Erae Touch II Shape Editor

A JUCE-Based Visual Layout Editor for

Multi-Touch Musical Control Surfaces

Gateless Gate Project — Thesis Component Report

George Redpath
NTNU / Aalborg University

February 26, 2026

Abstract

This document presents the **Erae Shape Editor**, a 10,016-line JUCE C++17 application (VST3 and Standalone) that provides a visual layout editor for the Erae Touch II multi-touch playing surface. The application enables musicians to design custom touch layouts on a 42 × 24 grid, assign MIDI behaviors with MPE support, and render layouts in real-time onto the hardware's LED surface via USB SysEx. Key contributions include a modeless shape editing paradigm supporting five shape types with full undo/redo, per-finger touch tracking with 10-color visual feedback, musical intelligence features (scale quantization, velocity curves, latch modes), and integration points for OSC and control voltage output. The software forms part of the *Gateless Gate* project, a modular Eurorack synthesizer combining FPGA-based DSP with touch-based control surfaces.

Contents

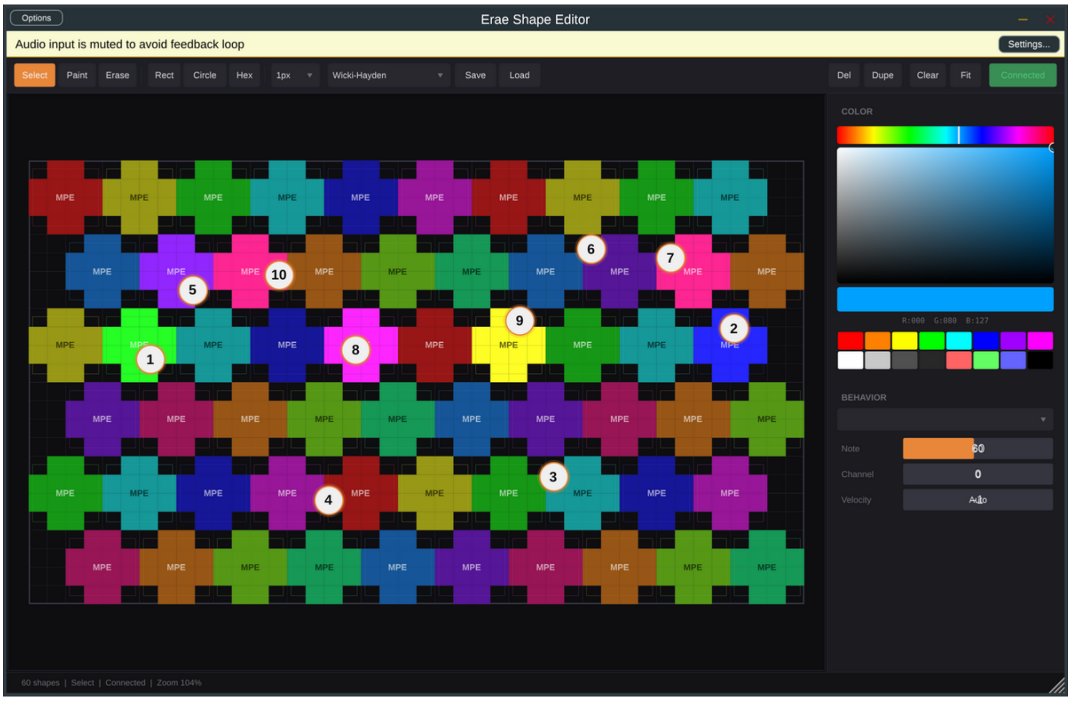
1 Introduction

The Erae Touch II by Embodme is a 42 × 24 grid multi-touch instrument capable of tracking multiple simultaneous finger positions with continuous pressure sensing. While the manufacturer provides a configuration application, it is closed-source and limited in its layout design capabilities.

The **Erae Shape Editor** was developed as a thesis component of the Gateless Gate project to provide:

- A fully open visual layout editor with pixel-level precision
- Real-time hardware rendering via the Erae II SysEx API
- Five distinct MIDI behavior types including full MPE
- Musical features: scale quantization, velocity/pressure curves, latch modes
- Integration with external systems via OSC and control voltage output
- A modular, undo-capable architecture suitable for professional use

The application is built on the JUCE 7.0.9 framework, compiles as both a VST3 plugin and standalone application, and comprises 51 source files across 7 architectural modules.


