项目详细设计结构说明

一、核心模块（core）

模块（文件夹）	子模块 / 文件	职责描述	优化要点说明
canvas	canvas_renderer.dart canvas_controller.dart canvas_interaction_manager.dart	画布渲染及基础交互管理（缩放、平移）	明确区分 Canvas 状态与交互管理，便于维护
node	node_model.dart node_renderer.dart node_controller.dart node_interaction_manager.dart	节点的状态、渲染、交互逻辑	引入 NodeController，分离数据逻辑与交互事件处理
edge	edge_model.dart edge_renderer.dart edge_controller.dart edge_interaction_manager.dart	边的状态、渲染、交互逻辑	引入 EdgeController 管理边状态，逻辑分离
anchor	anchor_model.dart anchor_renderer.dart anchor_interaction_manager.dart	锚点状态、渲染、交互逻辑	明确锚点独立性，便于扩展自定义 Anchor UI
selection	selection_manager.dart box_selection_handler.dart selection_renderer.dart	选中逻辑（单选、多选、框选）及高亮渲染	设计成独立的状态机，提升扩展性
interactions	mouse_event_dispatcher.dart keyboard_event_dispatcher.dart interaction_mode_controller.dart interaction_state_machine.dart	鼠标/键盘事件统一处理、交互模式管理	新增 InteractionStateMachine 提高事件处理清晰度

二、插件（plugins）

插件模块	文件	职责描述	优化要点说明
context_menu	context_menu_plugin.dart	右键菜单插件	实现标准插件接口，提高统一性与复用性
hover_highlight	hover_highlight_plugin.dart	悬停高亮	插件模式统一，允许热插拔、动态启用或禁用
quick_connection	quick_connection_plugin.dart	快速连接插件	优化插拔流程，避免与 Edge 逻辑耦合

插件模块	文件	职责描述	优化要点说明
custom_plugins	用户自定义插件模板	标准插件模板	便于开发人员快速创建自定义插件

三、Hooks（hooks）

Hooks模块	文件	职责描述	优化要点说明
node_hooks	use_node_hook.dart	节点生命周期事件	增加细粒度 hook (`onDragStart`, `onDragEnd`)
edge_hooks	use_edge_hook.dart	边生命周期事件	增加 `onEdgeCreate`、`onEdgeDelete` 钩子
canvas_hooks	use_canvas_hook.dart	画布生命周期事件	提供画布加载完成、布局改变事件

四、状态管理（state_management）

状态模块	文件	职责描述	优化要点说明
canvas_state	canvas_state.dart canvas_state_provider.dart	画布整体状态管理	细分 viewportState, scaleState，优化渲染性能
node_state	node_state.dart node_state_provider.dart	节点状态管理	增加状态快照（snapshot），实现 undo/redo 功能
edge_state	edge_state.dart edge_state_provider.dart	边状态管理	同 Node，实现状态快照，便于撤销与重做
interaction_state	interaction_state.dart	当前交互模式状态	使用状态机模式，便于模式扩展与清晰管理

五、单元与交互测试（tests）

测试模块	文件	职责描述	优化要点说明
canvas_tests	canvas_controller_test.dart	画布相关单元与 widget 测试	细分状态测试与交互测试
node_tests	node_model_test.dart node_interaction_test.dart	节点状态、渲染、交互功能测试	覆盖交互边界情况和异常情况
edge_tests	edge_model_test.dart edge_interaction_test.dart	边状态、渲染、交互功能测试	增加连线创建的边界测试

六、工具类（utils）

工具模块	文件	职责描述	优化要点说明

工具模块	文件	职责描述	优化要点说明
geometry_utils	geometry_utils.dart	几何计算、碰撞检测、坐标变换	增加空间索引（如 Quadtree）优化碰撞检测性能
color_utils	color_utils.dart	颜色管理工具	支持主题模式切换（亮/暗模式）
performance_utils	performance_utils.dart	性能监控与分析工具	实时性能监控插件，快速定位性能瓶颈

七、配置管理（config）

配置模块	文件	职责描述	优化要点说明
interaction_config	interaction_config.dart	交互相关配置	提供可序列化配置，允许保存与恢复用户设置
theme_config	theme_config.dart	UI 主题配置	支持动态主题切换，优化视觉体验
graph_config	graph_config.dart	全局画布、节点、边、锚点的配置	分模块定义，提供模块级别单独配置

八、UI组件（ui_components）

UI组件模块	文件	职责描述	优化要点说明
buttons	标准按钮组件	通用按钮组件	定义标准样式按钮，统一视觉风格
menus	菜单组件	菜单组件（右键菜单、节点菜单）	统一风格与接口，方便复用
toolbars	工具栏组件	顶部操作栏	支持灵活布局，允许用户自定义

九、示例与文档（examples & documentation）

示例与文档模块	文件	职责描述	优化要点说明
examples	basic_workflow_example.dart advanced_workflow_example.dart	功能与插件示例展示	提供基础和高级示例指导开发
documentation	getting_started.md plugins_development_guide.md hooks_usage.md	快速开始、插件与 Hooks 开发指南	提供清晰指导，确保开发者易于上手

📌 总体结构设计目标:

- **高内聚低耦合**：各模块独立明确，减少相互依赖。
- **可扩展性**：便于新功能的快速开发和现有功能的扩展。
- **可测试性**：清晰定义各模块职责，易于测试与维护。
- **可插拔性**：提供统一插件机制，允许用户快速自定义插件。

- **文档清晰**：为开发者提供明确、易懂的文档指引，降低开发门槛。