Design Document: Maurer Rose Interpolator v15.9

Author: Gemini

Date: August 7, 2025

Version: 1.6

1. Introduction & Vision

1.1. Intent

The **Maurer Rose Interpolator** is an interactive web-based application designed for the procedural generation and artistic exploration of geometric forms known as Maurer Roses. The primary goal is to provide a tool that is both a mathematical instrument and a digital artist's canvas. It allows users to intuitively manipulate the parameters that define these intricate figures and, most importantly, to create fluid, aesthetically pleasing animations by interpolating between two distinct roses.

The application is built to be highly interactive, providing real-time visual feedback for every parameter change. It serves as a platform for creative coding, mathematical visualization, and the generation of unique animated content.

1.2. Core Concepts

The application is founded on the mathematical principles of rhodonea curves and their extension into Maurer Roses, as described in the papers by Peter M. Maurer and Gregg Helt.

- **Rhodonea Curves:** The base "flower" shapes are defined by the polar equation r = sin(k * θ). This application extends the formula to r = c + A * sin((n/d) * θ), allowing for fractional values of k (as n/d) and providing controls for an offset (c) and amplitude (A).
- **Maurer Rose Construction:** Instead of a continuous curve, a Maurer Rose is a polyline created by connecting points on the rhodonea curve at discrete angular steps. The application implements this by defining a total number of divisions for the curve's full period (Z) and a stepping value (D).
- **Group Theory & Cosets:** The application correctly implements the group theory principles outlined by Maurer. The number of lines required for a rose to close is calculated as Z / gcd(D, Z), and the number of parallel patterns, or cosets, is gcd(D, Z). This mathematical foundation ensures predictable and structured pattern generation.

- **Interpolation:** The central feature is the ability to animate a smooth transition between two user-defined roses (Rose A and Rose B). This is achieved through several distinct mathematical strategies, from simple parameter blending to precise, point-by-point resampling based on the Least Common Multiple (LCM) of the line counts.

2. UI and Feature Breakdown

The user interface is organized into three primary columns: **Rose A**, **Interpolation**, and **Rose B**. This layout allows for the independent configuration of the two source roses and a central panel for controlling their interaction.

2.1. Rose Panels (A & B)

These panels are identical and provide a comprehensive set of controls for defining a single Maurer Rose.

- **Live Preview Canvas:** A dedicated canvas provides immediate visual feedback as parameters are adjusted.
- **Core Parameters:**
  - n, d: Define the fundamental shape of the underlying rhodonea curve.
  - c, A: Control the offset and amplitude of the curve.
  - Rotation: Rotates the final figure.
- **Sync Controls:** A "link" icon (🔗) next to each core parameter allows its value to be synchronized with the corresponding control in the other panel. When active (indicated by a green color), changing one control updates both, facilitating the creation of symmetrical transformations.
- **Divisions & Stepping:**
  - Use Custom Divisions: Toggles between a default mode where divisions are based on the rhodonea's period and a custom mode.
  - Total Divisions: Sets the total number of divisions (Z) for the curve's period.
  - Degrees/Divisions: Sets the angular step size (D).
  - **Prime Factorization Display:** A text label provides an instant mathematical analysis of the D value, displaying "Prime" or its prime factors (e.g., "2 x 3 x 5").
- **Styling & Display:**
  - Controls for line thickness and opacity.
  - Toggles to show/hide the base rhodonea curve, the Maurer points, and the start/end points of the walk.
- **Cosets:**
  - Show All Cosets: Renders all possible parallel patterns based on the gcd(D, Z).