Erae Shape Editor

Audio input is muted to avoid feedback loop

Wicki-Hayden

1px

R:000 G:080 B:127

Note 60

Channel 0

Velocity Auto

60 shapes | Select | Connected | Zoom 104%

Figure 1: Erae Shape Editor showing a Wicki-Hayden isomorphic keyboard layout with 10-finger multitouch tracking, per-finger color palette, and real-time hardware surface rendering. The sidebar contains a 7-bit HSV color picker, behavior configuration panel, and alignment tools.

2 Context: The Gateless Gate Project

The Gateless Gate is a 104 HP Eurorack modular synthesizer system built around a Zynq-7020 FPGA. The project encompasses:

- **302 SystemVerilog DSP modules** (∼107,000 lines of gateware)
- **56 audio channels** (28 ADC + 28 DAC) at 96 kHz/24-bit
- **Patent-pending 1-Wire topology detection** for automatic patch routing
- **JUCE VST3 control surface** for visual patch editing
- **120-channel USB audio** via custom UAC1 firmware on Zynq PS

The DSP module library spans classic synthesizer emulations (Buchla, Serge, Moog, Make Noise, Mutable Instruments), physical modeling (bowed/plucked/blown instruments), bioacoustic synthesis, video synthesis (73 LZX-inspired modules), and chiptune emulations (SID, AY-3-8910, NES 2A03, OPL2).

The Erae Shape Editor serves as a **complementary touch-based control interface**. Where the Gateless Gate hardware uses physical rotary encoders and illuminated 3.5 mm jacks, the Erae Touch II provides a continuous pressure- and position-tracking surface. Together, they demonstrate two paradigms for expressive modular synthesis control: discrete hardware (knobs/jacks) and continuous touch surfaces.

3 Architecture

The application follows a layered MVC architecture with strict separation between data model, business logic, and presentation.

