“Scapeshift Modular Framework” (SMF)*

A modular framework for building hardware-controlled generative musical instruments

By Tim Exile

Whitepaper Part One - Outline and Principles

**working title*

Overview

I'm writing this "whitepaper" mainly to get my head straight about the unmet need, the why now?, imagined user journeys, design principles, commercial model, community strategy, technical concepts and implementation details of SMF.

But this also serves two other purposes: firstly to share my plans with the community to spark conversations which might lead to better plans before I start building in anger. Secondly, as the basis of a prompt or series of prompts to implement this in something like Claude Code at a future time.

The first implementation of SMF will be in Native Instruments' Reaktor. There are many advantages to building this in Reaktor.

1) I know it like the back of my hand and have full certainty that I can implement everything laid out here with high quality.
2) As an environment for building user-customisable instruments, I think it's second to none.
3) It's a rock solid distribution platform with decades in production.
4) A lot of my community are already Reaktor users.

Building in Reaktor is by far the fastest and most reliable way to get SMF out in the world in the first instance. This is worth the other trade-offs such as the extra friction of needing to use Reaktor and in some cases Kushview Element too.

In the long run, my goal is to develop this as an independent platform. But that is a much more complicated task and will require a lot of risk and investment. I first want to prove the viability of the concept in Reaktor before embarking on a bigger quest.

Unmet Needs

Core Need: The Problem and Solution

Building your own instrument is way too complicated. You have to be a very patient nerd to do it. If you're lucky enough to be a very patient nerd, a world of creative and technical fulfillment awaits. I want to make that world available to way more people.

When I launched Scapeshift in November 2024, there were two very common responses.

1) Many people desired, or had tried and failed, to build their own generative instruments - whether in Ableton, Max(4Live), PureData, Reaktor, Supercollider, Eurorack, VCVRack or Bitwig Grid.

2) While people loved the interaction of Scapeshift, they found the choice of sounds limiting. Hats off to my esteemed Patreon Remi Fox-Novak for saying it was like playing “Tim Exile, the game”. He nailed it.

My goal is to enable you to build “You, the game”. When I say “You, the game”, I really mean “You, the musical instrument”, but “You, the game” is snappier so I’ll use that for the rest of this document.

The key question is what’s missing that has stopped you building “You, the game” so far? Here is my hypothesis:

You need to be able to build a fully-fledged “You, the game” v1 within 30 minutes of first using the system.

None of the existing solutions can achieve that. Existing solutions either:

1) Require months of learning and building to get anywhere near an implementation of “You, the game”, or
2) Quickly become unwieldy once the system you’re building gets large enough to feel like “You, the game”.

Building “You, the game” is a path of discovery. There is no substitute for the long, slow work of learning a framework and acquiring the skills to make it your own. And in fact this process can be very rewarding in its own right.

But the journey must start with an experience that gives you a taste of the destination, otherwise only those with the strongest faith that the journey is worth it will set out. This is where we’re at today. If the journey can start with an actual experience, many more will set out on the path.

This is why the primary goal of building the SMF framework is the 30 minute rule. You MUST be able to assemble a complete customised instrument within 30 minutes of setting out.

Secondary Needs: Skill Acquisition, Creative Development and the Formation of Community And Culture

Building a system for ease of use alone is not enough. It will result in building “You, the game” being another fleeting, socially-isolated experience to be consumed then disposed of.

I want SMF to sustain an unending journey of learning, personal creative development and the formation of community and culture.

Learning and Personal Creative Development

The art and craft of instrument-building is a creative practice, similar to learning how to produce in a DAW. It's both technical and creative. There is no limit to the depths to be explored in either of these dimensions and the two paths of discovery go hand in hand.

The SMF ecosystem must have a clear, linear, shallow and open-ended learning curve that facilitates progress in both these dimensions. It must start with instant-gratification experiences that require minimal prior expertise, and progress all the way to a deep understanding of the entire framework and an ever-growing collection of self-made instruments, acquired performance skills and musical experiences or recordings, as the user wishes.

Designing the system with specific skill-levels of interaction are key. See imagined user journeys and design principles for more detail.

Documentation and educational content will also be key.

Formation of Community and Culture

I'm inspired by the way the Eurorack community has formed. It's a healthy ecosystem of module manufacturers, deep-divers, youtubers, collectors and the curious. There's a strong DIY spirit. Module manufacturers are happy to educate others on how to build their own modules and the leaders in the space are happy to spill the tea on their approaches to live performance. There's a sense of 'the weirder the better'. Exploring obscure techniques and aesthetics is almost a sport.