    - Number & Selection: Allows for manual selection of a subset of cosets and the method used to choose them (e.g., sequentially, distributed).
    - Color by Range: Applies a color gradient across the rendered cosets.
- **Generative Features ("Matches"):**
    - **Match Type:** A dropdown menu to select the algorithm for finding new degrees values.
        - Lines to Closure: Finds other degrees values that produce the same number of lines.
        - Prime: Finds prime numbers.
        - Relatively Prime: Finds numbers that are coprime to the target.
        - Relatively Prime, not Prime: Finds coprime numbers that are not themselves prime.
        - Twin Primes: Finds pairs of primes (p, p+2).
        - Cousin Primes: Finds pairs of primes (p, p+4).
        - Hexy Primes: Finds pairs of primes (p, p+6).
    - **Relative To:** Determines the source for the target value used in the matching algorithm ("This Rose" or "Other Rose").
    - **Navigation Buttons (<, Rnd, >):** Apply the selected matching algorithm to find and set new degrees values.
- **Individual Segment Coloring:** An artistic feature that colors each segment of the rose based on a chosen property. This section is only visible when "Stroke Segments Individually" is checked.
    - **Methods:** Includes coloring by Length, Distance to Origin, Angle, Min Distance to Center, and the more advanced Angle (Quadrant) and Angle (Octant) methods, which create oscillating color patterns.
    - **Color Range:** Two color pickers define the gradient used for these methods.
- **Irrational Rose Controls:** These options introduce quasi-randomness by adding Euler's number (e) to the core parameters, creating complex, non-repeating patterns.
    - **Add e to n/d/deg:** Checkboxes to add e to the respective parameters.
    - **Cycles Multiplier:** A slider that appears when an irrational option is selected. It multiplies the number of lines drawn, allowing for the creation of longer and more intricate non-repeating patterns.

2.2. Interpolation Panel (Expanded)

This central panel is the animation and rendering hub.

- **Main Animation Canvas:** Displays the final interpolated animation.
- **Animation & Recording Controls:**
    - Play/Pause, Speed, and Ease In/Out controls for the animation timeline.

    - Format and Bitrate selectors for video recording.
    - Start/Stop Recording and Download Recording buttons.
- **Relative Combos (Generative Animation):** This is a powerful feature for creating ever-changing, generative animations that evolve over time.
    - **Functionality:** When the "Relative Combos" checkbox is enabled, the animation doesn't just loop between a static Rose A and Rose B. Instead, at the end of each animation cycle (when the slider reaches 0% or 100%), the application automatically selects a new "Match" for one or both roses, based on the "Match Type" and "Picker" settings in the side panels.
    - **RoseA/B Picker:** These dropdowns control how the next "Match" is chosen for each respective rose at the end of a cycle.
        - Static: The rose's parameters will not change.
        - Next: Selects the next match from the pre-calculated list in sequential order.
        - Previous: Selects the previous match from the list.
        - Random: Selects a random match from the list.
    - **Use Case:** By setting a "Match Type" (e.g., "Lines to Closure") in the side panels and then enabling "Relative Combos" with the pickers set to "Random", you can create a continuous animation that morphs between an endless variety of structurally similar but visually distinct roses.
- **Interpolation Slider & 50/50 Button:** The primary slider controls the morphing weight between Rose A and Rose B. The "50/50" button provides a shortcut to center the slider for a perfect mix.
- **Interpolation Methods (Expanded):** A dropdown allows the user to select the mathematical strategy for the morph, each with different visual characteristics.
    - **Exact (LCM) Methods:** These are the most mathematically precise methods. They calculate the Least Common Multiple (LCM) of the line counts of Rose A and Rose B. Both roses are then upsampled to this new, common line count before interpolation. This ensures a perfect, one-to-one vertex correspondence, resulting in a very smooth and coherent morph.
        - **LCM - Curved**: Upsamples by recalculating points along the original rhodonea curve. This is the most mathematically pure approach.
        - **LCM - Linear**: Upsamples by linearly interpolating between the existing vertices of the original rose. This is faster and often visually indistinguishable from the curved method.
        - **LCM - Linear + Cosets**: Extends the LCM method to work with multiple cosets, providing precise morphing for complex, multi-layered patterns. This method allows for selecting a subset of the total available cosets and choosing how they are selected (sequentially, distributed, etc.).

