Bringing Back The Sound Chip

The Case For Using Real-Time Synthesizers in Game Soundtracks

Thomas J. Webb November 2021

What do I Mean by Real-Time Synthesizers?

Software synthesizers as part of the game's audio engine, generating audio tracks for the soundtrack.

What is “Bringing Back the Sound Chip?”

- An homage to a time when games used sound chips built in to the systems they ran on
- A history we can learn lessons from
- A metaphor

What This Talk Isn’t About

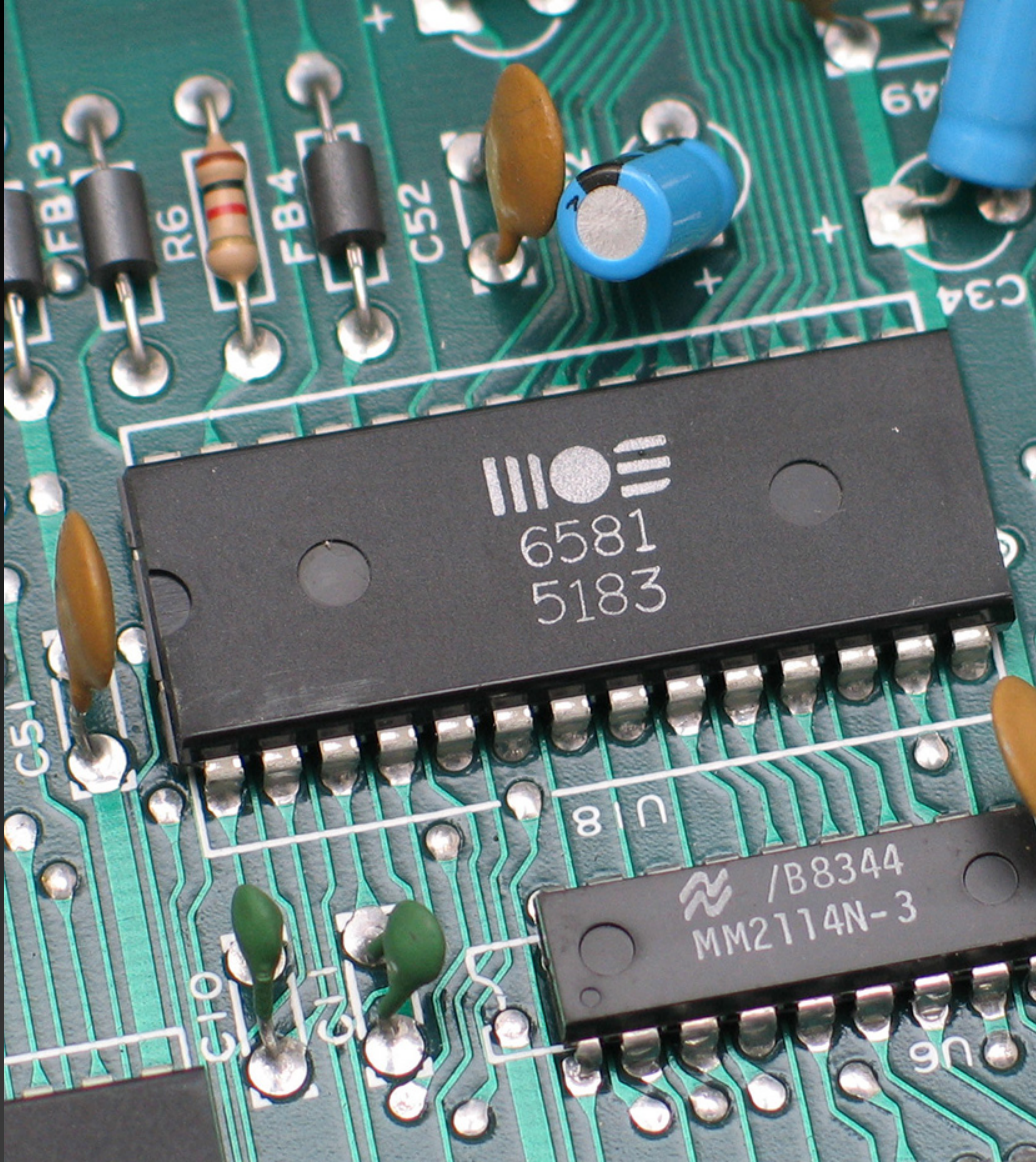
- Already common dynamic music techniques
- Synthesizers used for sound effects/non-musical audio
- Synthesizers used in the content creation
- Using track and bus effects to create dynamism

