Canopy Roadmap

A desktop widget framework built on top of Arbor for Lean 4.

Project Status

Canopy is currently a **scaffold project** consisting of only a namespace stub and version string. The project exists as a placeholder for a higher-level widget framework that will build upon Arbor's renderer-agnostic widget primitives.

Based on analysis of the arbor project and related applications (afferent, chroma), this roadmap outlines the features and improvements needed to make canopy a fully functional desktop widget framework.

Feature Proposals

[Priority: High] Stateful Widget Abstractions

Description: Create higher-level stateful widget types that wrap Arbor's declarative widgets with state management, lifecycle hooks, and automatic ID tracking.

Rationale: Arbor provides low-level display-only widgets with manual ID management. Applications like chroma must manually track widget IDs and register handlers separately. Canopy should provide stateful widget abstractions that automatically wire up event handlers.

Affected Files: - `Canopy/Widget/Stateful.lean` (new) - `Canopy/Widget/Component.lean` (new)

Estimated Effort: Large

Dependencies: None

Proposed Design:

```
-- Example API
structure Component (Model Msg : Type) where
  init : Model
  view : Model -> WidgetBuilder
  update : Msg -> Model -> Model
  subscriptions : Model -> Array Subscription

-- Auto-wiring of handlers
def button (label : String) (onClick : Msg) : ComponentBuilder Msg Widget
```

[Priority: High] Focus Management System

Description: Implement keyboard focus traversal, focus rings, and focus state tracking across the widget tree.

Rationale: Arbor's event system handles pointer events well but lacks focus management for keyboard navigation. Desktop applications require Tab/Shift-Tab focus cycling, focus indicators, and focus-aware keyboard event routing.

Affected Files: - `Canopy/Focus/State.lean` (new) - `Canopy/Focus/Traversal.lean` (new) - `Canopy/Focus/Ring.lean` (new)

Estimated Effort: Large

Dependencies: Stateful Widget Abstractions

Proposed API:

```
structure FocusState where
  focusedId : Option WidgetId
  focusRing : Array WidgetId  -- Ordered focusable widgets
  tabIndex : HashMap WidgetId Int

def focusNext (state : FocusState) : FocusState
def focusPrev (state : FocusState) : FocusState
def handleFocusKey (key : Key) (state : FocusState) : FocusState
```

[Priority: High] Theme System

Description: Create a centralized theming system with color schemes, typography scales, spacing tokens, and component-level style overrides.

Rationale: Arbor's BoxStyle is per-widget. Applications need consistent theming across all widgets. Terminus demonstrates this with its Style types. Canopy should provide a Theme structure that propagates design tokens through the widget tree.

Affected Files: - `Canopy/Theme/Core.lean` (new) - `Canopy/Theme/Colors.lean` (new) - `Canopy/Theme/Typography.lean` (new) - `Canopy/Theme/Spacing.lean` (new) - `Canopy/Theme/Presets.lean` (new)

Estimated Effort: Medium

Dependencies: None

Proposed Design:

```
structure Theme where
  colors : ColorScheme
  typography : TypographyScale
  spacing : SpacingScale
  borderRadius : Float
  shadows : ShadowScheme

structure ColorScheme where
  primary : Color
  secondary : Color
  background : Color
  surface : Color
  error : Color
  onPrimary : Color
  onSecondary : Color
  onBackground : Color
  onSurface : Color
  onError : Color
```

[Priority: High] Common Widget Library

Description: Provide pre-built, themed versions of common UI widgets: buttons, text fields, checkboxes, radio buttons, sliders, dropdowns, dialogs, and tooltips.

Rationale: Arbor provides primitive building blocks (text, box, flex, grid). Every application currently rebuilds common widgets from scratch. Canopy should provide a consistent set of ready-to-use components.

Affected Files: - `Canopy/Widgets/Button.lean` (new) - `Canopy/Widgets/TextField.lean` (new) - `Canopy/Widgets/Checkbox.lean` (new) - `Canopy/Widgets/Radio.lean` (new) - `Canopy/Widgets/Slider.lean` (new) - `Canopy/Widgets/Dropdown.lean` (new) - `Canopy/Widgets/Dialog.lean` (new) - `Canopy/Widgets/Tooltip.lean` (new)

Estimated Effort: Large

Dependencies: Theme System, Focus Management System

[**Priority: Medium**] **Animation System**

Description: Add declarative animation primitives for smooth transitions between states, spring physics, and easing functions.

Rationale: Modern UI frameworks provide animation capabilities. Arbor widgets are static. Canopy should provide an animation layer that interpolates between widget states over time.

