WebGL绘制可交互式多边形网格与简单动画

WebGL绘制可交互式多边形网格与简单动画
- 运行环境
- 代码结构
- 附加功能&项目亮点
- 性能调优

> 13302010039 童仲毅

运行环境

浏览器版本限制：

1. 支持HTML5 Canvas以及WebGL的浏览器。查看浏览器兼容性。
2. 支持JavaScript (ECMAScript 6)的浏览器。查看浏览器兼容性。

代码结构

- `config.js` config对象，配置项 `canvasSize` `vertex_pos` `vertex_color` `polygons` `VERTEX_TOLERANCE`。
- `worker.js` Worker对象，主要的逻辑代码。
- `line.js` LineShader对象，绘制边线。
- `triangle.js` TriangleShader对象，填充三角形。
- `lib.js` 压缩合并后的依赖文件。

附加功能&项目亮点

- 旋转时编辑（说好的加分！）

  处于旋转状态时，暂停可编辑图形，按E依然将角度重置到初状态。由于转过的角度已知，通过一个旋转矩阵将当前状态点击的坐标转换成初状态时坐标，即可判断当前移动的顶点，和移动到的位置在初状态下的表示。之所以没有旋转时直接重置所有坐标，一是性能考虑，二是保证可以按E将角度重置到初状态。

- 良好的项目结构和代码风格

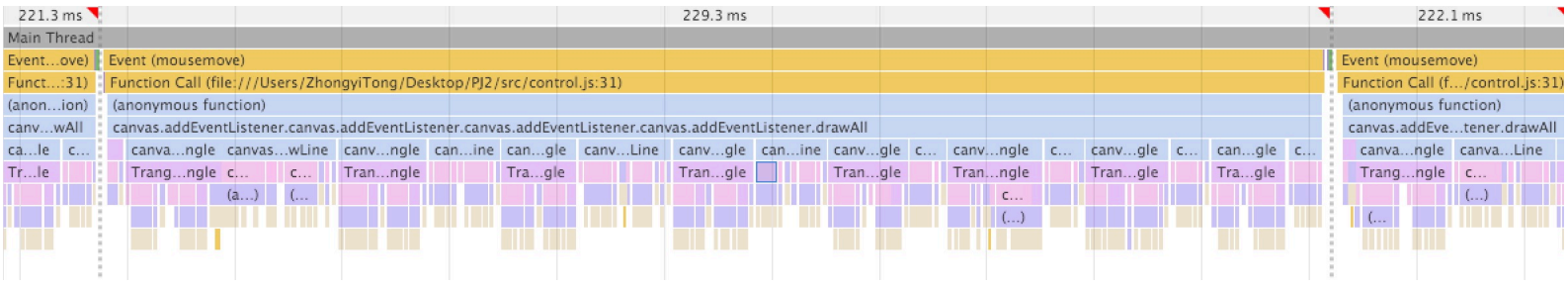
  项目全部使用ES6语法编写，分成三个类：`Worker`，`TriangleShader`和`LineShader`。

- 性能调优（见下节）

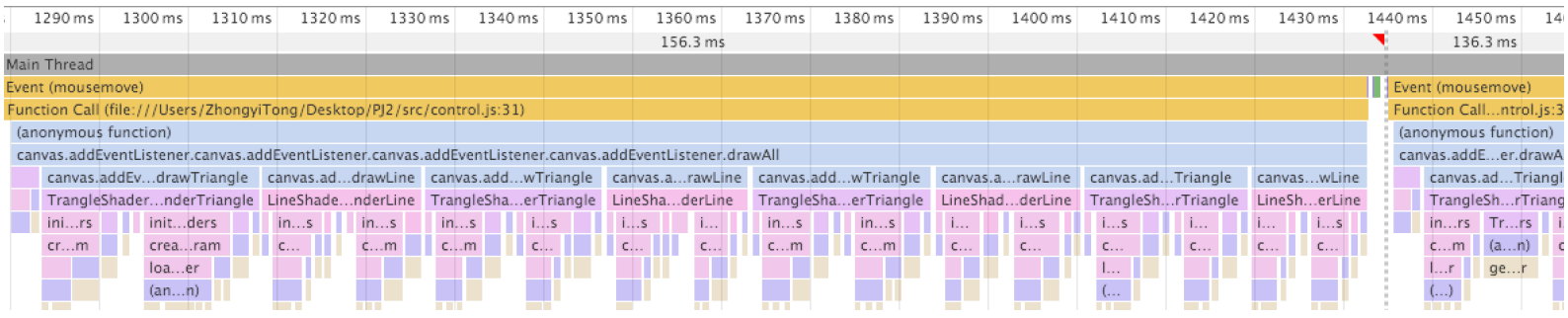
性能调优

这个PJ本身很简单，但由于对WebGL不熟悉，最初的实现在性能上很渣，因此后面在性能调优上花了很多时间...Chrome开发者工具是一个非常好用的东西，Teamline选项卡可以对代码进行profile。