    - **Joined Cosets**: This method treats all selected cosets of a rose as a single, continuous path. It then performs an LCM resampling on the total number of segments in these joined paths. This is useful for creating morphs between figures with different numbers of cosets, as it creates a unified path for each rose before interpolation.
    - **Use Cosets -- Merge**: This method takes the required number of cosets from each rose to reach the LCM and merges their point data into a single, continuous path before interpolating. This can create interesting sweeping and unwinding effects.
    - **Resample (Approx)**: A fast, approximate method that serves as a fallback when an LCM-based approach is not feasible (e.g., the LCM is too large). It resamples both roses to a fixed, high number of points (e.g., 720) and then interpolates between them. The morph is less mathematically "correct" but visually smooth for most cases.
    - **Parameter Morph**: This method takes a completely different approach. Instead of morphing the vertex data of the two roses, it directly interpolates the *parameters* (n, d, A, c, rot, etc.) that define them. At each frame of the animation, it calculates a new, intermediate rose based on these blended parameters. This creates a "breathing" or "evolving" effect, where the entire structure of the rose transforms, rather than just its points.
    - **Use Exact Match (Override):** This toggle is a special override feature.
        - **Availability:** This option is only enabled when **Rose A and Rose B have the exact same "Lines to close" value**. This is a crucial condition because it means both figures are constructed from the same number of line segments, allowing for a perfect, one-to-one mapping between their vertices.
        - **Functionality (When On):** When this toggle is checked and the line counts match, it **overrides whatever interpolation method is selected in the dropdown menu**. Instead of using LCM, Resampling, or Parameter Morphing, the application performs a direct, linear interpolation between the corresponding points of the two roses (e.g., Vertex #1 of Rose A morphs directly to Vertex #1 of Rose B). This results in the smoothest and most mathematically "pure" morph possible.
        - **Functionality (When Off/Disabled):** If the toggle is unchecked, or if it's disabled because the line counts do not match, the application falls back to using the interpolation method currently selected in the main dropdown menu.
- **Maurer Render Styles:** This dropdown controls how the segments connecting the Maurer points are rendered.

    - **Straight Lines**: The default and classic method, connecting each point with a straight line.
    - **Semicircular Arcs**: Connects each pair of points with a semicircular arc drawn on one side of the direct line between them.
    - **Semicircular Flipped**: Draws the same semicircular arc but on the opposite side.
    - **Full Circles**: Draws a full circle centered at the midpoint of the two points, with a radius equal to half the distance between them.
- **Coloring by Line Segment:** An artistic feature that colors each segment of the interpolated rose based on a chosen property.
    - **Methods:** Includes coloring by Length, Distance to Origin, Angle, Min Distance to Center, and the more advanced Angle (Quadrant) and Angle (Octant) methods, which create oscillating color patterns.
    - **Color Range:** Two color pickers define the gradient used for these methods.
- **Visual Effects & Display Options:**
    - **Show Percentages**: Toggles the visibility of the A/B percentage display below the interpolation slider.
    - **Auto-Scale**: When enabled, automatically scales the interpolated rose to fit the canvas, which is useful when morphing between roses of very different sizes.
    - **Scale Thickness**: Determines whether the line thickness should scale proportionally with the auto-scaling feature.
    - **Auto-Brightness**: Automatically adjusts the brightness of the interpolated rose based on the number of lines being drawn, helping to maintain a consistent visual intensity.
    - **Trails**: Enables a motion blur effect by leaving a fading history of previous frames, with controls for the decay rate and blend mode.

3. Implementation Architecture (Expanded)

The application is a self-contained single-page web application written in HTML, CSS, and modern JavaScript, designed for high interactivity and real-time feedback.