Affected Files: - `Canopy/Animation/Core.lean` (new) - `Canopy/Animation/Easing.lean` (new) - `Canopy/Animation/Spring.lean` (new) - `Canopy/Animation/Transition.lean` (new)

Estimated Effort: Large

Dependencies: Stateful Widget Abstractions

Proposed Design:

```
inductive Animation (a : Type) where
  | instant : a -> Animation a
  | tween : a -> a -> Duration -> Easing -> Animation a
  | spring : a -> a -> SpringConfig -> Animation a
  | sequence : Array (Animation a) -> Animation a
  | parallel : Array (Animation a) -> Animation a
```

[**Priority: Medium**] **Accessibility Layer**

Description: Add accessibility metadata, ARIA-like roles, screen reader text, and keyboard shortcut handling.

Rationale: Desktop applications should be accessible. Widgets need semantic roles (button, textbox, checkbox), labels for screen readers, and keyboard shortcut bindings.

Affected Files: - `Canopy/Accessibility/Role.lean` (new) - `Canopy/Accessibility/Label.lean` (new) - `Canopy/Accessibility/Announce.lean` (new)

Estimated Effort: Medium

Dependencies: Focus Management System

[**Priority: Medium**] **Form Handling**

Description: Provide form state management, validation, and submission handling for multi-field input forms.

Rationale: Terminus has a Form widget for terminal UIs. Canopy should provide similar functionality for desktop forms with validation, error display, and submission.

Affected Files: - `Canopy/Form/State.lean` (new) - `Canopy/Form/Validation.lean` (new) - `Canopy/Form/Field.lean` (new) - `Canopy/Form/Builder.lean` (new)

Estimated Effort: Medium

Dependencies: Common Widget Library

[Priority: Medium] Layout Helpers

Description: Provide higher-level layout combinators that compose Trellis flex/grid layouts with common patterns.

Rationale: Common layout patterns (sidebar + content, header + body + footer, split panes) require repetitive Trellis configuration. Canopy should provide named helpers.

Affected Files: - `Canopy/Layout/Patterns.lean` (new) - `Canopy/Layout/Responsive.lean` (new)

Estimated Effort: Small

Dependencies: None

Proposed Helpers:

```
def sidebarLayout (sidebar content : WidgetBuilder) : WidgetBuilder
def headerBodyFooter (header body footer : WidgetBuilder) : WidgetBuilder
def splitPane (left right : WidgetBuilder) (ratio : Float) : WidgetBuilder
def cardGrid (columns : Nat) (cards : Array WidgetBuilder) : WidgetBuilder
```

[Priority: Medium] Drag and Drop

Description: Add drag-and-drop support with drag sources, drop targets, and transfer data.

Rationale: Desktop applications commonly need drag-and-drop for reordering lists, moving items between containers, and file dropping. Arbor has pointer capture for dragging but no structured DnD API.

Affected Files: - `Canopy/DnD/Source.lean` (new) - `Canopy/DnD/Target.lean` (new) - `Canopy/DnD/State.lean` (new)

Estimated Effort: Large

Dependencies: Stateful Widget Abstractions

[Priority: Low] Context Menu System

Description: Add right-click context menu support with nested menus and keyboard navigation.

Rationale: Desktop applications expect right-click menus. This requires overlay rendering, focus trapping, and menu positioning logic.

Affected Files: - `Canopy/Menu/Context.lean` (new) - `Canopy/Menu/Item.lean` (new) - `Canopy/Menu/Overlay.lean` (new)

Estimated Effort: Medium

Dependencies: Focus Management System

[Priority: Low] Clipboard Integration

Description: Add clipboard read/write operations for cut, copy, and paste.

Rationale: Text editing and data manipulation require clipboard access. This needs FFI to system clipboard APIs.

Affected Files: - `Canopy/Clipboard/Core.lean` (new) - `Canopy/Clipboard/FFI.lean` (new)

Estimated Effort: Medium

Dependencies: None (but may need native FFI code)

[Priority: Low] Undo/Redo System

Description: Provide a command pattern-based undo/redo stack for reversible operations.

Rationale: Applications with editing capabilities need undo/redo. A generic command stack can be shared across widgets.

Affected Files: - `Canopy/History/Command.lean` (new) - `Canopy/History/Stack.lean` (new)

Estimated Effort: Small

Dependencies: None

[Priority: Low] Window Manager Integration

Description: Provide utilities for managing multiple windows, window state (minimized, maximized), and window-level events.

Rationale: Complex applications may have multiple windows (preferences, dialogs, tool palettes). Canopy could provide a window management layer above afferent's single-window API.

Affected Files: - `Canopy/Window/Manager.lean` (new) - `Canopy/Window/State.lean` (new)

Estimated Effort: Large

Dependencies: None (but depends on afferent FFI extensions)

Code Improvements

[Priority: High] Add Proper Module Structure

Current State: The project has only two files: `Canopy.lean` and `Canopy/Core.lean` with minimal content.

