep 0.5
  end
end

# Melody with effects
with_fx :reverb, room: 0.6, mix: 0.4 do
  with_fx :echo, phase: 0.5, decay: 3, mix: 0.3 do
    4.times { melody 0.8 }
  end
end
```

---

You now have the building blocks for dark, musical melodies. The key is balance: dark enough to fit the genre, musical enough to be memorable.

# Synth Selection for Leads

Leads carry the melody. The right synth choice determines whether your melody sounds warm, aggressive, ethereal, or dark.

## The Primary Lead: Prophet

Most leads in *Sonic Byte* use `:prophet`:

```
use_synth :prophet
play :d4, amp: 0.4, attack: 0.05, decay: 0.2,
      sustain: 0.3, release: 0.4, cutoff: 90
```

### Why Prophet?

- **Warm** — Analog-style warmth
- **Musical** — Good for melodic lines
- **Versatile** — Works dark or bright
- **Cuts through** — Present without harsh

## Lead Synth Options

### `:prophet` (Warm, Musical)

```
use_synth :prophet
play :d4, cutoff: 90, res: 0.15
```

Best for: Main melodies, emotional lines

## **:saw (Raw, Bright)**

```
use_synth :saw  
play :d4, cutoff: 85
```

Best for: Shimmer layers (octave up)

## **:mod\_saw (Aggressive, Moving)**

```
use_synth :mod_saw  
play :d4, mod_phase: 0.1, mod_range: 5, cutoff: 110
```

Best for: Hooks, alarms, aggressive accents

## **:dark\_ambience (Atmospheric)**

```
use_synth :dark_ambience  
play :d4, attack: 0.1, release: 1
```

Best for: Pad layers, atmosphere under leads

## **:hollow (Ethereal)**

```
use_synth :hollow  
play :d4, attack: 0.3, release: 1.5, cutoff: 85
```

Best for: High texture, whisper layers

# **Layering Leads**

Single synths can sound thin. Layer for richness:

## Prophet + Shimmer

```
define :lead do |n, dur=0.5, v=1|
  # Main layer
  use_synth :prophet
  play n, amp: 0.4*v, attack: 0.05, decay: dur*0.25,
        sustain: dur*0.4, release: dur*0.5, cutoff: 90

  # Shimmer (octave up, quiet)
  use_synth :saw
  play n+12, amp: 0.1*v, attack: 0.08,
        sustain: dur*0.3, release: dur*0.4, cutoff: 75
end
```

## Prophet + Dark Atmosphere

```
define :lead do |n, dur=0.5, v=1|
  use_synth :prophet
  play n, amp: 0.4*v, attack: 0.05, decay: dur*0.25,
        sustain: dur*0.4, release: dur*0.5, cutoff: 88

  use_synth :dark_ambience
  play n, amp: 0.12*v, attack: 0.1, release: dur*0.8
end
```

## Attack Shapes

Attack time changes the feel:

```
attack: 0      # Instant - percussive, aggressive
attack: 0.03    # Fast - punchy but softer
attack: 0.05    # Standard - musical
attack: 0.1     # Soft - expressive
attack: 0.3+    # Slow - pad-like
```

## By Track Type

Track	Attack	Feel
Aggressive	0.02-0.03	Punchy
Standard	0.05	Musical
Atmospheric	0.08-0.1	Soft
Ethereal	0.1+	Floating

## Cutoff for Leads

Higher cutoff than bass:

```
# Bass cutoff: 70-85 typical
# Lead cutoff: 85-100 typical

use_synth :prophet
play :d4, cutoff: 90 # Bright enough to cut through
```

## By Section

Section	Lead Cutoff
Intro/Build	85-88
Main	90-92
Peak	92-95
Break	85 (+ heavy reverb)

## Lead Synth by Track

Track	Lead Synth	Character
1. System Override	:prophet	Balanced

Track	Lead Synth	Character
2. Nerve Damage	:mod_saw	Aggressive
3. Chrome Cathedral	:pulse	Atmospheric
4. Skull Fracture	:mod_saw	Alarm-like
5. Midnight Protocol	:prophet	Warm, triumphant
6. Void Walker	:prophet	Dark, sparse
7. Core Meltdown	:prophet + :saw	Rich, ethereal
8. Terminal Velocity	:prophet	Emotional

## Quick Reference

```

# Warm lead
use_synth :prophet
play :d4, amp: 0.4, attack: 0.05, cutoff: 90

# Aggressive lead
use_synth :mod_saw
play :d4, amp: 0.35, mod_phase: 0.1, mod_range: 5, cutoff: 110

# Shimmer layer (add to prophet)
use_synth :saw
play :d5, amp: 0.1, cutoff: 75 # Octave up

# Atmospheric layer
use_synth :dark_ambience
play :d4, amp: 0.12, attack: 0.1, release: 0.8

# Complete layered lead
define :lead do |n, dur=0.5, v=1|
  use_synth :prophet
  play n, amp: 0.4*v, attack: 0.05, cutoff: 90
  use_synth :saw
  play n+12, amp: 0.1*v, cutoff: 75
end

```

# Writing Dark Melodies

Dark melodies walk the line between menacing and musical. They should feel dark but remain satisfying to hear.

## The Minor Key Foundation

All tracks in *Sonic Byte* use minor keys:

Track	Key	Character
1, 8	D Minor	Dark, powerful
2	E Minor	Aggressive
3	A Minor	Atmospheric
4	F Minor	Very dark
5	C Minor	Emotional
6	B Minor	Deep
7	G Minor	Climactic

## Dark vs Happy

**Too happy (avoid):**

```
lead :d4; sleep 0.5
lead :fs4; sleep 0.5    # F# = major 3rd = happy
lead :a4; sleep 0.5
```

**Dark but musical:**

```

lead :d4; sleep 0.5
lead :f4; sleep 0.5      # F = minor 3rd = dark
lead :a4; sleep 0.5

```

The difference: F vs F#. Minor third (F) sounds dark; major third (F#) sounds happy.

## Key Dark Intervals

From any root note, these intervals add darkness:

Interval	Semitones	Character
Minor 2nd	1	Tense, dissonant
Minor 3rd	3	Dark, sad
Tritone	6	Very tense
Minor 6th	8	Dark, yearning
Minor 7th	10	Bluesy, unresolved

## Examples in D Minor

```

lead :d4; lead :eb4    # Minor 2nd - tension
lead :d4; lead :f4    # Minor 3rd - dark
lead :d4; lead :ab4    # Tritone - extreme tension
lead :d4; lead :bb4    # Minor 6th - yearning
lead :d4; lead :c5    # Minor 7th - unresolved

```

# Melodic Patterns

## Rising (Builds energy)

```
define :mel_rise do |v=1|
  lead :d4, 0.5, v; sleep 0.5
  lead :f4, 0.5, v*0.95; sleep 0.5
  lead :a4, 0.5, v; sleep 0.5
  lead :d5, 1, v; sleep 1
end
```

## Falling (Releases tension)

```
define :mel_fall do |v=1|
  lead :d5, 0.5, v; sleep 0.5
  lead :c5, 0.5, v*0.9; sleep 0.5
  lead :a4, 0.5, v*0.95; sleep 0.5
  lead :f4, 0.5, v*0.85; sleep 0.5
  lead :d4, 1, v; sleep 1
end
```

## Tension and Resolution

```
define :mel_tension do |v=1|
  lead :d4, 0.5, v; sleep 0.5
  lead :f4, 0.5, v; sleep 0.5
  lead :bb4, 0.75, v; sleep 0.75 # Bb = tension
  lead :a4, 0.5, v*0.9; sleep 0.5
  lead :d4, 1, v; sleep 1.25    # Resolve to root
end
```

## Velocity as Expression

Vary velocity for musical expression:

```

lead :d4, 0.5, v; sleep 0.5      # Strong
lead :f4, 0.5, v*0.85; sleep 0.5 # Softer
lead :a4, 0.75, v; sleep 0.75    # Strong (peak)
lead :g4, 0.5, v*0.9; sleep 0.5  # Medium
lead :f4, 0.5, v*0.8; sleep 0.5 # Softer
lead :d4, 1, v; sleep 1          # Strong (resolution)

```

**Pattern:** Strong on important notes (root, peak), softer on passing notes.

## Rhythmic Considerations

### Stay on Grid

Melodies should feel rhythmic:

```

# Good: Clean beat divisions
lead :d4, 0.5; sleep 0.5      # Half beat
lead :f4, 0.5; sleep 0.5      # Half beat
lead :a4, 1; sleep 1          # Full beat

# Awkward: Irregular timing
lead :d4, 0.7; sleep 0.7      # Doesn't groove
lead :f4, 0.4; sleep 0.4      # Feels random

```

### Leave Space

Don't fill every beat:

```

lead :d4, 0.5, v; sleep 0.5
lead :f4, 0.5, v; sleep 0.5
lead :a4, 1, v; sleep 1
sleep 1                      # Rest - space to breathe
lead :g4, 0.5, v; sleep 0.5
lead :d4, 1, v; sleep 1.5

```

# Call and Response

Create conversation between phrases:

```
define :mel1 do |v=1| # "Question"
  lead :d4, 0.5, v; sleep 0.5
  lead :f4, 0.5, v; sleep 0.5
  lead :a4, 0.75, v; sleep 1
  sleep 0.25
end

define :mel2 do |v=1| # "Answer"
  lead :a4, 0.5, v; sleep 0.5
  lead :g4, 0.5, v*0.9; sleep 0.5
  lead :f4, 0.5, v; sleep 0.5
  lead :d4, 1, v; sleep 1.5
end

# Usage
4.times do
  mel1 0.8; mel2 0.75
end
```

# Darkness Through Notes

## The b6 Note

The flat 6 (b6) is particularly dark:

```
# In D minor, Bb is the b6
lead :d4; sleep 0.5
lead :bb4; sleep 0.5    # Very dark
lead :a4; sleep 0.5    # Resolves
lead :d4; sleep 0.5
```

## The Tritone

Six semitones from root — maximum tension:

```
# In D, Ab is the tritone
lead :d4; sleep 0.5
lead :ab4; sleep 0.5    # Extreme tension!
lead :g4; sleep 0.5    # Resolves down
```

Use sparingly.

## Quick Reference

```
# D Minor scale notes
:d4, :e4, :f4, :g4, :a4, :bb4, :c5, :d5

# Dark intervals from D
:f4    # Minor 3rd - dark
:bb4   # Minor 6th - very dark (b6)
:c5    # Minor 7th - unresolved

# Avoid (sounds happy)
:fs4  # Major 3rd
:b4   # Major 6th

# Melodic pattern template
define :melody do |v=1|
  lead :d4, 0.5, v; sleep 0.5      # Root
  lead :f4, 0.5, v*0.9; sleep 0.5  # Minor 3rd
  lead :a4, 0.75, v; sleep 0.75   # 5th (peak)
  lead :g4, 0.5, v*0.85; sleep 0.5 # Passing
  lead :d4, 1, v; sleep 1.5       # Resolve
end
```

# Arpeggios

Arpeggios are broken chords — playing chord notes one at a time rather than simultaneously. They add movement and hypnotic energy without complex melody writing.

## Basic Arpeggio

```
define :arp do |v=1|
  use_synth :pulse
  notes = [:d4, :f4, :a4, :d5, :a4, :f4, :d4, :a3]

  notes.each do |n|
    play n, amp: 0.25*v, attack: 0.01, decay: 0.1,
      release: 0.1, cutoff: 100
    sleep 0.5
  end
end
```

## The Pattern

D minor chord: D, F, A

Arpeggio: D → F → A → D(up) → A → F → D → A(down)

Goes up through the chord, then back down.

# Synth Choice

## :pulse (Classic)

```
use_synth :pulse
play :d4, pulse_width: 0.35, cutoff: 100
```

Hollow, rhythmic, synthwave feel.

## :dsaw (Aggressive)

```
use_synth :dsaw
play :d4, detune: 0.1, cutoff: 95
```

Wider, more present.

## :saw (Bright)

```
use_synth :saw
play :d4, cutoff: 90
```

Raw, cutting.

# Arpeggio Patterns

## Up and Down (Standard)

```
[:d4, :f4, :a4, :d5, :a4, :f4, :d4, :a3]
# 1    3    5    8    5    3    1    5(low)
```

## **Up Only**

[:d4, :f4, :a4, :d5, :f5, :a5, :d6, :a5]

Constantly rising = building energy.

## **Down Only**

[:d5, :a4, :f4, :d4, :a3, :f3, :d3, :a2]

Falling = releasing energy.

## **With Octaves**

[:d4, :d5, :f4, :f5, :a4, :a5, :d5, :d6]

Jumping octaves adds interest.

## **Pulsing**

[:d4, :d4, :f4, :f4, :a4, :a4, :d5, :d5]

Repeated notes create drive.

## **Arpeggio by Key**

### **D Minor**

[:d4, :f4, :a4, :d5, :a4, :f4, :d4, :a3]

## A Minor

```
[:a3, :c4, :e4, :a4, :e4, :c4, :a3, :e3]
```

## G Minor

```
[:g3, :bb3, :d4, :g4, :d4, :bb3, :g3, :d3]
```

## C Minor

```
[:c4, :eb4, :g4, :c5, :g4, :eb4, :c4, :g3]
```

# Parameters

## Short Notes (Rhythmic)

```
play n, attack: 0.01, decay: 0.1, release: 0.1
```

## Longer Notes (Flowing)

```
play n, attack: 0.02, decay: 0.15, release: 0.2
```

## pulse\_width (for :pulse synth)

```
pulse_width: 0.5    # Square wave - hollow  
pulse_width: 0.35   # Thinner - our standard  
pulse_width: 0.2    # Very thin - reedy
```

# Multiple Arpeggios

Create variation with different patterns:

```
define :arp1 do |v=1| # Main pattern
  use_synth :pulse
  [:d4, :f4, :a4, :d5, :a4, :f4, :d4, :a3].each do |n|
    play n, amp: 0.25*v, release: 0.1, cutoff: 100
    sleep 0.5
  end
end

define :arp2 do |v=1| # Higher variation
  use_synth :pulse
  [:a4, :d5, :f5, :a5, :f5, :d5, :a4, :f4].each do |n|
    play n, amp: 0.22*v, release: 0.1, cutoff: 95
    sleep 0.5
  end
end

# Usage: alternate
4.times do
  arp1 0.7; arp2 0.65
end
```

# Arpeggios with Effects

Arpeggios sound great with reverb and delay:

```
with_fx :reverb, room: 0.7, mix: 0.45 do
  with_fx :echo, phase: 0.75, decay: 4, mix: 0.35 do
    4.times { arp1 0.6 }
  end
end
```

## Ethereal (For breaks)

```
with_fx :reverb, room: 0.9, mix: 0.6 do
  with_fx :echo, phase: 1, decay: 6, mix: 0.5 do
    arp1 0.4
  end
end
```

## When to Use Arpeggios

- **Intros/Outros** — Atmospheric, hypnotic
- **Builds** — Rising arpeggios build energy
- **Behind melodies** — Rhythmic bed
- **Instead of chords** — More movement than pads

## Quick Reference

```
# Basic arpeggio
define :arp do |v=1|
  use_synth :pulse
  [:d4, :f4, :a4, :d5, :a4, :f4, :d4, :a3].each do |n|
    play n, amp: 0.25*v, attack: 0.01, decay: 0.1,
      release: 0.1, cutoff: 100, pulse_width: 0.35
    sleep 0.5
  end
end

# With effects
with_fx :reverb, room: 0.7, mix: 0.45 do
  with_fx :echo, phase: 0.75, decay: 4, mix: 0.35 do
    4.times { arp 0.6 }
  end
end

# Pattern formulas (for any minor key):
# root, 3rd, 5th, octave, 5th, 3rd, root, 5th-below
```

# Generative Patterns

This is what DAWs can't do.

In a traditional DAW, every note is placed by hand. In code, you can write *rules* that generate music. Patterns that evolve. Melodies that never repeat exactly. Music that surprises even you.

## The Power of .choose

The simplest generative technique — random selection:

```
# A melody that writes itself
use_synth :prophet
8.times do
  play scale(:d4, :minor).choose, release: 0.3
  sleep 0.5
end
```

Every time you run it, different notes. All from D minor. All musical.

## Controlled Randomness

Pure random sounds chaotic. **Controlled random** sounds intentional.

## Weighted Probability

Want the root note more often? Duplicate it in the array:

```

# 50% D, 25% F, 25% A
notes = [:d2, :d2, :f2, :a2]

8.times do
  play notes.choose, release: 0.3
  sleep 0.5
end

```

The root (D) appears twice, so it's chosen twice as often. The pattern stays grounded.

## Conditional Randomness

Use `one_in()` for occasional variations:

```

define :bass_gen do |v=1|
  use_synth :tb303
  n = :d2

  # 25% chance to jump to the fifth
  n = :a2 if one_in(4)

  # 10% chance to go an octave up
  n = n + 12 if one_in(10)

  play n, amp: 0.8*v, cutoff: rrand(70, 90), release: 0.3
end

16.times do
  bass_gen
  sleep 0.5
end

```

Mostly plays D2. Sometimes A2. Occasionally jumps up an octave. The cutoff varies randomly within a range. Every run is different, but it always *grooves*.

# Generative Basslines

## The Drunk Walk

Move up or down by small intervals:

```
define :drunk_bass do
  use_synth :tb303

  n = :d2

  8.times do
    play n, cutoff: 80, release: 0.3
    sleep 0.5

    # Move up or down by 1-3 semitones, or stay
    n = n + [-3, -2, -1, 0, 0, 1, 2, 3].choose

    # Keep it in a reasonable range
    n = :d2 if n < :a1
    n = :d3 if n > :a3
  end
end
```

The bass “wanders” but stays in range. Run it multiple times — each version is different but plausible.

## Probability-Driven Patterns

```
define :gen_bassline do |v=1, c=80|
  use_synth :tb303

  4.times do
    # Root note - always
    play :d2, amp: 0.8*v, cutoff: c, release: 0.3
    sleep 0.5

    # Ghost note - 70% chance
    if rand < 0.7
      play :d2, amp: 0.5*v, cutoff: c-15, release: 0.2
    end
    sleep 0.5

    # Movement note - pick from scale
    play [:f2, :g2, :a2, :c3].choose, amp: 0.7*v, cutoff: c,
release: 0.3
    sleep 0.5

    # Return or variation
    play (one_in(4) ? :a2 : :d2), amp: 0.75*v, cutoff: c, release:
0.3
    sleep 0.5
  end
end
```

The pattern has structure (root → ghost → movement → return) but the specifics vary.

# Generative Melodies

## Scale-Based

```
define :gen_melody do |v=1|
  use_synth :prophet

  notes = scale(:d4, :minor_pentatonic) # D F G A C

  8.times do
    # Pick a note, maybe jump an octave
    n = notes.choose
    n = n + 12 if one_in(5)

    # Vary the duration
    dur = [0.25, 0.5, 0.5, 0.75].choose

    play n, amp: 0.4*v, release: dur * 0.8
    sleep dur
  end
end

with_fx :reverb, room: 0.7 do
  4.times { gen_melody 0.7 }
end
```

## Contour-Controlled

Sometimes you want shape, not just random notes:

```

define :rising_phrase do |v=1|
  use_synth :prophet
  notes = scale(:d4, :minor)

  # Start low, end high
  start_idx = rrnd_i(0, 2)    # Low note
  end_idx = rrnd_i(5, 7)      # High note

  4.times do |i|
    # Interpolate between start and end
    idx = start_idx + ((end_idx - start_idx) * i / 3.0).round
    play notes[idx], amp: 0.4*v, release: 0.4
    sleep 0.5
  end
end

```

The phrase always rises, but the exact notes vary.

## Generative Rhythms

### Euclidean Rhythms

Spread hits evenly across beats:

```

define :euclidean do |hits, steps|
  # Attempt to spread hits evenly across steps
  pattern = []
  steps.times do |i|
    if (i * hits) % steps < hits
      pattern << true
    else
      pattern << false
    end
  end
  pattern
end

# 5 hits spread across 8 steps
pattern = euclidean(5, 8)  # [true, false, true, false, true, false,
                           true, true]

loop do
  pattern.each do |hit|
    kick 1 if hit
    sleep 0.5
  end
end

```

Try different combinations: `euclidean(3, 8)` , `euclidean(7, 16)` ,  
`euclidean(5, 12)` .

## Probability Drums

```
define :gen_drums do |k=1, s=1, h=1|
  in_thread do
    4.times do
      kick k if rand < 0.9           # 90% chance
      sleep 0.5
      kick k*0.6 if rand < 0.3       # 30% ghost kick
      sleep 0.5
    end
  end

  in_thread do
    4.times do
      sleep 0.5
      snare s if rand < 0.85        # 85% chance
      sleep 0.5
      snare s*0.4 if rand < 0.15     # 15% ghost snare
    end
  end

  in_thread do
    8.times do
      hat h * rrand(0.6, 1.0)       # Random velocity
      sleep 0.5
    end
  end

  sleep 4
end

8.times { gen_drums 1, 0.9, 0.7 }
```

The groove is recognizable but never exactly the same twice.

## Rings and Ticks

For patterns that cycle but evolve:

```

# A ring loops forever
notes = ring(:d4, :f4, :a4, :g4, :f4, :e4, :d4, :c4)

# tick advances through the ring
16.times do
  play notes.tick, release: 0.3
  sleep 0.25
end

# Combine with randomness
16.times do
  n = notes.tick
  n = n + 12 if one_in(6) # Sometimes octave up
  play n, release: 0.3
  sleep 0.25
end

```

## When to Use Generative Patterns

### Good for:

- Background elements (arps, textures)
- Live performance (always fresh)
- Inspiration (let the code suggest ideas)
- Long-form music (patterns that don't bore)

### Careful with:

- Main hooks (might lose memorability)
- Tight arrangements (unpredictable timing)
- Drop moments (need precision, not randomness)

# Quick Reference

```
# Random from array
[:d2, :f2, :a2].choose

# Random from scale
scale(:d4, :minor).choose

# Weighted probability (D twice as likely)
[:d2, :d2, :f2, :a2].choose

# Conditional (25% chance)
one_in(4)

# Random in range
rrand(70, 90)      # Float between 70 and 90
rrand_i(0, 4)      # Integer 0, 1, 2, 3, or 4

# Cycling patterns
notes = ring(:d4, :f4, :a4)
notes.tick          # Advances each call

# Probability check
if rand < 0.7      # 70% chance
  # do something
end
```

## Try This

Take any pattern from the album and add one element of randomness. Start small:

- Randomize the velocity slightly
- Occasionally skip a note
- Pick cutoff from a range instead of a fixed value

Listen to how it changes the feel. Sometimes chaos. Sometimes life.

# Song Structure

A track isn't just sounds — it's a journey. Structure transforms loops into music that holds attention for 3+ minutes.

## The Standard Sections

Most tracks in *Sonic Byte* follow this structure:

INTRO → BUILD → MAIN A → BREAK → MAIN B/PEAK → OUTRO

Each section serves a purpose:

Section	Bars	Purpose
Intro	6-8	Establish mood, DJ-friendly start
Build	8	Add elements, create anticipation
Main A	12-16	Full groove, introduce hooks
Break	4-6	Strip down, create tension
Main B/Peak	12-16	Maximum energy, everything combined
Outro	4-8	Wind down, DJ-friendly ending

## Section-by-Section

### Intro (24-32 beats)

**Goal:** Set the mood without giving everything away.

**Typical elements:**

- Drums (often just kick, or kick + hats)
- Maybe filtered bass
- Atmospheric sounds

```
# INTRO: 8 bars
in_thread do
  8.times { drums 0.7, 0, 0.4 } # Kicks + hats, no snare
end

in_thread do
  with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
    control fx, cutoff: 75
    8.times { bassline 0.5, 50 } # Filtered bass
  end
end

sleep 32
```

## Build (32 beats)

**Goal:** Add elements progressively, create anticipation.