Overview

1. History of Sound Chips
2. Why Don’t Modern Games Use Real-Time Synthesizers in Music?
3. Why You Should Consider it
4. Implementation Considerations
5. Conclusions

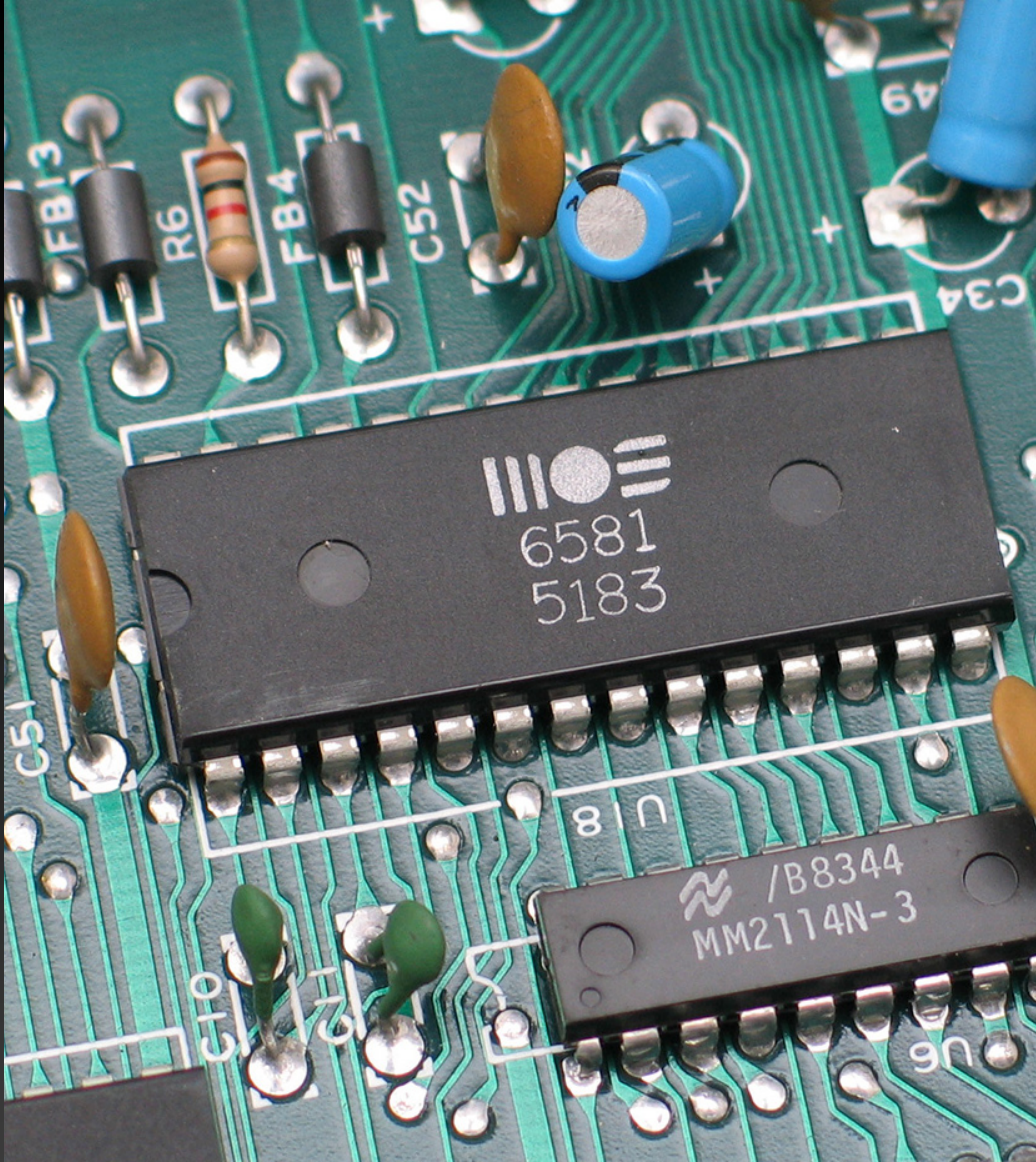
History of Sound Chips

- Programmable Sound Generators (PSGs)
- Subtractive (MOS SID chip)
- Frequency Modulation (FM)
- Wavetable
- Sample-Based
- CD-ROM and PCM



History of Sound Chips

- There were ways to get arbitrary audio on old chips such as the SID
- Some systems (Game Boy, Famicon) had analog input



Why Don’t Modern Games Use Synthesizers?

