项目申请书

项目名称： mermaid 可视化能力扩展

项目主导师： 木头人

项目编号： 251b90076

申请人： 单禹嘉

日期： 2025.05.25

邮箱 :shanyujia0626@gmail.com

github:eleliauk

目录

项目背景

Mermaid 是一个基于 JavaScript 的图表和图形可视化工具，它允许使用文本和代码创建可视化图表。目前 Mermaid 已经成为许多知名项目和平台的默认图表渲染工具，例如 GitHub、GitLab 等。然而，随着用户对可视化需求的不断增长，现有的图表实现存在一些局限性。

1. **现有图表功能受限**
   - 基于 D3.js 的实现，图表类型和功能相对基础
   - 缺乏高级交互特性和动画效果
   - 性能和可扩展性有待提升
2. **VChart 的优势**
   - 提供丰富的图表类型和组件
   - 强大的自定义和扩展能力
   - 优秀的渲染性能和动画效果
   - 完善的主题和样式系统
3. **VTable 的优势**
   - 支持复杂的表格布局
   - 丰富的单元格类型
   - 强大的性能和可扩展性
   - 完善的样式定制能力

本项目旨在通过整合 VChart 和 VTable 的能力，显著提升 Mermaid 的可视化表现力，为用户提供更强大、更灵活的图表和表格功能。

项目需求

1. 替换原有图表实现
   - 将mermaid原有的饼图、桑基图、xy图表从d3替换为VChart实现
   - 保持原有API兼容性
   - 提升图表展示效果和交互体验
2. 迁移VChart图表
   - 将VChart其他图表类型集成到mermaid中
   - 提供统一的配置和使用方式
   - 扩展mermaid的图表能力
3. 添加VTable表格能力