```
# BUILD: 8 bars
in_thread do
  8.times { drums 0.85, 0.7, 0.55 } # Add snares
end

in_thread do
  8.times { bassline 0.7, 65 } # Bass more present
end

in_thread do
  with_fx :reverb, room: 0.7, mix: 0.5 do
    sleep 16
    4.times { arp 0.5 } # Arp enters halfway
  end
end

sleep 32
hit 1 # Impact transitioning to main
```

## Main A (48-64 beats)

**Goal:** Full groove, introduce melodic hooks.

```
# MAIN A: 12 bars
in_thread do
  12.times { drums 1, 0.9, 0.7 }
end

in_thread do
  12.times { bassline 1, 80 } # Full bass
end

in_thread do
  6.times { arp 0.7 }
end

in_thread do
  with_fx :reverb, room: 0.6, mix: 0.4 do
    with_fx :echo, phase: 0.5, decay: 3, mix: 0.3 do
      sleep 16 # Melody enters after 4 bars
      4.times { melody 0.8 }
    end
  end
end

sleep 48
```

## Break (16-24 beats)

**Goal:** Create tension through subtraction. The silence makes the return more powerful.

```

# BREAK: 4-6 bars
in_thread do
  # Just hats, maybe some atmosphere
  12.times { hat 0.35; sleep 0.5 }
  sleep 12
end

in_thread do
  with_fx :reverb, room: 0.9, mix: 0.6 do
    with_fx :echo, phase: 1, decay: 6, mix: 0.5 do
      melody 0.5 # Distant, atmospheric melody
    end
  end
end

in_thread do
  sleep 12
  # Riser before drop
  use_synth :noise
  play :c2, amp: 0.4, attack: 12, release: 0.1, cutoff: 60
end

sleep 24
hit 1.5 # Big impact before peak

```

## Main B / Peak (48-64 beats)

**Goal:** Maximum energy. Everything at full power.

```

# PEAK: 12-14 bars
in_thread do
  14.times { drums 1.15, 1.05, 0.8 } # Louder than Main A
end

in_thread do
  7.times { bassline 1.1, 85; bassline 1.1, 88 } # Brighter
end

in_thread do
  7.times { arp 0.85 }
end

in_thread do
  with_fx :reverb, room: 0.5, mix: 0.3 do
    with_fx :echo, phase: 0.5, decay: 2.5, mix: 0.25 do
      7.times { melody 0.95 } # Strong melody
    end
  end
end

sleep 56

```

## Outro (16-32 beats)

**Goal:** Wind down gracefully. Leave space for DJ mixing.

```

# OUTRO: 6 bars
in_thread do
  6.times do |i|
    drums (1-i*0.12), (0.85-i*0.1), (0.7-i*0.08)
  end
end

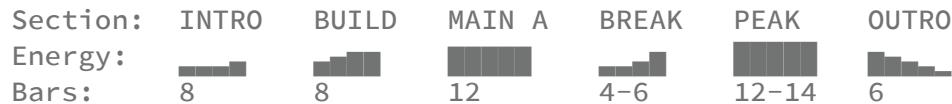
in_thread do
  with_fx :reverb, room: 0.9, mix: 0.6 do
    with_fx :echo, phase: 1, decay: 8, mix: 0.5 do
      melody 0.4; sleep 8
      arp 0.3
    end
  end
end

sleep 24

```

## Energy Mapping

Visualize energy levels:



## Transitions

### The Hit

Impact sounds signal section changes:

```

define :hit do |v=1|
  sample :bd_boom, amp: 2*v, rate: 0.4
  sample :drum_splash_hard, amp: 0.7*v, rate: 0.6
end

# Usage
8.times { drums 0.9, 0.7, 0.6 }
hit 1.2 # IMPACT
8.times { drums 1, 0.9, 0.8 }

```

## The Riser

Build tension before drops:

```

in_thread do
  sleep 24 # Wait until 6 beats before drop
  use_synth :noise
  play :g2, amp: 0.4, attack: 8, release: 0.1, cutoff: 55
end

```

## Filter Sweeps

Opening filters = building energy:

```

in_thread do
  with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
    control fx, cutoff: 110
    8.times { bassline 1 }
  end
end

```

## Subtraction Before Addition

Remove elements before a drop — silence creates impact:

```
# Bar before drop: just hats  
8.times { hat 0.4; sleep 0.5 }  
  
# Impact  
hit 1.5  
  
# Drop: everything  
drums 1.2, 1.1, 0.85
```

## Variations by Track Type

### Aggressive Track (Skull Fracture)

INTRO(6) → BUILD(8) → MAIN(12) → FAKE DROP(2) → PEAK(14) → OUTRO(6)

The “fake drop” is a brief silence before the real peak.

### Atmospheric Track (Chrome Cathedral)

INTRO(8) → BUILD(10) → MAIN A(12) → BREAK(8) → MAIN B(12) → LONG OUTRO(8)

Longer transitions, more breathing room.

### Climax Track (Core Meltdown)

INTRO(8) → BUILD(8) → MAIN A(12) → TENSION(6) → MASSIVE DROP(14) → OUTRO(6)

Longer tension section, bigger payoff.

## The Complete Pattern

Here's the full arrangement skeleton:

```

use_bpm 100

# [Sound definitions...]
# [Pattern definitions...]

# === ARRANGEMENT ===

# INTRO: 8 bars
in_thread do
  8.times { drums 0.7, 0, 0.4 }
end
in_thread do
  with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
    control fx, cutoff: 80
    8.times { bassline 0.5, 50 }
  end
end
sleep 32

# BUILD: 8 bars
in_thread do
  8.times { drums 0.85, 0.7, 0.55 }
end
in_thread do
  8.times { bassline 0.75, 68 }
end
in_thread do
  sleep 16; 4.times { arp 0.5 }
end
sleep 32
hit 1

# MAIN A: 12 bars
in_thread do
  12.times { drums 1, 0.9, 0.7 }
end
in_thread do
  12.times { bassline 1, 80 }
end
in_thread do
  with_fx :reverb, room: 0.6, mix: 0.4 do
    sleep 16; 4.times { melody 0.8 }
  end
end
sleep 48

# BREAK: 6 bars

```

```

in_thread do
  12.times { hat 0.35; sleep 0.5 }
end
in_thread do
  with_fx :reverb, room: 0.9, mix: 0.6 do
    melody 0.5
  end
end
sleep 24
hit 1.5

# PEAK: 14 bars
in_thread do
  14.times { drums 1.15, 1.05, 0.8 }
end
in_thread do
  14.times { bassline 1.1, 85 }
end
in_thread do
  with_fx :reverb, room: 0.5, mix: 0.3 do
    7.times { melody 0.95 }
  end
end
sleep 56

# OUTRO: 6 bars
in_thread do
  6.times { |i| drums (1-i*0.12), (0.85-i*0.1), (0.7-i*0.08) }
end
in_thread do
  with_fx :reverb, room: 0.9, mix: 0.6 do
    melody 0.4
  end
end
sleep 24

```

---

Structure is what separates a loop from a track. Master these patterns, then break them intentionally.

# Intro and Outro

The intro and outro frame your track. They're also essential for DJ mixing — clean intros and outros let tracks blend smoothly.

## The Intro

### Purpose

1. **Set the mood** — Establish atmosphere before full energy
2. **Introduce elements** — Bring in sounds gradually
3. **DJ-friendly** — Provide bars for mixing in

### Standard Intro (6-8 bars)

```
# INTRO: 8 bars
in_thread do
  8.times { drums 0.7, 0, 0.4 } # Kicks + hats, no snare
end

in_thread do
  with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
    control fx, cutoff: 80
    8.times { bassline 0.5, 50 } # Filtered bass
  end
end

sleep 32
```

### Intro Elements

**What to include:**

- Kicks (usually)
- Hi-hats (often)
- Filtered bass (sometimes)
- Atmospheric textures

### **What to exclude:**

- Snares (save for build/main)
- Full bass (filtered only)
- Melodies (save the hook)

## **Intro Variations**

### **Minimal (Tracks 1, 4):**

```
8.times { drums 0.7, 0, 0.4 } # Just kicks and hats
```

### **Atmospheric (Track 3):**

```
in_thread do
  6.times { drums 0.6, 0, 0.5 }
end
in_thread do
  with_fx :reverb, room: 0.85, mix: 0.55 do
    2.times { arp 0.4 }
  end
end
```

### **Building (Track 7):**

```
in_thread do
  4.times { drums 0.5, 0, 0.3 }
  4.times { drums 0.7, 0, 0.45 }
end
in_thread do
  with_fx :lpf, cutoff: 45, cutoff_slide: 32 do |fx|
    control fx, cutoff: 75
    8.times { bassline 0.4, 45 }
  end
end
```

# The Outro

## Purpose

1. **Wind down** — Release energy gradually
2. **Provide closure** — Signal the track is ending
3. **DJ-friendly** — Provide bars for mixing out

## Standard Outro (4-8 bars)

```
# OUTRO: 6 bars
in_thread do
  6.times do |i|
    drums (1-i*0.12), (0.85-i*0.1), (0.7-i*0.08)
  end
end

in_thread do
  with_fx :reverb, room: 0.9, mix: 0.6 do
    with_fx :echo, phase: 1, decay: 8, mix: 0.5 do
      melody 0.4
    end
  end
end

sleep 24
```

## Outro Techniques

### Fade drums:

```
6.times do |i|
  drums (1-i*0.12), (0.85-i*0.1), (0.7-i*0.08)
end
```

Volume decreases each bar.

### **Filter closing:**

```
with_fx :lpf, cutoff: 80, cutoff_slide: 24 do |fx|
    control fx, cutoff: 45
    6.times { bassline 0.7 }
end
```

### **Increase reverb:**

```
with_fx :reverb, room: 0.95, mix: 0.65 do
    with_fx :echo, phase: 1.25, decay: 10, mix: 0.55 do
        arp 0.3
    end
end
```

Long decay (10) means notes ring out after playing stops.

## **Outro Variations**

### **Quick fade (aggressive tracks):**

```
4.times do |i|
    drums (0.9-i*0.2), 0, (0.6-i*0.12)
end
```

### **Long fade (atmospheric tracks):**

```
8.times do |i|
    drums (0.8-i*0.08), (0.7-i*0.07), (0.6-i*0.06)
end
```

### **Definitive ending (Track 8):**

```
# Final note, held long, fading
with_fx :reverb, room: 0.95, mix: 0.65 do
    lead :d4, 4, 0.3 # Root note, 4 beats, quiet
end
```

## Intro/Outro Duration

Track Type	Intro	Outro
Aggressive	6 bars	4 bars
Standard	8 bars	6 bars
Atmospheric	8-10 bars	8 bars
Finale	8 bars	8+ bars

## Quick Reference

```
# Standard intro (8 bars)
in_thread do
  8.times { drums 0.7, 0, 0.4 }
end
in_thread do
  with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
    control fx, cutoff: 80
    8.times { bassline 0.5, 50 }
  end
end
sleep 32

# Standard outro (6 bars)
in_thread do
  6.times { |i| drums (1-i*0.12), (0.85-i*0.1), (0.7-i*0.08) }
end
in_thread do
  with_fx :reverb, room: 0.9, mix: 0.6 do
    melody 0.4
  end
end
sleep 24
```

# Builds and Drops

The build is a promise. The drop is keeping it.

A good build makes the audience *need* the drop. They know it's coming. They're waiting for it. And when it finally hits, it's not just sound — it's release. All that tension, resolved in one moment.

Get this wrong and people check their phones. Get it right and they throw their hands up.

## The Basic Cycle

LOW ENERGY → BUILD → TENSION → DROP → HIGH ENERGY

Each phase has specific techniques:

## The Build

### 1. Filter Opening

The most common build technique — start filtered, open up:

```
in_thread do
  with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
    control fx, cutoff: 110 # Opens over 32 beats
    8.times { bassline 0.8 }
  end
end
```

The sound gets brighter as energy builds.

## 2. Adding Elements

Layer in elements progressively:

```
# Bar 1-2: Kicks only
in_thread do
  8.times { kick 0.7; sleep 1 }
end

# Bar 3-4: Add hats
in_thread do
  sleep 8
  8.times { hat 0.5; sleep 0.5 }
end

# Bar 5-6: Add snares
in_thread do
  sleep 16
  snare 0.6; sleep 2; snare 0.6; sleep 2
  snare 0.7; sleep 2; snare 0.7; sleep 2
end

# Bar 7-8: Add bass
in_thread do
  sleep 24
  2.times { bassline 0.7 }
end

sleep 32
```

## 3. Rising Volume

Gradually increase amplitude:

```
8.times do |i|
  drums (0.6 + i*0.05), (0.5 + i*0.05), (0.4 + i*0.04)
end
```

Volume rises from 0.6 to 0.95 over 8 bars.

## 4. Increasing Density

More notes = more energy:

```
# Sparse (early build)
kick; sleep 1; kick; sleep 1; kick; sleep 1; kick; sleep 1

# Dense (late build)
kick; sleep 0.5; kick; sleep 0.5
```

## 5. Risers

Noise or synth rising in pitch/volume:

```
define :riser do |dur, v=1|
  use_synth :noise
  play :g2, amp: 0.4*v, attack: dur*0.9, release: dur*0.1,
    cutoff: 50, cutoff_slide: dur
  with_fx :lpf, cutoff: 50, cutoff_slide: dur do |fx|
    control fx, cutoff: 120
  end
end

# Use at end of build
in_thread do
  sleep 24 # Wait until 8 beats before drop
  riser 8, 0.8
end
```

## The Tension

The moment before the drop — maximum anticipation:

## 1. Strip the Kick

Remove the foundation:

```
# TENSION: 4 bars - no kick
in_thread do
  16.times { hat 0.4; sleep 0.5 }
end

in_thread do
  with_fx :hpf, cutoff: 90 do # Remove bass frequencies
    4.times { snare 0.5; sleep 2 }
  end
end

sleep 16
```

## 2. High-Pass Everything

Remove low frequencies to create yearning for bass:

```
with_fx :hpf, cutoff: 100 do
  with_fx :reverb, room: 0.9, mix: 0.6 do
    melody 0.5
  end
end
```

## 3. Increase Reverb

Sound floats, untethered:

```
with_fx :reverb, room: 0.95, mix: 0.7 do
  with_fx :echo, phase: 1, decay: 6, mix: 0.55 do
    lead :d4, 2, 0.5
  end
end
```

## 4. The Pause

Sometimes, silence:

```
# Last beat before drop
sleep 0.75
# Silence...
sleep 0.25
hit 1.5 # DROP
```

# The Drop

## 1. The Hit

Impact sound at the moment of drop:

```
define :hit do |v=1|
  sample :bd_boom, amp: 2*v, rate: 0.4
  sample :drum_splash_hard, amp: 0.7*v, rate: 0.6
end

# End of tension
hit 1.5
# DROP begins
drums 1.2, 1.1, 0.85
```

## 2. Everything Returns

All elements at full power:

```

# DROP: 12 bars
in_thread do
  12.times { drums 1.15, 1.05, 0.8 }
end

in_thread do
  12.times { bassline 1.1, 88 } # Brighter cutoff
end

in_thread do
  6.times { melody 0.9 }
end

sleep 48

```

### 3. Louder Than Before

The drop should be louder than the section before the build:

Section	Drums	Bass
Main A	1.0	1.0
Build	0.6→0.95	0.7→0.9
Tension	0 (no kick)	0
<b>Drop</b>	<b>1.15</b>	<b>1.1</b>

### 4. Filter Fully Open

Bass cutoff at maximum brightness:

```

# Before drop
bassline 0.9, 75 # Cutoff 75

# At drop
bassline 1.1, 90 # Cutoff 90 - brighter!

```

# Complete Build-Drop Example

```
# BUILD: 8 bars
in_thread do
  with_fx :lpf, cutoff: 50, cutoff_slide: 28 do |fx|
    control fx, cutoff: 100
    8.times { |i| drums (0.65+i*0.04), (0.5+i*0.05), (0.45+i*0.04) }
  end
end

in_thread do
  with_fx :lpf, cutoff: 55, cutoff_slide: 28 do |fx|
    control fx, cutoff: 85
    8.times { bassline (0.6+i*0.04) }
  end
end

in_thread do
  sleep 24
  riser 8, 0.7
end

sleep 32

# TENSION: 4 bars
in_thread do
  with_fx :hpfilter, cutoff: 95 do
    16.times { hat 0.35; sleep 0.5 }
  end
end

in_thread do
  with_fx :reverb, room: 0.9, mix: 0.65 do
    melody 0.5
  end
end

sleep 15.75
# Brief silence
sleep 0.25

# DROP
hit 1.5

in_thread do
  12.times { drums 1.15, 1.05, 0.85 }
```

```
end

in_thread do
  12.times { bassline 1.1, 90 }
end

in_thread do
  with_fx :reverb, room: 0.5, mix: 0.3 do
    6.times { melody 0.95 }
  end
end

sleep 48
```

## Drop Variations

### The Fake Drop

Build expectation, then deny it:

```
# FAKE DROP: Silence instead of drop
sleep 16 # Tension...
# Listeners expect drop here
sleep 4 # More silence!
hit 1.8 # REAL drop (bigger because delayed)
```

Used in: Skull Fracture

### The Half Drop

Partial return of elements:

```

# HALF DROP: Just drums, no bass
in_thread do
  4.times { drums 1, 0.9, 0.7 }
end
sleep 16

# FULL DROP: Everything
in_thread do
  12.times { drums 1.15, 1.05, 0.85 }
end
in_thread do
  12.times { bassline 1.1, 90 }
end

```

## The Slow Drop

Elements fade in rather than slam:

```

# SLOW DROP: 4 bars of building into full
in_thread do
  4.times { |i| drums (0.7+i*0.12), (0.6+i*0.12), (0.5+i*0.1) }
  12.times { drums 1.15, 1.05, 0.85 }
end

```

Used in: Chrome Cathedral (atmospheric track)

## Quick Reference

### Build techniques:

- Filter opening (lpf cutoff slide up)
- Adding elements progressively
- Rising volume
- Increasing note density
- Risers

### Tension techniques:

- Remove kick
- High-pass filter
- Increase reverb
- Silence/pause

**Drop techniques:**

- Hit/impact sound
- All elements return
- Louder than before
- Filter fully open

# Tension and Release

The tension-release cycle is fundamental to music. Creating tension makes release satisfying; constant intensity becomes exhausting.

## What Creates Tension

### 1. Removing Elements (Subtraction)

The most powerful technique — take things away:

```
# MAIN section: Full elements
in_thread do
  drums 1, 0.9, 0.7
  bassline 1, 80
  melody 0.8
end

# BREAK section: Stripped
in_thread do
  # No kick, no bass
  hat 0.4
  with_fx :reverb, room: 0.9 do
    melody 0.5
  end
end
```

When elements return, the contrast is powerful.

### 2. High-Pass Filtering

Remove bass frequencies to create yearning:

```
with_fx :hpf, cutoff: 90 do
  # Everything sounds thin, incomplete
  melody 0.6
  drums 0.5, 0.4, 0.5 # Even drums lose weight
end
```

### 3. Increasing Reverb

Sound becomes untethered, floating:

```
# Normal
with_fx :reverb, room: 0.6, mix: 0.35 do
  melody 0.8
end

# Tension
with_fx :reverb, room: 0.95, mix: 0.65 do
  melody 0.5 # Sounds distant, yearning
end
```

### 4. Harmonic Tension

Use dissonant notes:

```
# In D minor, these create tension:
lead :bb4 # b6 - dark, unresolved
lead :e4 # 2nd - wants to resolve
lead :ab4 # tritone - maximum tension
```

### 5. Risers

Building noise or synth:

```

define :riser do |dur, v=1|
  use_synth :noise
  play :g2, amp: 0.4*v, attack: dur*0.9, release: dur*0.1,
        cutoff: 50, cutoff_slide: dur
end

# 8-beat riser before drop
riser 8, 0.8

```

## What Creates Release

### 1. Elements Returning

After subtraction, addition feels powerful:

```

# After stripped break...
hit 1.5 # Impact!
drums 1.2, 1.1, 0.85 # Everything back, louder
bassline 1.1, 90      # Brighter than before

```

### 2. Filter Opening

Bass returning after high-pass:

```

# Tension: high-passed
with_fx :hpf, cutoff: 90 do
  4.times { melody 0.5 }
end

# Release: full frequency
melody 0.9
bassline 1, 85 # Bass is back!

```

### 3. Resolution to Root

Melodies resolving to the key's root note:

```
# Tension
lead :bb4; sleep 0.5 # Dark, unresolved
lead :a4; sleep 0.5 # Still tense

# Release
lead :d4; sleep 1    # Root note = home
```

### 4. The Hit/Impact

A punctuation mark signaling release:

```
define :hit do |v=1|
  sample :bd_boom, amp: 2*v, rate: 0.4
  sample :drum_splash_hard, amp: 0.7*v
end

# After tension...
hit 1.5
# Release begins
```

# The Tension-Release Pattern

```
# MAIN A: Full energy (48 beats)
in_thread do
  12.times { drums 1, 0.9, 0.7 }
end
in_thread do
  12.times { bassline 1, 80 }
end
sleep 48

# TENSION: Stripped (16-24 beats)
in_thread do
  with_fx :hpfilter, cutoff: 90 do
    16.times { hat 0.35; sleep 0.5 }
  end
end
in_thread do
  with_fx :reverb, room: 0.9, mix: 0.6 do
    melody 0.5
  end
end
in_thread do
  sleep 12
  riser 8, 0.8
end
sleep 24
hit 1.5

# RELEASE/MAIN B: Full energy, louder (48 beats)
in_thread do
  12.times { drums 1.15, 1.05, 0.8 }
end
in_thread do
  12.times { bassline 1.1, 88 }  # Brighter!
end
sleep 48
```

# Tension Duration

Track Type	Tension Duration	Effect
Aggressive	2-4 bars	Quick, punchy
Standard	4-6 bars	Balanced
Atmospheric	6-8 bars	Building dread
Climactic	6 bars	Maximum anticipation

# Tension Intensity

## Light Tension

Just reduce some elements:

```
drums 0.6, 0.5, 0.6 # Quieter, but present  
bassline 0.7, 70     # Filtered
```

## Medium Tension

Remove kick, high-pass:

```
with_fx :hpf, cutoff: 80 do  
  drums 0, 0.4, 0.5 # No kick  
end
```

## Heavy Tension

Remove most elements:

```
# Just hats and reverbed melody
hat 0.35
with_fx :reverb, room: 0.95, mix: 0.7 do
  melody 0.4
end
```

## Quick Reference

```
# Creating tension:
# 1. Remove kick
drums 0, s, h

# 2. High-pass filter
with_fx :hpf, cutoff: 90 do ... end

# 3. More reverb
with_fx :reverb, room: 0.95, mix: 0.65 do ... end

# 4. Riser
riser 8, 0.8

# Creating release:
# 1. Hit/impact
hit 1.5

# 2. Everything returns, louder
drums 1.2, 1.1, 0.85
bassline 1.1, 90

# 3. Resolve to root note
lead :d4, 1, v # Home
```

# DJ-Friendly Arrangements

Even if you never DJ, structuring tracks for DJ use makes them more professional. These principles create cleaner, more impactful music.

## Why DJ-Friendly?

1. **Clean transitions** — Intro/outro without melody clutter
2. **Predictable structure** — 8, 16, 32-bar phrases
3. **Energy flow** — Clear builds and drops
4. **Mixable** — Tracks can blend together

## Phrase Length

DJs count in 8-bar phrases. Structure your sections accordingly:

Bars	Beats	Common Use
4	16	Short break, transition
8	32	Intro, outro, build
12-16	48-64	Main sections

```
# Good: 8-bar intro
8.times { drums 0.7, 0, 0.4 } # 32 beats

# Good: 12-bar main
12.times { drums 1, 0.9, 0.7 } # 48 beats

# Avoid: 7-bar section (awkward for DJs)
7.times { drums 1, 0.9, 0.7 } # 28 beats - hard to mix
```

## Clean Intro

The intro should let DJs mix in smoothly:

### Do:

- Start with drums (kick, hats)
- Filter any bass/melodic elements
- Keep it rhythmically simple

### Don't:

- Start with full melody
- Use complex rhythms
- Change tempo

```
# DJ-friendly intro
in_thread do
  8.times { drums 0.7, 0, 0.4 } # Kicks + hats only
end
in_thread do
  with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
    control fx, cutoff: 75
    8.times { bassline 0.5, 50 } # Heavily filtered
  end
end
sleep 32
```

## Clean Outro

The outro should let DJs mix out:

## Do:

- End with drums fading
- Remove melodic elements early
- Provide 4-8 bars of minimal content

## Don't:

- End abruptly
- Have melody playing to the last beat
- Use complex fills at the end

```
# DJ-friendly outro
in_thread do
  # Melody ends early
  with_fx :reverb, room: 0.9, mix: 0.55 do
    melody 0.4 # Once, then done
  end
end

in_thread do
  # Drums continue and fade
  6.times do |i|
    drums (1-i*0.15), 0, (0.6-i*0.08)
  end
end

sleep 24
```

## Predictable Changes

Changes should happen on expected beats:

```

# Changes happen at bar boundaries
8.times { drums 0.7, 0, 0.4 }    # Bar 1-8: Intro
hit 1
8.times { drums 0.85, 0.7, 0.6 } # Bar 9-16: Build
hit 1.1
12.times { drums 1, 0.9, 0.7 }   # Bar 17-28: Main

```

DJs can predict when changes happen (every 8 bars).

## Energy Markers

Help DJs identify sections with clear audio cues:

### The Hit

```

define :hit do |v=1|
  sample :bd_boom, amp: 2*v, rate: 0.4
  sample :drum_splash_hard, amp: 0.7*v
end

# Mark section changes
8.times { drums 0.8, 0.6, 0.5 }
hit 1.2 # Clear marker: new section coming
8.times { drums 1, 0.9, 0.7 }

```

### The Break

Clear tension before drops:

```

# BREAK: Obviously different
in_thread do
  # No kick - clear signal
  16.times { hat 0.35; sleep 0.5 }
end
sleep 16
hit 1.5
# DROP: Everything returns

```

## The 4-Bar Warning

Professional tracks often “warn” 4 bars before major changes:

```

# Main section
8.times { drums 1, 0.9, 0.7 }

# Warning: Something's changing
# (subtle fill, filter movement, added element)
in_thread do
  with_fx :lpf, cutoff: 85, cutoff_slide: 16 do |fx|
    control fx, cutoff: 60 # Filter starts closing
    4.times { drums 1, 0.9, 0.7 }
  end
end
sleep 16

# Change happens
hit 1.2
# New section...

```

# Standard DJ-Friendly Structure

INTRO	8 bars	Drums only, filtered bass
BUILD	8 bars	Add elements, filter opens
MAIN A	12 bars	Full groove
BREAK	4 bars	Stripped, tension
MAIN B	12 bars	Full groove, peak energy
OUTRO	6 bars	Drums fade, melody ends

Total: ~50 bars, ~3:20 at 100 BPM

## Quick Reference

```
# DJ-friendly phrase lengths
8.times { ... }    # Intro, build, outro
12.times { ... }   # Main sections
4.times { ... }    # Breaks, transitions

# Clean intro (drums + filtered bass)
in_thread do
  8.times { drums 0.7, 0, 0.4 }
end
in_thread do
  with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
    control fx, cutoff: 75
    8.times { bassline 0.5, 50 }
  end
end

# Clean outro (fade drums, melody ends early)
in_thread do
  melody 0.4  # Once
end
in_thread do
  6.times { |i| drums (1-i*0.15), 0, (0.6-i*0.08) }
end

# Mark changes with hits
hit 1.2
```

# Dynamics and Energy

Dynamics are the volume relationships in your music. Good dynamics create excitement; poor dynamics create fatigue.

## The Dynamic Range

Dynamic range is the difference between quietest and loudest moments:

Track Type	Quietest	Loudest	Range
Pop (compressed)	0.7	1.0	Small
<i>Sonic Byte</i>	0.3	1.2	Medium-Large
Classical	0.1	1.0	Very Large

More range = more emotional impact.

## Section Dynamics

Each section has a baseline energy level:

Section	Drum Level	Bass Level	Overall
Intro	0.6-0.7	0.5	Low
Build	0.7→0.95	0.6→0.9	Rising
Main	1.0	1.0	Full
Break	0.3-0.5	0	Very Low
Peak	1.1-1.2	1.1	Maximum
Outro	1.0→0.3	0.8→0	Falling

# Building Energy

## Progressive Volume

```
8.times do |i|
  drums (0.6 + i*0.05), (0.5 + i*0.05), (0.4 + i*0.04)
end
# Volume: 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95
```

## Adding Elements

```
# Bar 1-2: Kicks only
in_thread do
  8.times { kick 0.7; sleep 1 }
end

# Bar 3-4: Add hats
in_thread do
  sleep 8
  8.times { hat 0.5; sleep 0.5 }
end

# Bar 5-6: Add snares
in_thread do
  sleep 16
  sleep 1; snare 0.6; sleep 1; snare 0.6; sleep 2
  sleep 1; snare 0.7; sleep 1; snare 0.7; sleep 2
end

# Bar 7-8: Add bass
in_thread do
  sleep 24
  2.times { bassline 0.8 }
end

sleep 32
```

## Filter Opening (Perceived Loudness)

```
with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
  control fx, cutoff: 110
  # Sound gets brighter = perceived louder
  8.times { bassline 0.8 }
end
```

## Releasing Energy

### Progressive Fade

```
6.times do |i|
  drums (1-i*0.12), (0.9-i*0.1), (0.7-i*0.08)
end
# Volume: 1.0, 0.88, 0.76, 0.64, 0.52, 0.4
```

## Removing Elements

```
# Bar 1-2: Full mix
in_thread do
  2.times { drums 1, 0.9, 0.7; bassline 1; melody 0.8 }
end

# Bar 3-4: Remove melody
in_thread do
  sleep 8
  2.times { drums 0.9, 0.8, 0.65; bassline 0.9 }
end

# Bar 5-6: Remove bass
in_thread do
  sleep 16
  2.times { drums 0.75, 0.6, 0.5 }
end

# Bar 7-8: Drums fading
in_thread do
  sleep 24
  drums 0.5, 0.4, 0.35
  drums 0.3, 0, 0.2
end
```

## Filter Closing

```
with_fx :lpf, cutoff: 85, cutoff_slide: 24 do |fx|
  control fx, cutoff: 50
  # Sound gets darker = perceived quieter
  6.times { bassline 0.8 }
end
```

## The Loudest Moment

Every track should have a clear peak — the loudest moment:

```

# PEAK section
in_thread do
  drums 1.15, 1.05, 0.85 # 15% louder than Main
end
in_thread do
  bassline 1.1, 90        # Brighter cutoff
end
in_thread do
  melody 0.95            # Strong melody
end

```

## Peak Timing

The peak usually happens:

- Around 2/3 through the track
- After a tension/break section
- Before the outro begins

## Contrast Creates Impact

The peak feels loudest because of **contrast** with quiet moments:

```

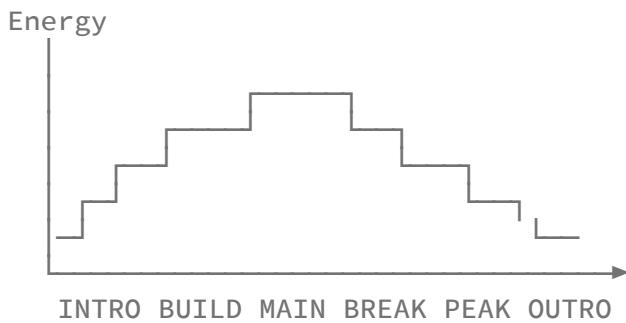
# BREAK: Very quiet (16 beats)
in_thread do
  16.times { hat 0.35; sleep 0.5 }
end
sleep 16

# PEAK: Full volume
hit 1.5
in_thread do
  drums 1.2, 1.1, 0.85 # Feels MASSIVE after quiet break
end

```

Without the quiet break, the peak would feel ordinary.

# Dynamic Curve Visualization



## Quick Reference

```
# Building energy
8.times { |i| drums (0.6+i*0.05), (0.5+i*0.05), (0.4+i*0.04) }

# Releasing energy
6.times { |i| drums (1-i*0.12), (0.9-i*0.1), (0.7-i*0.08) }

# Section levels
# Intro:  drums 0.7, 0, 0.4
# Build:  drums 0.85, 0.7, 0.55
# Main:   drums 1, 0.9, 0.7
# Break:   drums 0, 0, 0.35 (or just hats)
# Peak:    drums 1.15, 1.05, 0.85
# Outro:   drums fade from 1 to 0.3
```

# Volume Automation

Volume changes over time create movement and energy. Static volumes sound lifeless.

## The Volume Parameter

Every sound function uses a volume parameter:

```
define :kick do |v=1|
    sample :bd_tek, amp: 2.2*v
end

kick 1      # Full volume (amp = 2.2)
kick 0.5    # Half volume (amp = 1.1)
kick 1.2    # Boosted (amp = 2.64)
```

## Progressive Volume Changes

### Building Up

```
8.times do |i|
    # i goes: 0, 1, 2, 3, 4, 5, 6, 7
    volume = 0.6 + i*0.05  # 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9,
0.95
    drums volume, volume*0.85, volume*0.7
end
```

## Fading Down

```
6.times do |i|
  volume = 1.0 - i*0.12 # 1.0, 0.88, 0.76, 0.64, 0.52, 0.4
  drums volume, volume*0.9, volume*0.7
end
```

## Custom Curves

```
# Exponential build (slow start, fast finish)
volumes = [0.5, 0.55, 0.62, 0.71, 0.82, 0.9, 0.96, 1.0]
volumes.each do |v|
  drums v, v*0.9, v*0.7
end

# S-curve (slow-fast-slow)
volumes = [0.5, 0.55, 0.65, 0.8, 0.9, 0.95, 0.98, 1.0]
```

## Section Volume Levels

### Intro

```
drums 0.7, 0, 0.4          # Kicks + hats, no snare
bassline 0.5, 50            # Quiet, filtered
```

### Build

```
drums 0.85, 0.7, 0.55      # Growing
bassline 0.75, 68           # More present
```

## Main

```
drums 1, 0.9, 0.7      # Full
bassline 1, 80          # Full
melody 0.8              # Present but not dominant
```

## Break

```
drums 0, 0, 0.35        # Just hats (or nothing)
# No bass
melody 0.5              # Quiet, reverbed
```

## Peak

```
drums 1.15, 1.05, 0.85   # Louder than main
bassline 1.1, 88          # Louder, brighter
melody 0.95                # Strong
```

## Outro

```
# Fading over 6 bars
6.times { |i| drums (1-i*0.12), (0.85-i*0.1), (0.7-i*0.08) }
```

## Element Balance

Not everything should be the same volume:

Element	Typical Range	Role
Kick	amp: 2.0-2.5	Foundation
Snare	amp: 0.8-1.1	Accent
Hat	amp: 0.2-0.3	Texture

Element	Typical Range	Role
Bass	amp: 0.6-0.9 (synth)	Body
Sub	amp: 1.0-1.2	Weight
Lead	amp: 0.35-0.45	Melody
Arp	amp: 0.2-0.3	Rhythm
Pad	amp: 0.25-0.4	Atmosphere

## Velocity Variation Within Patterns

Add life with subtle variation:

```
define :bassline do |v=1, c=80|
  bass :d2, v, c; sleep 0.5
  bass :d2, v*0.7, c-10; sleep 0.5      # Ghost note
  bass :f2, v*0.9, c; sleep 0.5        # Slightly softer
  bass :d2, v*0.8, c; sleep 0.5        # Medium
  bass :a2, v, c+5; sleep 0.5          # Accent (full)
  bass :d2, v, c; sleep 1.5            # Resolution (full)
end
```

## Automation Patterns

### Swell

```
# Quiet → loud → quiet
volumes = [0.4, 0.6, 0.8, 1.0, 0.8, 0.6, 0.4]
volumes.each do |v|
  melody v
end
```

## Pulse

```
# Alternating loud/soft
8.times do |i|
  v = i.even? ? 1.0 : 0.7
  drums v, v*0.9, v*0.7
end
```

## Ramp

```
# Steady increase
16.times do |i|
  v = 0.5 + (i * 0.03125) # 0.5 to 1.0 over 16 bars
  drums v, v*0.9, v*0.7
end
```

## Quick Reference

```
# Building volume
8.times { |i| drums (0.6+i*0.05) }

# Fading volume
6.times { |i| drums (1-i*0.12) }

# Section volumes
# Intro: 0.5-0.7
# Build: 0.7-0.95
# Main: 1.0
# Break: 0.3-0.5
# Peak: 1.1-1.2
# Outro: 1.0>0.3

# Velocity variation in patterns
v, v*0.7, v*0.9, v*0.8, v # Full, ghost, medium, medium, full
```

# Filter Sweeps

Filter sweeps are one of the most powerful tools in electronic music. They create movement, build energy, and signal transitions.

## The Basic Sweep

```
with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
    control fx, cutoff: 110 # Sweeps from 50 to 110 over 32 beats
    8.times { bassline 1 }
end
```

## How It Works

1. `cutoff: 50` — Start position (dark, muffled)
2. `cutoff_slide: 32` — Transition time (32 beats = 8 bars)
3. `control fx, cutoff: 110` — Target position (bright)

## Opening Filter (Building Energy)

The classic build technique:

```

# BUILD section
with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
  control fx, cutoff: 110

  in_thread do
    8.times { drums 0.85, 0.7, 0.55 }
  end

  in_thread do
    8.times { bassline 0.75, 60 }
  end

  sleep 32
end

```

Sound gets brighter = energy builds.

## Closing Filter (Releasing Energy)

For outros or transitions down:

```

# OUTRO section
with_fx :lpf, cutoff: 85, cutoff_slide: 24 do |fx|
  control fx, cutoff: 45

  6.times { |i|
    drums (1-i*0.1), (0.85-i*0.08), (0.7-i*0.06)
  }
end

```

Sound gets darker = energy releases.

## Sweep Speed

Beats	Bars	Feel
8	2	Fast, urgent

Beats	Bars	Feel
16	4	Medium
32	8	Slow, building
64	16	Very gradual

```

# Fast sweep (2 bars)
with_fx :lpf, cutoff: 60, cutoff_slide: 8 do |fx|
  control fx, cutoff: 100
  2.times { bassline 1 }
end

# Slow sweep (8 bars)
with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
  control fx, cutoff: 110
  8.times { bassline 1 }
end

```

## High-Pass Sweeps

Remove bass frequencies for tension:

```

# BREAK: High-pass removes bass
with_fx :hpf, cutoff: 40, cutoff_slide: 16 do |fx|
  control fx, cutoff: 100 # Bass disappears

  4.times { drums 0.7, 0.5, 0.6 }
end

# Then release
with_fx :hpf, cutoff: 100, cutoff_slide: 4 do |fx|
  control fx, cutoff: 20 # Bass returns quickly

  drums 1.1, 1, 0.8 # DROP
end

```

# Combined Sweeps

Use both LPF and HPF:

```
# Narrowing to mid frequencies only
with_fx :lpf, cutoff: 120, cutoff_slide: 16 do |lpf|
  with_fx :hpf, cutoff: 30, cutoff_slide: 16 do |hpf|
    control lpf, cutoff: 80    # Remove highs
    control hpf, cutoff: 90    # Remove lows

