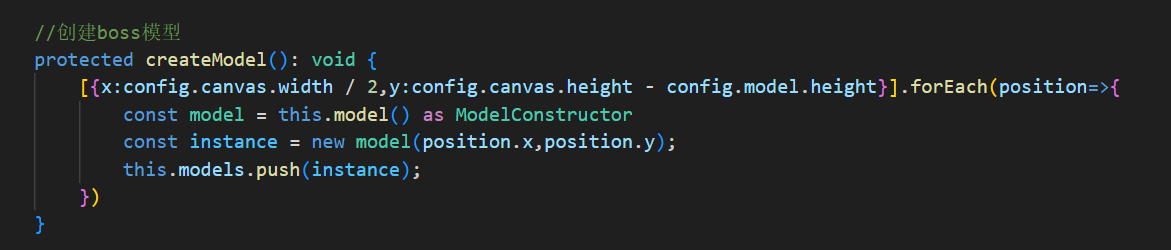
绘制boss模型

Boss需要绘制在画布的下方的中间，是一个固定位置，所以我们需要在boos的画布类中重写createModel方法。在创建模型的时候就给boss模型的位置定好就行了。



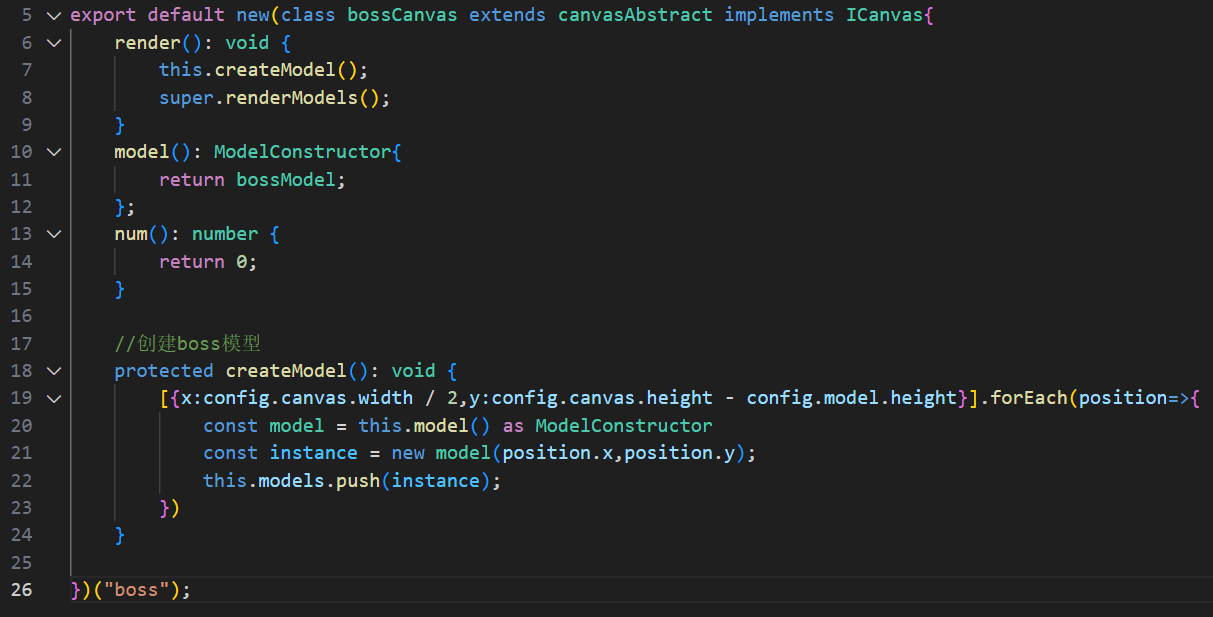
创建boss模型的方法

绘制boss围墙

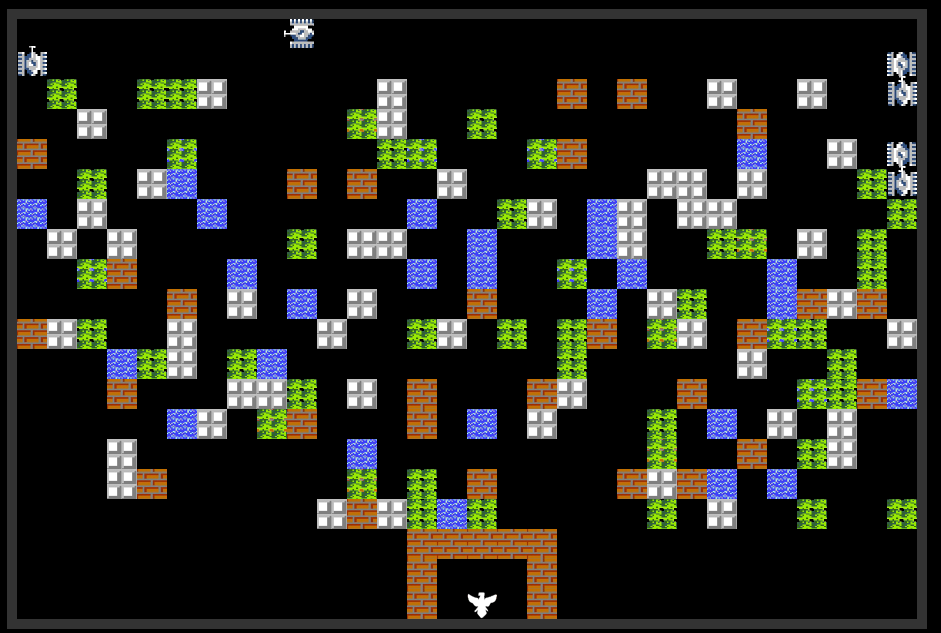
Boss的围墙也是红色的砖墙，所以我们可以在wall的画布类中单独写一个用于创建围墙模型的方法，并在方法中将创建好的模型压入到wall画布类实例的IModel数组中。这样方便后续做子弹的碰撞检测，因为它不必再额外创建一个画布类。绘制围墙的算法特别简单，是一个固定的讨论。这里我直接将其是如何绘制的贴出来。