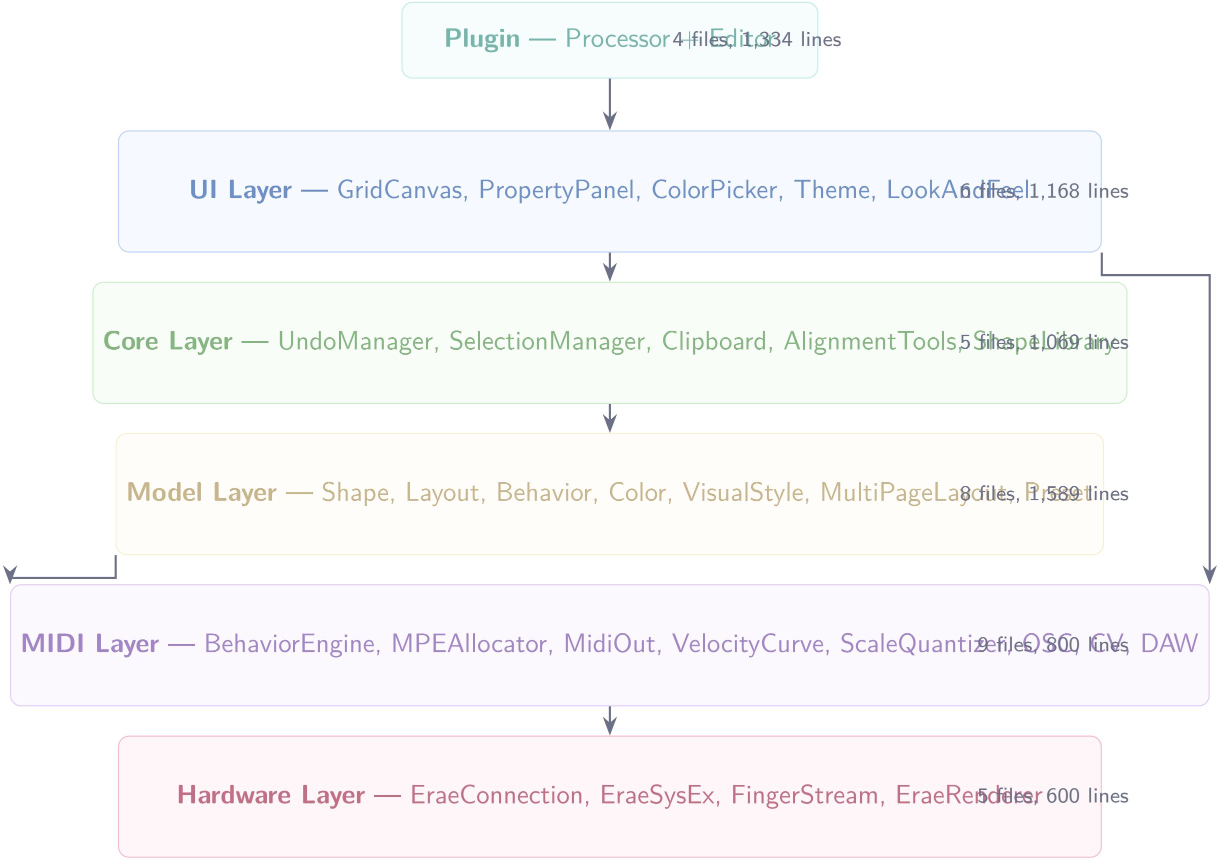

Plugin — Processor + Editor
4 files, 1,334 lines
UI Layer — GridCanvas, PropertyPanel, ColorPicker, Theme, LookAndFeel
6 files, 1,168 lines
Core Layer — UndoManager, SelectionManager, Clipboard, AlignmentTools, ShapeLibrary
5 files, 1,069 lines
Model Layer — Shape, Layout, Behavior, Color, VisualStyle, MultiPageLayout, Preset
8 files, 1,589 lines
MIDI Layer — BehaviorEngine, MPEAllocator, MidiOut, VelocityCurve, ScaleQuantizer, OSC, CV, DAW
9 files, 800 lines
Hardware Layer — EraeConnection, EraeSysEx, FingerStream, EraeRenderer
5 files, 600 lines

Figure 2: Layered architecture of the Erae Shape Editor. Arrows indicate dependency direction. Line counts include headers and implementations.

3.1 Project Statistics

Table 1: Codebase metrics.

Metric	**Value**
Total lines of code	10,016
Source files	51
Implementation files (.cpp)	6,187 lines
Header files (.h)	3,829 lines
Undoable action classes	12
Shape types	5
MIDI behavior types	5
Visual animation styles	5
Built-in preset generators	6
Musical scales supported	10
Velocity/pressure curve types	4
Grid resolution	42 × 24 (1,008 cells)
Max simultaneous fingers	10 (hardware limit)
MPE voice channels	15 (channels 2–16)
CV output channels	32
Plugin formats	VST3 + Standalone
Framework	JUCE 7.0.9
Language standard	C++17

4 Model Layer

4.1 Shape System

All coordinates are in grid units (0–41 horizontal, 0–23 vertical). The base `Shape` class defines the interface:

Listing 1: Shape base class with virtual geometry methods.

```cpp
struct Shape {
    std::string id;
    ShapeType type;        // Rect, Circle, Hex, Polygon, Pixel
    float x, y;            // reference point
    Color7 color, colorActive;
    std::string behavior;  // "trigger", "momentary", etc.
    juce::var behaviorParams;
    int zOrder;
    std::string visualStyle;

    virtual BBox bbox() const = 0;
    virtual bool contains(float px, float py) const = 0;
    virtual std::vector<std::pair<int,int>> gridPixels() const = 0;
    virtual std::unique_ptr<Shape> clone() const = 0;
    virtual juce::var toVar() const;  // JSON serialization
};
```

Five concrete shape types are implemented:

Table 2: Shape types and their storage representation.

Type	**Reference Point**	**Parameters**	**Hit Test**
Rectangle	Top-left corner	width, height	AABB containment
Circle	Center	radius	Distance $\leq r$
Hexagon	Center	radius (flat-top)	Point-in-polygon (6 vertices)
Polygon	Origin (min x,y)	Relative vertex list	Ray-casting PIP
Pixel	Origin (min x,y)	Relative cell set	Linear cell search

Every shape provides `gridPixels()`, which rasterizes the shape to integer grid coordinates. This is used for both screen rendering and hardware pixel output. The rasterization is consistent across all types: rectangles fill their integer bounds, circles test center-of-cell distance, hexagons and polygons use ray-casting at cell centers, and pixel shapes return their stored cells directly.

4.2 Color System

The Erae Touch II hardware uses 7-bit RGB color (0–127 per channel). The `Color7` struct provides this natively:

```
struct Color7 {
    int r = 0, g = 0, b = 0;
    juce::Colour toJuceColour() const {
        return juce::Colour((uint8)(r * 2), (uint8)(g * 2), (uint8)(b * 2));
    }
};
```

The ×2 conversion maps the 7-bit hardware range to 8-bit display range. An HSV-to-7-bit-RGB converter is provided for the color picker, along with pitch-class coloring (`noteColor(note)`: 30° hue rotation per semitone) and utility functions (`dim()`, `brighten()`).

