

# MRuby-Zest - A new GUI toolkit for audio programs

Mark McCurry

June 5th, 2018

# Zyn-Fusion

The screenshot displays the Zyn-Fusion software interface, which is a graphical user interface for audio programs. The interface is dark-themed and features a central Envelope editor with a blue line graph showing the attack time curve. The graph starts at a low level, rises to a peak, and then gradually decays. The Envelope editor is divided into three sections: GENERAL, ENVELOPE, and LFO. The GENERAL section includes controls for VOL, SCALE, PAN, STRETCH, STRENGTH, and TIME. The ENVELOPE section includes controls for A.DT, D.DT, S.VAL, R.DT, and STRETCH. The LFO section includes controls for FREQ, DEPTH, START, STR, DELAY, A.R., and F.R. The interface also features a keyboard at the bottom, a MIDI CC control, and various utility buttons like FILE, MIDI, LEARN, FINE, VOLUME, NRPN, and KEY SHIFT. The top left corner shows the ZYN logo and the title 'ATTACK TIME'. The top right corner shows a warning icon. The bottom right corner shows the MIDI CC control with a dropdown menu set to '01: MOD.WHEEL'.

# Another GUI Toolkit? Why?

# Another GUI Toolkit? Why?

- ▶ Qt
- ▶ GTK
- ▶ AVTK
- ▶ roltk
- ▶ DPF
- ▶ JUCE
- ▶ fltk

# Challenges

Toolkit's:

- ▶ Maturity
- ▶ Suitability for use in plugins
- ▶ Development speed

# Zyn Timeline

- ▶  $\approx$  3 months of time
- ▶  $\approx$  30 views to implement

# Zyn Timeline

- ▶  $\approx$  3 months of time
- ▶  $\approx$  30 views to implement
- ▶ Not a lot of time

# Zyn Timeline

- ▶ I'm bad at GUI Programming



# Zyn Timeline

- ▶ I'm bad at GUI Programming
- ▶ (and I hope I'm not alone)

# Zyn Timeline

- ▶ Target DRY
- ▶ Target fast feedback loop
- ▶ Target long term maintainability

# Zyn Timeline

New framework is a reasonable investment

- ▶ From scratch look and feel
- ▶ Take advantage of Zyn metadata model
- ▶ Provide something that can be enhanced long term

# Borrowing Ideas

- ▶ Qt's QML
- ▶ QML's built in hotloading
- ▶ rtosc's metadata system

# Qt's QML

Provides easy way to:

- ▶ Build widget trees
- ▶ Define custom behavior for edge cases
- ▶ Constrain how you expect widgets to be extended

# Qt's QML

- ▶ QML's organization is great, but javascript is not-fun
- ▶ Provides a means for organizing widgets
- ▶ Makes widget extension easy

# MRuby - Not Ruby



## QML before - a parsing standpoint

```
MouseArea {
    id: ma
    property var styleData

    anchors.fill: parent
    onPressed: {
        //javascript
        parent.currentRow = styleData.row
        parent.selection.select(styleData.row)
    }
    onClicked: {
        //javascript
        console.log(styleData.value)
    }
}
```



## QML after - a parsing standpoint

```
MouseArea {
    id: ma
    property var styleData

    anchors.fill: parent
    onPressed: lambda {
        #Ruby
        parent.currentRow = styleData.row
        parent.selection.select styleData.row
    }
    onClicked: lambda {
        #Ruby
        puts styleData.value.inspect
    }
}
```

# So What is MRuby-Zest

- ▶ Uses QML's syntax in a MRuby environment
- ▶ Builds off of rtosc's exported metadata for quick dev

## So What is MRuby-Zest

- ▶ Uses QML's syntax in a MRuby environment
- ▶ Builds off of rtosc's exported metadata for quick dev
- ▶  $\approx 1,000$  commits so far
- ▶  $\approx 15$  kloc of QML (widgets)
- ▶  $\approx 6$  kloc ruby
- ▶  $\approx 7$  kloc C

# Components of the toolkit

- ▶ mruby-qml-parse
- ▶ mruby-qml-spawn

# Components of the toolkit

- ▶ mruby-qml-parse
- ▶ mruby-qml-spawn
- ▶ mruby-zest

# Components of the toolkit

- ▶ mruby-qml-parse
- ▶ mruby-qml-spawn
- ▶ mruby-zest
- ▶ osc-bridge

# Components of the toolkit

- ▶ mruby-qml-parse
- ▶ mruby-qml-spawn
- ▶ mruby-zest
- ▶ osc-bridge
- ▶ mruby-widget-lib

