优化项目结构

为什么要优化项目结构？

其实到这个部分的时候，我们的tank游戏大部分的工作已经完成。还是我们还要完善游戏的机制和细节，优化项目结构的目的在于晚上游戏的机制，我们需要在一个文件中控制游戏的开始与结束，同时给我们的tank游戏加上音效。

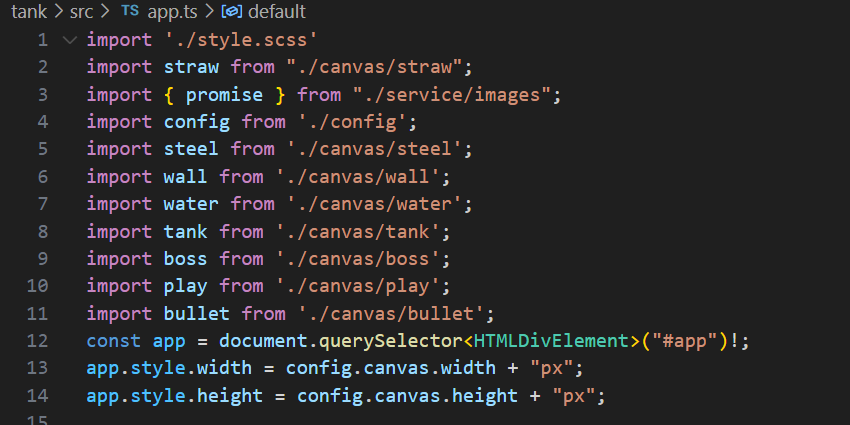
具体要做的工作有3点

1. 如何控制游戏的开始
2. 如何结束游戏
3. 如何完成模型爆炸效果的渲染

接下来我将介绍如何实现。

控制游戏的开始

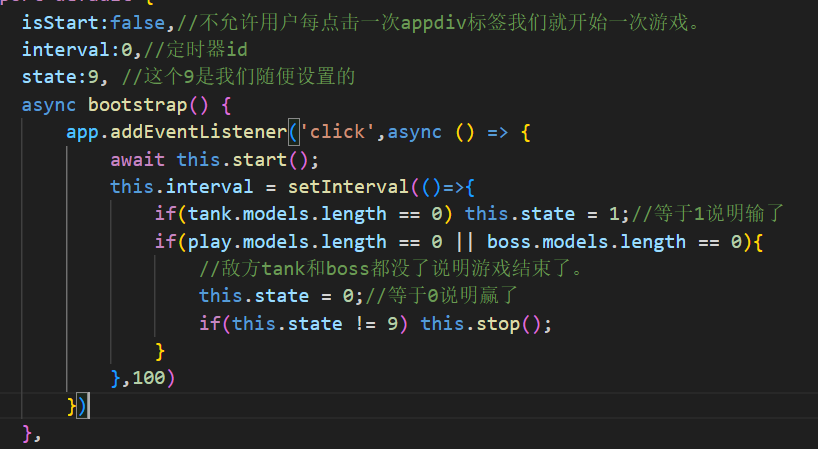
我们可以单独创建一个app.ts文件，然后先将我们先前在main.ts中的一部分代码移动过去。



然后我们在app.ts文件中默认导出一个对象，游戏的开始和结束我们都通过这个对象导出的方法来实现。

游戏的开始

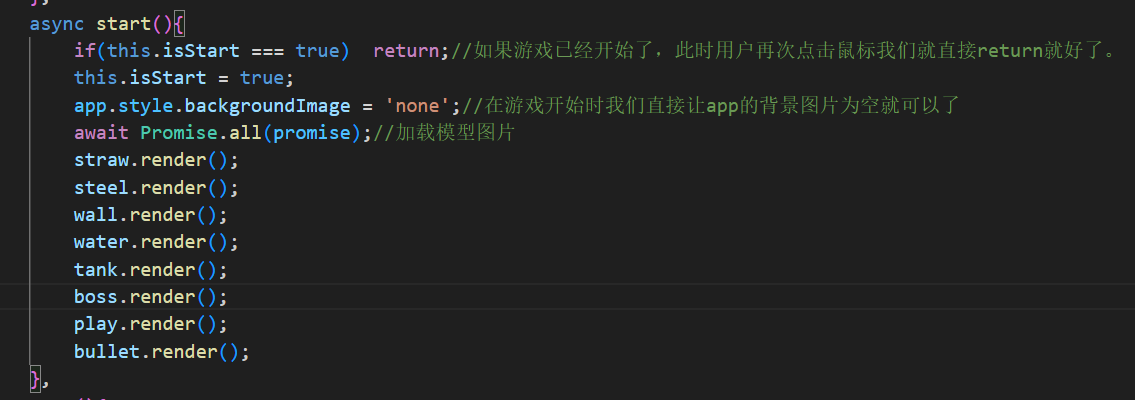
我们可以在style.scss中直接给app这个div标签设置一个背景图片（因为此时还未正式开始渲染画布，所以页面上会先显示app元素的背景），然后在app.ts中设置一个bootstrap方法，通过在这个方法给app元素添加一个鼠标点击事件用于开始游戏，在开始游戏后，我们需要设置一个定时器函数，用于不断的检查游戏是否结束，一旦游戏结束的条件满足，我们就正式结束游戏。



Bootstrap方法用于开始游戏

渲染画布并真正的开始游戏

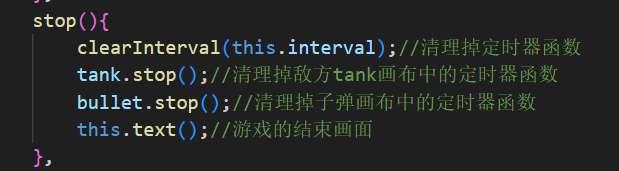
我们先前在main.ts文件中通过调用画布实例的render方法渲染不同的画布。我依旧通过这种方式实现游戏的真正开始，这里我们还是老套路，我们在start方法种先判断用户是否已经点击过app元素开始了游戏，如果已经点击过，我们直接return就可以，否则我们还是按照在main.ts种的老套路即可，但是由于我们一开始在style.scss文件中给app设置了一个初始的背景的原因，所以我们需要在正式渲染画布之前把app上的背景图片给清除。



Start方法。

游戏结束停止渲染画布

不可能游戏都结束了画布还在不停的渲染，所以在正式在画布上提示“游戏结束”的字样的时候，我们需要关闭定时器函数，停止渲染画布。停止渲染画布就是使用clearInterval方法清理定时器函数。我们先前在开启定时器函数的时候保存过定时器函数Id。所以这不是什么难事。



定制渲染画布清理定时器函数

绘制“游戏结束”

这个逻辑非常简单就是在app这个div标签中再追加一个用于显示“游戏结束”四个字的画布而已。



绘制“游戏结束”的方法。