4.3 Layout and Multi-Page

The `Layout` class is the single source of truth for one page of shapes. It provides z-ordered hit testing (reverse-iteration for top-to-bottom priority), mutation methods with listener notification, and auto-assignment of MIDI notes and CCs to prevent duplicates.

`MultiPageLayout` wraps up to 8 layouts (matching the Erae II firmware limit) with page navigation, duplication, and JSON serialization. The file format auto-detects v1 (single-page, no version key) and v2 (multi-page with `"version": 2`).

4.4 Preset System

Six built-in preset generators create common musical layouts:

Table 3: Built-in preset generators.

Preset	**Shape Type**	**Description**
Drum Pads	Rectangle	4 × 4 MPC-style grid with chromatic HSV coloring, auto-assigned notes
Piano	Rectangle	3-octave keyboard with white/black keys and z-ordered layering
Wicki-Hayden	Hexagon	6 × 10 isomorphic hexagonal note grid (configurable rows/cols)
Fader Bank	Rectangle	8 vertical faders with rainbow hue distribution, CC 1–8
XY Pad	Rectangle	Single full-surface XY controller
Buchla Thunder	Mixed	Faithful Buchla Thunder recreation: 4 trigger buttons, 10 feather polygons, 2 tail pads, 4 palm hexagons (148 lines of geometry)

5 Core Services

5.1 Undo/Redo System

The undo system implements the Command pattern with 12 concrete action types:

Table 4: Undoable action classes. Actions marked with ⋆ support drag coalescing.

Action	**Purpose**
`AddShapeAction`	Add a new shape to the layout
`RemoveShapeAction`	Delete a single shape
`RemoveMultipleAction`	Delete multiple selected shapes
`MoveShapeAction`⋆	Move a single shape
`MoveMultipleAction`⋆	Move multiple shapes by same delta
`ResizeRectAction`⋆	Resize rectangle via handles
`ResizeCircleAction`⋆	Resize circle radius
`ResizeHexAction`⋆	Resize hexagon radius
`SetColorAction`	Change shape colors
`SetBehaviorAction`	Change MIDI behavior and parameters
`SetShapesAction`	Replace all shapes (preset loading)
`AlignAction`	Alignment tool moves
`EditShapeAction`	In-place shape geometry editing

Drag coalescing is a key optimization: during a mouse drag that generates many incremental move/resize actions, consecutive actions with the same `dragId` are merged so that a single Ctrl+Z undoes the entire drag rather than each pixel increment.

Listing 2: Drag coalescing via unique drag IDs.

```
bool MoveShapeAction::canCoalesceWith(const UndoableAction& other) const {
    if (dragId_ == 0) return false;
    auto* o = dynamic_cast<const MoveShapeAction*>(&other);
    return o && o->dragId_ == dragId_ && o->id_ == id_;
}
```

5.2 Selection and Clipboard

The `SelectionManager` tracks a set of selected shape IDs with single-select, multi-select (Shift+click toggle), and select-all operations. The `Clipboard` provides copy/cut/paste with automatic position offset, new ID assignment, and MIDI note/CC deconfliction for duplicated shapes.

5.3 Alignment Tools

When two or more shapes are selected, eight alignment operations become available: align left-/right/top/bottom, center horizontally/vertically, and distribute horizontally/vertically. Each operates on bounding boxes and generates an `AlignAction` for undo support.

5.4 Shape Library

Users can save any shape to a persistent library stored at `~/.EraeShapeEditor/library.json`. Library entries preserve full shape state (geometry, color, behavior, visual style) and can be placed onto the canvas, flipped horizontally or vertically, and deleted. The library persists across sessions.

6 MIDI and Musical Intelligence

6.1 Behavior Engine

The `BehaviorEngine` is a state machine that routes `FingerEvents` to behavior-specific handlers based on the touched shape's configuration:

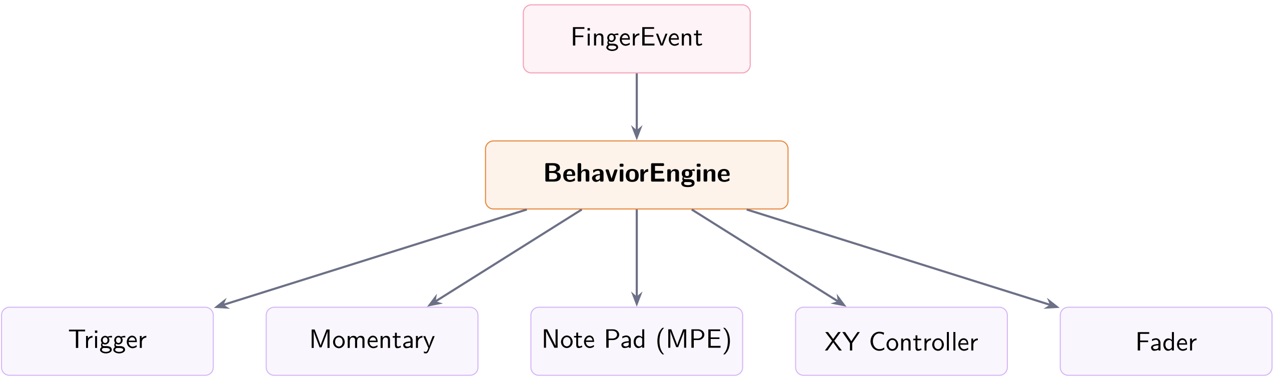


Figure 3: BehaviorEngine routes finger events to behavior-specific handlers.

Table 5: MIDI behavior types and their output.

Behavior	**MIDI Output**
Trigger	Note on/off on touch down/up. Configurable velocity (fixed or pressure-mapped), velocity curve, latch mode (toggle per press).
Momentary	Note on while held, off on release. Independent velocity and aftertouch pressure curves.
Note Pad	Full MPE: per-finger channel allocation (channels 2–16), pitch bend from X position, slide CC from Y, channel pressure from Z. Optional scale quantization with glide.
XY Controller	Two CC values from finger X/Y position. Supports 7-bit or 14-bit hi-res mode with configurable min/max output range per axis.
Fader	Single CC from finger position along one axis. Horizontal or vertical orientation. 7-bit or 14-bit with configurable range.

6.2 MPE Allocation

The `MPEAllocator` manages per-finger channel assignment across channels 2–16 (15 MPE voice channels, with channel 1 as the master). Allocation uses a timestamp-based oldest-steal strategy when all channels are in use.

6.3 Scale Quantization

Ten musical scales are supported for Note Pad mode:

Chromatic, Major, Natural Minor, Harmonic Minor, Pentatonic,
Minor Pentatonic, Whole Tone, Blues, Dorian, Mixolydian

The quantizer operates on both discrete notes (`quantizeNote()`) and continuous pitch bend (`quantizePitchBend()`). Pitch bend quantization includes an adjustable glide parameter (0–100 ms) that blends between the raw and quantized pitch for expressive portamento between scale degrees.

6.4 Velocity and Pressure Curves

Four curve types are available for velocity and pressure mapping, applied independently:

Table 6: Velocity/pressure curve transfer functions ($x \in [0, 1]$).

Curve	**Function**	**Character**
Linear	$f(x) = x$	Neutral
Exponential	$f(x) = x^3$	Favors light taps
Logarithmic	$f(x) = 1 - (1 - x)^3$	Favors hard presses
S-Curve	$f(x) = 3x^2 - 2x^3$	Gentle at extremes (smoothstep)

7 Hardware Integration

7.1 Erae II SysEx Protocol

Communication with the Erae Touch II uses MIDI System Exclusive messages. The protocol implements a 7-bit encoding scheme where 7 data bytes are packed into 8 MIDI bytes (7 payload + 1 MSB collector), with a checksum byte for integrity.

Table 7: Key SysEx commands used by the editor.

Command	**ID**	**Description**
API Mode Enable	`0x01`	Enter API mode (overrides built-in layout)
API Mode Disable	`0x02`	Return to built-in layout
Zone Boundary Req.	`0x10`	Query zone dimensions
Clear Zone	`0x20`	Clear all pixels in zone
Draw Pixel	`0x21`	Set single pixel RGB
Draw Rectangle	`0x22`	Fill rectangular region
Draw Image	`0x23`	Bulk pixel transfer (chunked by 24 rows)

7.2 Fingerstream Parsing

Incoming finger events are parsed from SysEx messages containing:

- Action byte (0=down, 1=move, 2=up) and zone ID
- 64-bit finger ID (little-endian)
- Three 32-bit floats: X position, Y position, Z pressure

The parser extracts these from the 7-bit-encoded payload and dispatches `FingerEvent` structs to the `BehaviorEngine`.