- Delivering PCM means you can use any techniques in recording or composing you wish with no performance considerations
- No one is doing it so it doesn’t occur to most people and there is little in the way of literature or frameworks
- Game engines often lack flexibility needed for real-time audio
- Heavy on the CPU
- Risk of underruns/pops and clicks
- Headlocked vs. environmental split
- Possible to get sufficient flexibility with existing PCM-based techniques

Why You Should Consider it Anyway

- Can make even short music loops more interesting
- Music can be directly responsive to game events and user input
- GPUs free up CPUs to do interesting things
- You can let the game and the gamer do some of the composition for you
- Consider analogy to cutscenes - they are now generally done in-engine
- Can add realism to in-universe (diegetic) music

Implementation Considerations

Real-Time vs. Offline

Real-Time

- Instant feedback to environment - next audio callback

Offline

- Simpler performance concerns
- Easy integration with existing engines and pipelines
- Doesn’t require as much audio programming-specific skills

Game Engine Audio vs. General Audio

Game Engine Audio API

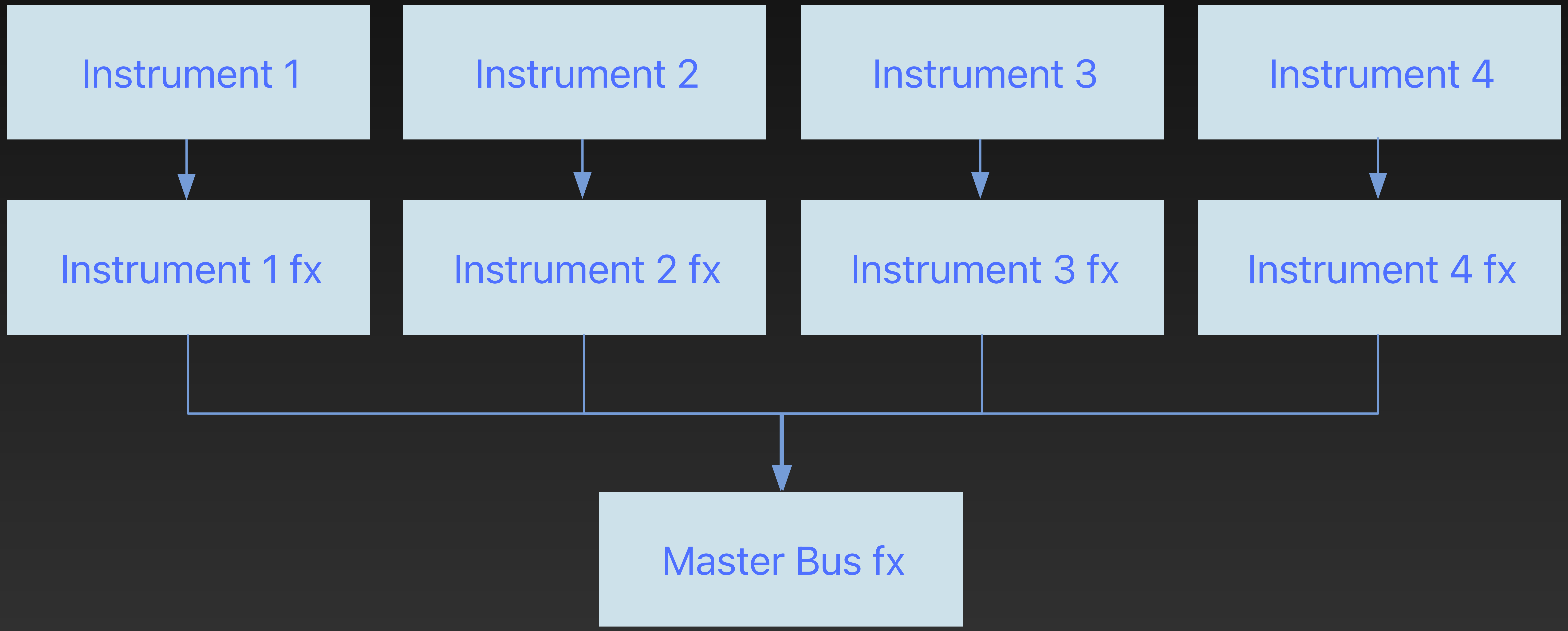
- Environmental/3D audio problems already solved for you
- Default way to use game engines
- Cross-platform issues already dealt with for you
- No need for specialized audio software engineers