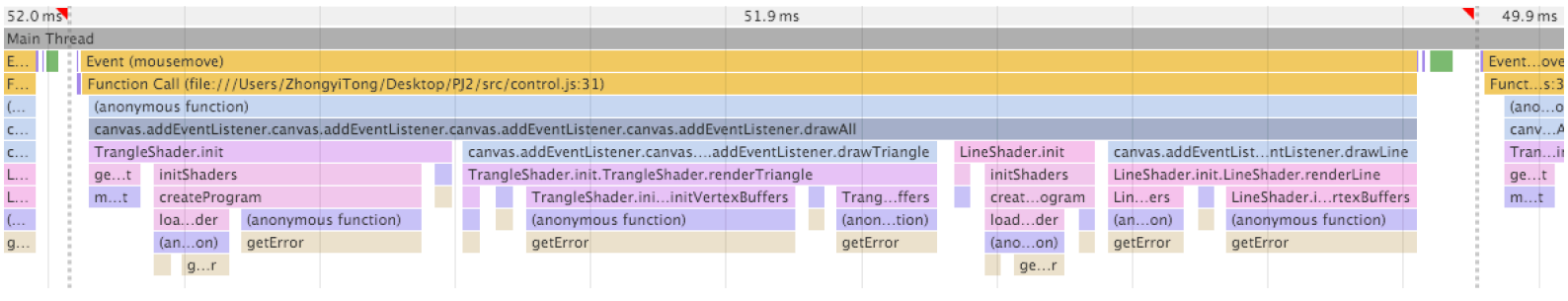
I. 最开始没有任何优化，程序每一帧遍历三角形和边线，分别调用TriangleShader和LineShader的绘制函数。Profile工具可以看出其中scripting占用了99%的时间，很明显代码写的有问题。



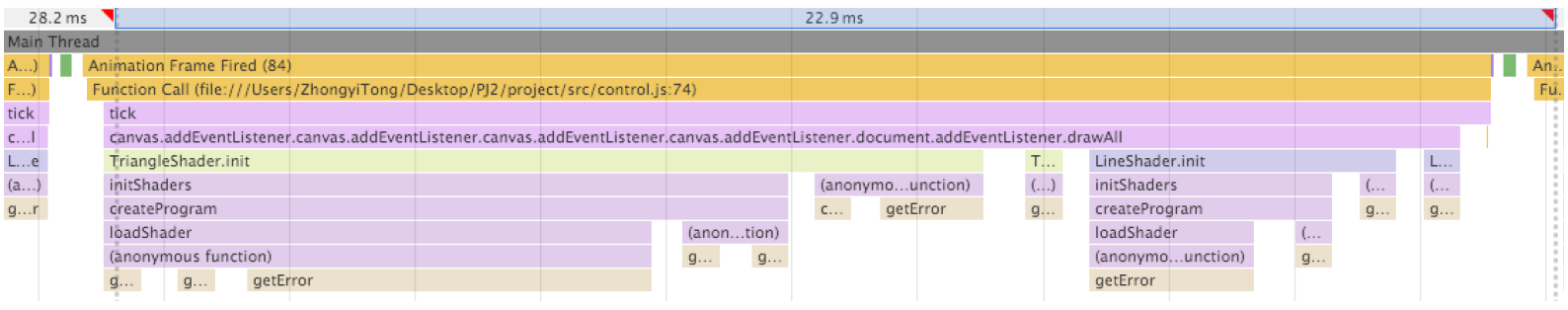
II. 首先存在的问题是，单个三角形和边线的绘制时间过长。由于每一帧总共需要调用8次三角形绘制和24次边线绘制，因此绘制函数对性能的影响会被放大。将部分初始化从绘制函数移到页面刚加载完成时，就能得到明显的性能提升。