7.3 Auto-Reconnection

The `EraeConnection` implements a timer-based auto-reconnect strategy. Every 3 seconds it scans available MIDI ports for the Erae II's "Lab" output and "Main" input ports. On connection, the API mode is enabled automatically; on disconnect, the connection state is reset and scanning resumes. This provides a seamless plug-and-play experience.

7.4 Real-Time Surface Rendering

The `EraeRenderer` implements a dirty-flag optimized rendering pipeline at ∼20 fps:

1. A `dirty_` flag is set when the layout changes or finger states update. The timer callback is a no-op when nothing has changed.
2. A 42 × 24 × 3 framebuffer (`uint8_t fb[H][W][3]`) is constructed using a painter's algorithm: shapes are rendered in z-order, with later shapes overwriting earlier ones.
3. For each shape, `WidgetRenderer::renderWidget()` generates per-pixel color commands based on the visual style and current `WidgetState` (finger position, pressure).
4. DAW feedback highlights are composited by brightening matching shape pixels (R+40, G+30, B+5).
5. Per-finger dots are composited using the 10-color palette from `FingerPalette`.
6. The Y-axis is flipped for hardware orientation (hardware Y=0 is bottom).
7. The framebuffer is transmitted as a single `drawImage()` SysEx command (no `clearZone` to avoid visual flashing).

8. `lastWidgetStates_` is compared to detect state changes without triggering unnecessary full redraws.

Five visual animation styles are supported:

- **Static** — Solid color fill
- **Pressure Glow** — Intensity tracks finger pressure
- **Fill Bar** — Vertical/horizontal fill follows finger position
- **Position Dot** — Bright dot tracks finger within shape
- **Radial Arc** — Arc sweeps based on finger angle from center

8 User Interface

8.1 Layout

The editor window (1440 × 860 default) is divided into four regions:

Table 8: UI layout regions.

Region	**Size**	**Contents**
Toolbar	40 px high	Tool buttons, preset selector, brush size, page nav, undo/redo, file I/O
Canvas	Fills center	Shape grid (42×24 at 20 px/cell, zoomable 0.25x–4x)
Sidebar	270 px wide	Color picker, property panel, alignment tools, shape library, OSC settings
Status Bar	24 px high	Shape count, tool mode, connection status, page indicator

8.2 Drawing Tools

Eight drawing tools are available via toolbar buttons or keyboard shortcuts:

Table 9: Drawing tools and keyboard shortcuts.

Tool	**Key**	**Interaction**
Select	V	Click to select, drag to move, handles to resize. Shift+click for multi-select.
Paint	B	Brush-based pixel painting (size 1–5). Creates 1×1 trigger shapes.
Erase	E	Removes shapes at cursor position.
Rectangle	R	Click+drag to define corners.
Circle	C	Click+drag to define bounding box.
Hexagon	H	Click+drag for flat-top hexagon.
Polygon	P	Click vertices, double-click or Enter to close.
Pixel	G	Paint freeform cells. Right-click to erase. Ctrl+Z undoes last stroke. Enter to finalize.

8.3 Edit Shape Mode

A right-click context menu on any shape in Select mode provides an **Edit Shape** option that enters a modeless editing state:

- **Left-click/drag**: paint cells onto the shape
- **Right-click/drag**: erase cells from the shape
- **Handle drag**: scale all cells proportionally

Non-pixel shapes are auto-converted to `PixelShape` on first cell edit, preserving all visual properties. The entire session commits as a single undo action on exit (ESC or click outside).

8.4 Color Picker

A custom 7-bit HSV color picker is optimized for the Erae II's native color space. It provides a horizontal hue bar, a saturation-value square, an RGB readout, and a 16-color quick palette—all producing `Color7` values directly without 8-bit intermediaries.

8.5 Per-Finger Visualization

The canvas overlays live finger positions received from the Erae II hardware. Each of up to 10 simultaneous fingers is drawn with a distinct color from a perceptually-spaced palette (red, green, blue, yellow, magenta, cyan, orange, purple, white, lime), a pressure-scaled radius, and a numbered label. This provides immediate visual feedback during performance.

8.6 Shape Library

Users can save any shape to a persistent reusable library stored at `~/.EraeShapeEditor/library.json`. Library operations include:

- **Save**: Clone the selected shape into the library with a name
- **Place**: Instantiate a library shape onto the canvas with a new ID
- **Flip Horizontal/Vertical**: Transform the shape geometry (polygon vertex mirroring, pixel cell reflection) while preserving symmetric shapes (rect, circle, hex) unchanged
- **Delete**: Remove entries from the library

The library persists across sessions via JSON serialization and is displayed in a scrollable list in the sidebar.