In the early days of the Reaktor community, it felt a bit like this - 100s of builders and tinkerers making the weirdest things, tens of thousands using them in their productions and a sense that the community had momentum and was a place to belong. That spirit has largely fallen away but I want to bring it back. I want SMF to be something that a community and culture can form around.

Documentation is key to this, as is providing places for the community to connect and opportunities for third party developers to create and sell premium modules for the SMF system (see commercial / licensing model).

Why Now

The art and craft of electronic instrument-building has been with us for decades, but it's been reserved for the technically adventurous and those with the time and money to put into climbing the steep and long learning curve to competency. As a result, it has been niche.

The opportunity to make instrument-building more accessible isn't new but it's more relevant than ever.

The most widely-adopted electronic music practice in the 21st century has been learning to produce in a DAW. The social pay-off for practicing this craft is publishing music online via streaming services and social networks, and the discovery of and access to communities this comes with. The pressure of attention scarcity has increasingly obstructed the social pay-off for over a decade. It has become harder and harder to get anyone to listen to music online and has therefore become harder to build and access community.

More recently, the capability of generative AI has made leaps. It can now produce output that sounds as convincing as the output of a producer with years of experience and expensive equipment. This undermines trust in music shared online, adding further pressure to the social payoff for producing music in a DAW.

As producing music in a DAW becomes less viable as a line of work and less rewarding as a hobby, other practices that are more demonstrably human will become more important. I believe that building and learning to play your own custom instrument is a strong contender to become a core practice for those who love music. This is why I'm building SMF now.

Imagined User Journeys

Here is a sequence of imagined user journeys. As with the rest of this document, I'm writing these to help myself get clear about the design requirements for SMF and tease out everything that needs to be considered to make this journey possible.

I don't intend to do a complete analysis of personas and use-cases. This is much better done "in the field" once something functioning is out there and real people are using it in ways I could never predict.

These imagined journeys follow a path from beginner up to the most advanced level of interacting with the ecosystem. In reality, I don't expect anyone to follow this exact path. Most will not progress this far, some will enter the ecosystem at a higher level. And yet more will find completely different journeys within the ecosystem.

Level 1

I hear about SMF from a friend, post or advert. I download, install and launch a free entry-level product (and Reaktor). I run it as a plugin in my DAW.

I see a clean and simple UI showing patterns to launch, some tools to manage patterns and macro knobs which control the entire structure of the music being generated.

I hit play. Like Scapeshift, it creates finished-sounding music right out of the box. I explore different patterns displayed in the UI and am impressed by the sonic range. I tweak the macro knobs and hear the entire structure of the composition respond to my inputs.

I open one of the instrument tabs and see more parameters displayed. I start to explore these parameters and realise that I have precise control over all of the elements contributing to the music.

I dial in something to my taste and save it down as a user pattern in the UI, repeating to save a number of my own patterns.

Using my controller keyboard and DAW, I then put together a sequence of patterns I've created and render it out as a track from within my DAW.

Level 2

A day later, I receive an email telling me that you can easily build your own instruments with SMF. It has a link to download some basic modules to get started for free.

I watch a 3 minute video explaining how it works. I download the modules, launch one of the template projects in Reaktor and open the folder of modules. I drop in a Drum Generator module to the Reaktor Structure window, connect a single wire from the 'Brain' module to the Drum Generator and wire the L & R audio outputs to the mixer module.

I hit play on the Panel window and immediately hear a drum pattern playing back. It's full of interesting details and modulations. I tweak some of the macro controls on the Brain interface in the Panel window and hear the pattern and sounds changing as I move the knobs.

I drag a Bass Generator module to the Reaktor Structure window and connect it up as I did the Drum Generator. As soon as I connect up the module, a Bass part joins the mix. I see the Bass Generator interface appear in the Panel window alongside the Brain and Drum Generator interfaces.

I adjust some of the parameters in the Bass Generator interface to dial in the preset to my liking. I use the pattern controls in the Brain to save this preset.

I continue adding modules until I have a complete ensemble of instruments that can recreate an entire track. Using the macro controls in the Brain interface, I control the overall sound of the entire structure, like a conductor conducts an orchestra.

Inside my DAW, I use my controller keyboard to build a sequence of patterns and record automation of the macro controls and bounce out a track made with my custom instrument.

I continue to explore the free modules and I follow the QR code to the premium modules shop, where I purchase a couple of premium modules to add to my system. I start to build a library of my own self-created instruments.

Level 3

A few days later I receive an email about ways to further customise my own instrument, by adding MIDI controller mappings and by controlling external instruments using SMF. I'm interested in both. I explore MIDI mappings first.

I watch a 3 minute video where I learn that I can purchase mappings for common hardware controllers. These mappings transform my custom SMF instruments into hands-on hardware-controlled performance instruments. It's as simple as dropping in the mapping module and connecting it up to the Brain module in the structure with a single wire.