    # Only mid frequencies remain
    4.times { melody 0.6 }
  end
end
```

## On Synths vs Effects

### On Effects (Affects Everything)

```
with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
  control fx, cutoff: 110
  # ALL sounds inside are filtered
  drums 1; bassline 1; melody 0.8
end
```

### On Synths (Affects One Sound)

```
use_synth :tb303
play :d2, cutoff: 60, cutoff_slide: 4
sleep 0.1
control cutoff: 100  # Only this note sweeps
```

# Sweep Patterns

## Intro Pattern

```
with_fx :lpf, cutoff: 45, cutoff_slide: 32 do |fx|
  control fx, cutoff: 80
  # Start very dark, open partially
end
```

## Build Pattern

```
with_fx :lpf, cutoff: 60, cutoff_slide: 32 do |fx|
  control fx, cutoff: 100
  # Open up for the drop
end
```

## Break Pattern

```
with_fx :hpf, cutoff: 30, cutoff_slide: 16 do |fx|
  control fx, cutoff: 95
  # Remove bass for tension
end
```

## Drop Pattern

```
# Instant open (no sweep)
with_fx :lpf, cutoff: 110 do
  # Full brightness immediately
end
```

## Outro Pattern

```
with_fx :lpf, cutoff: 85, cutoff_slide: 24 do |fx|
  control fx, cutoff: 50
  # Close down as track ends
end
```

## Quick Reference

```
# Opening filter (building)
with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
  control fx, cutoff: 110
  8.times { bassline 1 }
end

# Closing filter (releasing)
with_fx :lpf, cutoff: 85, cutoff_slide: 24 do |fx|
  control fx, cutoff: 50
  6.times { bassline 1 }
end

# High-pass for tension
with_fx :hpf, cutoff: 40, cutoff_slide: 16 do |fx|
  control fx, cutoff: 100
  4.times { drums 0.7, 0.5, 0.6 }
end

# Cutoff values:
# 40-60: Very dark, muffled
# 60-80: Dark, warm
# 80-100: Balanced, present
# 100-120: Bright, aggressive
```

# Using Effects for Energy

Effects aren't just polish — they're compositional tools that shape energy and emotion.

## Effects and Energy Relationship

More Effect	Energy Change
More reverb	Lower energy (floating, distant)
Less reverb	Higher energy (present, punchy)
More delay	Lower energy (spacious, hypnotic)
Less delay	Higher energy (tight, direct)
Lower cutoff	Lower energy (dark, muffled)
Higher cutoff	Higher energy (bright, aggressive)

## Section-Based Effects

### Intro (Atmospheric)

```
with_fx :reverb, room: 0.8, mix: 0.5 do
  with_fx :echo, phase: 1, decay: 5, mix: 0.45 do
    arp 0.5
  end
end
```

High reverb + long delay = distant, building.

## Build (Opening)

```
with_fx :reverb, room: 0.65, mix: 0.4 do
  with_fx :echo, phase: 0.75, decay: 4, mix: 0.35 do
    melody 0.7
  end
end
```

Moderate effects = presence increasing.

## Main (Present)

```
with_fx :reverb, room: 0.6, mix: 0.35 do
  with_fx :echo, phase: 0.5, decay: 3, mix: 0.3 do
    melody 0.8
  end
end
```

Tighter effects = full energy, present.

## Break (Floating)

```
with_fx :reverb, room: 0.95, mix: 0.65 do
  with_fx :echo, phase: 1.5, decay: 8, mix: 0.55 do
    melody 0.5
  end
end
```

Maximum effects = tension, floating, yearning.

## Peak (Punchy)

```
with_fx :reverb, room: 0.5, mix: 0.25 do
  with_fx :echo, phase: 0.5, decay: 2.5, mix: 0.25 do
    melody 0.95
  end
end
```

Minimal effects = maximum impact, energy.

## Outro (Fading)

```
with_fx :reverb, room: 0.9, mix: 0.6 do
  with_fx :echo, phase: 1.25, decay: 10, mix: 0.55 do
    melody 0.4
  end
end
```

Long decay = notes fade into infinity.

# Effect Parameter Ranges

## Reverb Room Size

Value	Character	Use
0.4-0.5	Tight	Peak, drops
0.6-0.7	Medium	Main sections
0.8-0.9	Large	Builds, intros
0.95-1.0	Huge	Breaks, outros

## Echo Decay

Value	Character	Use
2-3	Short	Tight, punchy
4-5	Medium	Standard
6-8	Long	Atmospheric
10+	Very long	Outros, fades

## Mix (Wet/Dry)

Value	Character	Use
0.2-0.3	Subtle	Peak energy
0.35-0.45	Balanced	Main
0.5-0.6	Prominent	Atmospheric
0.65+	Drenched	Breaks, ethereal

## Effect Automation

### Reverb Building

```
# Increasing reverb over a section
[:0.5, :0.6, :0.7, :0.8].each do |room|
  with_fx :reverb, room: room, mix: 0.4 do
    melody 0.7
  end
end
```

## Delay Increasing

```
# Longer delays as section progresses
[3, 4, 5, 6].each do |decay|
  with_fx :echo, phase: 0.75, decay: decay, mix: 0.4 do
    arp 0.6
  end
end
```

## Contrast for Impact

The break's heavy effects make the drop feel more powerful:

```
# BREAK: Maximum effects
with_fx :reverb, room: 0.95, mix: 0.6 do
  with_fx :echo, phase: 1.5, decay: 8, mix: 0.5 do
    melody 0.5 # Floating, distant
  end
end

hit 1.5

# DROP: Minimal effects
with_fx :reverb, room: 0.5, mix: 0.25 do
  with_fx :echo, phase: 0.5, decay: 2, mix: 0.2 do
    drums 1.15, 1.05, 0.85 # Punchy, immediate
    melody 0.95
  end
end
```

# Element-Specific Effects

## Drums (Usually Dry)

```
# Little to no reverb on kicks
with_fx :reverb, room: 0.4, mix: 0.15 do
  drums 1, 0.9, 0.7
end
```

## Bass (Dry)

```
# No reverb on bass (keeps it tight)
bassline 1, 80 # No effects wrapper
```

## Leads (Wet)

```
# Reverb + delay on leads
with_fx :reverb, room: 0.6, mix: 0.4 do
  with_fx :echo, phase: 0.5, decay: 3, mix: 0.3 do
    melody 0.8
  end
end
```

## Arps (Very Wet)

```
# Heavy effects on arps
with_fx :reverb, room: 0.75, mix: 0.5 do
  with_fx :echo, phase: 0.75, decay: 4, mix: 0.4 do
    arp 0.6
  end
end
```

# Quick Reference

```
# Low energy (atmospheric)
with_fx :reverb, room: 0.9, mix: 0.6 do
  with_fx :echo, phase: 1, decay: 6, mix: 0.5 do
    # Floating, distant
  end
end

# Full energy (present)
with_fx :reverb, room: 0.6, mix: 0.35 do
  with_fx :echo, phase: 0.5, decay: 3, mix: 0.3 do
    # Present, driving
  end
end

# Maximum energy (punchy)
with_fx :reverb, room: 0.5, mix: 0.25 do
  with_fx :echo, phase: 0.5, decay: 2, mix: 0.2 do
    # Tight, impactful
  end
end

# Effects by section:
# Intro: room 0.8, decay 5
# Build: room 0.65, decay 4
# Main: room 0.6, decay 3
# Break: room 0.95, decay 8
# Peak: room 0.5, decay 2.5
# Outro: room 0.9, decay 10
```

# Going Live

The tracks in this album are arranged linearly — intro to outro, scripted from start to finish. But Sonic Pi was built for **live coding**: changing music while it plays.

This chapter bridges the gap between composition and performance.

## Static vs Live

### Static (Album Style)

```
define :drums do |k=1, s=1, h=1|
  in_thread do
    4.times { kick k; sleep 1 }
  end
  # ...
  sleep 4
end

# Plays once and stops
8.times { drums 1, 0.9, 0.7 }
```

### Live (Performance Style)

```
live_loop :drums do
  kick 1
  sleep 1
end
```

The `live_loop` runs forever until you stop it. Change the code and press Run — it updates on the next loop.

# Converting to Live Loops

## Basic Conversion

**From:**

```
define :drums do |k=1|
  4.times { kick k; sleep 1 }
end

8.times { drums 1 }
```

**To:**

```
live_loop :drums do
  kick 1
  sleep 1
end
```

## With Control Variables

```
# Global controls – change these while playing
set :kick_vol, 1
set :bpm, 100

live_loop :drums do
  use_bpm get(:bpm)
  kick get(:kick_vol)
  sleep 1
end
```

Now you can change the kick volume by running just this line:

```
set :kick_vol, 0.5
```

The loop picks up the change immediately.

# Multiple Synchronized Loops

```
use_bpm 100

live_loop :kick do
  sample :bd_tek, amp: 2
  sleep 1
end

live_loop :snare do
  sleep 1
  sample :sn_dub, amp: 0.9
  sleep 1
end

live_loop :hats do
  sample :drum_cymbal_closed, amp: 0.25, rate: 2.2
  sleep 0.5
end
```

Each loop runs independently but they stay synchronized.

# Building Arrangement Live

Use `set` and `get` to control song state:

```

set :section, :intro

live_loop :drums do
  case get(:section)
  when :intro
    kick 0.7
  when :main
    kick 1
    sleep 0.5
    kick 0.6
    sleep 0.5
    # Skip the final sleep
    next
  when :break
    # Silence
  end
  sleep 1
end

live_loop :bass do
  case get(:section)
  when :intro
    # Filtered
    bass :d2, 0.5, 60
  when :main
    bass :d2, 1, 80
  when :break
    # Silence
  end
  sleep 1
end

```

Change the section during performance:

```
set :section, :main    # Run this to switch
```

## The sync Command

Wait for another loop before playing:

```
live_loop :drums do
  kick 1
  sleep 1
end

live_loop :bass, sync: :drums do
  # Waits for :drums to complete one cycle before starting
  bass :d2, 1, 80
  sleep 1
end
```

## Performance Tips

### Start Simple

Begin with just kicks. Add elements one at a time.

```
# Start here
live_loop :kick do
  sample :bd_tek, amp: 2
  sleep 1
end

# Then add this (press Run again)
live_loop :hats do
  sample :drum_cymbal_closed, amp: 0.2, rate: 2.2
  sleep 0.5
end
```

### Use Comments to Mute

```
live_loop :bass do
  # bass :d2, 1, 80  # Commented out = muted
  sleep 1
end
```

## Keep a “Kill” Buffer

```
# Run this to stop everything  
stop
```

## Volume Faders

```
set :master, 1.0  
  
live_loop :drums do  
  kick get(:master)  
  sleep 1  
end  
  
# Fade down  
set :master, 0.7  
set :master, 0.5  
set :master, 0
```

## Example: Track 01 as Live Loops

```
use_bpm 100

set :energy, 0.5 # Start low

# === SOUNDS ===
define :kick do |v=1|
    sample :bd_tek, amp: 2*v, rate: 0.9
end

define :snare do |v=1|
    sample :sn_dub, amp: v, rate: 0.85
end

define :hat do |v=1|
    sample :drum_cymbal_closed, amp: 0.25*v, rate: 2.2
end

define :bass do |n, v=1, c=80|
    use_synth :dsaw
    play n, amp: 0.6*v, cutoff: c, detune: 0.15, release: 0.3
    use_synth :sine
    play n-12, amp: v, release: 0.3
end

# === LOOPS ===
live_loop :kicks do
    kick get(:energy)
    sleep 1
end

live_loop :snare do
    sleep 1
    snare get(:energy) * 0.9
    sleep 1
end

live_loop :hats do
    hat get(:energy) * 0.7
    sleep 0.5
end

live_loop :bassline do
    e = get(:energy)
    c = 60 + (e * 30) # Cutoff follows energy
```

```

bass :d2, e, c
sleep 0.5
bass :d2, e*0.6, c-15
sleep 0.5
bass :f2, e*0.85, c
sleep 1
bass :d2, e, c
sleep 1
bass :a2, e*0.9, c+5
sleep 1
end

# === PERFORMANCE ===
# Run these lines one at a time during performance:

# set :energy, 0.7    # Building
# set :energy, 1.0    # Full
# set :energy, 0.3    # Break
# set :energy, 1.2    # Peak
# set :energy, 0       # Stop

```

## Further Reading

Live coding is deep. This chapter just scratches the surface.

The [Sonic Pi tutorial](#) has excellent sections on:

- `live_loop` in detail
- `sync` and `cue` for coordination
- `tick` and `ring` for patterns
- Time state with `set` and `get`

For live performance inspiration, search YouTube for “Sonic Pi live coding” or check out [Algorave](#).

# Track Walkthroughs

Part IV examines each track in detail. You'll see how the concepts from Parts I-III come together in complete compositions.

---



**Listen to the album on SoundCloud** while you read. Hearing the music makes everything click.

---

## How to Use This Section

Each track walkthrough covers:

1. **Overview** — Purpose in the album, key characteristics
2. **Sound Design** — The specific sounds and why they were chosen
3. **Patterns** — Drum, bass, and melodic patterns
4. **Arrangement** — Section-by-section breakdown
5. **Hacker Challenges** — Modifications for you to try

## The Tracks

#	Track	BPM	Key	Focus
01	System Override	100	D min	Album opener, establishing the sound
02	Nerve Damage	105	E min	Industrial aggression, double-time kicks
03	Chrome Cathedral	98	A min	Atmosphere, space, reverb techniques
04	Skull Fracture	108	F min	Maximum aggression, hook sounds

#	Track	BPM	Key	Focus
05	Midnight Protocol	102	C min	Synthwave melody, arpeggios
06	Void Walker	95	B min	Slow power, half-time feel
07	Core Meltdown	106	G min	Album climax, combined techniques
08	Terminal Velocity	100	D min	Cinematic finale, return to origin

## Suggested Reading Order

If you want to learn progressively:

1. Start with **Track 01** (System Override) — the simplest structure
2. Then **Track 03** (Chrome Cathedral) — introduces atmosphere
3. Then **Track 05** (Midnight Protocol) — introduces melody focus
4. Then **Track 07** (Core Meltdown) — combines everything

If you want specific techniques:

- Double-time drums → Track 02 (Nerve Damage)
- Atmospheric effects → Track 03 (Chrome Cathedral)
- Hook/signature sounds → Track 04 (Skull Fracture)
- Arpeggios → Track 05 (Midnight Protocol)
- Half-time/slow power → Track 06 (Void Walker)
- Ethereal leads → Track 07 (Core Meltdown)
- Fade-out arrangement → Track 08 (Terminal Velocity)

## Running the Code

Each track's complete code is available as a `.rb` file:

1. Open Sonic Pi
2. Copy the track code into a buffer
3. Press Run (or Cmd+R / Alt+R)
4. Listen and follow along with the walkthrough

## Experimentation Encouraged

As you read each walkthrough, try:

- **Changing one parameter** — What happens if cutoff is 60 instead of 80?
- **Swapping patterns** — What if Track 01's drums used Track 02's pattern?
- **Adjusting arrangement** — What if the break was longer?
- **Combining elements** — Can you merge ideas from different tracks?

The best way to learn is to break things, then fix them.

---

Let's begin with the album opener: System Override.

# 01: System Override

**BPM:** 100 | **Key:** D Minor | **Duration:** ~2:30

The album opener. System Override establishes the sound with aggressive industrial textures and relentless 4-on-the-floor beats.

## The Vision

As the first track, System Override needs to:

1. **Establish the aesthetic** — Dark, aggressive, industrial
2. **Set energy expectations** — This album will hit hard
3. **Introduce key sounds** — Bass, drums, leads we'll develop
4. **Be relatively simple** — Save complexity for later tracks

## Sound Design

### Kick Drum

```
define :kick do |v=1|
  sample :bd_tek, amp: 2*v, rate: 0.9
  sample :bd_boom, amp: 0.5*v, rate: 1.2, cutoff: 70
end
```

Two layers:

- **bd\_tek** — The main punch, pitched down slightly ( `rate: 0.9` )
- **bd\_boom** — Adds low-end weight

## Snare

```
define :snare do |v=1|
  sample :sn_dub, amp: v, rate: 0.85
end
```

Simple but effective. The `rate: 0.85` gives it a slightly deeper, more industrial character.

## Hi-Hat

```
define :hat do |v=1|
  sample :drum_cymbal_closed, amp: 0.3*v, rate: 2.2, release: 0.05
end
```

High `rate` makes it tight and metallic. Short `release` keeps it crisp.

## Bass

```
define :bass do |n, v=1, c=80|
  use_synth :dsaw
  play n, amp: 0.6*v, attack: 0.01, decay: 0.15,
         sustain: 0.1, release: 0.15, cutoff: c, detune: 0.15

  use_synth :sine
  play n-12, amp: v, attack: 0.02, sustain: 0.3, release: 0.2
end
```

Two layers: aggressive dsaw on top, pure sine sub underneath.

## Lead

```
define :lead do |n, dur=0.25, v=1|
  use_synth :prophet
  play n, amp: 0.4*v, attack: 0.02, decay: dur*0.5,
    sustain: dur*0.3, release: dur*0.4, cutoff: 95

  use_synth :dark_ambience
  play n, amp: 0.15*v, attack: 0.05, release: dur*0.8
end
```

The `dark_ambience` layer adds atmosphere without overwhelming the melody.

## Patterns

### Drum Pattern

```
define :drums do |k=1, s=1, h=1|
  in_thread do
    4.times { kick k; sleep 1 }
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s; sleep 2
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  sleep 4
end
```

Classic 4-on-the-floor: kick on every beat, snare on 2 and 4, hats on eighth notes.

## Bassline

```
define :bassline do |v=1, c=80|
  bass :d2, v, c; sleep 0.75
  bass :d2, v*0.7, c-10; sleep 0.75
  bass :f2, v*0.9, c; sleep 0.5
  bass :d2, v*0.8, c; sleep 0.5
  bass :a2, v, c+5; sleep 0.5
  bass :d2, v, c; sleep 1
end
```

The pattern:

- Starts on root (D)
- Ghost note with lower velocity and cutoff
- Moves to F (minor third)
- Returns to D
- Jumps to A (fifth)
- Resolves to D

## Melodic Hook

```
define :hook do |v=1|
  lead :d4, 0.25, v; sleep 0.25
  lead :f4, 0.25, v*0.8; sleep 0.25
  lead :e4, 0.25, v*0.6; sleep 0.25
  lead :d4, 0.25, v*0.9; sleep 0.5
  lead :a4, 0.5, v; sleep 0.5
  lead :g4, 0.25, v*0.7; sleep 0.5
  lead :d4, 0.75, v; sleep 1
end

define :hook2 do |v=1|
  lead :a4, 0.5, v; sleep 0.5
  lead :g4, 0.25, v*0.8; sleep 0.25
  lead :f4, 0.25, v*0.7; sleep 0.25
  lead :e4, 0.5, v*0.9; sleep 0.5
  lead :d4, 0.25, v; sleep 0.25
  lead :c4, 0.25, v*0.6; sleep 0.25
  lead :d4, 1, v; sleep 1.25
end
```

Two complementary phrases that play call-and-response.

## Arrangement

INTRO	8 bars	Kicks only, then hats
BUILD	8 bars	Add snares, bass enters low-passed
MAIN A	12 bars	Full groove, hooks enter
BREAK	4 bars	Strip kick, tension with reverb
MAIN B	12 bars	Return harder, full energy
OUTRO	4 bars	Drums only, fading for mix

### Intro (32 beats)

```
in_thread do
  8.times { drums 0.8, 0, 0 } # Kicks only
end
in_thread do
  sleep 16
  4.times { drums 0, 0, 0.5 } # Add hats
end
sleep 32
```

Notice: drums 0.8, 0, 0 — kick at 80%, no snare (0), no hats (0).

### Build (32 beats)

```
in_thread do
  8.times { drums 0.9, 0.7, 0.6 }
end
in_thread do
  8.times { bassline 0.7, 65 } # Low cutoff = filtered
end
sleep 32
hit 1
```

The bass enters with cutoff at 65 (darker). The `hit` function provides impact at the end.

## Main A (48 beats)

```
in_thread do
  12.times { drums 1, 0.9, 0.75 }
end
in_thread do
  12.times { bassline 1, 85 } # Full cutoff now
end
in_thread do
  sleep 8
  4.times { hook 0.7; sleep 1; hook2 0.6; sleep 1 }
end
sleep 48
```

Full energy. The hook enters after 8 beats, letting the groove establish first.

## Break (16 beats)

```
with_fx :reverb, room: 0.8 do
  with_fx :hpf, cutoff: 85 do
    in_thread do
      16.times { hat 0.4; sleep 0.5 }
    end
    in_thread do
      hook 0.6; sleep 4; hook2 0.5
    end
  end
end
sleep 16
hit 1.2
```

The high-pass filter (`hpf`) removes bass frequencies, creating tension. Big hit at the end signals the return.

## Main B (48 beats)

```
in_thread do
  12.times { drums 1.1, 1, 0.85 } # Louder than Main A
end
in_thread do
  12.times { bassline 1.1, 90 }    # Brighter cutoff
end
in_thread do
  6.times { hook 0.85; sleep 1; hook2 0.75; sleep 1 }
end
sleep 48
```

Everything is slightly louder and brighter than Main A — this is the peak.

## Outro (16 beats)