- 集成VTable到mermaid
- 实现表格的解析和渲染

项目参考

mermind

d3

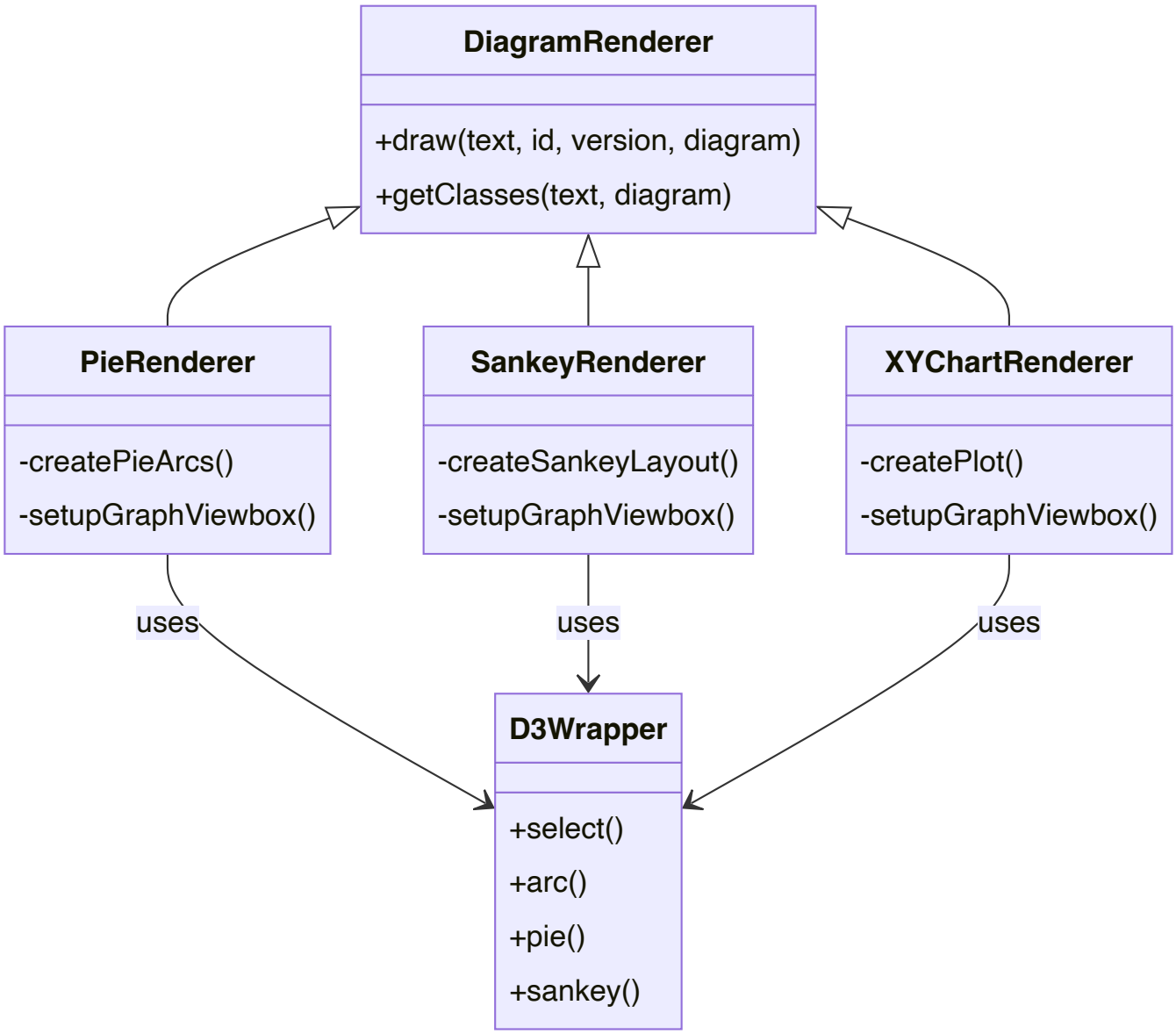
antv

项目技术方案

1. 当前Mermaid图表实现分析

1.1 渲染器架构

Mermaid目前采用基于D3.js的渲染方案，主要包含以下核心组件:



1. **DiagramRenderer接口**

- 定义draw()方法用于图表渲染
- 提供getClasses()方法处理样式类
- 负责整体渲染流程控制

2. **各类图表独立实现**

- 饼图(pieRenderer.ts)：使用D3.js的pie布局和arc生成器
- 桑基图(sankeyRenderer.ts)：基于d3-sankey插件实现
- XY图表(xyChartRenderer.ts)：自定义实现基础图表元素渲染