As a Launchpad Mk3 user, I purchase the Launchpad Mk3 mapping. I insert the module into the Structure, wire it up to the terminal in the Brain module and immediately I have hands-on access to a range of functions which make playing my instrument tactile and performative.

I now explore SMF's ability to control external plugins. I watch a 3 minute video and learn there's a SMF module that can generate MIDI data to control plugins. I download and install the free modular VST host Kushview Element as instructed in the video.

I open Reaktor as a plugin inside Kushview Element and follow the link to purchase the MIDI out module, insert it into the structure and connect it to the Brain. I open Serum inside Kushview Element and wire it up to Reaktor's output. In Reaktor, I select modulation channels to map to controls in Serum and set up a number of Serum controls to be modulated by SMF. I patch Serum back into Reaktor and into the Mixer module to access SMF's built-in mixing algorithms.

I continue to add more plugins, building up an entire instrument from third party softsynths triggered by SMF and controlled by my Launchpad Mk3.

Level 4

A week later, I receive an email telling me I can build my own custom SMF instruments. I watch a 10 minute video about how to get started.

I open Reaktor and explore the 'components' folder and follow the basic tutorial I saw in the video. After 30 minutes I've created my first self-built custom instrument. I feel there's a huge amount of depth here that is within my grasp, with a little more structured help

In the original email about building my own custom instruments, there's a link to a video course about building your own custom SMF instruments. I decide to purchase the paid course.

Over the next few weeks, I learn all the techniques and capabilities that are required to build my own custom SMF instruments. My imagination is set alight with the types of things I can build.

I spend the next few months creating a range of different instruments. Because it's so easy to combine these instruments and control them within SMF, I'm able to use WIP instruments in a performance setting immediately. I feel like my creativity has been unleashed. I'm now able to explore instrument-building, sound design and performance techniques in near-realtime and in parallel.

Level 5

A few months later, I receive an email about how to become an SMF developer. I'm quite happy with some of the instruments I've built over the past few months so I'm curious. I watch a 20 minute video about some of the key concepts that an SMF developer needs to know.

I decide to purchase the 5-hour 'SMF advanced developer course'. In this, I learn all about how to develop high-quality user-interfaces and sound-design and how to package up my SMF instruments into a production-ready product.

I spend the next month working on bringing my best instrument creation up to scratch. I submit it to the SMF store for listing and get it approved. It goes on sale and a month later I start earning royalties.

Design Principles

These are principles to guide technical and design decisions. They're provisional and are tools to help the planning process. Part 2 of this paper will be much more technical. My goal is to reference these design principles in all of the technical definitions to make sure that the technical decisions I've made will deliver against these principles.

I've split them into Building and Playing. Building principles guide how the process of building must feel. Playing principles guide how the results must feel to play.

Building

Building should be…

Skill-appropriate: always meets the user at their skill level, or assume zero skill if unable

Play-biased: yields playable results in the shortest time possible

Creativity-biased: is an expression of creative rather than technical intent

Learning-biased: implicitly reveals how the system is working and rewards deeper investigation

Modular: is achieved by adding, arranging and connecting modules

Impact-maximised: single building actions should have maximal creative impact, with a bias towards bundling up multiple actions under the hood.

Componentised: Modules are made up of nested components that are well-described and map perfectly to the learning curve of the overall system.

Extensible: Supports future developments which add more capabilities to the system

Playing

The instruments you build should be…

Orchestral: feature a full set of pattern generators and instruments capable of creating complete musical structures

Mixed: create a sound which is balanced and cohesive regardless of the instrumentation

Deterministic: any musical structure created by the system must always be reproducible

Surprising: it must be possible to provoke the system to generate surprising but musically relevant variations on the current musical structure

Conductable: it must be possible to control the entire musical structure with simple controls like how a conductor would conduct an orchestra

Tweakable: it must be possible to discover and access all the deeper parameters of the structure to make fine-tuned adjustments to the output

Journaled: it must be possible to save, browse and recall snapshots of the entire structure

Curatable: it must be possible to curate and group snapshots

Performable: it must be possible to execute a live performance with curated snapshots and controls

Sequenceable: it must be possible to create sequenced compositions with the available controls

Hardware-controlled: it must be controllable by a wide range of off-the-shelf MIDI controllers, without the need to manually set up mappings

Commercial model / licensing

SMF will be creative commons non-commercial. Users will be free to modify purchased SMF modules and create their own SMF modules from scratch without any licensing cost.

Some SMF products, modules and components will be offered for free, other products, modules and components will be for purchase.

Some day, I plan for there be an SMF store. SMF creators will be able to submit their creations to be sold on the SMF store.