- **Structure:** The layout is defined with standard HTML elements (, <canvas>, <input>, etc.) and styled exclusively with the **Tailwind CSS** utility-first framework. This allows for rapid development of a responsive, modern interface without custom CSS files. The UI is divided into three main containers representing the two control panels and the central interpolation panel, which are arranged in a row on larger screens and stack vertically on smaller screens using Tailwind's responsive breakpoints.

- **Core Logic (<script type="module">):** All application logic resides within a single script module to ensure simplicity and portability.
    - **State Management:** The application's state is managed through two primary mechanisms, avoiding the complexity of a formal state management library:
        1. **DOM as Source of Truth:** The current values of all sliders, checkboxes, and select menus are considered the primary "source of truth" for user-configurable parameters.
        2. **JavaScript State Variables:** A collection of top-level variables (paramsA, paramsB, syncState, isPlaying, isRecording, etc.) hold the processed state derived from the DOM, as well as the application's runtime status (e.g., whether the animation is playing). This keeps the logic centralized and easy to trace.
    - **Event-Driven update() Loop:** The application's core is an event-driven architecture centered on the update() function.
        - **Event Listeners:** Listeners are attached to all UI controls (input for sliders, change for selects/checkboxes). Any user interaction triggers the main update() function. A special handler, handleRelativeChange(), is used for the "Matches" buttons to prevent jarring visual shifts by temporarily pausing the animation during the parameter update.
        - **The update() Function:** This monolithic function acts as the central controller. On every call, it executes a precise sequence:
            1. **Read Panels:** It calls readPanelParams() for both Rose A and B. This sub-function reads all related DOM controls, performs necessary type conversions (e.g., parseInt, parseFloat), calculates dependent values like the rhodonea period, and returns a complete parameter object for that rose.
            2. **Handle Syncing:** If any parameters are synced (tracked in the syncState object), it ensures the values are propagated between paramsA and paramsB.
            3. **Calculate Dependencies:** It computes values that depend on both panels, such as the Least Common Multiple (LCM) of the line counts, which is crucial for determining which interpolation methods are available.
            4. **Generate Point Data:** Based on the selected interpolation method, it calls the appropriate point generation function (e.g., generateLcm2Points, resample, generateRosePoints for parameter morphing).
            5. **Render Visuals:** It calls the drawPath and drawPoints functions for all three canvases, passing the newly generated point data and styling

parameters.

- **Rendering Pipeline (drawPath, drawPoints):** These are pure rendering functions that take point data and styling information and draw to the 2D canvas context.
  - They handle the coordinate system transformation from the mathematical model (centered at 0,0 with a scale of ~4 units) to the canvas coordinates (centered at width/2, height/2 with a pixel-based scale).
  - The drawPath function contains the branching logic (if/else if) for the various **Color by Line Segment** methods. For performance, it calculates the color value for each segment *just-in-time* during the rendering loop.
- **Animation (requestAnimationFrame):** The animation is driven by a standard animate() loop for browser-optimized performance.
  - When isPlaying is true, animate() is called via requestAnimationFrame.
  - It calculates the next animationPhase based on the speedSlider value and applies an easing function if toggled.
  - Crucially, it only updates the interp slider's value and then calls the main update() function. This ensures that the entire application state is recalculated consistently for each frame, preventing desynchronization.
- **Recording (MediaRecorder API):** Video recording leverages the browser's native capabilities.
  - A MediaRecorder instance is created using a stream captured directly from the interpolation canvas (canvasInterpolated.captureStream(30)), which provides a 30 FPS video stream.
  - The user-selected mimeType (e.g., 'video/mp4') and videoBitsPerSecond are passed directly into the MediaRecorder constructor options, instructing the browser on how to encode the video. This avoids the need for client-side transcoding libraries.
  - Recorded data is collected in the recordedChunks array and compiled into a Blob when recording stops. This Blob is then converted to an object URL for download via a temporary <a> tag.