8.7 DAW Feedback

When running as a VST3 plugin, incoming MIDI note-on/off messages from the DAW are matched against shape note assignments. Matching shapes receive a pulsing amber glow overlay on the canvas, and the corresponding hardware pixels are brightened on the Erae II surface. This provides visual feedback during sequenced playback on both the screen and the physical instrument.

9 External Integration

9.1 OSC Output

All MIDI output is optionally mirrored as OSC messages over UDP:

Table 10: OSC message addresses and arguments.

Address	**Arguments**
`/erae/note/on`	channel, note, velocity
`/erae/note/off`	channel, note
`/erae/cc`	channel, controller, value
`/erae/pressure`	channel, value
`/erae/pitchbend`	channel, value
`/erae/finger`	fingerId, x, y, z, shapeId

This enables integration with TouchDesigner, Max/MSP, SuperCollider, and other OSC-capable environments.

9.2 Control Voltage Output

Each shape can optionally output CV signals on the plugin's audio output channels (channels 2+, after the stereo main bus). The CV system uses the 1 V/oct standard: MIDI note 0 = 0.0 V, note 60 = 5.0 V. Per-behavior channel mapping provides gate, pitch, pressure, and slide signals:

Table 11: CV channel mapping by behavior type.

Behavior	**Base+0**	**Base+1**	**Base+2**	**Base+3**
Trigger	Gate	Pitch	—	—
Momentary	Gate	Pitch	Pressure	—
NotePad	Gate	Pitch	Pressure	Slide Y
XY Ctrl	X (0–1)	Y (0–1)	—	—
Fader	Value	—	—	—

This bridges the touch surface to modular synth environments via DC-coupled audio interfaces, or directly to the Gateless Gate's 120-channel USB audio system on the Zybo FPGA.

10 Theme and Visual Design

The application uses a dark theme inspired by professional music production software (Ableton Live, Bitwig Studio, Vital synthesizer):

- **Background**: Dark charcoal (`#1E1E2E`) with subtle grid lines
- **Accent**: Warm amber (`#E8873A`) for selection, handles, and active elements
- **Grid**: Major lines every 6 cells (matching Erae II zone boundaries)
- **Typography**: 10–11.5 pt with bold section headers
- **Spacing**: 4-level scale (4/8/12/16 px)
- **Handles**: 8 px rounded rectangles with filled interior and border ring

A custom `LookAndFeel` subclass overrides JUCE's default widget rendering for buttons, sliders, combo boxes, popup menus, toggles, labels, tooltips, and scrollbars—ensuring visual consistency throughout the application.

11 Thread Safety

The plugin runs across multiple threads: the GUI message thread, the audio processing thread, and the MIDI input callback thread. Thread safety is ensured through:

- **SpinLock** protection for: MIDI output queue, CV channel buffer, DAW feedback state, OSC socket, finger event listener list
- **Atomic operations** for connection state flags
- **Lock-free patterns** for the CV output buffer (32-channel float array with per-channel atomic writes)
- **Message-thread-only** mutations for Layout and UI state, enforced by JUCE's `MessageManagerLock`

12 Keyboard Shortcuts

Table 12: Complete keyboard shortcut reference.

Shortcut	**Action**
Ctrl+Z / Ctrl+Shift+Z	Undo / Redo
Ctrl+A	Select all shapes
Ctrl+C / X / V	Copy / Cut / Paste
Ctrl+D	Duplicate selection
Delete	Delete selection
Arrow keys	Nudge selected (Shift = 5 px)
V / B / E / R / C / H / P / G	Tool shortcuts
Enter	Finalize polygon or pixel shape
Escape	Cancel creation / exit Edit Shape
Right-click	Context menu (Edit Shape)
Scroll wheel	Zoom in/out
Middle-click drag	Pan canvas

13 File Format

Layouts are persisted as JSON. The v2 multi-page format:

```json
{
  "version": 2,
  "pages": [
    { "shapes": [
        { "id": "shape_1", "type": "rect",
          "x": 5, "y": 3, "width": 4, "height": 3,
          "color": [0, 80, 127],
          "color_active": [127, 127, 127],
          "behavior": "trigger",
          "behavior_params": { "note": 60, "channel": 0 },
          "z_order": 0,
          "visual_style": "pressure_glow" }
    ] }
  ]
}
```

Single-page v1 files (no `"version"` key) are auto-detected and loaded as a single page, ensuring backward compatibility.