```
in_thread do
  4.times do |i|
    drums (0.9-i*0.2), (0.7-i*0.15), (0.6-i*0.12)
  end
end
sleep 16
```

The `|i|` creates a counter (0, 1, 2, 3). Each iteration gets quieter, creating a fade.

## Key Takeaways

1. **Layer sounds** — Two samples for kick, two synths for bass
2. **Use velocity variation** — `v*0.7, v*0.8` creates groove
3. **Filter automation** — Cutoff changes add movement
4. **Structural contrast** — Break strips energy before peak
5. **Clean outro** — Drums only for DJ mixing

# Hacker Challenges

Don't just read — **hack**. Try these modifications:

1. **Half-Time Feel:** Change the drum pattern so the kick only hits on beats 1 and 3. How does the energy change? Does it feel heavier or lighter?
2. **Randomize the Ghost Notes:** In the bassline, replace the fixed ghost note with randomness:

```
bass [:d2, :f2].choose, v*0.7, c-10
```

Run it several times. Does it still groove?

3. **Glitch the Break:** Wrap the break section in a bitcrusher:

```
with_fx :bitcrusher, bits: 8, sample_rate: 8000 do
  # break code here
end
```

Does the digital destruction fit the industrial aesthetic?

4. **Extend the Hook:** Write a `hook3` that answers `hook` and `hook2`. What notes feel like a natural continuation?
5. **Swap the Bass Synth:** Replace `:dsaw` with `:tb303`. Adjust the cutoff and add `res: 0.3`. Which version is more aggressive?

## Full Code

The complete track code is available in `01_system_override.rb`.

---

Next: [Track 02: Nerve Damage — Pushing aggression with industrial textures.](#)

# 02: Nerve Damage

**BPM:** 105 | **Key:** E Minor | **Duration:** ~3:00

Industrial crusher. Nerve Damage pushes the aggression established in Track 1, introducing double-time kicks and grinding textures.

## The Vision

Track 2's role is to say "Track 1 wasn't a fluke — this album means business." It needs to:

1. **Escalate intensity** — Faster, harder than System Override
2. **Introduce industrial textures** — Grinding, mechanical sounds
3. **Maintain groove** — Aggressive but still danceable

## What Makes It Unique

### Double-Time Kicks

While Track 1 used standard four-on-the-floor, Nerve Damage doubles the kick rate:

```

define :drums_x do |k=1, s=1, h=1|
  in_thread do
    8.times do
      kick k
      sleep 0.5 # Kick every half beat = 8 per bar
    end
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s; sleep 2
  end
  in_thread do
    16.times { hat h; sleep 0.25 } # Sixteenth note hats
  end
  sleep 4
end

```

**The effect:** Relentless, driving energy. The 8 kicks per bar create a steamroller feel.

## The “Grind” Bass

Named for its grinding, industrial character:

```

define :grind do |n, v=1, c=75|
  use_synth :tb303
  play n, amp: 0.8*v, attack: 0.01, decay: 0.2,
    sustain: 0.1, release: 0.15, cutoff: c, res: 0.3, wave: 0
  use_synth :sine
  play n-12, amp: 1.1*v, attack: 0.01, sustain: 0.25, release: 0.15
end

```

### Key choices:

- res: 0.3 — Higher resonance than Track 1 for more “squench”
- wave: 0 — Saw wave for brightness and grit
- Named grind not bass — reflects its character

## Industrial Snare

Layered for mechanical punch:

```
define :snare do |v=1|
    sample :sn_dub, amp: v, rate: 0.8
    sample :drum_snare_hard, amp: 0.4*v, rate: 0.9
end
```

The second layer adds metallic crack.

## Sound Design

### Kick (same as Track 1)

```
define :kick do |v=1|
    sample :bd_tek, amp: 2.2*v, rate: 0.9
    sample :bd_boom, amp: 0.5*v, rate: 1.2, cutoff: 70
end
```

Consistency across album — the kick is our anchor.

### Hi-Hat (tighter)

```
define :hat do |v=1|
    sample :drum_cymbal_closed, amp: 0.22*v, rate: 2.4, release: 0.04
end
```

Higher rate (2.4 vs 2.2) and shorter release — tighter, more mechanical.

## Signature Riff

A grinding melodic hook:

```

define :riff do |v=1|
  use_synth :mod_saw
  play :e3, amp: 0.35*v, attack: 0, decay: 0.08,
    sustain: 0.04, release: 0.06, mod_phase: 0.12, mod_range: 5,
  cutoff: 105
  sleep 0.25
  play :e3, amp: 0.25*v, attack: 0, decay: 0.06,
    sustain: 0.03, release: 0.05, mod_phase: 0.15, mod_range: 6,
  cutoff: 100
  sleep 0.25
  play :g3, amp: 0.3*v, attack: 0, decay: 0.07,
    sustain: 0.04, release: 0.06, mod_phase: 0.12, mod_range: 5,
  cutoff: 108
  sleep 0.5
end

```

The `mod_saw` with modulation creates an alarm-like, industrial texture.

## Patterns

### Bassline

```

define :bassline do |v=1, c=75|
  grind :e2, v, c; sleep 0.5
  grind :e2, v*0.6, c-8; sleep 0.5
  grind :g2, v*0.85, c; sleep 0.5
  grind :e2, v*0.75, c-5; sleep 0.5
  grind :d2, v*0.9, c; sleep 0.5
  grind :e2, v, c+5; sleep 0.5
  grind :b2, v*0.8, c; sleep 0.5
  grind :e2, v, c; sleep 0.5
end

```

**E Minor movement:** E (root) → G (minor 3rd) → D (7th) → B (5th)

## Riff Pattern

```
define :riff_pattern do |v=1|
  2.times { riff v }
  sleep 1.5
  riff v*0.8
  sleep 0.5
end
```

The riff plays twice, pauses, then once more — creating rhythmic interest.

## Arrangement

INTRO (32) → BUILD (32) → MAIN A (48) → BREAK (16) → MAIN B (48) → OUTRO (24)

## The Intensity Curve

Section	Kicks	Snare	Hats	Bass	Riff
Intro	8/bar	—	light	filtered	—
Build	8/bar	enters	building	opening	teases
Main A	8/bar	full	full	full	full
Break	—	—	only	—	reverbed
Main B	8/bar	full	full	brighter	full
Outro	fading	fading	fading	—	echoing

## Key Moment: The Build

```
# BUILD: 8 bars
in_thread do
  8.times { drums_x 0.85, 0.7, 0.55 }
end

in_thread do
  with_fx :lpf, cutoff: 55, cutoff_slide: 32 do |fx|
    control fx, cutoff: 95
  8.times { bassline 0.75, 60 }
  end
end

in_thread do
  sleep 16
  with_fx :reverb, room: 0.7, mix: 0.5 do
    4.times { riff_pattern 0.5 }
  end
end

sleep 32
hit 1.1
```

The bass filter opens from 55→95 while the riff enters with reverb — building anticipation.

## Key Techniques

### 1. Double-Time for Intensity

Doubling the kick rate is the simplest way to increase energy:

```
# Normal: 4 kicks per bar
4.times { kick; sleep 1 }

# Double: 8 kicks per bar
8.times { kick; sleep 0.5 }
```

## 2. Velocity Variation on Fast Patterns

With 8 kicks per bar, static velocity sounds like a machine gun. Add variation:

```
in_thread do
  8.times do |i|
    v = [1, 0.7, 0.85, 0.75, 0.95, 0.7, 0.8, 0.9][i]
    kick v
    sleep 0.5
  end
end
```

## 3. Industrial Texture with mod\_saw

The `mod_saw` synth creates mechanical, alarm-like sounds:

```
use_synth :mod_saw
play :e3,
  mod_phase: 0.12, # Speed of modulation
  mod_range: 5,     # Depth (semitones)
  cutoff: 105
```

Lower `mod_phase` = faster wobble = more aggressive.

## 4. BPM Escalation

Track 1: 100 BPM → Track 2: 105 BPM

The 5 BPM increase is subtle but felt. It maintains groove while adding urgency.

## Mixing Considerations

With double-time kicks, mix carefully:

- **Kick volume:** Slightly lower than Track 1 (2.2 vs 2.5) to prevent overwhelming

- **Hat volume:** Lower (0.22 vs 0.25) — there are twice as many
- **Bass:** Same level, but filter more aggressively in intros

## Hacker Challenges

1. **Quad-Time Madness:** Push the kicks even further — 16 per bar instead of 8. At what point does it stop being music and become noise? Where's the line?
2. **Swing the Hats:** Instead of perfect 16th notes, try:

```
16.times do |i|
    hat h
    sleep i.even? ? 0.27 : 0.23 # Slight shuffle
end
```

Does it feel more human or just sloppy?

3. **Filter the Riff:** The `mod_saw` riff is bright and cutting. Wrap it in a low-pass filter that opens during the build. Does it create more anticipation?
4. **Remove the Sub Layer:** Delete the `:sine` sub from the bass. What's lost? Is the track still powerful or does it feel hollow?
5. **Create a Counter-Riff:** Write a second riff pattern that plays against the first. Can you create call-and-response between two `mod_saw` lines?

## Full Code

The complete track code is available in `02_nerve_damage.rb`.

---

Next: [Track 03: Chrome Cathedral](#) — atmospheric contrast.

# 03: Chrome Cathedral

**BPM:** 98 | **Key:** A Minor | **Duration:** ~3:15

Atmospheric cyberpunk. After two aggressive tracks, Chrome Cathedral provides breathing room with spacious reverbs and hypnotic arpeggios.

## The Vision

Track 3 is the album's first "breath." It needs to:

1. **Provide contrast** — Slower, more spacious than Tracks 1-2
2. **Maintain darkness** — Atmospheric, not soft
3. **Showcase effects** — Reverb and delay take center stage
4. **Be hypnotic** — Repetition with subtle evolution

## What Makes It Unique

### Slow, Spacious BPM

At 98 BPM, Chrome Cathedral is noticeably slower:

Track	BPM	Feel
System Override	100	Driving
Nerve Damage	105	Urgent
<b>Chrome Cathedral</b>	<b>98</b>	<b>Hypnotic</b>

The slower tempo creates space for reverb tails and atmospheric textures.

## Arpeggio-Driven

Instead of aggressive leads, Chrome Cathedral uses hypnotic arpeggios:

```
define :arp1 do |v=1|
  use_synth :pulse
  notes = [:a3, :c4, :e4, :a4, :e4, :c4, :a3, :e3]
  notes.each do |n|
    play n, amp: 0.22*v, attack: 0.01, decay: 0.12,
           release: 0.12, cutoff: 98, pulse_width: 0.35
    sleep 0.5
  end
end
```

**The pattern:** A minor arpeggio up and down — simple, hypnotic.

## Heavy Reverb Processing

Everything is drenched in space:

```
with_fx :reverb, room: 0.85, mix: 0.55 do
  with_fx :echo, phase: 0.75, decay: 5, mix: 0.45 do
    4.times { arp1 0.7 }
  end
end
```

Compare to Track 1's tighter processing:

- Track 1: room: 0.6, mix: 0.4
- Track 3: room: 0.85, mix: 0.55

## Warm Prophet Bass

Softer than the TB303 aggression:

```

define :bass do |n, v=1, c=72|
  use_synth :prophet
  play n, amp: 0.55*v, attack: 0.03, decay: 0.25,
    sustain: 0.2, release: 0.25, cutoff: c, res: 0.18
  use_synth :sine
  play n-12, amp: 1.0*v, attack: 0.02, sustain: 0.3, release: 0.25
end

```

### Key differences from Track 1:

- Prophet instead of TB303/dsaw — warmer tone
- Lower cutoff (72 vs 80) — darker
- Longer attack (0.03 vs 0.01) — softer entry

## Sound Design

### Pad Layer

Atmospheric background texture:

```

define :pad do |notes, v=1|
  use_synth :dark_ambience
  play notes, amp: 0.3*v, attack: 2, decay: 1,
    sustain: 3, release: 4, cutoff: 70
end

```

Long ADSR values create evolving, washy textures.

## Drums (Stripped Back)

```
define :drums do |k=1, s=1, h=1|
  in_thread do
    kick k; sleep 1.5
    kick k*0.6; sleep 0.5
    kick k*0.8; sleep 1
    kick k; sleep 1
  end
  in_thread do
    sleep 1.5; snare s*0.7; sleep 2.5
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  sleep 4
end
```

### Key differences:

- Syncopated kicks (not straight four-on-floor)
- Only one snare per bar (beat 2.5)
- Same hi-hat pattern, but lower volume

## Whisper Texture

High, ethereal layer:

```
define :whisper do |n, v=1|
  use_synth :hollow
  play n, amp: 0.15*v, attack: 0.5, decay: 0.3,
         sustain: 0.4, release: 1.5, cutoff: 85
end
```

The `hollow` synth adds ghostly presence.

# Patterns

## Bassline (Minimal)

```
define :bassline do |v=1, c=72|
  bass :a1, v, c; sleep 2
  bass :a1, v*0.5, c-8; sleep 1
  bass :e2, v*0.8, c; sleep 1
end
```

Only 3 notes per bar — space is the point.

## Second Arpeggio (Variation)

```
define :arp2 do |v=1|
  use_synth :pulse
  notes = [:a4, :g4, :e4, :c4, :e4, :g4, :a4, :c5]
  notes.each do |n|
    play n, amp: 0.2*v, attack: 0.01, decay: 0.1,
      release: 0.15, cutoff: 95, pulse_width: 0.4
    sleep 0.5
  end
end
```

Descending then ascending — complements arp1.

# Arrangement

INTRO (32) → BUILD (32) → MAIN A (48) → AMBIENT (24) → MAIN B (40) → OUTRO (32)

## The “Ambient” Section

Instead of a traditional break, Chrome Cathedral has an ambient interlude:

```
# AMBIENT: 6 bars - pure atmosphere
in_thread do
  with_fx :reverb, room: 0.95, mix: 0.7 do
    pad [:a2, :e3, :a3], 0.6
    sleep 8
    pad [:a2, :c3, :e3], 0.5
    sleep 8
    pad [:a2, :e3, :g3], 0.55
  end
end

in_thread do
  with_fx :reverb, room: 0.9, mix: 0.6 do
    with_fx :echo, phase: 1.5, decay: 8, mix: 0.5 do
      sleep 4
      whisper :a5, 0.4
      sleep 6
      whisper :e5, 0.35
      sleep 6
      whisper :c5, 0.4
    end
  end
end

sleep 24
```

**No drums, no bass** — just pads and whispers floating in reverb.

## Long Outro

```
# OUTRO: 8 bars - fading into space
in_thread do
  4.times do |i|
    drums (0.7-i*0.12), (0.5-i*0.1), (0.55-i*0.1)
  end
  4.times { |i| hat (0.35-i*0.08); sleep 0.5 }
end

in_thread do
  with_fx :reverb, room: 0.95, mix: 0.65 do
    with_fx :echo, phase: 1, decay: 10, mix: 0.55 do
      2.times { arp1 0.4 }
      arp2 0.3
    end
  end
end

sleep 32
```

The decay: 10 on the echo means notes ring out long after the track ends.

## Key Techniques

### 1. Space Through Subtraction

Less is more. Compare note density:

Track	Notes per bar (approx)
Nerve Damage	25+
Chrome Cathedral	12-15

### 2. Reverb as Instrument

Reverb isn't just polish — it's compositional:

```

# DRY: Sounds thin, mechanical
play :a4

# WET: Sounds like a cathedral
with_fx :reverb, room: 0.9, mix: 0.6 do
  play :a4
end

```

### 3. Syncopated Drums for Float

Straight four-on-floor is driving. Syncopation floats:

```

# Driving (Tracks 1-2)
4.times { kick; sleep 1 }

# Floating (Track 3)
kick; sleep 1.5
kick; sleep 0.5
kick; sleep 1
kick; sleep 1

```

### 4. Pulse Width for Character

The `pulse` synth's `pulse_width` parameter changes its tone:

```

pulse_width: 0.5    # Square wave, hollow
pulse_width: 0.35   # Thinner, more nasal
pulse_width: 0.2    # Very thin, reedy

```

Chrome Cathedral uses 0.35-0.4 for a slightly hollow, retro sound.

## The Cyberpunk Aesthetic

“Chrome Cathedral” evokes:

- Chrome — metallic, reflective (the pulse arpeggios)

- Cathedral — vast, reverberant (the heavy effects)
- Cyberpunk — neon-lit darkness, melancholy technology

The title guides the sound design. Name your tracks evocatively.

## Hacker Challenges

- 1. Drown It in Reverb:** Set reverb room: 1.0 and mix: 0.8 . At what point does atmosphere become mud? Find the threshold.
- 2. Generative Arpeggio:** Replace the fixed note array with:

```
8.times do
  play scale(:a3, :minor_pentatonic).choose, amp: 0.22,
  release: 0.12
  sleep 0.5
end
```

Let the arpeggio write itself. Does it still feel intentional?

- 3. Add a Pad Chord Progression:** The track uses single pad chords. Try a simple progression:

```
[[:a2,:e3,:a3], [:f2,:c3,:f3], [:g2,:d3,:g3],
[:a2,:e3,:a3]].each do |chord|
  pad chord, 0.5
  sleep 8
end
```

Does movement help or hurt the hypnotic quality?

- 4. Speed It Up:** Change to 110 BPM. Does Chrome Cathedral still feel atmospheric, or does it become a different track entirely?
- 5. Kill the Reverb in the Drop:** After the ambient section, cut reverb to room: 0.3 . Does the contrast make the return feel more powerful?

## Full Code

The complete track code is available in `03_chrome_cathedral.rb`.

---

Next: [Track 04: Skull Fracture](#) — maximum aggression.

# 04: Skull Fracture

**BPM:** 108 | **Key:** F Minor | **Duration:** ~2:45

Maximum aggression. Skull Fracture is the album's fastest, hardest track — pure adrenaline and violence.

## The Vision

After the atmospheric break of Chrome Cathedral, Skull Fracture slams the door:

1. **Maximum energy** — Fastest BPM, hardest hits
2. **Signature hook sound** — The “skull” alarm
3. **Short and brutal** — No fat, all impact
4. **First peak** — This is the album's first climax

## What Makes It Unique

### Fastest BPM

108 BPM is the album's maximum:

Track	BPM	Delta from previous
Chrome Cathedral	98	—
<b>Skull Fracture</b>	<b>108</b>	<b>+10</b>

The 10 BPM jump is jarring — intentionally.

## The “Skull” Hook

A distorted alarm sound that defines the track:

```
define :skull do |v=1|
    use_synth :mod_saw
    play :f4, amp: 0.4*v, attack: 0, decay: 0.06,
          sustain: 0.02, release: 0.04, mod_phase: 0.08,
          mod_range: 8, cutoff: 115
    sleep 0.125
    play :f4, amp: 0.35*v, attack: 0, decay: 0.05,
          sustain: 0.02, release: 0.04, mod_phase: 0.1,
          mod_range: 7, cutoff: 112
    sleep 0.125
    play :ab4, amp: 0.38*v, attack: 0, decay: 0.06,
          sustain: 0.02, release: 0.05, mod_phase: 0.08,
          mod_range: 8, cutoff: 118
    sleep 0.25
end
```

### Key characteristics:

- Very fast notes (0.125 beat = 32nd notes)
- High `mod_range` (8) — extreme modulation
- Fast `mod_phase` (0.08) — rapid wobble
- High `cutoff` (115+) — bright, cutting

**The effect:** An alarm-like sound that cuts through everything.

## Triple-Layer Bass

Maximum weight:

```

define :bass do |n, v=1, c=82|
  use_synth :dsaw
  play n, amp: 0.6*v, attack: 0, decay: 0.15,
         sustain: 0.08, release: 0.12, cutoff: c, detune: 0.22
  use_synth :tb303
  play n, amp: 0.45*v, attack: 0, decay: 0.18,
         sustain: 0.05, release: 0.1, cutoff: c-8, res: 0.35, wave: 1
  use_synth :sine
  play n-12, amp: 1.25*v, attack: 0.01, sustain: 0.22, release: 0.18
end

```

### Three layers:

1. dsaw — Width and aggression (high detune: 0.22)
2. tb303 (square wave) — Grit and punch
3. sine — Sub weight (louder than other tracks: 1.25)

### Punishing Drums

```

define :drums_brutal do |k=1, s=1, h=1|
  in_thread do
    kick k; sleep 0.5
    kick k*0.7; sleep 0.5
    kick k*0.9; sleep 0.5
    kick k*0.6; sleep 0.5
    kick k; sleep 0.5
    kick k*0.75; sleep 0.5
    kick k*0.85; sleep 0.5
    kick k; sleep 0.5
  end
  in_thread do
    sleep 1; snare s*1.1; sleep 1; snare s; sleep 2
  end
  in_thread do
    16.times { hat h; sleep 0.25 }
  end
  sleep 4
end

```

**8 kicks per bar** with heavy velocity variation — relentless but groovy.

# Sound Design

## Kick (Heavier)

