Tuna 应用 UI 架构总结

整体架构

Tuna是一个macOS菜单栏应用程序，使用SwiftUI和AppKit构建。主要功能是音频控制和语音转文字。

应用入口点

- **AppDelegate**: 负责应用初始化、状态栏图标设置和弹出窗口管理
- **主要UI状态管理**:
  - 弹出窗口行为: 可切换固定/非固定模式
  - 事件监控: 捕获应用外点击事件以关闭弹出窗口

核心UI组件

1. 菜单栏视图 (MenuBarView)

文件位置: `Sources/Tuna/MenuBarView.swift` **尺寸**: 固定宽度400px，高度自适应（最小380px） **布局结构**: - 顶部区域: - 标题栏: 展示应用名称和固定按钮 - 标签页导航: 设备(Devices)、语音(Whispen)、统计(Stats) - **内容区域**: - 自适应高度的滚动视图，最大高度为容器高度的70% - 根据选定标签显示不同内容 - **底部按钮栏**: - 退出、关于、设置按钮 - 视觉层级较低，使用较小字体和半透明效果

视觉风格: - 使用毛玻璃效果(NSVisualEffectView)背景 - 暗色主题 - 圆角边缘

2. 设备卡片 (DeviceCards)

状态: 显示在"Devices"标签页 组件: - **OutputDeviceCard**: 输出设备选择和控制 - 设备下拉选择器 - 音量控制滑块 - **InputDeviceCard**: 输入设备选择和控制 - 设备下拉选择器 - 音量等级显示

3. 语音转写界面 (DictationView)

文件位置: `Sources/Tuna/Views/TunaDictationView.swift` **状态**: 显示在"Whispen"标签页 **主要组件**: - 录音控制按钮 - 使用红色(critical)表示录音状态 - 文本输入/编辑区域 - 转写状态指示器 - 文本操作菜单(复制/粘贴/格式化)

4. 快速录音窗口 (QuickDictationView)

文件位置: `Sources/Tuna/Views/TunaDictationView.swift` **功能**: 支持通过快捷键激活的简化录音界面 **特点**: - 精简布局 - 仅显示必要控制元素 - 响应全局快捷键

5. 设置视图 (TunaSettingsView)

文件位置: `Sources/Tuna/TunaSettingsView.swift` **布局**: - 标签页导航: 一般设置、Whispen设置、智能切换、支持 - 每个标签页包含相关设置卡片 - 使用统一的视觉风格

视觉组件: - 玻璃卡片风格(GlassCard修饰器)，内边距从16pt增加到20pt - 现代化开关样式(ModernToggleStyle) - 强调色: 薄荷绿(DesignTokens.Colors.accent)和危险红(DesignTokens.Colors.critical)

设计令牌系统

Tuna应用现在使用集中管理的设计令牌来确保UI组件的一致性。

文件位置

`Sources/Tuna/DesignTokens.swift`

主要设计常量

- **网格单位**: 4pt基本网格单位
- **卡片内边距**: 20pt (5 × 网格单位)
- 组件间距: 12pt (3 × 网格单位)
- 圆角半径: 卡片12pt，按钮6pt

颜色系统

- **主强调色(accent)**: 薄荷绿 - Color(red: 0.3, green: 0.9, blue: 0.7)
- **次强调色(critical)**: 红色 - Color(red: 1, green: 0.35, blue: 0.35)
- 输出设备色(output): 蓝色 - Color(red: 0.2, green: 0.6, blue: 1.0)
- 输入设备色(input): 红色 - Color(red: 1.0, green: 0.4, blue: 0.4)

交互模式

1. 菜单栏交互

- 点击状态栏图标显示/隐藏主界面
- 支持窗口钉住模式(点击其他区域不会关闭)
- 标签页导航支持动画过渡

2. 快捷键集成

- 全局快捷键支持(通过KeyboardShortcutManager)
- 快速激活语音转写功能
- 自定义快捷键设置

3. 上下文菜单

- 文本区域支持右键菜单
- 录音控制和文本操作选项

文件组织

主要UI文件

- **MenuBarView.swift**: 主菜单视图
- **TunaSettingsView.swift**: 设置界面
- **TunaDictationView.swift**: 语音转写界面
- **AppDelegate.swift**: 应用初始化和管理
- **DesignTokens.swift**: 设计令牌系统
- **SharedStyles.swift**: 共享样式定义

