Specification: Unified 2D/3D Plot Component System

Overview: This document outlines the design of a modular, extensible plotting system capable of rendering real-time 2D and 3D data visualizations using OpenGL. The system will support waveform displays, spectrum analyzers (including 3D spectrum cubes), and statistical data plots (e.g., histograms, distributions). The architecture emphasizes shared logic, performance, and scalability.

Goals:

* Provide a unified API for rendering both 2D and 3D data.
* Enable real-time responsiveness with GPU acceleration (OpenGL).
* Support dynamic zooming, markers, labeling, and theming.
* Remain agnostic to data semantics (e.g., X = time, X = frequency, etc.).
* Allow visualization of live, high-throughput data (e.g. audio buffers).

Component Breakdown:

1. PlotComponentBase (Abstract Base Class)
   * Responsibilities:
     + Define shared interface for 2D and 3D plot components.
     + Manage axis range, labels, markers, title, grid visibility.
     + Provide infrastructure for theme-aware rendering.
   * Key Methods:
     + setXRange(float min, float max)
     + setYRange(float min, float max)
     + setZRange(float min, float max) [3D only]
     + setAutoFitX(bool)
     + setAutoFitY(bool)
     + setTitle(const juce::String&)
     + setAxisLabels(const juce::String& x, const juce::String& y, const juce::String& z = {})
     + setShowGrid(bool)
     + setPadding(float percent)
     + setLineColor(juce::Colour)
     + setMarkerColor(juce::Colour)
     + clearMarkers()
2. Plot2DComponent (Derived Class)
   * Uses OpenGL or JUCE Graphics for 2D rendering.
   * Draws connected XY data points, markers, grid lines, and labels.
   * Supports:
     + setXYData(std::vector x, std::vector y)
     + setHorizontalMarker(int id, float y, const juce::String& label)
     + setVerticalMarker(int id, float x, const juce::String& label)
     + setZoom(float xZoom, float yZoom)
     + setLineThickness(float)
3. Plot3DComponent (Derived Class)
   * Uses OpenGL 3D pipeline.
   * Supports camera manipulation and interactive orbit/zoom.
   * Draws X/Y/Z data as point clouds, surfaces, or wireframes.
   * Supports:
     + setXYZData(std::vector x, std::vector y, std::vector z)
     + setCameraPosition(glm::vec3)
     + setLookAt(glm::vec3 target)
     + setProjectionType(Orthographic | Perspective)
     + rotateTo(float azimuth, float elevation)
4. Data Buffer Management
   * Must support real-time streaming:
     + Efficient updates with minimal redraw overhead.
     + VBO-backed buffer transfer for GPU acceleration.
     + Optional circular buffer mode.
   * Optional shared utility:
     + DataBuffer2D / DataBuffer3D abstraction to encapsulate data structure and lifecycle.
5. Theming and Style
   * Integrate with ThemeManager and ThemeColorId:
     + Grid colors
     + Marker colors
     + Active/inactive waveform lines
     + Background
6. Extensibility
   * Future plots may support:
     + Multiple datasets per view (e.g. overlays)
     + Annotated zones (highlighted ranges)
     + Export-to-image (PNG/SVG)
     + Mouse interaction (click/drag markers, tooltips)
7. Usage Expectations
   * External components control all interaction (no internal buttons/sliders).
   * API-driven configuration ensures reusability across contexts (oscilloscope, spectrum, analysis).

Next Steps:

* Draft header for PlotComponentBase with all shared methods.
* Begin OpenGL-backed implementation for Plot2DComponent.
* Create minimal DataBuffer2D abstraction for test feeding.