```
define :kick do |v=1|
  sample :bd_tek, amp: 2.5*v, rate: 0.85
  sample :bd_zum, amp: 0.7*v, rate: 1.1
end
```

### Differences from earlier tracks:

- Lower rate (0.85 vs 0.9) — deeper pitch
- bd\_zum instead of bd\_boom — more attack
- Higher amp (2.5 vs 2.2) — louder

## Snare (Cracking)

```
define :snare do |v=1|
  sample :sn_dub, amp: v*1.1, rate: 0.75
  sample :drum_snare_hard, amp: 0.5*v, rate: 0.85
end
```

Lower rates = deeper, more impactful.

## Power Stab

Chord stabs for emphasis:

```
define :stab do |v=1|
  use_synth :dsaw
  play [:f2, :c3, :f3], amp: 0.5*v, attack: 0,
    decay: 0.15, release: 0.1, cutoff: 90, detune: 0.2
end
```

F minor power chord — dark and heavy.

# Patterns

## Bassline (Aggressive)

```
define :bassline do |v=1, c=82|
  bass :f2, v, c; sleep 0.5
  bass :f2, v*0.6, c-10; sleep 0.25
  bass :f2, v*0.7, c-5; sleep 0.25
  bass :ab2, v*0.9, c; sleep 0.5
  bass :f2, v*0.8, c; sleep 0.5
  bass :c2, v, c+5; sleep 0.5
  bass :db2, v*0.85, c; sleep 0.5
  bass :f2, v, c; sleep 1
end
```

**F Minor movement:** F (root) → Ab (minor 3rd) → C (5th) → Db (b6 — dark!)

## Skull Pattern

```
define :skull_pattern do |v=1|
  skull v; sleep 0.5
  skull v*0.8; sleep 0.5
  sleep 0.5
  skull v*0.9; sleep 0.5
  sleep 1
  skull v; sleep 1
end
```

The hook appears, disappears, reappears — creating anticipation.

# Arrangement

INTRO (24) → BUILDUP (24) → MAIN (40) → FAKE DROP (8) → PEAK (40) → OUTRO (16)

## Short and Brutal

At ~2:45, Skull Fracture is the album's shortest track. No fat:

- Intro: 6 bars (not 8)
- No ambient section
- Short outro

## The Fake Drop

A technique for extra impact:

```
# FAKE DROP: 2 bars - silence before real peak
in_thread do
  4.times { hat 0.3; sleep 0.5 }
  sleep 4
  hit 1.5
end

sleep 8
```

After the main section, everything stops except quiet hi-hats. Listeners expect the outro — instead, they get the biggest hit of the track.

## The Peak

```
# PEAK: 10 bars - MAXIMUM
in_thread do
  10.times { drums_brutal 1.2, 1.15, 0.85 }
end

in_thread do
  5.times { bassline 1.15, 88; bassline 1.15, 92 }
end

in_thread do
  with_fx :reverb, room: 0.5, mix: 0.25 do
    10.times { skull_pattern 1 }
  end
end

in_thread do
  sleep 8
  5.times { stab 0.9; sleep 4; stab 0.85; sleep 4 }
end

sleep 40
```

Everything at maximum:

- Drums at 1.2 (20% louder than normal)
- Bass cutoff at 88-92 (brightest in album)
- All elements playing

## Key Techniques

### 1. The Hook Sound

Every great track has a signature. Skull Fracture's is the alarm:

```
use_synth :mod_saw
play :f4, mod_phase: 0.08, mod_range: 8, cutoff: 115
```

Design sounds that are:

- Instantly recognizable
- Different from other tracks
- Tied to the track's concept ("skull" = sharp, alarming)

## 2. Fake Drops

Subvert expectations:

Expected: MAIN → OUTRO

Actual: MAIN → SILENCE → BIGGEST HIT → PEAK → OUTRO

The silence makes the peak feel even bigger.

## 3. Note Density = Energy

Section	Notes per bar
Intro	~15
Main	~30
Peak	~40

More notes = more energy (but don't sacrifice clarity).

## 4. F Minor Darkness

F Minor is one of the darkest keys. The Db (b6) note is particularly unsettling:

```
# Very dark
bass :db2  # b6 in F minor

# Less dark
bass :c2   # 5th in F minor
```

Use b6 for maximum darkness.

# The Skull Concept

The title guides everything:

- **Sound:** Alarm-like (warning of danger)
- **Rhythm:** Brutal, skull-cracking impacts
- **Feel:** Relentless assault

Name → Sound → Success.

## Hacker Challenges

1. **Design Your Own Hook:** The “skull” alarm is `mod_saw` with fast modulation. Create a completely different signature sound using `:mod_tri` or `:mod_pulse`. What makes a hook memorable?
2. **Extend the Fake Drop:** Instead of 2 bars of silence, try 4. Does more tension = more impact, or does it lose momentum?
3. **Pitch the Hook Down:** Transpose the skull sound down an octave ( `:f3` instead of `:f4` ). Does it become more menacing or just muddy?
4. **Randomize the Stabs:** Instead of fixed timing:

```
4.times do
  stab (0.7 + rand(0.3))
  sleep [1, 1.5, 2].choose
end
```

Does unpredictability add energy or chaos?

5. **Layer a Scream:** Add a distorted, high-pitched synth layer during the peak. How much is too much? When does “aggressive” become “annoying”?

## Full Code

The complete track code is available in `04_skull_fracture.rb`.

---

Next: [Track 05: Midnight Protocol — melodic relief.](#)

# 05: Midnight Protocol

**BPM:** 102 | **Key:** C Minor | **Duration:** ~3:20

Synthwave triumph. After Skull Fracture's violence, Midnight Protocol provides melodic relief with uplifting arpeggios and triumphant leads.

## The Vision

Track 5 is the album's emotional pivot:

1. **Melodic focus** — Strong, memorable melodies
2. **Synthwave influence** — 80s-inspired arpeggios
3. **Triumphant feel** — Dark but not depressing
4. **Groove over brutality** — Still dark, but you can breathe

## What Makes It Unique

### Synthwave Arpeggios

Carpenter Brut-inspired patterns:

```
define :arp1 do |v=1|
  use_synth :pulse
  notes = [:c4, :eb4, :g4, :c5, :g4, :eb4, :c4, :g3]
  notes.each do |n|
    play n, amp: 0.25*v, attack: 0.01, decay: 0.12,
      release: 0.12, cutoff: 102, pulse_width: 0.3
    sleep 0.5
  end
end
```

**C minor arpeggio** — dark but with upward motion that feels hopeful.

## Triumphant Lead Melody

The lead is more melodic than any other track:

```
define :lead do |n, dur=0.5, v=1|
  use_synth :prophet
  play n, amp: 0.45*v, attack: 0.06, decay: dur*0.3,
    sustain: dur*0.45, release: dur*0.55, cutoff: 92, res: 0.12
  use_synth :saw
  play n+12, amp: 0.12*v, attack: 0.08,
    sustain: dur*0.3, release: dur*0.4, cutoff: 78
end
```

### Key choices:

- Longer attack (0.06) — more singing quality
- Octave shimmer layer — adds brightness/hope
- Higher cutoff (92) — brighter than darker tracks

## The Melodies

Four phrases that tell a story:

```

define :mel1 do |v=1| # Rising hope
  lead :c4, 0.5, v; sleep 0.5
  lead :eb4, 0.5, v*0.95; sleep 0.5
  lead :g4, 1, v; sleep 1
  lead :c5, 1.5, v; sleep 2
end

define :mel2 do |v=1| # Gentle descent
  lead :c5, 0.75, v; sleep 0.75
  lead :bb4, 0.5, v*0.9; sleep 0.5
  lead :g4, 0.75, v*0.95; sleep 0.75
  lead :eb4, 0.5, v*0.85; sleep 0.5
  lead :c4, 1.5, v; sleep 1.5
end

define :mel3 do |v=1| # Emotional peak
  lead :g4, 0.5, v; sleep 0.5
  lead :ab4, 0.75, v; sleep 0.75 # Ab = emotional tension
  lead :g4, 0.5, v*0.9; sleep 0.5
  lead :f4, 0.75, v*0.95; sleep 0.75
  lead :eb4, 1.5, v; sleep 1.5
end

define :mel4 do |v=1| # Resolution
  lead :eb4, 0.5, v; sleep 0.5
  lead :f4, 0.5, v*0.9; sleep 0.5
  lead :g4, 1, v; sleep 1
  lead :c4, 2, v; sleep 2
end

```

**The arc:** Rise → Descent → Tension (Ab) → Resolution to root (C)

## Groovy, Not Brutal Drums

Back to four-on-the-floor, but with swing:

```

define :drums do |k=1, s=1, h=1|
  in_thread do
    kick k; sleep 1
    kick k*0.75; sleep 0.75
    kick k*0.85; sleep 0.25
    kick k; sleep 1
    kick k*0.8; sleep 1
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s*0.9; sleep 2
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  sleep 4
end

```

**Ghost kick** at beat 2.75 creates groove without aggression.

## Sound Design

### Warm Bass

Prophet-based for warmth:

```

define :bass do |n, v=1, c=78|
  use_synth :prophet
  play n, amp: 0.6*v, attack: 0.02, decay: 0.22,
         sustain: 0.15, release: 0.2, cutoff: c, res: 0.2
  use_synth :sine
  play n-12, amp: 1.05*v, attack: 0.01, sustain: 0.28, release: 0.2
end

```

No TB303 or dsaw — pure analog warmth.

## Pad Layer

Sustained chords for fullness:

```
define :pad do |notes, v=1|
  use_synth :prophet
  play notes, amp: 0.25*v, attack: 1.5, decay: 0.5,
         sustain: 2, release: 2.5, cutoff: 72
end
```

Plays full chords: pad [:c3, :eb3, :g3]

## Bright Hi-Hats

```
define :hat do |v=1|
  sample :drum_cymbal_closed, amp: 0.28*v, rate: 2.0, release: 0.06
end
```

Slightly lower rate (2.0 vs 2.2+) — less metallic, more musical.

## Patterns

### Bassline (Melodic)

```
define :bassline do |v=1, c=78|
  bass :c2, v, c; sleep 1.5
  bass :c2, v*0.6, c-8; sleep 0.5
  bass :eb2, v*0.85, c; sleep 1
  bass :f2, v*0.9, c; sleep 0.5
  bass :g2, v*0.8, c+5; sleep 0.5
end
```

**The movement:** C → Eb → F → G — ascending bassline = building energy.

## Second Arpeggio

```
define :arp2 do |v=1|
  use_synth :pulse
  notes = [:g4, :c5, :eb5, :g5, :eb5, :c5, :g4, :c4]
  notes.each do |n|
    play n, amp: 0.22*v, attack: 0.01, decay: 0.1,
           release: 0.1, cutoff: 98, pulse_width: 0.35
    sleep 0.5
  end
end
```

Higher register — plays call-and-response with arp1.

## Arrangement

INTRO (32) → BUILD (32) → MAIN A (48) → BREAK (16) → MAIN B (48) →  
OUTRO (32)

## The Build: Arp Entry

```
# BUILD: 8 bars
in_thread do
  8.times { drums 0.85, 0.7, 0.6 }
end

in_thread do
  8.times { bassline 0.75, 70 }
end

in_thread do
  with_fx :reverb, room: 0.6, mix: 0.4 do
    sleep 16
    2.times { arp1 0.5; arp2 0.45 }
  end
end

in_thread do
  with_fx :reverb, room: 0.7, mix: 0.5 do
    sleep 24
    pad [:c3, :eb3, :g3], 0.4
  end
end

sleep 32
hit 1
```

Elements enter in layers: drums → bass → arps → pad.

## The Break: Melody Focus

```
# BREAK: 4 bars - spotlight on melody
in_thread do
  8.times { hat 0.35; sleep 0.5 }; sleep 8
end

in_thread do
  with_fx :reverb, room: 0.85, mix: 0.55 do
    with_fx :echo, phase: 0.75, decay: 5, mix: 0.45 do
      mel3 0.6; mel4 0.55
    end
  end
end

in_thread do
  with_fx :reverb, room: 0.9, mix: 0.6 do
    pad [:ab2, :c3, :eb3], 0.35 # Ab for tension
  end
end

sleep 16
hit 1.1
```

**Ab chord** under the melody creates tension before resolution in Main B.

## Key Techniques

### 1. Synthwave Arp Formula

Classic synthwave arpeggios follow this pattern:

```
# Root position up and down
[:c4, :eb4, :g4, :c5, :g4, :eb4, :c4, :g3]
# 1     b3    5     8     5     b3    1     5(low)
```

Variations:

```
# Extended range  
[:c4, :eb4, :g4, :c5, :eb5, :c5, :g4, :eb4]  
  
# With passing tones  
[:c4, :d4, :eb4, :g4, :c5, :g4, :eb4, :d4]
```

## 2. Triumphant ≠ Happy

Midnight Protocol is triumphant but still dark:

### Dark elements:

- Minor key (C minor)
- Dark pad tones
- Sub-bass weight

### Triumphant elements:

- Ascending melodies
- Bright arp tones
- Longer, singing lead notes

The balance is key.

## 3. Ab in C Minor

The Ab note (b6) adds emotional weight:

```
# Standard C minor: C D Eb F G Ab Bb  
lead :ab4 # Creates yearning, emotional tension
```

Use it sparingly for maximum impact (mel3 uses it at the peak).

## 4. Groove Through Ghost Notes

The ghost kick at beat 2.75 creates swing:

```
kick k; sleep 1
kick k*0.75; sleep 0.75    # Ghost: softer, slightly early
kick k*0.85; sleep 0.25    # Lands on beat 3
```

This makes the track “breathe” instead of hammer.

## The Midnight Aesthetic

“Midnight Protocol” evokes:

- **Midnight** — The darkest hour, but also possibility
- **Protocol** — Digital, systematic, precise (the arpeggios)
- **Synthwave** — Neon-lit nights, driving through darkness

A moment of beauty in a brutal album.

## Hacker Challenges

1. **Major Key Experiment:** Change the arpeggio from C minor to C major ( `:c4, :e4, :g4, :c5...` ). How dramatically does the mood shift? Can dark clubbing work in major keys?
2. **Double-Time Arp:** Run the arpeggio at 16th notes ( `sleep 0.25` ) instead of 8th notes. Does it add energy or become too busy?
3. **Strip the Shimmer:** Remove the octave-up `:saw` layer from the lead. What’s lost? Is the shimmer essential or decorative?
4. **Write a Counter-Melody:** Create `mel5` that plays *against* the main melodies — different rhythm, complementary notes. Can two melodies coexist without fighting?
5. **Synthwave the Drums:** Replace the industrial kick layers with a cleaner, punchier 80s-style kick:

```
sample :bd_haus, amp: 2, rate: 1.0
```

Does the track become more or less “Midnight Protocol”?

## Full Code

The complete track code is available in `05_midnight_protocol.rb`.

---

Next: [Track 06: Void Walker](#) — dark power at slow tempo.

# 06: Void Walker

**BPM:** 95 | **Key:** B Minor | **Duration:** ~3:30

Dark power at slow tempo. Void Walker proves that slow doesn't mean weak — each hit lands with devastating weight.

## The Vision

Track 6 builds tension before the album's climax:

1. **Slowest BPM** — 95 creates hypnotic menace
2. **Maximum weight** — Each element hits harder
3. **Space = power** — Less notes, more impact
4. **Building dread** — Preparing for Core Meltdown

## What Makes It Unique

### The Slowest Tempo

At 95 BPM, Void Walker is 13 BPM slower than Skull Fracture:

Track	BPM	Feel
Skull Fracture	108	Frantic aggression
Midnight Protocol	102	Driving groove
<b>Void Walker</b>	<b>95</b>	<b>Menacing weight</b>

Slow tempo = more time between hits = each hit feels heavier.

## Triple-Layer Dark Bass

The deepest, heaviest bass in the album:

```
define :bass do |n, v=1, c=72|
  use_synth :prophet
  play n, amp: 0.6*v, attack: 0.02, decay: 0.28,
    sustain: 0.18, release: 0.22, cutoff: c, res: 0.22
  use_synth :dsaw
  play n, amp: 0.4*v, attack: 0.01, decay: 0.22,
    release: 0.18, cutoff: c-12, detune: 0.2
  use_synth :sine
  play n-12, amp: 1.2*v, attack: 0.02, sustain: 0.32, release: 0.25
end
```

### Key choices:

- Prophet for warmth + dsaw for grit
- Lower cutoff (72) — darker than other tracks
- Louder sine sub (1.2) — physical weight
- Longer decay/release — notes bloom and sustain

## Dark Ambience Stabs

Power chord stabs using `dark_ambience`:

```
define :stab do |notes, v=1|
  use_synth :dark_ambience
  play notes, amp: 0.4*v, attack: 0.04, decay: 0.35,
    sustain: 0.12, release: 0.45
end

define :stab_pattern do |v=1|
  stab [:b2, :fs3, :b3], v; sleep 2
  stab [:b2, :fs3, :b3], v*0.5; sleep 0.5
  stab [:d3, :a3, :d4], v*0.75; sleep 1.5
end
```

**B minor power chords** — dark and powerful.

## Pulse Arpeggios

Hypnotic, minimal arps:

```
define :arp do |v=1|
  use_synth :pulse
  notes = [:b3, :d4, :fs4, :b4, :fs4, :d4, :b3, :fs3]
  notes.each do |n|
    play n, amp: 0.22*v, attack: 0.01, decay: 0.15,
           release: 0.12, cutoff: 95, pulse_width: 0.35
    sleep 0.5
  end
end
```

B minor arpeggio — simple but hypnotic at slow tempo.

## Whisper Textures

High, ethereal layer:

```
define :whisper do |n, v=1|
  use_synth :hollow
  play n, amp: 0.18*v, attack: 0.4, decay: 0.3,
        sustain: 0.5, release: 1.2, cutoff: 82
end
```

Adds ghostly presence without taking focus.

## Sound Design

### Heavy Kick

```
define :kick do |v=1|
  sample :bd_tek, amp: 2.4*v, rate: 0.85
  sample :bd_zum, amp: 0.6*v, rate: 1.0, cutoff: 65
end
```

**Lowest rate** (0.85) for the deepest pitch. The slow tempo lets each kick breathe.

## Snare (Impactful)

```
define :snare do |v=1|
    sample :sn_dub, amp: v*1.05, rate: 0.78
end
```

Lower rate = deeper, more impactful.

## Drums Pattern (Half-Time Feel)

```
define :drums do |k=1, s=1, h=1|
    in_thread do
        kick k; sleep 2
        kick k*0.65; sleep 0.5
        kick k*0.8; sleep 1.5
    end
    in_thread do
        sleep 2; snare s; sleep 2
    end
    in_thread do
        8.times { hat h; sleep 0.5 }
    end
    sleep 4
end
```

**Key difference:** Kick on beat 1 and 3.5, snare on beat 3 only.

This creates a **half-time feel** — even slower than the tempo suggests.

# Patterns

## Bassline (Sparse)

```
define :bassline do |v=1, c=72|
  bass :b1, v, c; sleep 2
  bass :b1, v*0.55, c-10; sleep 0.5
  bass :d2, v*0.8, c; sleep 1
  bass :b1, v*0.7, c; sleep 0.5
end
```

Only 4 notes per bar — space is power.

## Lead (Sparse, Emotional)

```
define :lead do |n, dur=1, v=1|
  use_synth :prophet
  play n, amp: 0.4*v, attack: 0.1, decay: dur*0.25,
    sustain: dur*0.5, release: dur*0.6, cutoff: 85, res: 0.15
end

define :melody do |v=1|
  lead :b4, 1.5, v; sleep 1.5
  lead :d5, 1, v*0.9; sleep 1
  lead :cs5, 1.5, v; sleep 1.5 # C# = major 2nd, tension
end
```

**Long notes** — at 95 BPM, a 1.5-beat note lasts nearly a full second.

# Arrangement

INTRO (32) → BUILD (32) → MAIN A (48) → DARK (24) → MAIN B (48) → OUTRO (28)

## The “Dark” Section

Instead of a break, a descent into darkness:

```
# DARK: 6 bars - descent
in_thread do
  12.times { hat 0.3; sleep 0.5 }
  sleep 12
end

in_thread do
  with_fx :reverb, room: 0.95, mix: 0.65 do
    with_fx :echo, phase: 1.5, decay: 8, mix: 0.55 do
      whisper :b5, 0.35; sleep 6
      whisper :fs5, 0.3; sleep 5
      whisper :d5, 0.35; sleep 5
    end
  end
end

in_thread do
  with_fx :reverb, room: 0.9, mix: 0.6 do
    sleep 8
    stab [:b1, :fs2, :b2], 0.4 # Deep stab
    sleep 8
    stab [:d2, :a2, :d3], 0.35
  end
end

sleep 24
hit 1.3
```

**No bass, no kick** — just whispers and distant stabs floating in reverb.

## Building Weight

Energy through volume, not speed:

Section	Kick amp	Bass amp	Stab amp
Intro	0.7	0.5	—
Build	0.85	0.7	0.4

Section	Kick amp	Bass amp	Stab amp
Main A	1.0	0.9	0.6
Dark	—	—	0.35
Main B	1.1	1.0	0.75
Outro	fade	fade	fade

## Long Outro