1.2 现有实现的局限性

1. **图表能力受限**

- 饼图仅支持基础饼图,缺乏环形图、南丁格尔玫瑰图等变体
- 桑基图交互性较弱,缺乏节点拖拽等高级特性
- XY图表功能简单,缺乏复杂的坐标轴配置和图表组合

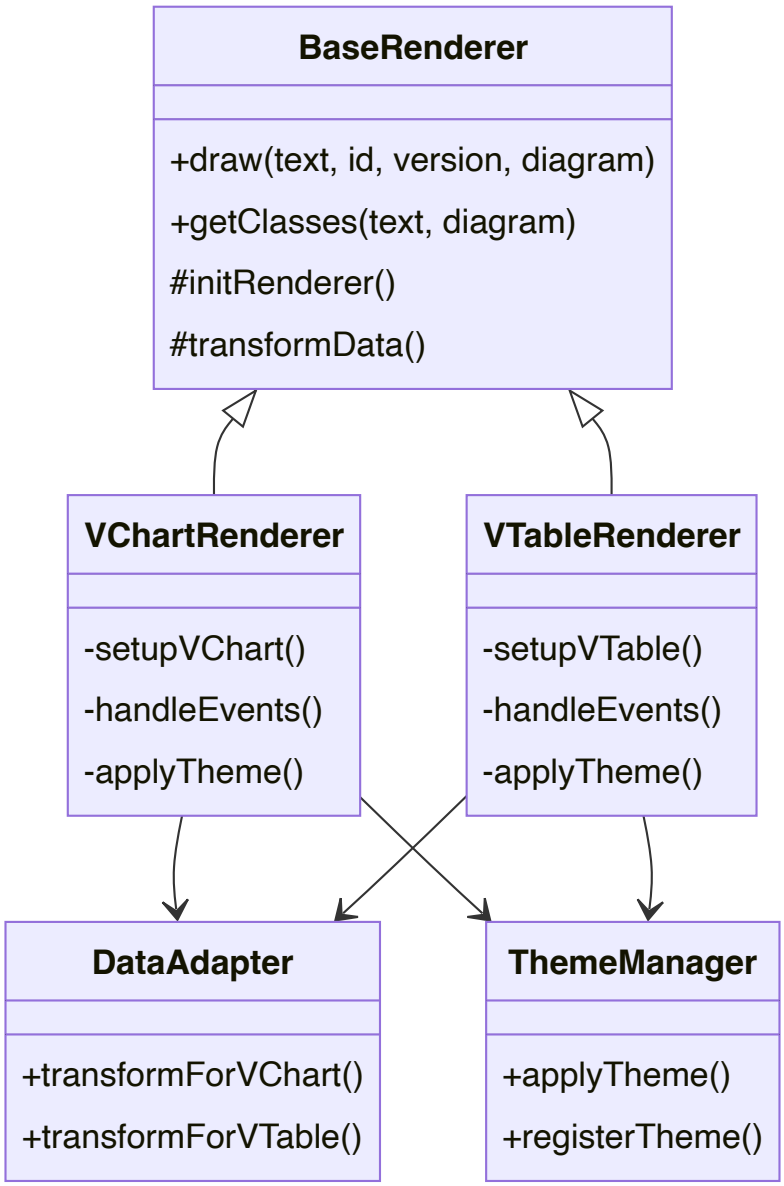
2. **性能问题**

- 大数据量下渲染性能较差
- 缺乏数据更新的增量渲染
- 动画效果不流畅

3. **可扩展性不足**

- 强耦合于D3.js
- 图表定制能力有限
- 主题样式难以统一

2. VChart/VTable集成方案



2.1 核心改造思路

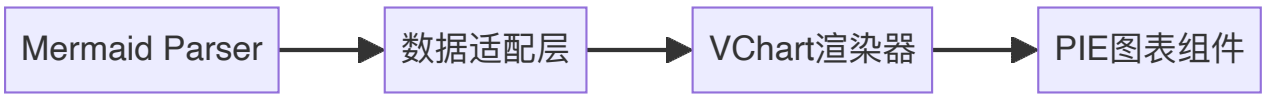
1. **保持API兼容**
   - 维持现有语法格式
   - 配置项平滑过渡
   - 渐进式替换策略
2. **能力增强**
   - 引入VChart丰富的图表类型
   - 提供更强大的定制能力
   - 支持复杂交互场景
3. **架构优化**

- 解耦渲染引擎
- 统一主题系统
- 提供插件机制

2.2 具体改造方案

2.2.1 替换原有图表

1. 饼图改造



- 保持原有语法和配置格式
- 使用VChart饼图组件替换D3实现
- 增加新特性(如环形图、动画等)

2. 桑基图改造

- 迁移到VChart的桑基图实现
- 增强节点交互能力
- 优化性能和动画效果

3. XY图表改造

- 使用VChart的Plot系统
- 支持更多图表类型组合
- 增强坐标轴配置能力

2.2.2 扩展新图表类型

1. 图表分类

- 基础图表(柱状图、折线图等)
- 关系图表(树图、网络图等)
- 统计图表(箱线图、热力图等)

2. 实现策略

- 分批次引入新图表
- 提供统一的配置模式
- 完善示例和文档

2.2.3 VTable集成

1. **表格语法设计**
   - 兼容现有markdown表格语法
   - 扩展高级表格特性配置
   - 支持样式定制
2. **渲染实现**
   - 独立的VTable渲染器
   - 表格组件定制封装
   - 主题样式统一

2.3 优化方向

1. **性能优化**
   - 按需加载组件
   - 大数据渲染优化
   - 缓存机制优化
2. **交互增强**
   - 图表联动
   - 缩放平移
   - 数据筛选
3. **可视化增强**
   - 主题定制
   - 动画效果
   - 响应式适配

时间规划

项目开发将按照开源之夏的时间节点进行规划，预计总工期约3个月。作为本科大二在读学生，暑期可以投入充足的时间进行开发。以下是具体的时间规划：

项目开发第一阶段（7月1日 - 7月31日）

1. **第1-2周：基础工作**
   - 深入学习Mermaid源码
   - 熟悉VChart/VTable API
   - 搭建开发环境
   - 设计适配层架构
2. **第3-4周：图表替换**
   - 实现VChart渲染器基类
   - 替换饼图实现并优化
   - 替换桑基图实现并优化
   - 替换XY图表实现并优化
   - 单元测试和文档

项目开发第二阶段（8月1日 - 8月31日）

1. **第5-6周：VChart图表迁移**
   - 设计通用配置模式
   - 迁移基础图表(柱状图、折线图等)
   - 迁移关系图表(树图、网络图等)
   - 迁移统计图表(箱线图、热力图等)
2. **第7-8周：VTable集成**
   - 设计表格语法规范
   - 实现VTable渲染器
   - 开发基础表格功能
   - 添加高级表格特性
   - 完善样式系统

项目开发第三阶段（9月1日 - 9月30日）

1. **第9-10周：优化与完善**
   - 性能优化
   - Bug修复

- 兼容性测试
- 交互体验改进

2. **第11-12周：收尾工作**
   - 完善技术文档
   - 编写使用教程
   - 制作示例demo
   - 提交PR并处理反馈

项目里程碑

1. 7月中旬：完成核心渲染器开发
2. 8月中旬：完成图表迁移
3. 9月初：完成VTable集成
4. 9月下旬：完成所有功能开发和文档

> 作为本科大二在读学生，有过丰富的React项目开发经验，可以保证暑假期间投入充足的时间进行开发。希望能通过这次机会为Visactor开源社区做出贡献，同时提升自己的技术能力。