Proposed Change: Create a proper module hierarchy reflecting planned features:

```
Canopy/
  Core.lean           -- Re-exports, version, core types
  Widget/             -- Stateful widget system
  Theme/              -- Theming
  Focus/              -- Focus management
  Animation/          -- Animations
  Layout/             -- Layout helpers
  Widgets/            -- Common widgets
  Form/               -- Form handling
  Accessibility/      -- A11y
```

```
  DnD/                  -- Drag and drop
```

Benefits: Clear code organization, easier navigation, explicit module boundaries

Affected Files: All project files

Estimated Effort: Small (structure only)

[Priority: High] Add Crucible Test Framework

Current State: The project has no test infrastructure.

Proposed Change: Add crucible dependency and create a test directory structure.

Benefits: Enables TDD for new features, regression testing

Affected Files: - `lakefile.lean` - Add crucible dependency and test target - `CanopyTests/Main.lean` (new)

Estimated Effort: Small

[Priority: Medium] Re-export Arbor and Trellis Types

Current State: Core.lean imports Arbor but doesn't re-export useful types.

Proposed Change: Selectively re-export commonly-used Arbor and Trellis types so users can import only Canopy.

Benefits: Cleaner imports for downstream users, single import for common use cases

Affected Files: - `Canopy/Core.lean`

Estimated Effort: Small

[Priority: Medium] Add TypeClass Hierarchy

Current State: No typeclasses defined.

Proposed Change: Define typeclasses for common widget behaviors:

```
class Focusable (w : Type) where
  canFocus : w -> Bool
  tabIndex : w -> Int

class Themed (w : Type) where
  applyTheme : Theme -> w -> w

class Validatable (a : Type) where
  validate : a -> ValidationResult
```

Benefits: Polymorphic widget behavior, consistent API

Affected Files: - `Canopy/Typeclass/Focusable.lean` (new) - `Canopy/Typeclass/Themed.lean` (new)

Estimated Effort: Medium

[Priority: Low] Add Comprehensive Documentation

Current State: Minimal documentation, placeholder README.

Proposed Change: Add docstrings to all public definitions and expand README with examples.

Benefits: Better developer experience, clearer API understanding

Affected Files: - All source files - `README.md`

Estimated Effort: Medium (ongoing)

Code Cleanup

[Priority: High] Remove Placeholder Comment

Issue: `Canopy/Core.lean` contains a placeholder comment that should be replaced with actual implementation.

Location: `/Users/Shared/Projects/lean-workspace/canopy/Canopy/Core.lean`, line 9

Action Required: Replace placeholder comment with initial core types or module re-exports.

Estimated Effort: Small

[Priority: Medium] Expand README

Issue: README is minimal and doesn't describe the project's goals or planned features.

Location: `/Users/Shared/Projects/lean-workspace/canopy/README.md`

Action Required: Add sections for: - Project goals and philosophy - Relationship to Arbor - Getting started guide - Roadmap summary - Contributing guidelines

Estimated Effort: Small

[Priority: Low] Add License File

Issue: Project has a LICENSE file but should verify it matches workspace conventions.

Location: `/Users/Shared/Projects/lean-workspace/canopy/LICENSE`

Action Required: Verify license is appropriate and consistent with related projects (arbor, afferent).

Estimated Effort: Trivial

Dependencies to Add

When implementing features, the following dependencies may be needed:

Feature	Dependency	Purpose
Testing	crucible	Test framework
Property Testing	plausible	Property-based testing
Colors	tincture	Color utilities (already via arbor)
Layout	trellis	CSS layout (already via arbor)

Implementation Order

Recommended sequence for implementing features:

1. **Phase 1: Foundation**
   - Add proper module structure
   - Add Crucible test framework
   - Re-export Arbor/Trellis types
   - Theme System (enables consistent styling)
2. **Phase 2: Core Widgets**
   - Stateful Widget Abstractions
   - Focus Management System
   - Common Widget Library (Button, TextField)
3. **Phase 3: Advanced Widgets**
   - Form Handling
   - Dropdown, Dialog, Tooltip
   - Context Menu System
4. **Phase 4: Rich Interactions**
   - Animation System
   - Drag and Drop
   - Accessibility Layer
5. **Phase 5: Platform Integration**
   - Clipboard Integration
   - Window Manager Integration
   - Undo/Redo System

Notes

- This project is designed to complement Arbor (low-level, renderer-agnostic) with high-level, stateful, themed widgets.
- The chroma project demonstrates the current pain point: manual widget ID tracking and separate handler registration.
- Terminus provides inspiration for terminal widgets but uses immediate-mode rendering; Canopy will use Arbor's declarative approach.
- Consider whether animations should be handled at the Arbor level (render command animations) or Canopy level (state interpolation).