General Audio API

- More control over audio settings
- More control over how sound is mixed
- Access to sound cards' pro audio features
- Low latency

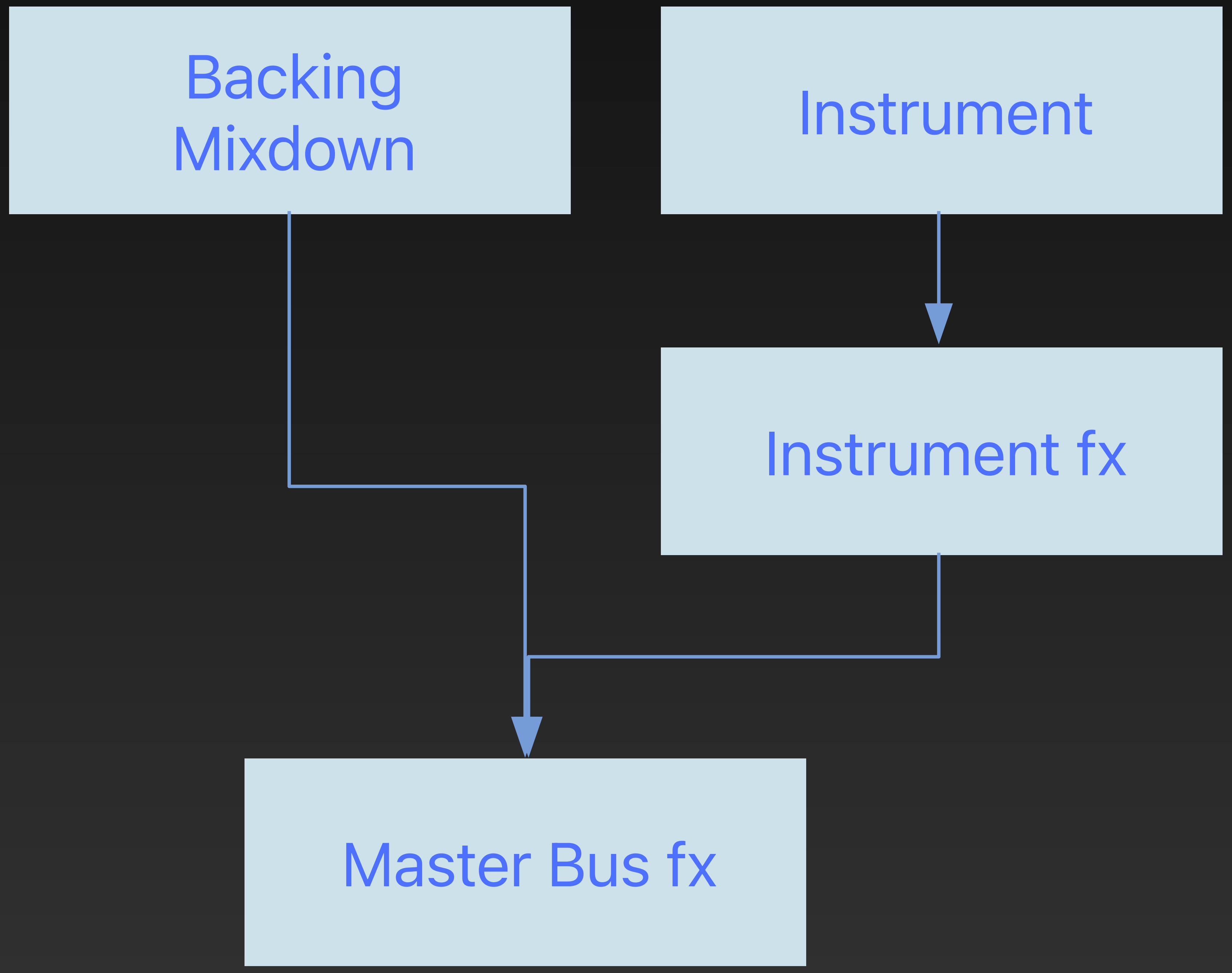
To Synthesize Everything or Not

Should You Ship Your DAW Project?



