Afferent Roadmap

This document tracks improvement opportunities, feature proposals, and code cleanup tasks for the Afferent graphics framework.

Feature Proposals

[Priority: High] Pattern and Image Fills

Description: Add support for pattern/texture fills in addition to solid colors and gradients. The FillStyle enum already has a commented placeholder for `pattern (p : Pattern)`.

Rationale: Pattern fills are essential for many graphics applications (tiled backgrounds, hatching, textures). The infrastructure exists but the feature is not implemented.

Affected Files: - `Afferent/Core/Paint.lean` (FillStyle enum) - `Afferent/Render/Tessellation.lean` (sampleFillStyle, vertex UV generation) - `native/src/metal/` (shader support for texture sampling in 2D pipeline)

Estimated Effort: Medium

Dependencies: Requires UV coordinate generation in tessellation and texture binding in the 2D rendering pipeline.

[Priority: High] Round Line Caps and Joins

Description: Implement proper round line caps and line joins for stroke rendering. Currently `LineCap.round` and `LineJoin.round` fall back to butt caps and miter joins respectively.

Rationale: Round caps and joins are commonly needed for smooth graphics and are part of the standard Canvas API.

Affected Files: - `Afferent/Render/Tessellation.lean` (expandPolylineToStroke function, lines 529, 555-558, 603-609)

Estimated Effort: Medium

Dependencies: None. Requires generating arc geometry for round elements.

[Priority: High] PBR Material Support for 3D

Description: Extend the 3D asset loading pipeline to support full PBR (Physically Based Rendering) materials including normal maps, metallic, and roughness textures. The LoadedAsset structure notes this as a future enhancement.

Rationale: Modern 3D content uses PBR workflows. The current system only loads diffuse textures.

Affected Files: - `Afferent/FFI/Asset.lean` (SubMesh structure, loadAsset function - lines 44-47) - `native/src/metal/` (shader updates for PBR)

Estimated Effort: Large

Dependencies: Shader modifications, additional texture slots.

[Priority: Medium] Dashed and Dotted Lines

Description: Add support for dashed and dotted line patterns in StrokeStyle.

Rationale: Dashed lines are a common requirement for charts, borders, selection indicators, and technical drawings.

Affected Files: - `Afferent/Core/Paint.lean` (StrokeStyle structure) - `Afferent/Render/Tessellation.lean` (stroke tessellation)

Estimated Effort: Medium

Dependencies: None.

[Priority: Medium] Shadow and Glow Effects

Description: Add shadow/glow capabilities to the Canvas API, similar to HTML5 Canvas shadowBlur, shadowColor, shadowOffsetX/Y.

Rationale: Shadows and glows are essential for modern UI design, depth perception, and visual effects.

Affected Files: - `Afferent/Canvas/State.lean` (CanvasState structure) - `Afferent/Canvas/Context.lean` (shadow rendering) - `native/src/metal/` (blur shader or multi-pass rendering)

Estimated Effort: Large

Dependencies: May require additional render passes or blur shader.

[Priority: Medium] Image/Texture Drawing in Canvas API

Description: Add drawImage/drawTexture functions to the Canvas API for drawing textures with transformations.

Rationale: While Renderer.drawSprites exists, there is no high-level Canvas API for texture drawing with transforms, clipping, and compositing.

Affected Files: - `Afferent/Canvas/Context.lean` (new drawImage functions) - `Afferent/FFI/Texture.lean` (may need additional FFI functions)

Estimated Effort: Medium

Dependencies: None.

[Priority: Medium] Animation Easing Library

Description: Add a library of standard easing functions (ease-in, ease-out, ease-in-out, cubic-bezier, spring) for animations.

Rationale: The framework has robust animation support (animated rendering, time-based loops) but lacks high-level easing utilities.

Affected Files: - New file: `Afferent/Animation/Easing.lean` - Integration with existing animation demos

Estimated Effort: Small

Dependencies: None.

[Priority: Medium] Multi-Window Support

Description: Enable creating and managing multiple windows from a single application.

Rationale: Some applications require multiple windows (toolbars, palettes, preview windows).