# OSC bridge

```
{
  "path"      : "/part[0,15]/kit[0,15]/padpars/GlobalFilter/basefreq",
  "shortname": "cutoff",
  "name"      : "basefreq",
  "tooltip"   : "Base cutoff frequency",
  "units"     : "Hz",
  "scale"     : "logarithmic",
  "type"      : "f",
  "range"     : [31.25,32000]
},
{
  "path"      : "/part[0,15]/kit[0,15]/padpars/GlobalFilter/freqtracking",
  "shortname": "f.track",
  "name"      : "freqtracking",
  "tooltip"   : "Frequency Tracking amount",
  "units"     : "%",
  "scale"     : "linear",
  "type"      : "f",
  "range"     : [-100,100],
  "default"   : "0.0f"
},
```



# QML Loading

- ▶ Class definitions
- ▶ Property definitions
- ▶ Method definitions
- ▶ Class instance specialization\*\*

## QML Loading - Live or at build

- ▶ Classes can be parsed and turned into .rb at build
- ▶ .qml files can be reloaded and re-instantiated at runtime

## Hotloading - setup

```
Knob {  
  function draw(vg) {  
    active_color = :blue  
  
    draw_outline()  
    draw_xxx()  
    ...  
    ...  
    ...  
    ...  
  }  
}
```



## Hotloading - making changes

```
Knob {  
  function draw(vg) {  
    active_color = :red  
  
    draw_outline()  
    draw_xxx()  
    ...  
    ...  
    ...  
    ...  
  }  
}
```



## Hotloading - update on saving

```
Knob {  
  function draw(vg) {  
    active_color = :red  
  
    draw_outline()  
    draw_xxx()  
    ...  
    ...  
    ...  
    ...  
  }  
}
```



# Widgets

The screenshot displays the ZYN audio software interface, which is a complex GUI for audio programming. The interface is organized into several sections:

- Top Bar:** Contains menu options (FILE, MIDI, ATTACK TIME), playback controls (STOP, PLAY, LEARN), and utility buttons (FINE, VOLUME, NRPN, KEY SHIFT, and a warning icon).
- Part Settings:** A row of tabs for different parameters: GLOBAL (selected), VOICE, OSC, MOD-OSC, MODULATION, VOICE LIST, and RESONANCE. It also includes a 'C P' button.
- Grid:** A 4x4 grid of buttons (1-16) for selecting different parts or voices.
- Browser:** A section with 'BROWSER', 'MIXER', and 'OTS' buttons, and another 4x4 grid of buttons (1-16).
- Envelope Editor:** A large central area showing a blue line graph representing the attack time envelope. The graph starts at a low level, rises to a peak, and then gradually decays. A vertical blue line indicates the current time position. To the right of the graph are buttons for 'FREE', 'ADD', and 'DELETE', and a 'SUSTAIN POINT' control with a value of '2' and a '288.1 MSEC' display.
- General and Envelope Controls:** Two columns of knobs and buttons. The 'GENERAL' column includes VOL, SCALE, PAN, STRETCH, STEREO, STRENGTH, and TIME. The 'ENVELOPE' column includes A.DT, D.DT, S.VAL, FRCR, LIN/LOG, R.DT, and STRETCH. The 'LFO' column includes FREQ, DEPTH, START, STR, DELAY, A.R., F.R., and SYNC.
- Bottom Section:** Features a piano roll keyboard, a 'PAD' button, and a row of controls for 'AMPLITUDE', 'FREQUENCY', 'FILTER', 'VELOCITY', 'VMND', 'OCTAVE', 'QWERTY', 'C.VAL', and 'MOD.WHEEL'.

# Widgets, Widgets

The screenshot displays the ZYN software interface, specifically the 'PART SETTINGS' section for a drum kit. The interface is dark-themed and features a grid of 16 drum kits, each with a unique name and associated MIDI notes. The 'KIT NAME' column lists various drum components, while the 'LAST NOTE' column shows the MIDI notes (Mn, R, Mx) for each. The 'MIN. KEY' and 'MAX. KEY' columns indicate the key range for each kit. The 'ADD', 'SUB', and 'PAD' columns provide controls for adding, subtracting, and padding the kit. The 'EFFECT ROUTE' column shows the effect chain for each kit, currently set to 'FX1'. The interface also includes a 'MIDI LEARN' section and a 'Piano Roll' at the bottom.

KIT NAME	MIN. KEY	LAST NOTE	MAX. KEY	ADD	SUB	PAD	EFFECT ROUTE
1 DRUMS KIT		Mn R Mx		EDIT	EDIT	EDIT	FX1
2 SNARE DRUM 2		Mn R Mx		EDIT	EDIT	EDIT	FX1
3 HIT HAT (CLOSED)		Mn R Mx		EDIT	EDIT	EDIT	FX1
4 CYMBAL		Mn R Mx		EDIT	EDIT	EDIT	FX1
5 BASS DRUM 1		Mn R Mx		EDIT	EDIT	EDIT	FX1
6 HARD SNARE		Mn R Mx		EDIT	EDIT	EDIT	FX1
7 HIT-HAT (PEDAL)		Mn R Mx		EDIT	EDIT	EDIT	FX1
8 SNARE DRUM 1		Mn R Mx		EDIT	EDIT	EDIT	FX1
9 SIDE STICK		Mn R Mx		EDIT	EDIT	EDIT	FX1
10 BASS DRUM 2		Mn R Mx		EDIT	EDIT	EDIT	FX1
11 TOM		Mn R Mx		EDIT	EDIT	EDIT	FX1
12 ...		Mn R Mx		EDIT	EDIT	EDIT	FX1
13 ...		Mn R Mx		EDIT	EDIT	EDIT	FX1
14 ...		Mn R Mx		EDIT	EDIT	EDIT	FX1
15 ...		Mn R Mx		EDIT	EDIT	EDIT	FX1
16 ...		Mn R Mx		EDIT	EDIT	EDIT	FX1