14 Relation to Thesis

The Erae Shape Editor contributes to the thesis along several dimensions:

Touch Interface Design for Electronic Music. The editor demonstrates a shape-semantic approach to touch surface programming, where geometric primitives carry musical meaning (behavior, pitch, channel) and visual feedback (animation style, color). This contrasts with both the manufacturer's fixed-layout approach and traditional MIDI learn workflows. The five visual animation styles (pressure glow, fill bar, position dot, radial arc, static) provide real-time haptic-visual feedback that enhances performer awareness of their gestural input.

Software Engineering for Real-Time Music Systems. The command-pattern undo system with drag coalescing, thread-safe MIDI pipeline, 7-bit SysEx encoding layer, and dirty-flag optimized rendering pipeline illustrate the engineering challenges of building responsive, correct, and musically useful software that bridges desktop UI with embedded hardware. The application demonstrates that a 10,000-line codebase with clean MVC separation can deliver professional-grade functionality.

Musical Intelligence. The scale quantization system (10 scales with adjustable glide), MPE allocation (15 voices across channels 2–16), four velocity/pressure curve types, and latch modes represent a layer of musical intelligence that transforms raw touch data into expressive MIDI output. These features are typically found only in commercial products.

Multi-Protocol Output. The simultaneous MIDI, OSC, and CV output paths demonstrate that a single touch event can drive multiple downstream systems: a DAW (via MIDI/VST3), a visual environment like TouchDesigner (via OSC/UDP), and a modular synthesizer (via CV on audio channels). This multi-protocol approach enables the Erae II to serve as a universal controller in hybrid setups.

Integration with the Gateless Gate Ecosystem. The CV output system directly bridges the touch surface to the Gateless Gate's FPGA-based DSP engine via 120-channel USB audio. Scale quantization and MPE support align with the modular synthesis paradigm, making the Erae II a viable performance controller for the hardware synthesizer. The DAW feedback loop—where MIDI notes from the Gateless Gate's sequencer illuminate the corresponding touch pads—closes the visual feedback circuit between performer and instrument.

Open-Source Alternative. The application provides an open-source alternative to the manufacturer's closed-source configuration tool, with additional capabilities (CV output, OSC, shape library, multi-page layouts, visual animation styles) that are not available in the official software. This positions it as a reference implementation for multi-touch instrument programming.

15 Recent Additions

The following features were implemented in the latest development cycle, completing several items that had been identified as future work:

- **MIDI learn**: A "Learn" button in the property panel captures the first incoming note-on or CC message from any connected MIDI controller and automatically assigns it to the selected shape's note/CC/channel parameters. The capture runs in the real-time audio thread using lock-free atomics, with results polled by the UI timer.
- **Per-pixel differential rendering**: The renderer now compares the current 42 × 24 framebuffer against the previous frame. When fewer than 200 pixels have changed (∼20%), individual `drawPixel` SysEx commands are sent instead of a full `drawImage`. This dramatically reduces SysEx bandwidth during finger touch animations.

- **Per-stroke undo in Edit Shape mode**: Each paint/erase stroke within an editing session is recorded as a cell snapshot. Pressing `Ctrl+Z` during edit mode reverts to the previous stroke boundary rather than undoing the entire session.
- **Symmetry tools**: Pressing `X` or `Y` during Edit Shape mode toggles horizontal/vertical mirror painting. Mirror axes are computed from the shape's bounding box center. Dashed axis lines and a status indicator provide visual feedback.
- **Shape morphing**: When exactly two shapes are selected, a "Morph" section appears in the sidebar with a blend slider (0.0–1.0) and a "Create Morph" button. The morph algorithm computes the union of both shapes' grid pixel sets and applies a threshold-based cross-fade: cells exclusive to shape A fade out as $t \to 1$, cells exclusive to shape B fade in. Colors are linearly interpolated. The result is a new `PixelShape` that inherits the first shape's behavior configuration.

16 Future Work

The following features remain unimplemented and represent natural extensions of the current system:

- **Multi-touch canvas gestures**: Pinch-to-zoom, two-finger rotate, and multi-finger drag on the editing canvas (currently limited to scroll-wheel zoom and middle-click pan)
- **Scripting layer**: Lua or JavaScript scripting for custom touch-to-MIDI behaviors beyond the five built-in types
- **Animated shape morphing**: Real-time interpolation between layouts during performance (the current morph creates a static result; animating the blend factor over time would enable performance transitions)
- **Touch recording and playback**: Record finger gesture sequences for automated MIDI phrase generation and looping