To Synthesize Everything or Not

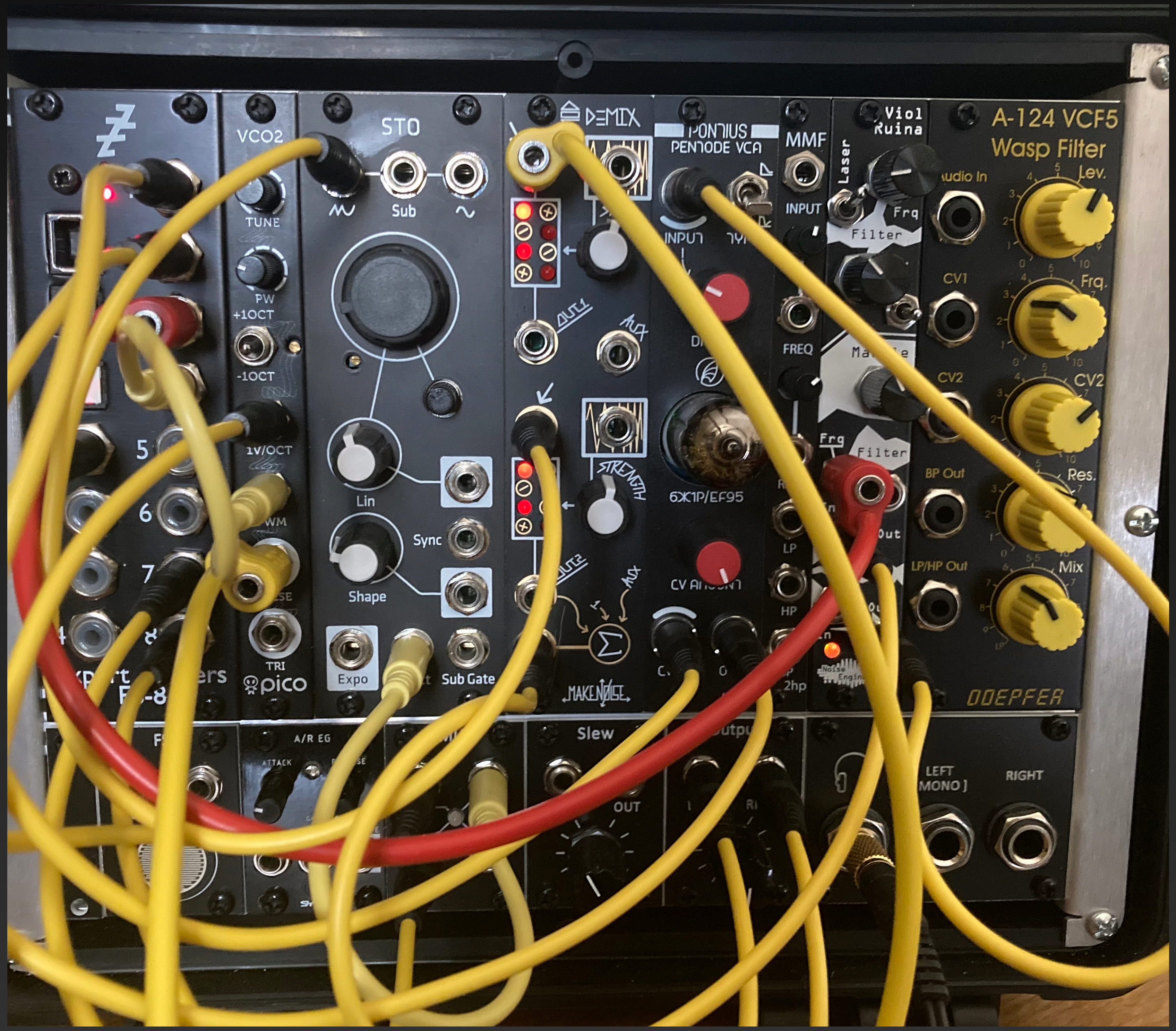
A Hybrid Approach



Synthesis Techniques

A Few Ideas

- FM/PM Synthesis
- Wavetable and Sample-based
- Subtractive
- Phase distortion and wave folding
- Analog modeling
- Physical modeling