视图模型和数据流

- **AudioManager**: 管理音频设备状态和控制
- **DictationManager**: 管理语音转写功能
- **TunaSettings**: 全局设置存储
- **TabRouter**: 标签页路由管理

视觉设计规范

颜色系统

- 主背景: 深灰色(Color(red: 0.18, green: 0.18, blue: 0.18))
- 主强调色: 薄荷绿(DesignTokens.Colors.accent)
- 次强调色: 红色(DesignTokens.Colors.critical)
- 文本颜色: 白色(主要)、半透明白色(次要)

字体规范

- 标题: 系统字体14pt, medium weight
- 正文: 系统字体13pt
- 小文本: 系统字体12pt
- 底部按钮: 系统字体11pt (降低视觉层级)

间距规范

- 卡片内边距: 20pt(水平), 20pt(垂直)
- 组件间距: 12pt(垂直)
- 按钮区内边距: 16pt(水平), 8pt(垂直)

元素风格

- 卡片: 圆角12pt, 半透明背景, 微弱边框
- 按钮: 无边框, hover状态显示背景
- 切换开关: 现代胶囊样式, 带动画效果

SF Symbols风格指南

应用中的图标使用统一的风格规范: - 主要操作按钮: 使用.fill变体 (例如: mic.circle.fill, stop.circle.fill) - 次要操作按钮: 使用常规变体 - 标签图标: 使用.fill变体表示当前选中状态 - 视觉层次: 重要操作使用更大尺寸图标

可访问性考虑

- 支持系统深色模式
- 图标包含辅助功能描述
- 使用标准系统控件提高兼容性
- 支持键盘导航和快捷键
- 所有互动元素都有足够的点击区域

已知UI问题

1. ~~标签页切换时内容区域高度闪烁~~ (已修复：使用GeometryReader和自适应高度)
2. 窗口固定状态在重启应用后需要再次点击才能恢复 (待实现：Persist pop-over pinning)
3. 弹出窗口偶尔会有灰色阴影问题
4. 文本编辑区域的水平内边距不一致 (待实现：Fix TextEditor horizontal padding)
5. 录音状态缺少实时视觉反馈 (待实现：Embed live waveform)

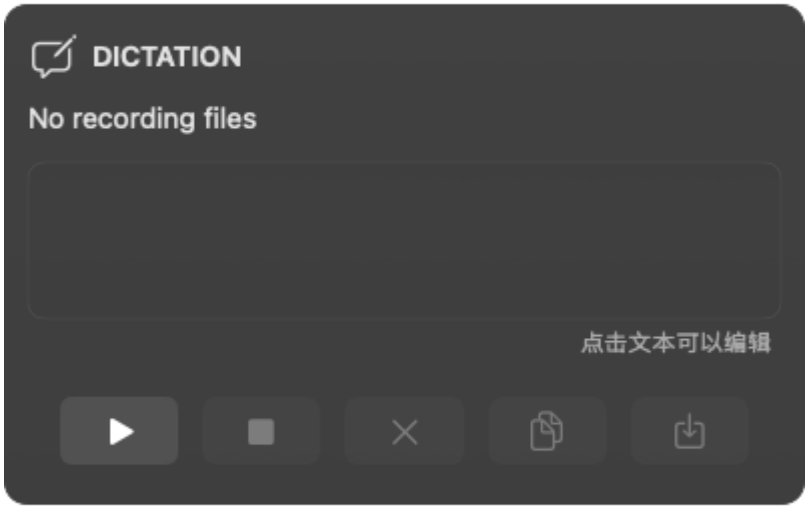
建议的UI改进

1. ✓ 替换固定内容区域高度为自适应布局 (已完成)
2. ✓ 增强卡片内边距，改善视觉呼吸空间 (已完成：从16pt增加到20pt)
3. ✓ 降低底部按钮栏的视觉层级 (已完成)
4. ✓ 统一SF Symbols图标风格 (已完成)