Affected Files: - `Afferent/FFI/Window.lean` - `Afferent/Canvas/Context.lean` - `native/src/metal/window.m`

Estimated Effort: Large

Dependencies: Significant FFI and native code changes.

[Priority: Low] Keyboard Event API Enhancement

Description: Add higher-level keyboard event handling with key names (not just key codes), text input support, and key repeat detection.

Rationale: Current API returns raw key codes which require manual mapping. Text input for text fields is not directly supported.

Affected Files: - `Afferent/FFI/Window.lean` (new FFI functions) - `native/src/metal/window.m` (text input delegates)

Estimated Effort: Medium

Dependencies: None.

[Priority: Low] Cursor Customization

Description: Add ability to change the mouse cursor (pointer, text, crosshair, custom image).

Rationale: Different cursor styles provide important UI feedback.

Affected Files: - `Afferent/FFI/Window.lean` (new FFI function) - `native/src/metal/window.m` (NSCursor handling)

Estimated Effort: Small

Dependencies: None.

[Priority: Low] Window Fullscreen and Resize API

Description: Add programmatic fullscreen toggle and window resize/position control.

Rationale: Applications often need fullscreen mode and window management.

Affected Files: - `Afferent/FFI/Window.lean` - `native/src/metal/window.m`

Estimated Effort: Small

Dependencies: None.

Code Improvements

[Priority: High] Non-Convex Polygon Tessellation

Current State: The tessellateConvexPath function uses simple fan triangulation which only works correctly for convex polygons.

Proposed Change: Implement proper ear-clipping or triangulation algorithm for non-convex polygons.

Benefits: Correct rendering of arbitrary polygon shapes (complex paths, concave shapes).

Affected Files: - `Afferent/Render/Tessellation.lean` (triangulateConvexFan function and callers)

Estimated Effort: Medium

[Priority: High] Proper arcTo Implementation

Current State: The arcTo command in path tessellation is simplified to just draw lines to p1 and p2 (line 116).

Proposed Change: Implement proper arcTo geometry that draws a line to p1 then an arc tangent to both lines with the given radius.

Benefits: Correct HTML5 Canvas API compatibility for rounded corners and path effects.

Affected Files: - `Afferent/Render/Tessellation.lean` (pathToPolygonWithClosed, lines 113-118)

Estimated Effort: Medium

[Priority: High] Arc Transform Handling

Current State: In CanvasState.transformPath, arc commands transform the center but not the radius or angles (line 132-133 comment notes this).

Proposed Change: Properly handle arc transforms including non-uniform scaling (may require converting arcs to beziers).

Benefits: Correct arc rendering under arbitrary transforms.

Affected Files: - `Afferent/Canvas/State.lean` (transformPath function, lines 131-133)

Estimated Effort: Medium

[Priority: Medium] Matrix4 Performance

Current State: Matrix4.multiply creates intermediate arrays and uses nested loops with getD.

Proposed Change: Implement matrix multiplication with inline expanded operations or SIMD intrinsics in native code.

Benefits: Faster 3D transforms, especially for scenes with many objects.

Affected Files: - `Afferent/Render/Matrix4.lean` (multiply function) - Optionally: new FFI for matrix operations

Estimated Effort: Small to Medium

[Priority: Medium] FPSCamera Clamp Function Visibility

Current State: The clamp helper function in FPSCamera is marked private but could be useful elsewhere.

Proposed Change: Move clamp to a shared utility module (e.g., Afferent.Core.Math or use Float.max/Float.min).

Benefits: Reduce code duplication, consistent utility functions.

Affected Files: - `Afferent/Render/FPSCamera.lean` (line 34) - New file or existing core module

Estimated Effort: Small

[Priority: Medium] Font Registry Thread Safety

Current State: FontRegistry uses a simple Array which may not be thread-safe for concurrent access.

Proposed Change: Consider using thread-safe data structures or document single-threaded usage requirement.

Benefits: Safer concurrent font registration and lookup.

Affected Files: - `Afferent/Text/Measurer.lean`

Estimated Effort: Small

[Priority: Medium] Batch Capacity Growth Strategy

Current State: Batch pre-allocates with capacity hints but growth strategy is implicit via Lean Array behavior.