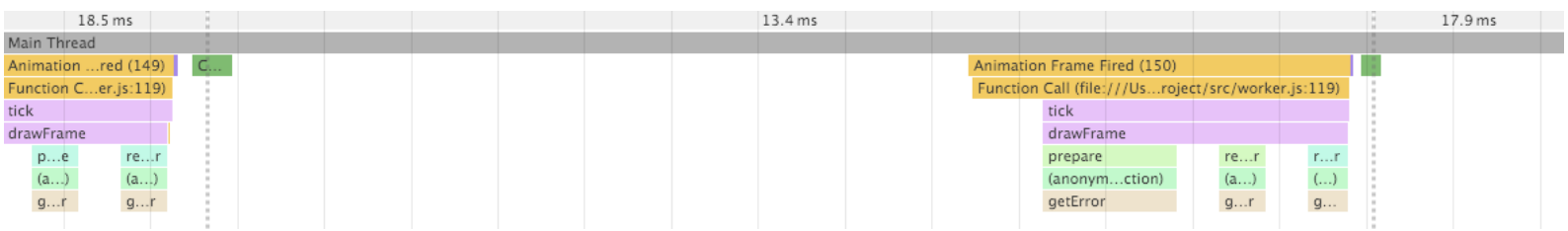
III. 观察JS调用栈，性能瓶颈存在于initShader，它占用了每一次绘制约50%的时间。而事实上只需要在切换Shader时重新init。于是将原来的循环拆成两个，每个循环分别绘制三角形和边线，这样就只需要在每个循环前调用initShader一次。



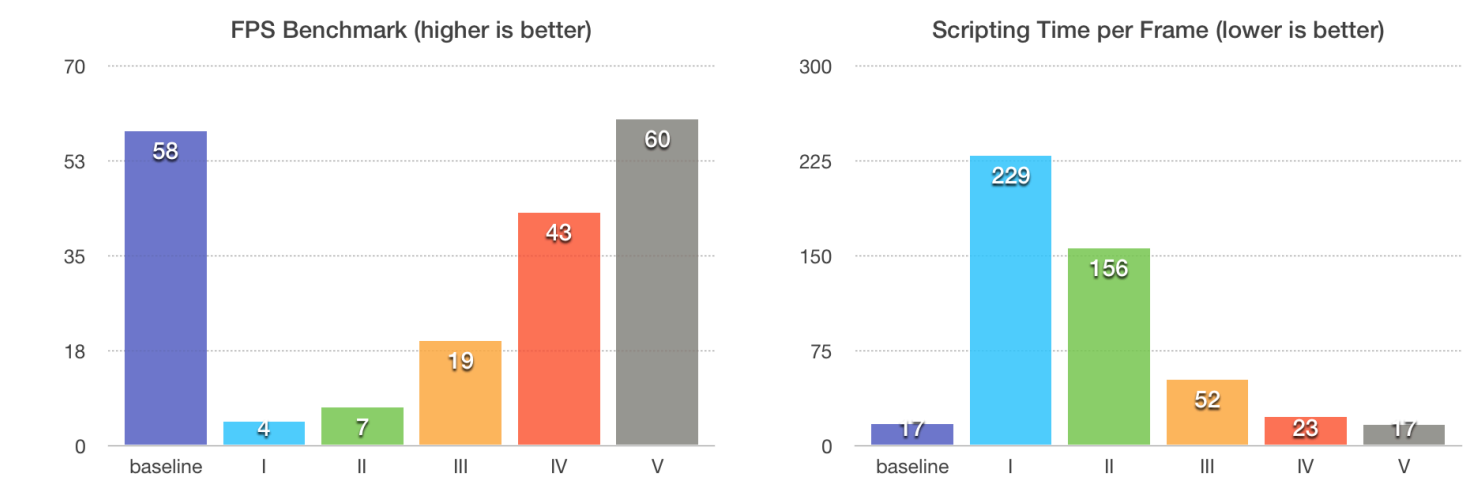
IV. 看WebGL API的时候想到，之前每次循环分别传入一个三角形或边线进行绘制。事实上，vertices数组可以是所有三角形或边线的集合，这样一次绘制大大减少了执行时间。



V. 现在其实性能已经比较满意了，然而调用栈那个initShader的函数看起来特别扎眼。于是就去翻了下webgl-debug的源码，发现其中主要做的事情是initBuffer，而这件事可以在最开始页面加载完的时候做，后面切换shader的时候只需要重新bindBuffer。这样一优化，程序跑的比谁都快。



下面用助教给的demo作为baseline，FPS作为benchmark来衡量程序性能。



总之就是这样了...主要写下做PJ的过程，其实中间还做了些愚蠢的事情，比如去研究好久GLSL想改写Fragment Shader实现边线和内容同时绘制，发现是理论上可以不过太坑了等等。过程还是挺有趣的。