At the bottom of the interface, there is a piano roll and a control section with buttons for 'VELOCITY', 'V.MD', 'OCTAVE', 'QWERTY', 'C. VAL', and 'MOD. WHEEL'. The 'QWERTY' button is currently selected.

# Widgets, Widgets, Widgets

The screenshot displays the ZYN audio software interface, which is a complex GUI for audio programming. The interface is organized into several sections:

- Top Bar:** Includes menu options like FILE, MIDI, and LEARN, along with transport controls (play, stop, learn) and a volume knob.
- Part Settings:** A vertical sidebar on the left with buttons for PART SETTINGS, BROWSER, MIXER, KITS, MIDI LEARN, EFFECTS, and SUB/PAD.
- OSC Panel:** The central control area for the oscillator, featuring:
  - BASE WAVEFORM:** A graph showing a sawtooth wave with a frequency spectrum plot above it.
  - MOD-OSC:** Controls for BASE FUNC. (SAW), WAVESHAPING (UNDISTORT), MAG. TYPE (LINEAR), and MODULATION (REV).
  - FILTER:** Controls for BF MOD. (NONE), LP2, and AS BASE.
  - ADAPT. HARM.:** Controls for R (PRE/POST), ADAPT. HARM., and CLEAR ALL.
  - SPECTRUM ADJ.:** Controls for AMOUNT, C. FREQ., POW, and TO SINE.
- FULL OSCILLATOR:** A graph on the right showing the full oscillator waveform and its frequency spectrum.
- Piano Roll:** A central area with a piano keyboard at the bottom and a grid above it for editing notes. The grid shows several notes placed across 32 steps.
- Bottom Bar:** Includes a piano keyboard, velocity and volume knobs, and MIDI CC controls (QWERTY, MOD.WHEEL).



# Even more widgets

The screenshot displays the ZYN software interface, specifically the 'HARMONIC DISTRIBUTION MODEL' window. The interface is dark-themed and features a variety of widgets and controls:

- Top Bar:** Includes a 'ZYN' logo, a menu bar with 'FILE', 'MIDI', and 'HARMONIC DISTRIBUTION MODEL', and control buttons for 'FINE', 'VOLUME', 'NRPN', and 'KEY SHIFT'.
- Left Panel:** Contains 'PART SETTINGS' with a 4x4 grid of buttons (1-16), a 'BROWSER' section, and 'MIXER' and 'KITS' sections with similar 4x4 grids. Below these are 'MIDI LEARN', 'EFFECTS', and 'PAD' sections with 'ADD', 'SUB', and 'PAD' buttons.
- Main Area:** Divided into three tabs: 'HARMONIC STRUCTURE' (selected), 'OSCILLATOR', and 'ENVELOPES & LFOS'. The 'HARMONIC STRUCTURE' tab shows a spectral plot with vertical lines and a waveform plot with a blue curve.
- Right Panel:** Contains several control sections:
  - OVERTONE POS.:** Includes 'BANDWIDTH', 'FORCE H.', 'POWER', 'SPECTRAL MODE', 'BW. SCALE', and 'BANDWIDTH' controls.
  - HARMONIC PROFILE:** Includes 'BASE TYPE' (P1), 'DOUBLE EX', 'FULL', 'AUTOSCALE', 'AMP. MLT.', 'GAUSS', 'AMP. MODE' (P1), and 'DIV1' (P2) controls.
  - HARMONIC CONTENT:** Includes 'C-4', '2', '3', and 'SAMPLE SIZE' (128K) controls.
- Bottom Panel:** Features a piano roll keyboard, 'VELOCITY', 'VIBO', 'OCTAVE', 'QWERTY', 'C. VAL', and 'MOD. WHEEL' controls.

# Future Work

- ▶ Translations
- ▶ More data visualizations
- ▶ Animations
- ▶ Automated Screenshot collection
- ▶ Exploiting the scripting capabilities more
- ▶ Separation from Zyn

# Conclusions

- ▶ MRuby-Zest powers Zyn-Fusion
- ▶ Adds hotloading and scripting to the plugin level UI design
- ▶ Builds off existing tools for streamlined dev
- ▶ It's new and ready to adapt

Questions?