Proposed Change: Add explicit capacity doubling or configurable growth for very large batches.

Benefits: Better memory allocation patterns for scenes with many shapes.

Affected Files: - `Afferent/Render/Tessellation.lean` (Batch namespace)

Estimated Effort: Small

[Priority: Low] Reduce Magic Numbers in Path.lean

Current State: Pi is defined as a literal constant (3.14159265358979323846) and the bezier approximation constant (0.5522847498) appears multiple times.

Proposed Change: Define named constants for pi and the bezier circle approximation factor.

Benefits: Improved readability, single source of truth.

Affected Files: - `Afferent/Core/Path.lean` (lines 165-166, 112, 127, 142)

Estimated Effort: Small

[Priority: Low] Pi Constant Consolidation

Current State: Pi is defined locally in multiple files (Path.lean, FPSCamera.lean, Seascape.lean).

Proposed Change: Define Float.pi or Afferent.pi in a single location and use it everywhere.

Benefits: Consistency, reduced duplication.

Affected Files: - `Afferent/Core/Path.lean` - `Afferent/Render/FPSCamera.lean` - `Demos/Seascape.lean`

Estimated Effort: Small

Code Cleanup

[Priority: High] Remove Unused Imports

Issue: Some files may import modules that are not used.

Location: Project-wide audit needed

Action Required: Run import analysis and remove unused imports.

Estimated Effort: Small

[Priority: Medium] Document Vertex Layout Constants

Issue: Vertex layouts (6 floats for 2D, 10 floats for 3D, 12 floats for textured 3D) are documented in comments but could benefit from named constants.

Location: - `Afferent/Render/Tessellation.lean` (multiple locations) - `Afferent/FFI/Renderer3D.lean` - `Afferent/FFI/Asset.lean`

Action Required: Define constants like VERTEX_SIZE_2D = 6, VERTEX_SIZE_3D = 10, VERTEX_SIZE_TEXTURED = 12.

Estimated Effort: Small

[Priority: Medium] Add More Test Coverage

Issue: Tests exist for tessellation and FFI safety but coverage could be expanded for Canvas, Widget rendering, and gradient sampling edge cases.

Location: - `Afferent/Tests/` directory - `AfferentTests.lean`

Action Required: - Add CanvasState transform composition tests - Add gradient edge case tests (empty stops, single stop) - Add Font loading and measurement tests

Estimated Effort: Medium

[Priority: Low] Normalize Doc Comments

Issue: Some functions have detailed doc comments while others have minimal or no documentation.

Location: Throughout codebase, especially FFI modules.

Action Required: Add doc comments to all public functions, standardize format.

Estimated Effort: Medium

[Priority: Low] Demo Code Cleanup

Issue: Demo files have some duplicated helper functions (e.g., pi definition, vector operations).

Location: - `Demos/Seascape.lean` - Other demo files

Action Required: Extract shared demo utilities to a common module.

Estimated Effort: Small

[Priority: Low] Clean Up Deprecated Patterns

Issue: Some code uses older Lean patterns that could be modernized.

Location: Project-wide

Action Required: Audit for deprecated patterns as Lean evolves.

Estimated Effort: Ongoing

Architecture Considerations

Renderer Abstraction for Non-Metal Backends

Currently the framework is tightly coupled to Metal on macOS. Consider a renderer abstraction layer to potentially support: - Vulkan (for cross-platform) - WebGPU (for browser targets) - OpenGL fallback

This would be a significant undertaking but would expand the framework's reach.

State Machine for Widget Events

The widget event system could benefit from a formal state machine for focus, hover, and pressed states to ensure consistent behavior across all interactive widgets.

Memory Budget System for Tile Cache

The map tile cache uses fixed counts for GPU and RAM limits. A memory-budget-based system would be more flexible across different devices.

Quick Wins

These items can be addressed quickly with minimal risk:

1. Define pi constant in one place
2. Add named constants for vertex layout sizes
3. Add shouldBeNear tolerance parameter to test helpers
4. Document all FFI function parameters
5. Add more gradient sampling tests

Last updated: 2025-12-21