5. ✓ 引入次要强调色用于破坏性/录音状态 (已完成)
6. 修复文本编辑器的水平内边距 (待实现)
7. 在录音按钮下添加实时波形显示 (待实现)
8. 在QuickDictationView标题中显示键盘快捷键提示 (待实现)
9. 在应用重启后保持弹出窗口的固定状态 (待实现)
10. 添加减少动画选项，支持无障碍需求 (待实现)

UI Snapshot Manifest

> **运行截图命令**
> `cursor run gen-screenshot`



Screens

- **MenuBarView**: 主菜单栏视图，固定宽度400px，高度自适应（最小380px）
- **TunaDictationView**: 语音转写界面，显示在"Whispen"标签页
- **QuickDictationView**: 快捷键激活的简化录音界面
- **AboutCardView**: 关于页面，展示应用理念，宽780px，高750px
- **TunaSettingsView**: 设置界面，包含多个标签页

Hierarchy

- **MenuBarView**
  - TitleBar: 应用名称 + 固定按钮
  - TabNavigation: 设备、语音、统计标签
  - ContentArea: 自适应高度滚动视图
  - BottomBar: 退出、关于、设置按钮 (视觉层级较低)
- **TunaDictationView**
  - RecordingControls: 录音控制按钮 (使用critical颜色表示录音状态)
  - TextEditor: 文本输入/编辑区
  - StatusIndicator: 转写状态指示器
  - ContextMenu: 文本操作选项
- **QuickDictationView**
  - SimplifiedControls: 精简录音界面
  - TextArea: 转写文本区域
  - StatusIndicator: 录音状态指示
- **DeviceCards**
  - OutputDeviceCard: 输出设备选择 + 音量滑块
  - InputDeviceCard: 输入设备选择 + 音量显示
- **TunaSettingsView**
  - TabBar: 一般、Whispen、智能切换、支持
  - SettingsCards: 每个标签对应设置卡片

Styles

- **Colors**
  - accent: DesignTokens.Colors.accent - 薄荷绿
  - critical: DesignTokens.Colors.critical - 红色
  - output: DesignTokens.Colors.output - 蓝色
  - input: DesignTokens.Colors.input - 红色
  - textPrimary: .white
  - textSecondary: .white.opacity(0.8)
  - cardBackground: 毛玻璃效果，半透明深灰+微弱光晕
- **Typography**
  - title: system(size: 14, weight: .medium)
  - body: system(size: 13)
  - small: system(size: 12)
  - footer: system(size: 11) - 底部按钮栏专用
  - monospace: monospacedSystemFont(ofSize: 12, weight: .regular)
- **Layout**
  - cardPadding: DesignTokens.cardPadding - 20pt
  - componentSpacing: DesignTokens.componentSpacing - 12pt
  - buttonPadding: 水平16pt，垂直8pt
  - cornerRadius: DesignTokens.cardRadius - 12pt (卡片)，
    DesignTokens.buttonRadius - 6pt (按钮)
- **Components**
  - GlassCard: 圆角12pt，半透明背景，微弱边框，轻微阴影，内边距20pt
  - ModernToggleStyle: 胶囊形滑块开关，带动画效果
  - BidirectionalSlider: 双向滑块，自定义轨道和滑块按钮
  - ShortcutTextField: 自定义快捷键输入字段
  - GreenButtonStyle: 绿色圆角按钮，带悬停和按压效果