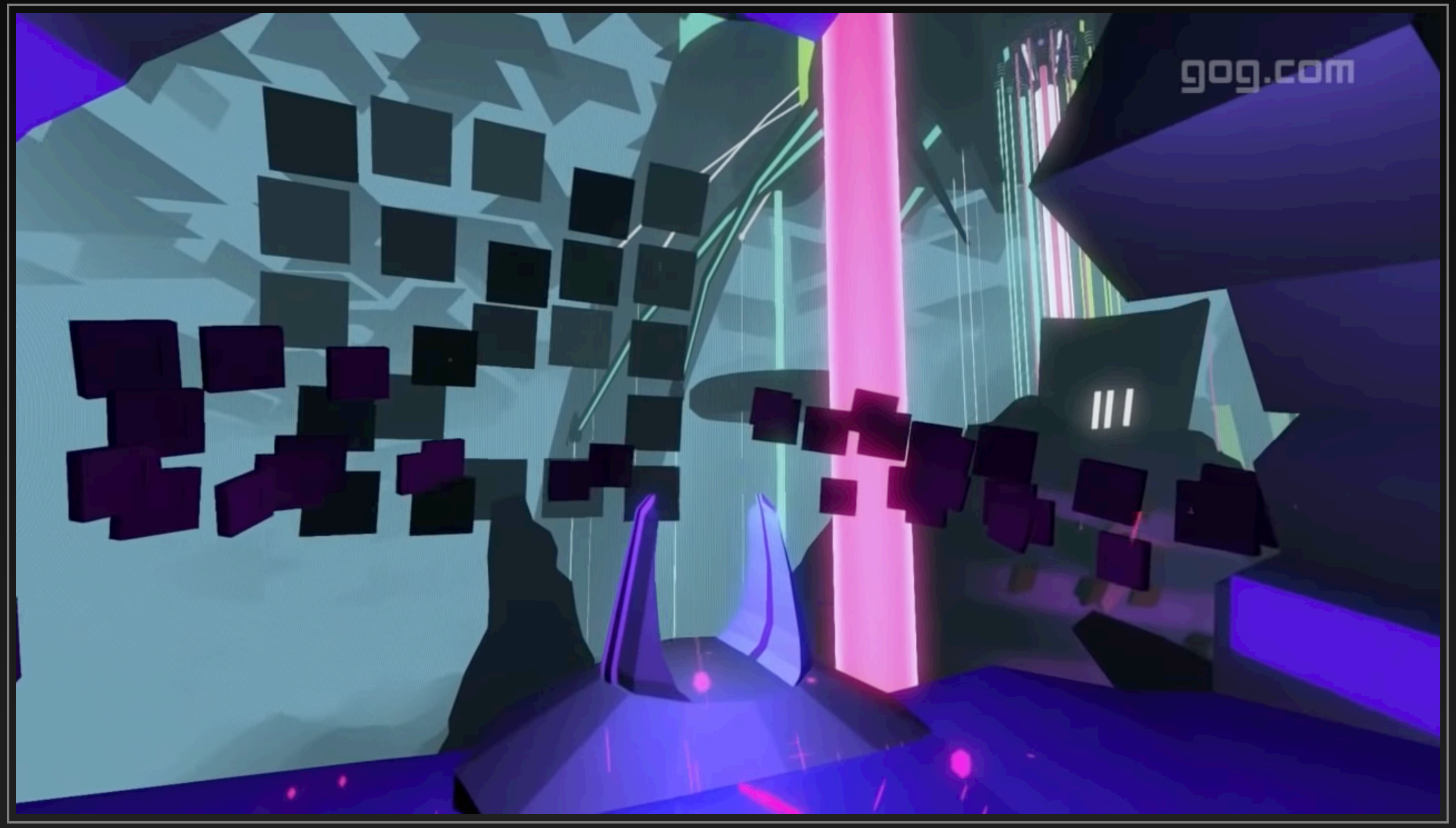


Free Synthesizer Code
Where to Get Started

- libpd/puredata (C) and Heavy (C++)
- libfmsynth (C)
- TinySoundFont (C)
- webaudio and pizzicato.js (js)
- grig.synth (haxe)
- …countless free example code

Modern Games That Use Synths

- Fract OSC
- Zya/Song Battles
- …that’s it (as far as I know)



Conclusions

Some Closing Thoughts & Recommendations

- Worth considering, whatever the game genre
- Various intermediate options exist
- If more people do this, it gets easier
- Weigh the risks

Thanks!

https://linktr.ee/thomasjwebb