```
# OUTRO: 7 bars - slow fade
in_thread do
  7.times do |i|
    drums (0.9-i*0.1), 0, (0.55-i*0.06)
  end
end

in_thread do
  with_fx :reverb, room: 0.95, mix: 0.65 do
    with_fx :echo, phase: 1.25, decay: 10, mix: 0.55 do
      2.times { arp 0.35 }
      arp 0.25
    end
  end
end

sleep 28
```

**10-second echo decay** — notes ring out long after playing stops.

## Key Techniques

### 1. Half-Time Feel

Even at 95 BPM, half-time makes it feel like 47.5:

```

# Normal feel (kick every beat)
kick; sleep 1; kick; sleep 1; kick; sleep 1; kick; sleep 1

# Half-time (kick beats 1 and 3.5)
kick; sleep 2; kick; sleep 0.5; sleep 1.5

```

The sparse kick pattern creates massive weight.

## 2. Space = Power

Compare note density:

Track	Bass notes/bar
Nerve Damage	8
Skull Fracture	8
<b>Void Walker</b>	<b>4</b>

Fewer notes = more impact per note.

## 3. Lower Everything

Void Walker uses lower rates/pitches than other tracks:

Element	Other tracks	Void Walker
Kick rate	0.9	0.85
Snare rate	0.85	0.78
Bass cutoff	78-82	72

Everything is deeper, darker.

## 4. B Minor Character

B Minor has a unique darkness:

```

# B natural minor: B C# D E F# G A
# The F# (5th) is strong and stable
# The G (b6) adds darkness

bass :b1; sleep 2      # Root - grounded
bass :fs2; sleep 1     # Fifth - powerful
bass :g2; sleep 1      # b6 - dark tension

```

## 5. Dynamic Range

Void Walker has the album's widest dynamic range:

- Quietest moments: whispers at 0.18 amp
- Loudest moments: kicks at 2.4 amp

**Contrast creates drama.**

## The Void Walker Concept

The title evokes:

- **Void** — Emptiness, space, darkness
- **Walker** — Slow, deliberate movement
- **Together** — Something moving through darkness, each step deliberate

The slow tempo and sparse arrangement embody walking through the void.

## Hacker Challenges

1. **Go Slower:** Drop to 85 BPM. At what tempo does “menacing” become “boring”? Where’s the floor?
2. **Remove All Reverb:** Strip the atmospheric effects entirely. Does the track still have power, or was the space doing all the work?

**3. Add a Pulse:** Introduce a quiet, steady 8th-note element (maybe a filtered arp or just a soft click). Does it add drive or fight the half-time feel?

**4. Drone Layer:** Add a sustained low note that never changes:

```
use_synth :dark_ambience  
play :b1, amp: 0.3, attack: 4, sustain: 100, release: 4
```

Does constant bass add weight or become oppressive?

**5. Double the Tempo Feel:** Keep BPM at 95 but change drums to 4-on-floor. Same tempo, completely different energy. Which is more “Void Walker”?

## Full Code

The complete track code is available in `06_void_walker.rb`.

---

Next: [Track 07: Core Meltdown](#) — the album climax.

# 07: Core Meltdown

**BPM:** 106 | **Key:** G Minor | **Duration:** ~3:30

The album's climax. Core Meltdown combines aggressive bass, beautiful ethereal melodies, and a massive drop to create the emotional peak.

## The Vision

As Track 7, Core Meltdown needs to:

1. **Be the climax** — Maximum combined energy
2. **Contrast intensity with beauty** — Dark bass + ethereal leads
3. **Feature a massive drop** — Tension → explosion
4. **Feel distinct** — Not just “louder Track 1”

## What Makes It Unique

### Beautiful Dark Melodies

Unlike the aggressive leads in other tracks, Core Meltdown features **ethereal melodies** with heavy effects:

```
define :lead do |n, dur=0.5, v=1|
  use_synth :prophet
  play n, amp: 0.4*v, attack: 0.08, decay: dur*0.25,
         sustain: dur*0.5, release: dur*0.7, cutoff: 85, res: 0.15
  use_synth :saw
  play n+12, amp: 0.1*v, attack: 0.1, sustain: dur*0.35,
         release: dur*0.5, cutoff: 72
end
```

## Key differences:

- Longer attack (0.08) — softer entry
- Lower cutoff (85) — warmer tone
- Octave shimmer layer — adds air

## Heavy Effects on Melodies

The melodies are drenched in reverb and echo:

```
# TENSION section - maximum ethereal
with_fx :reverb, room: 1.0, mix: 0.75 do
    with_fx :echo, phase: 1.5, decay: 8, mix: 0.6 do
        mel3 0.55; mel4 0.5; mel1 0.45
    end
end
```

Compare to Main section:

```
# MAIN section - tighter
with_fx :reverb, room: 0.65, mix: 0.4 do
    with_fx :echo, phase: 0.5, decay: 3.5, mix: 0.35 do
        3.times { mel1 0.75; mel3 0.7; mel2 0.75; mel4 0.7 }
    end
end
```

## The Melodies Themselves

Four melodic phrases that flow and resolve:

```

define :mel1 do |v=1|
  lead :g4, 1, v; sleep 1
  lead :bb4, 0.75, v*0.95; sleep 0.75
  lead :c5, 0.75, v; sleep 0.75
  lead :d5, 1.5, v; sleep 1.5
end

define :mel2 do |v=1|
  lead :d5, 0.75, v; sleep 0.75
  lead :c5, 0.5, v*0.9; sleep 0.5
  lead :bb4, 0.75, v*0.95; sleep 0.75
  lead :a4, 0.5, v*0.85; sleep 0.5
  lead :g4, 1.5, v; sleep 1.5
end

```

**mel1** rises (G→Bb→C→D) — building hope

**mel2** falls (D→C→Bb→A→G) — resolving

## Dark Synth Bass

The bass uses Prophet + Sine layering for warmth with weight:

```

define :bass do |n, v=1, c=75|
  use_synth :prophet
  play n, amp: 0.65*v, attack: 0.02, decay: 0.25,
    sustain: 0.15, release: 0.2, cutoff: c, res: 0.25
  use_synth :dsaw
  play n, amp: 0.4*v, attack: 0.01, decay: 0.2,
    release: 0.15, cutoff: c-10, detune: 0.18
  use_synth :sine
  play n-12, amp: 1.15*v, attack: 0.01, sustain: 0.28, release: 0.2
end

```

Three layers:

1. Prophet — warm character
2. Dsaw — grit and width
3. Sine — sub foundation

## Dark Ambience Stabs

Chord stabs using `dark_ambience` for texture:

```
define :dark_stab do |notes, v=1|
  use_synth :dark_ambience
  play notes, amp: 0.35*v, attack: 0.05, decay: 0.3,
        sustain: 0.1, release: 0.4
end

define :stab_pat do |v=1|
  dark_stab [:g2,:d3,:g3], v; sleep 1.5
  dark_stab [:g2,:d3,:g3], v*0.5; sleep 0.5
  dark_stab [:bb2,:f3,:bb3], v*0.8; sleep 1
  dark_stab [:g2,:d3,:g3], v; sleep 1
end
```

## Arrangement

INTRO (32) → BUILD (32) → MAIN A (48) → TENSION (24) → DROP (56) → OUTRO (24)

### The Tension Section

The key to the massive drop:

```

# TENSION: 6 bars - stripped, building
in_thread do
  12.times { hat 0.35; sleep 0.5 }; sleep 12
end

in_thread do
  with_fx :reverb, room: 0.9, mix: 0.6 do
    with_fx :echo, phase: 1, decay: 5, mix: 0.5 do
      mel3 0.5; mel4 0.45; mel1 0.4
    end
  end
end

in_thread do
  sleep 12; riser 12, 0.9 # Noise riser
end

sleep 24
hit 1.7 # MASSIVE impact

```

### Elements:

1. Only hi-hats (no kick or snare)
2. Ethereal, reverbed melodies
3. Noise riser building for 12 beats
4. Hit at 1.7 amplitude — the biggest in the album

## The Massive Drop

After the tension, everything slams back:

```

# DROP: 14 bars - MAXIMUM
in_thread do
  14.times { drums_intense 1.25, 1.1, 0.85 }
end

in_thread do
  7.times { bass2 1.2, 85; bass1 1.2, 88 }
end

in_thread do
  7.times { stab_pat 0.9 }
end

in_thread do
  with_fx :reverb, room: 0.7, mix: 0.4 do
    with_fx :echo, phase: 0.75, decay: 4, mix: 0.3 do
      7.times { mell 0.9; mel2 0.85 }
    end
  end
end

sleep 56

```

### What makes it hit:

- Drums at 1.25 (louder than anywhere else)
- Bass cutoff at 85-88 (brightest)
- All elements playing together
- Contrast with the stripped tension section

## Key Techniques

### 1. Contrast Creates Impact

The TENSION section removes:

- Kick drum
- Snare
- Bass

So when they return in the DROP, the contrast is extreme.

## 2. Effects Tell the Story

Section	Reverb Room	Echo Decay	Feel
Build	0.65	3.5	Grounded
Main	0.6	3	Present
Tension	0.9	5	Floating
Drop	0.7	4	Powerful

## 3. The Riser

```
define :riser do |dur, v=1|
  use_synth :noise
  play :g2, amp: 0.4*v, attack: dur*0.9, release: dur*0.1,
    cutoff: 55, cutoff_slide: dur
end
```

A noise sweep that builds tension before the drop. The `cutoff_slide` makes it rise in frequency.

## 4. Hit Scaling

Throughout the album, hits increase:

Track	Transition Hit
1-2	1.0
3-4	1.1-1.2
5-6	1.2-1.3
7	<b>1.7</b>
8	0.9-1.0

Core Meltdown's hit is the album's loudest — marking it as the climax.

## The Balance

Core Meltdown succeeds because it balances:

- **Aggression** (bass, drums, stabs) with **Beauty** (ethereal melodies)
- **Tension** (stripped sections) with **Release** (massive drop)
- **Complexity** (multiple layers) with **Space** (reverb/delay)

This is the album's emotional peak — everything we've built toward.

## Hacker Challenges

1. **Extend the Tension:** Double the tension section length. Does more anticipation = bigger payoff, or does it lose momentum?
2. **Kill the Melody in the Drop:** Remove the ethereal melodies from the drop section entirely — just drums, bass, stabs. Is it more or less powerful?
3. **Louder Hit:** Push the transition hit to `amp: 2.5`. At what point does impact become distortion? Find the edge.
4. **Generative Melody:** Replace one melody with:

```
8.times do
  lead scale(:g4, :minor).choose, 0.5, 0.7
  sleep [0.5, 0.75, 1].choose
end
```

Can generative melodies feel as emotional as composed ones?

5. **Add a Riser Before Every Section:** Put a noise riser before each transition. Does it add energy or become predictable?

## Full Code

The complete track code is available in `07_core_meltdown.rb`.

---

Next: [Track 08: Terminal Velocity](#) — the cinematic finale.

# 08: Terminal Velocity

**BPM:** 100 | **Key:** D Minor | **Duration:** ~3:15

Cinematic finale. Terminal Velocity returns to where we began — 100 BPM, D Minor — but now with the weight of the entire journey behind it.

## The Vision

The final track must:

1. **Return home** — Same key and tempo as Track 1
2. **Feel like resolution** — The journey's end
3. **Cinematic scope** — Bigger, more emotional than Track 1
4. **Fade into darkness** — End the album definitively

## What Makes It Unique

### Circular Structure

Track 1 and Track 8 share:

- **100 BPM**
- **D Minor**

But Track 8 sounds different because:

- We've heard 7 tracks of development
- The melodies are more emotional
- The arrangement is more cinematic
- The ending fades rather than loops

## Noisecream-Inspired Intensity

Driving, intense, but **not happy**:

```
define :bass do |n, v=1, c=78|
    use_synth :tb303
    play n, amp: 0.75*v, attack: 0.01, decay: 0.18,
        sustain: 0.1, release: 0.15, cutoff: c, res: 0.32, wave: 0
    use_synth :dsaw
    play n, amp: 0.45*v, attack: 0.01, decay: 0.15,
        release: 0.12, cutoff: c-8, detune: 0.18
    use_synth :sine
    play n-12, amp: 1.15*v, attack: 0.01, sustain: 0.25, release: 0.2
end
```

**TB303 + dsaw + sine** — the full aggressive stack returns.

## Power Stabs

D minor chord stabs for emphasis:

```
define :stab do |v=1|
    use_synth :dsaw
    play [:d2, :a2, :d3], amp: 0.55*v, attack: 0,
        decay: 0.2, release: 0.15, cutoff: 88, detune: 0.22
end
```

## Tight Melodic Lines

Fast, rhythmic melodies (learned from Noisecream — we wanted “intense”):

```
define :lead do |n, dur=0.25, v=1|
    use_synth :prophet
    play n, amp: 0.4*v, attack: 0.03, decay: dur*0.3,
        sustain: dur*0.4, release: dur*0.5, cutoff: 92
end
```

**Fast attack (0.03) and short default duration (0.25)** for rhythmic precision.

## The Melodies

Three phrases designed to be tight and driving:

```
define :mel1 do |v=1| # Driving, tense
  lead :d4, 0.5, v; sleep 0.5
  lead :f4, 0.5, v*0.95; sleep 0.5
  lead :a4, 0.5, v; sleep 0.5
  lead :g4, 0.5, v*0.9; sleep 0.5
  lead :f4, 0.5, v*0.85; sleep 0.5
  lead :d4, 0.5, v; sleep 0.5
  lead :e4, 0.5, v*0.9; sleep 0.5 # E creates tension
  sleep 0.5
end

define :mel2 do |v=1| # Dark with Bb
  lead :a4, 0.5, v; sleep 0.5
  lead :bb4, 0.5, v*0.95; sleep 0.5 # Bb = darkness
  lead :a4, 0.5, v*0.9; sleep 0.5
  lead :g4, 0.5, v*0.85; sleep 0.5
  lead :f4, 0.5, v*0.9; sleep 0.5
  lead :e4, 0.5, v*0.8; sleep 0.5
  lead :d4, 1, v; sleep 1
end

define :mel3 do |v=1| # Descending resolution
  lead :d5, 0.5, v; sleep 0.5
  lead :c5, 0.5, v*0.95; sleep 0.5
  lead :bb4, 0.5, v*0.9; sleep 0.5
  lead :a4, 0.5, v*0.85; sleep 0.5
  lead :g4, 0.5, v*0.9; sleep 0.5
  lead :a4, 0.5, v*0.8; sleep 0.5
  lead :d4, 1, v; sleep 1
end
```

**All notes on 0.5 beat grid — rhythmically tight, no rubato.**

# Sound Design

## Drums (Driving but Clean)

```
define :drums do |k=1, s=1, h=1|
  in_thread do
    kick k; sleep 0.75
    kick k*0.65; sleep 0.25
    kick k*0.85; sleep 1
    kick k; sleep 0.75
    kick k*0.7; sleep 0.25
    kick k*0.9; sleep 1
  end
  in_thread do
    sleep 1; snare s; sleep 1; snare s*0.95; sleep 2
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  sleep 4
end
```

Syncopated kicks create groove without double-time aggression.

## Kick and Snare

```
define :kick do |v=1|
  sample :bd_tek, amp: 2.3*v, rate: 0.9
  sample :bd_boom, amp: 0.55*v, rate: 1.15
end

define :snare do |v=1|
  sample :sn_dub, amp: v, rate: 0.82
end
```

Similar to Track 1, but slightly heavier (amp: 2.3 vs 2.2).

# Arrangement

INTRO (32) → BUILD (32) → MAIN (48) → PEAK (48) → DESCENT (24) → OUTRO (32)

## The “Descent” Section

Unique to the finale — a gradual descent before the outro:

```
# DESCENT: 6 bars - drums fading
in_thread do
  6.times do |i|
    drums (1-i*0.1), (0.9-i*0.08), (0.7-i*0.06)
  end
end

in_thread do
  with_fx :reverb, room: 0.75, mix: 0.45 do
    3.times { mel3 0.7; mel2 0.65 }
  end
end

in_thread do
  with_fx :lpf, cutoff: 85, cutoff_slide: 24 do |fx|
    control fx, cutoff: 55 # Filter closing
    6.times { bassline 0.85, 78 }
  end
end

sleep 24
```

### Key elements:

- Drums fade 10% per bar
- Bass filter closes (85→55)
- Melody continues but feeling more distant

## The Outro

A definitive ending:

```
# OUTRO: 8 bars - fading into darkness
in_thread do
  4.times do |i|
    drums (0.6-i*0.12), 0, (0.45-i*0.08)
  end
  # Last 4 bars: no drums
end

in_thread do
  with_fx :reverb, room: 0.95, mix: 0.65 do
    with_fx :echo, phase: 1, decay: 10, mix: 0.55 do
      mel3 0.45; sleep 4
      mel2 0.35; sleep 4
      lead :d4, 4, 0.3 # Final note: root, long, quiet
    end
  end
end

sleep 32
```

**The final note:** D4, held for 4 beats, fading into reverb. The album ends on its home note.

## Key Techniques

### 1. Circular Key Structure

Starting and ending in D Minor creates closure:

Track 1: D Minor → [journey through 6 other keys] → Track 8: D Minor

The return feels like resolution, not repetition.

## 2. Tight Rhythmic Melodies

All melody notes on the 0.5-beat grid:

```
lead :d4, 0.5; sleep 0.5 # Half beat  
lead :f4, 0.5; sleep 0.5 # Half beat
```

No 0.75 or 0.25 — keeps it driving and rhythmic.

This was a deliberate correction from earlier versions that felt “out of rhythm.”

## 3. The Descent Arc

Most tracks go: MAIN → BREAK → PEAK → OUTRO

Terminal Velocity goes: MAIN → PEAK → **DESCENT** → OUTRO

The descent gives listeners time to process before the end.

## 4. Filter Closing

Opening filters = building energy. **Closing filters = releasing energy:**

```
with_fx :lpf, cutoff: 85, cutoff_slide: 24 do |fx|  
  control fx, cutoff: 55 # Slides DOWN  
  # ...  
end
```

The closing filter signals “this is ending.”

## 5. The Final Note

End on the root note, held long:

```
lead :d4, 4, 0.3 # Root (D), 4 beats, quiet
```

This provides definitive closure. The album doesn't just stop — it resolves.

## The Terminal Velocity Concept

The title works on multiple levels:

- **Physics:** Terminal velocity = maximum falling speed
- **Finale:** Terminal = final, ending
- **Velocity:** Speed, momentum from the whole album

The track "falls" into darkness at maximum weight, then resolves.

## Track 1 vs Track 8

Element	Track 1	Track 8
BPM	100	100
Key	D Minor	D Minor
Role	Opening	Closing
Melodies	Aggressive	Emotional
Ending	Loops/fades for DJ	Resolves definitively
Stabs	Present	More prominent
Reverb	Moderate	Building to massive

Same skeleton, different soul.

## Hacker Challenges

1. **Different Key:** The album starts and ends in D minor. Change Terminal Velocity to D *major*. Does the circular structure still work? Does it feel like resolution or betrayal?

2. **Extended Fade:** Double the outro length with an even slower fade. Does it feel more cinematic or just drag?
3. **Remove the Final Note:** Delete the held D4 at the end. Let it just... stop. Which ending is more powerful?
4. **Add One More Element:** The descent strips elements away. What if you *added* something unexpected — a new sound heard only in the final minute?
5. **Loop It:** Remove the outro entirely and make the main section loop seamlessly. If this track played forever, would it work? What makes an ending feel necessary?

## Full Code

The complete track code is available in `08_terminal_velocity.rb`.

---

## Album Complete

You've now walked through all 8 tracks of *Sonic Byte*:

1. **System Override** — Establishing the sound
2. **Nerve Damage** — Industrial aggression
3. **Chrome Cathedral** — Atmospheric breath
4. **Skull Fracture** — Maximum violence
5. **Midnight Protocol** — Melodic triumph
6. **Void Walker** — Slow, heavy power
7. **Core Meltdown** — Climax
8. **Terminal Velocity** — Cinematic resolution

Each track taught different techniques. Together, they form a journey.

Now go make your own.

# Quick Reference

Copy-paste templates for common patterns.

# Track Skeleton

```
use_bpm 100

# === SOUNDS ===
define :kick do |v=1|
    sample :bd_tek, amp: 2.2*v, rate: 0.9
    sample :bd_boom, amp: 0.5*v, rate: 1.2
end

define :snare do |v=1|
    sample :sn_dub, amp: v, rate: 0.85
end

define :hat do |v=1|
    sample :drum_cymbal_closed, amp: 0.25*v, rate: 2.2, release: 0.05
end

define :bass do |n, v=1, c=80|
    use_synth :tb303
    play n, amp: 0.8*v, attack: 0.01, decay: 0.2,
        sustain: 0.1, release: 0.15, cutoff: c, res: 0.3
    use_synth :sine
    play n-12, amp: v, attack: 0.01, sustain: 0.25, release: 0.2
end

define :lead do |n, dur=0.5, v=1|
    use_synth :prophet
    play n, amp: 0.4*v, attack: 0.05, decay: dur*0.25,
        sustain: dur*0.45, release: dur*0.5, cutoff: 90
end

define :hit do |v=1|
    sample :bd_boom, amp: 2*v, rate: 0.4
end

# === PATTERNS ===
define :drums do |k=1, s=1, h=1|
    in_thread do
        4.times { kick k; sleep 1 }
    end
    in_thread do
        sleep 1; snare s; sleep 1; snare s; sleep 2
    end
    in_thread do
        8.times { hat h; sleep 0.5 }
```

