

BaroMod Workbanch

Web-UI的潛淵症
模組集成工具

木星機械教神龕
2026年2月14日

1. 技术架构

核心框架: React 18
构建工具: Vite
图形引擎: Konva.js
状态管理: Zustand
XML处理: fast-xml-parser
文件打包: JSZip+FileSaver.js
UI组件库: Ant Design

2. 软件功能模块设计

我们将界面分为三个主要区域: 左侧可视编辑区, 右侧属性配置区, 底部代码预览/导出区。

2.1 核心流程

1. 新建项目: 输入Mod名称、ID。
2. 资源导入: 拖入图片。
3. Sprite 定义: 在图片上框选区域, 自动生成 SourceRect。
4. 组件配置: 添加/修改物品属性 (如 Holdable, Engine, Hull 等)。
5. 导出: 生成符合游戏目录结构的 Zip 包。

2.2 关键交互细节

多状态预览: 潜渊症的贴图通常在一张图上包含多个状态 (普通状态、损坏状态)。Canvas 需要支持“切片列表”, 允许用户框选多个区域并命名。

锚点 (Origin) 模拟: 在 Canvas 选框中心显示一个可拖动的“十字准星”, 计算相对坐标 (0.0 - 1.0), 这决定了物品拿在手里的位置。

3. 数据结构设计

这是整个应用的核心。我们需要定义一个中间层 JSON (Store), 它既能渲染 UI, 又能生成 XML。

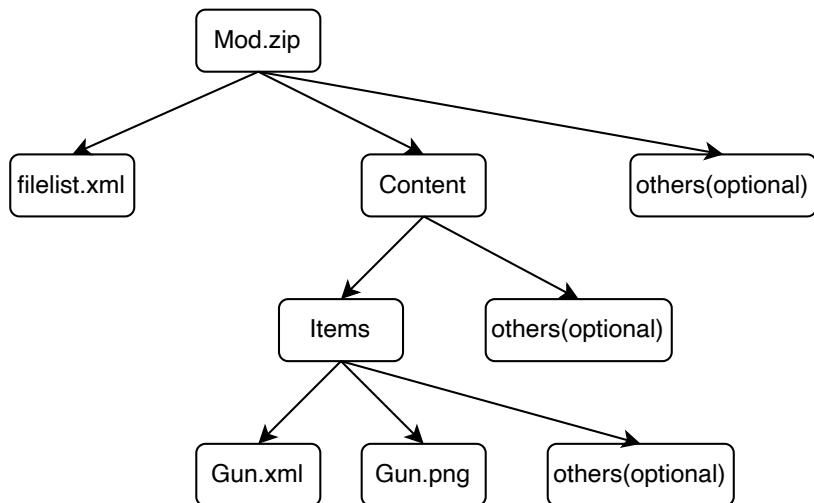
4. 核心功能实现逻辑

4.1 难点一: Canvas 框选与 SourceRect 计算
场景: 用户在 1024x1024 的图上画了一个 100x100 的框。
Konva 实现逻辑:
1. 加载图片到 Canvas。

2. 添加一个 Rect 对象，设为可拖拽 (draggable: true) 和可缩放 (transformable: true)。
3. 监听 onDragEnd 和 onTransformEnd 事件。
4. 计算。
 - 4.2 难点二：Origin (锚点) 计算
 - 4.3 难点三：XML 生成器

5. 目录结构与导出

当用户点击“下载 Mod”时，JSZip 应该生成如下结构，这是《潜渊症》识别 Mod 的标准结构：



6. 开发路线图

Phase 1:

搭建 React + Vite 环境。

实现图片上传 + Canvas 显示。

实现矩形框选，并在控制台打印出正确的 `sourcerect` 数据。

简单的文本框输入 ID 和 Name。

导出 XML 功能。

Phase 2:

引入 XMLBuilder，实现完整的 Item 结构生成。

实现 `origin` 十字准星拖拽。

右侧属性面板，支持添加 "Holdable" 等组件。

Phase 3:

反向解析：允许用户粘贴一段现有的 XML，自动在 Canvas 上还原框选位置（这对修改官方物品很有用）。

本地存储：利用 localStorage 保存未完成的草稿。