绘制围墙的方法



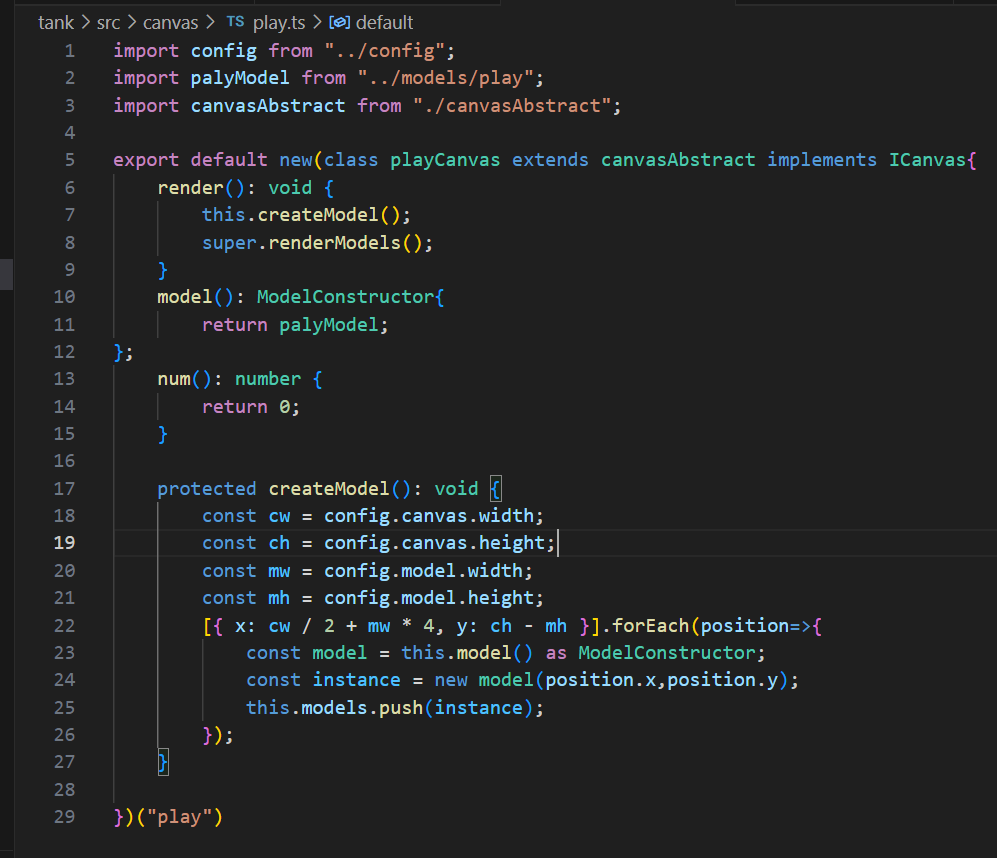
Boss画布类完整代码



效果图

创建我方tank实例

创建我方tank的实例交给我方tank的画布类来完成。这个实例在画布上的坐标我们就不取随机位置了，让其固定在boos围墙的右边生成即可。



在paly画布类中我们需要重写createModel方法。

让我方坦克动起来

让我方tank动起来其实非常简单，我们需要做的就是在当按下键盘上的“上下左右”按钮的时候，我们先根据用户按下的按钮改变tank的炮头方向（其实就是切换图片），然后再将tank的坐标按照上减下加（y坐标）左减右加（x坐标）的规则（drawImage方法规定画布的左上角坐标是[0,0]）改变后再重新渲染画布即可。



改变我方tank炮头方向的方法。



实现我方tank移动的方法

但是我们还需要注意一点：不可能每按下一次键盘上的方向键，我们就给页面(document对象)绑定一个键盘事件处理函数。一般来说我们只需要绑定一次就可以了，所以我们可以在玩家坦克的模型类中设置一个变量，用于控制。当这个变量的值为false的时候我允许向页面上添加事件，而当这个变量为true后不再允许向页面上添加事件。还有一点，在绑定事件处理函数的时候需要注意this的指向问题，普通函数内部的this指向根据函数的执行（调用）方式不同而不同，绑定在document上的事件处理函数，在通过键盘事件触发这个函数时，函数的this就指向绑定的对象。而箭头函数没有自己的this指向，依据函数的执行上下文而来，且在某些情况下箭头函数内的this指向undefined（箭头函数作为定时器函数、数组的迭代函数等），只不过在非严格模式下，当发现this指针为undefined的时候，Js会自动帮你将this绑定为globalThis。(全局this，在没有nodeJs和webworker之前，其实全局对象就是指window,但有了nodeJs和webworker之后，Js全局对象就有了3个，分别是window、process和self，所以Js又提供了一个globalThis属性，让你可以在不必考虑当前Js的执行环境的情况下直接获取当前执行环境下的全局对象)



实现我方tank的移动。