```

    end
    sleep 4
end

define :bassline do |v=1, c=80|
  bass :d2, v, c; sleep 1
  bass :d2, v*0.7, c-10; sleep 1
  bass :f2, v*0.9, c; sleep 1
  bass :d2, v, c; sleep 1
end

define :melody do |v=1|
  lead :d4, 0.5, v; sleep 0.5
  lead :f4, 0.5, v*0.9; sleep 0.5
  lead :a4, 1, v; sleep 1
  lead :g4, 0.5, v*0.85; sleep 0.5
  lead :d4, 1, v; sleep 1.5
end

# === ARRANGEMENT ===

# INTRO: 8 bars
in_thread do
  8.times { drums 0.7, 0, 0.4 }
end
in_thread do
  with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
    control fx, cutoff: 80
    8.times { bassline 0.5, 50 }
  end
end
sleep 32

# BUILD: 8 bars
in_thread do
  8.times { drums 0.85, 0.7, 0.55 }
end
in_thread do
  8.times { bassline 0.75, 68 }
end
sleep 32
hit 1

# MAIN: 12 bars
in_thread do
  12.times { drums 1, 0.9, 0.7 }
end
in_thread do

```

```

    12.times { bassline 1, 80 }
end
in_thread do
  with_fx :reverb, room: 0.6, mix: 0.4 do
    sleep 16
    4.times { melody 0.8 }
  end
end
sleep 48

# OUTRO: 6 bars
in_thread do
  6.times { |i| drums (1-i*0.12), (0.85-i*0.1), (0.7-i*0.08) }
end
sleep 24

```

## Common Effects Chains

### Standard Lead

```

with_fx :reverb, room: 0.6, mix: 0.4 do
  with_fx :echo, phase: 0.5, decay: 3, mix: 0.3 do
    # lead code
  end
end

```

### Ethereal/Break

```

with_fx :reverb, room: 0.95, mix: 0.6 do
  with_fx :echo, phase: 1, decay: 6, mix: 0.5 do
    # lead code
  end
end

```

## Tight/Peak

```
with_fx :reverb, room: 0.5, mix: 0.25 do
  with_fx :echo, phase: 0.5, decay: 2, mix: 0.2 do
    # lead code
  end
end
```

## Filter Build

```
with_fx :lpf, cutoff: 50, cutoff_slide: 32 do |fx|
  control fx, cutoff: 110
  # bass code for 32 beats
end
```

## Timing Values

Value	Name	Per Bar
4	Whole note	1
2	Half note	2
1	Quarter note	4
0.5	Eighth note	8
0.25	Sixteenth note	16
0.75	Dotted eighth	~5.3

## Volume Guidelines

Element	Typical Range
Kick sample	2.0 - 2.5
Snare	0.8 - 1.1

Element	Typical Range
Hi-hat	0.2 - 0.3
Bass synth	0.6 - 0.9
Sub sine	1.0 - 1.2
Lead	0.35 - 0.45
Arp	0.2 - 0.3
Pad	0.25 - 0.4

## Section Energy Levels

Section	Drums	Bass	Lead
Intro	0.7, 0, 0.4	0.5	-
Build	0.85, 0.7, 0.55	0.75	0.5
Main	1, 0.9, 0.7	1	0.8
Break	0, 0, 0.35	-	0.5
Peak	1.15, 1.05, 0.8	1.1	0.95
Outro	fade	fade	0.4

## Minor Scale Notes

Key	1	2	3	4	5	6	7
D min	D	E	F	G	A	Bb	C
E min	E	F#	G	A	B	C	D
A min	A	B	C	D	E	F	G
G min	G	A	Bb	C	D	Eb	F
C min	C	D	Eb	F	G	Ab	Bb
B min	B	C#	D	E	F#	G	A

<b>Key</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>
F min	F	G	Ab	Bb	C	Db	Eb

# Synth Cheat Sheet

Quick reference for synths used in *Sonic Byte*.

## Primary Synths

### :tb303

**Character:** Acid bass, squelchy, aggressive

**Best for:** Bass, acid lines

```
use_synth :tb303
play :d2,
    amp: 0.8,
    attack: 0.01,
    decay: 0.2,
    sustain: 0.1,
    release: 0.15,
    cutoff: 80,      # 60-100 typical
    res: 0.3,        # 0.2-0.5 typical
    wave: 0          # 0=saw, 1=square
```

### :prophet

**Character:** Warm, fat, analog

**Best for:** Bass, leads, pads

```
use_synth :prophet
play :c4,
  amp: 0.5,
  attack: 0.05,
  decay: 0.2,
  sustain: 0.3,
  release: 0.4,
  cutoff: 85,
  res: 0.2
```

## :dsaw

**Character:** Aggressive, wide, detuned

**Best for:** Bass layers, stabs, aggressive leads

```
use_synth :dsaw
play :d2,
  amp: 0.5,
  attack: 0.01,
  decay: 0.2,
  release: 0.15,
  cutoff: 85,
  detune: 0.2      # 0.1-0.3 typical
```

## :sine

**Character:** Pure, clean, no harmonics

**Best for:** Sub bass layer

```
use_synth :sine
play :d1,          # Usually octave below main bass
  amp: 1,
  attack: 0.01,
  sustain: 0.3,
  release: 0.2
```

## :pulse

**Character:** Hollow, rhythmic, retro

**Best for:** Arpeggios, rhythmic patterns

```
use_synth :pulse
play :d4,
  amp: 0.25,
  attack: 0.01,
  decay: 0.1,
  release: 0.1,
  cutoff: 100,
  pulse_width: 0.3 # 0.1-0.5 typical
```

## :dark\_ambience

**Character:** Atmospheric, eerie, evolving

**Best for:** Pads, texture layers

```
use_synth :dark_ambience
play :d3,
  amp: 0.35,
  attack: 1.5,
  sustain: 2.5,
  release: 3,
  cutoff: 75
```

## :hollow

**Character:** Ethereal, breathy, haunting

**Best for:** Pads, high texture

```

use_synth :hollow
play :d4,
  amp: 0.25,
  attack: 0.3,
  decay: 0.5,
  sustain: 0.2,
  release: 0.8,
  cutoff: 90

```

## :mod\_saw / :mod\_dsaw

**Character:** Modulated, moving, alarm-like

**Best for:** Hooks, alarm sounds, special effects

```

use_synth :mod_saw
play :g4,
  amp: 0.35,
  attack: 0,
  decay: 0.08,
  sustain: 0.04,
  release: 0.06,
  mod_phase: 0.15,  # Speed of modulation
  mod_range: 7,      # Depth of modulation
  cutoff: 110

```

## Quick Comparison

Synth	Warmth	Aggression	Best Use
:tb303	Medium	High	Acid bass
:prophet	High	Low	Leads, warm bass
:dsaw	Low	High	Aggressive bass/stabs
:sine	Neutral	None	Sub layer
:pulse	Low	Medium	Arps
:dark_ambience	Medium	Low	Atmospheric pads

Synth	Warmth	Aggression	Best Use
:hollow	High	None	Ethereal textures
:mod_saw	Low	High	Hooks, effects

## Parameter Ranges

### cutoff (Filter Frequency)

- **40-60:** Very dark, muffled
- **60-75:** Dark, subby
- **75-90:** Balanced, present
- **90-110:** Bright, cutting
- **110+:** Very bright, harsh

### res (Resonance)

- **0.1-0.2:** Subtle color
- **0.3-0.4:** Classic analog feel
- **0.5-0.6:** Pronounced, acid-style
- **0.7+:** Screaming (use carefully)

### detune (for :dsaw)

- **0.05-0.1:** Subtle thickening
- **0.15-0.25:** Classic detuned sound
- **0.3+:** Very wide, aggressive

# **Common Layering Combinations**

## **Heavy Bass**

:tb303 (character) + :sine (sub)

## **Warm Bass**

:prophet (character) + :sine (sub)

## **Aggressive Bass**

:dsaw (aggression) + :tb303 (grit) + :sine (sub)

## **Rich Lead**

:prophet (main) + :saw (shimmer, octave up)

## **Dark Lead**

:prophet (main) + :dark\_ambience (texture)

## **Atmospheric Pad**

:dark\_ambience (body) + :hollow (high texture)

# Scale Reference

All tracks in *Sonic Byte* use minor keys for their dark character.

## Natural Minor Scales

### D Minor (Tracks 1 & 8)

D E F G A Bb C D  
1 2 b3 4 5 b6 b7 1

**Common bass notes:** D2, F2, G2, A2, Bb1

**Common melody notes:** D4, F4, G4, A4, C5, D5

### E Minor (Track 2)

E F# G A B C D E  
1 2 b3 4 5 b6 b7 1

**Common bass notes:** E2, G2, A2, B2

**Common melody notes:** E4, G4, A4, B4, D5

### A Minor (Track 3)

A B C D E F G A  
1 2 b3 4 5 b6 b7 1

**Common bass notes:** A1, C2, D2, E2, G2

**Common melody notes:** A4, C5, D5, E5, G5

## F Minor (Track 4)

F G Ab Bb C Db Eb F  
1 2 b3 4 5 b6 b7 1

**Common bass notes:** F2, Ab2, Bb2, C2

**Common melody notes:** F4, Ab4, Bb4, C5, Eb5

## C Minor (Track 5)

C D Eb F G Ab Bb C  
1 2 b3 4 5 b6 b7 1

**Common bass notes:** C2, Eb2, F2, G2, Bb2

**Common melody notes:** C4, Eb4, F4, G4, Bb4, C5

## B Minor (Track 6)

B C# D E F# G A B  
1 2 b3 4 5 b6 b7 1

**Common bass notes:** B1, D2, E2, F#2

**Common melody notes:** B4, D5, E5, F#5

## G Minor (Track 7)

G A Bb C D Eb F G  
1 2 b3 4 5 b6 b7 1

**Common bass notes:** G2, Bb2, C2, D2, F2

**Common melody notes:** G4, Bb4, C5, D5, F5

# Sonic Pi Note Names

```
# Octave notation
:c2    # C in octave 2 (bass range)
:d3    # D in octave 3 (low mid)
:a4    # A in octave 4 (middle)
:e5    # E in octave 5 (high)

# Sharps and flats
:fs4   # F sharp
:bf3   # B flat (also :bb3)
:cs2   # C sharp
:ef4   # E flat (also :eb4)
```

## Common Intervals

Interval	Semitones	Character	Example from D
Minor 2nd	1	Tense, dissonant	D → Eb
Major 2nd	2	Neutral	D → E
Minor 3rd	3	Dark, sad	D → F
Major 3rd	4	Bright, happy	D → F#
Perfect 4th	5	Strong, open	D → G
Tritone	6	Very tense	D → Ab
Perfect 5th	7	Powerful, stable	D → A
Minor 6th	8	Dark	D → Bb
Major 6th	9	Warm	D → B
Minor 7th	10	Bluesy, dark	D → C
Major 7th	11	Bright tension	D → C#
Octave	12	Same note, higher	D → D

# Dark vs Bright Intervals

## Darkest (use for tension):

- Minor 2nd (1 semitone)
- Tritone (6 semitones)
- Minor 6th (8 semitones)

## Dark but stable (main vocabulary):

- Minor 3rd (3 semitones)
- Perfect 4th (5 semitones)
- Perfect 5th (7 semitones)
- Minor 7th (10 semitones)

## Avoid in dark music:

- Major 3rd (sounds happy)
- Major 6th (sounds warm)
- Major 7th (sounds jazzy)

# Chord Construction

## Minor Triad

```
# D minor chord: D F A
play [:d4, :f4, :a4]
# or
play chord(:d4, :minor)
```

## Minor 7th

```
# D minor 7: D F A C
play chord(:d4, :m7)
```

## Power Chord (5th)

```
# D5: D A (no third - neither major nor minor)
play [:d3, :a3]
```

## Bass Note Relationships

When writing bass:

### Safe jumps:

- Root to 5th: Strong, powerful
- Root to 4th: Moving, transitional
- Root to minor 3rd: Dark color

### Avoid:

- Root to major 3rd: Sounds too happy
- Large jumps without purpose

```
# Good bass movement in D minor
bass :d2; sleep 1
bass :a2; sleep 0.5    # Fifth - strong
bass :g2; sleep 0.5    # Fourth - movement
bass :f2; sleep 1      # Minor third - dark
bass :d2; sleep 1      # Return to root
```

## Melody Tips by Key

### D Minor

- Emphasize: D, F, A (minor triad)
- Use Bb for darkness
- Resolve to D

## **G Minor**

- Emphasize: G, Bb, D
- Use Eb for extra darkness
- The Bb-A-G resolution is powerful

## **E Minor**

- Emphasize: E, G, B
- The F#-G movement is classic
- Resolve descending to E

# Troubleshooting

When code doesn't work, here's where to look.

## No Sound

### Check the basics first

1. **Volume up?** System volume, Sonic Pi volume (top right)
2. **Correct audio output?** Preferences → Audio → Output Device
3. **Did you press Run?** Cmd+R (Mac) or Alt+R (Windows/Linux)

### Check your code

```
# Does this make sound?  
sample :bd_tek
```

If yes, the problem is in your code. If no, it's a system/audio issue.

## Common silent failures

```
# Missing sleep – code runs instantly and "finishes"
4.times do
  play :c4
  # sleep 0.5 ← forgot this
end

# Volume of zero
play :c4, amp: 0

# Synth that doesn't exist
use_synth :not_a_real_synth
play :c4

# Note too low/high to hear
play :c0    # Too low
play :c9    # Too high
```

## Timing Problems

### Pattern ends too soon

```
define :drums do |k=1, s=1, h=1|
  in_thread do
    4.times { kick k; sleep 1 }
  end
  in_thread do
    8.times { hat h; sleep 0.5 }
  end
  # MISSING: sleep 4
end

8.times { drums 1, 0.9, 0.7 } # Loops overlap!
```

**Fix:** Always end multi-threaded functions with `sleep` equal to the pattern length.

## Elements out of sync

```
# Problem: Threads drift
in_thread do
  loop { kick; sleep 1.001 } # Tiny error accumulates
end

# Fix: Use live_loop for long-running patterns
live_loop :kick do
  sample :bd_tek
  sleep 1 # Always exactly 1 beat
end
```

## Sound Problems

### Muddy/cluttered sound

- Too many elements at once
- Bass and kick fighting
- Too much reverb/delay

#### Fixes:

```
# Reduce element count
# Cut bass when kick hits
# Lower reverb mix: mix: 0.3 not mix: 0.6
```

### Harsh/piercing sound

- Cutoff too high
- Resonance too high
- Too much high frequency content

#### Fixes:

```
# Lower cutoff: 80 instead of 110
# Lower resonance: res: 0.2 instead of res: 0.6
# Add low-pass filter
with_fx :lpf, cutoff: 100 do
    # code
end
```

## Thin/weak sound

- Missing sub frequencies
- Single-layer sounds
- Cutoff too low

### Fixes:

```
# Add sine sub layer
use_synth :sine
play :d1, amp: 1.1

# Layer synths
# Raise cutoff

# Check your rate isn't too high
sample :bd_tek, rate: 0.9 # Lower = more weight
```

## Error Messages

### “Syntax error”

Usually a typo. Look for:

- Missing end statements
- Unclosed strings or brackets
- Misspelled keywords

```
# Wrong
define :kick do |v=1
  sample :bd_tek
end

# Right
define :kick do |v=1| # Missing |
  sample :bd_tek
end
```

## “Unknown synth”

Check the synth name. Use `puts synth_names` to see all available.

```
# Wrong
use_synth :prophet_5

# Right
use_synth :prophet
```

## “Buffer too large”

Sonic Pi has a buffer size limit. The code is too long.

### Fixes:

- Split into multiple buffers
- Remove comments
- Condense repeated code into functions
- Use shorter variable names

## “Note out of range”

MIDI notes go from 0 to 127.

```
play 150 # Too high  
play -5 # Too low
```

## Performance Issues

### Crackly/glitchy audio

- CPU overloaded
- Too many effects
- Too many simultaneous sounds

#### Fixes:

```
# Reduce polyphony  
# Use fewer effects  
# Simplify patterns  
# Close other applications
```

### Late sounds / timing drift

- Code taking too long to execute
- Complex calculations inside loops

#### Fixes:

```

# Pre-calculate values outside loops
notes = scale(:d4, :minor) # Calculate once

live_loop :melody do
  play notes.choose # Just access
  sleep 0.5
end

# NOT this:
live_loop :melody do
  play scale(:d4, :minor).choose # Calculates every time
  sleep 0.5
end

```

## Quick Diagnostic

When something's wrong, isolate the problem:

```

# 1. Does basic sound work?
sample :bd_tek

# 2. Does your synth work?
use_synth :tb303
play :d2, cutoff: 80

# 3. Does your function work alone?
kick 1

# 4. Does your pattern work alone?
drums 1, 0.9, 0.7

# 5. Do multiple patterns work together?
# Add one at a time until it breaks

```

## Still Stuck?

1. Check the Sonic Pi tutorial
2. Search the Sonic Pi forum

3. Simplify until it works, then add complexity back

The bug is almost always simpler than you think.

# Resources

Further learning and inspiration.

## Sonic Pi

### Official Resources

- [Sonic Pi Website](#) — Download and documentation
- [Sonic Pi Tutorial](#) — Built-in tutorial
- [Sonic Pi Reference](#) — Full documentation

### Community

- [Sonic Pi Forum](#) — Community discussions
- [Sonic Pi GitHub](#) — Source code, issues

## Music Theory

### For Electronic Producers

- [Learning Music \(Ableton\)](#) — Interactive basics
- [Music Theory for Electronic Producers](#) — Genre-focused

### Synthesis

- [Syntorial](#) — Learn synthesis by ear

- **Sound on Sound Synth Secrets** — Deep synthesis theory

## Dark Electronic Artists

### Darksynth

- **Perturbator** — Godfather of darksynth
- **Carpenter Brut** — Cinematic, aggressive
- **Dan Terminus** — Complex, progressive
- **GosT** — Extreme, metal-influenced

### Dark Clubbing / Midtempo

- **Gesaffelstein** — Minimal, powerful
- **REZZ** — Hypnotic, dark bass
- **1788-L** — Heavy, cyberpunk
- **Blanck Mass** — Experimental, intense

### Industrial Electronic

- **Irving Force** — Cinematic, metal-influenced
- **Noisecream** — Game soundtrack energy
- **Author & Punisher** — Mechanical, brutal
- **3TEETH** — Industrial rock/electronic

### Recommended Albums

- Perturbator — *Dangerous Days*
- Carpenter Brut — *Trilogy*
- Irving Force — *Godmode*
- Gesaffelstein — *Aleph*

- REZZ — *Mass Manipulation*

## Sound Design

### YouTube Channels

- **In The Mix** — Production techniques
- **Venus Theory** — Experimental sound design
- **Andrew Huang** — Creative approaches

### Books

- *Dance Music Manual* by Rick Snoman — Comprehensive production
- *Mixing Secrets* by Mike Senior — Professional mixing

## Live Coding

### Communities

- [TOPLAP](#) — Live coding community
- [Algorave](#) — Algorithmic dance music

### Other Live Coding Environments

- **TidalCycles** — Haskell-based, pattern-focused
- **FoxDot** — Python-based
- **Overtone** — Clojure-based
- **SuperCollider** — The foundation of Sonic Pi's audio engine

# Music Tools

## DAWs (for further production)

- **Ableton Live** — Industry standard for electronic
- **FL Studio** — Popular, pattern-based
- **Bitwig** — Modern, modular

## Mastering

- **LANDR** — AI-assisted mastering
- **iZotope Ozone** — Professional mastering suite

## This Album

## GitHub Repository

The complete code for *Sonic Byte* is available at:

[github.com/antonarhipov/sonic-byte](https://github.com/antonarhipov/sonic-byte)

## Files

- 01\_system\_override.rb
- 02\_nerve\_damage.rb
- 03\_chrome\_cathedral.rb
- 04\_skull\_fracture.rb
- 05\_midnight\_protocol.rb
- 06\_void\_walker.rb
- 07\_core\_meltdown.rb

- `08_terminal_velocity.rb`

## Continue Learning

### Next Steps After This Tutorial

1. **Modify the tracks** — Change parameters, see what happens
2. **Write your own track** — Use the skeleton, create new patterns
3. **Explore live coding** — Try `live_loop` for performance
4. **Add to your palette** — Discover more synths and samples
5. **Study more music** — Analyze tracks you love
6. **Share your work** — Post to the Sonic Pi community

### Practice Projects

1. **Remix a track** — Take one of the 8 tracks, make it your own
  2. **Write a B-side** — Create Track 9 in the same style
  3. **Genre experiment** — Apply these techniques to a different genre
  4. **Live set** — Convert tracks to `live_loops` for performance
  5. **Collaboration** — Share code, build together
- 

*Making music with code is a journey. This tutorial is just the beginning.*