自定义组件

DevicePreferenceRow

设备选择行，包含图标、标题和下拉箭头 DevicePreferenceRow预览

BidirectionalSlider

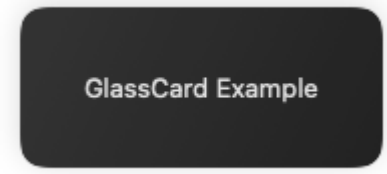
自定义双向滑块，支持从-50到50的值范围

ShortcutTextField

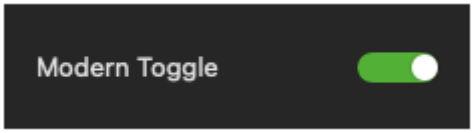
快捷键输入字段，自定义键盘事件处理

GlassCard

现代化玻璃卡片修饰器，带反光效果和微弱边框，内边距已增加到20pt



ModernToggleStyle

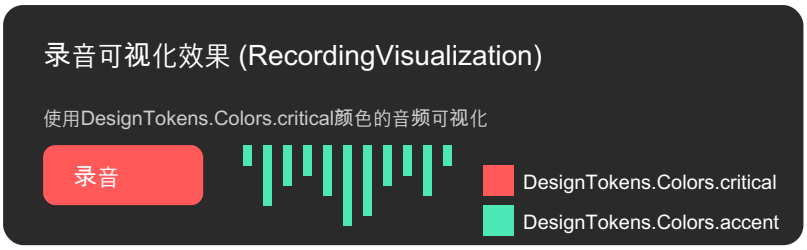


现代开关样式，带弹簧动画效果

动态状态

RecordingVisualization

录音状态可视化，带动画效果，现在使用critical颜色



录音可视化效果

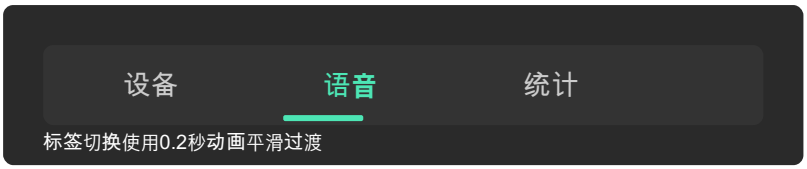
技术说明： - 使用DesignTokens.Colors.critical（红色）替代硬编码RGB值 - AudioVisualizerBar组件实现音频条形图效果 - 录音按钮在录音状态下使用critical颜色提供视觉反馈 - 每个音频条使用随机高度变化模拟声音输入

实现代码：

```
// 录音按钮背景色
.background(dictationManager.state == .recording ?
            DesignTokens.Colors.critical.opacity(0.7) : // 使用
        DesignTokens颜色
            (dictationManager.state == .paused ?
        Color.orange.opacity(0.7) :
        DesignTokens.Colors.accent.opacity(0.7)))

// 音频可视化条
struct AudioVisualizerBar: View {
    let isRecording: Bool
    @State private var height: CGFloat = 5
    @State private var timer: Timer?

    var body: some View {
        RoundedRectangle(cornerRadius: 1)
            .fill(DesignTokens.Colors.accent)
            .frame(width: 2, height: height)
            .onAppear { startAnimation() }
            .onDisappear { stopAnimation() }
    }

    private func startAnimation() {
        timer = Timer.scheduledTimer(withTimeInterval: 0.15,
        repeats: true) { _ in
            withAnimation(.linear(duration: 0.1)) {
                height = CGFloat.random(in: 2...24)
            }
        }
    }
}
```

TabNavigation

标签切换使用withAnimation，带0.2秒过渡动画



标签导航动画

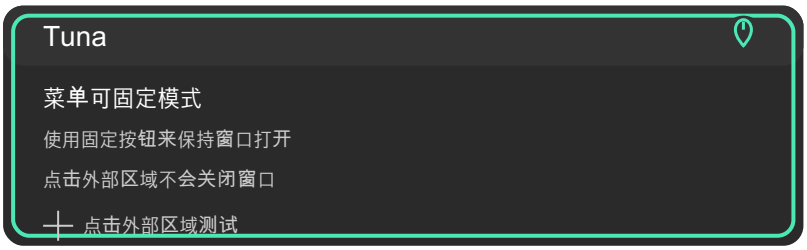
技术说明： - 使用SwiftUI的withAnimation函数实现标签切换 - 选中标签文本变为accent颜色 - 底部指示器滑动到当前选中标签下方

实现**代码**：

```
// 标签切换动画
withAnimation(.easeInOut(duration: 0.2)) {
    selectedTab = newTab
}

// 标签视图
HStack(spacing: 20) {
    ForEach(TabItem.allCases, id: \.self) { tab in
        Text(tab.title)
            .font(.system(size: 14, weight: selectedTab ==
        tab ? .medium : .regular))
            .foregroundColor(selectedTab == tab ?
        DesignTokens.Colors.accent : .white.opacity(0.8))
            .padding(.vertical, 8)
            .onTapGesture {
                withAnimation(.easeInOut(duration: 0.2)) {
                    selectedTab = tab
                }
            }
    }
}
```

PopoverPinning

菜单可固定，防止点击外部关闭



弹出窗口固定功能

技术说明： - 通过NSPopover的behavior属性控制固定状态 - 用户点击固定按钮后，弹出窗口不会被点击外部区域的操作关闭 - 按钮图标从"pin"变为"pin.fill"提供视觉反馈

实现**代码**：

```
// 固定按钮视图
Button(action: togglePinning) {
    Image(systemName: isPinned ? "pin.fill" : "pin")
        .font(.system(size: 12))
        .foregroundColor(.white.opacity(0.8))
}
.buttonStyle(PlainButtonStyle())
.help(isPinned ? "取消固定" : "固定窗口")

// 切换固定状态
private func togglePinning() {
    isPinned.toggle()
    if isPinned {
        popover.behavior = .applicationDefined
    } else {
        popover.behavior = .transient
    }
}
```

ToggleAnimation

开关切换使用0.2秒弹簧动画效果

开关切换动画

技术说明： - 使用自定义ModernToggleStyle实现开关效果 - 弹簧动画提供轻微弹性效果，增强物理感 - 开关状态变化时背景颜色从灰色渐变到accent颜色

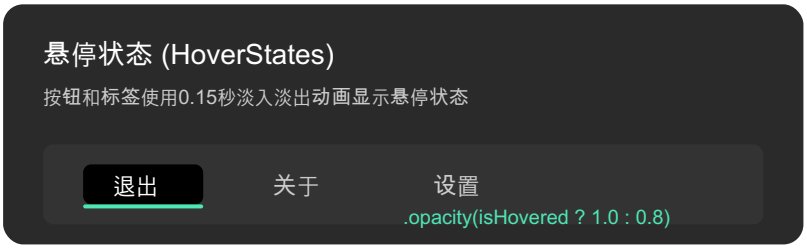
实现代码：

```
struct ModernToggleStyle: ToggleStyle {
    func makeBody(configuration: Configuration) -> some View {
        Button(action: { configuration.isOn.toggle() }) {
            HStack {
                configuration.label
                Spacer()
                RoundedRectangle(cornerRadius: 16)
                    .fill(configuration.isOn ?
        DesignTokens.Colors.accent : Color.gray.opacity(0.3))
                    .frame(width: 50, height: 29)
                    .overlay(
                        Circle()
                            .fill(Color.white)
                            .shadow(radius: 1)
                            .padding(2)
```

```
                            .offset(x: configuration.isOn ? 10 :
        -10)
                    )
                }
            }
            .buttonStyle(PlainButtonStyle())
            .animation(.spring(response: 0.2, dampingFraction: 0.8),
            value: configuration.isOn)
        }
}
```

HoverStates

按钮和标签使用0.15秒淡入淡出动画显示悬停状态



悬停状态动画

技术说明： - 使用SwiftUI的onHover修饰符实现悬停效果 - 鼠标悬停时按钮不透明度增加，并添加背景色 - 标签悬停时可能显示下划线或高亮效果

实现代码：

```
// 按钮悬停效果
Button(action: action) {
    Text(title)
        .font(.system(size: 11))
        .foregroundColor(.white.opacity(0.8))
}
.buttonStyle(PlainButtonStyle())
.onHover { isHovered in
    withAnimation(.easeInOut(duration: 0.15)) {
        self.isHovered = isHovered
    }
}
.background(isHovered ? Color.white.opacity(0.1) : Color.clear)
.opacity(isHovered ? 1.0 : 0.8)
.cornerRadius(4)
```

已知问题

1. 某些显示器缩放设置下可能出现对齐问题
2. 窗口固定状态在重启应用后需要再次点击才能恢复 (待实现)
3. 弹出窗口偶尔会有灰色阴影问题
4. 文本编辑区域的水平内边距不一致 (待实现)

5. 录音状态缺少实时视觉反馈 (待实现)
6. DesignTokens系统需要完善，目前有部分循环引用和导入问题

最近UI改进

1. ✅ 替换固定内容区域高度为自适应布局
2. ✅ 增加卡片内边距从16pt到20pt，改善视觉呼吸空间
3. ✅ 降低底部按钮栏的视觉层级，缩小字体和降低不透明度
4. ✅ 统一SF Symbols图标风格，使用.fill变体表示主要操作
5. ✅ 引入次要强调色(critical)用于破坏性操作和录音状态
6. ✅ 创建设计令牌系统(DesignTokens)，集中管理设计常量
