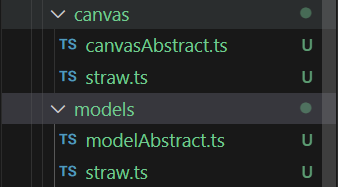
在了解了canvas后，我们知道这个项目是完全可以使用canvas实现的。但要如何在画布上绘制我们想要的图案呢，又应该如何管理这些图案呢？

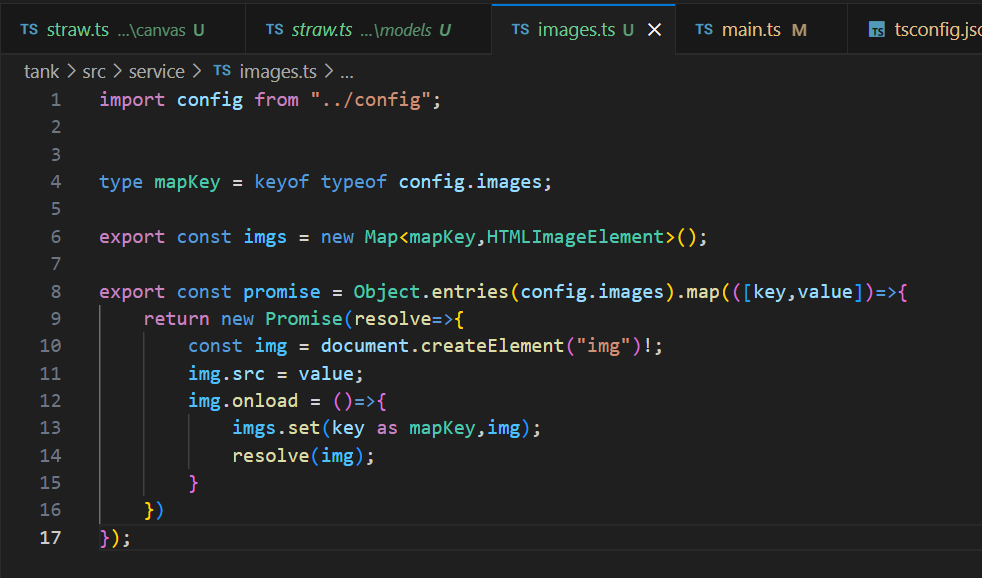
我直接来说思路好了，我们可以创建一个canvas类，不同的canvas用来绘制不同的元素，而具体的元素我们也使使用class类来创建，我们就叫其模型类好了。



但是现在请想一下，strawCanvas类中应该有那些属性和方法？而model模型类中又应该有那些方法？他们之间应该如何配合来完成在画布上绘制元素这个工作呢？我们可以先将这些问题放在一边，因为我们需要先考虑的是：模型的图片如何加载，模型的位置（在画布上的位置）应该如何生成。

我们先来说模型的图片应该如何加载，这里我们可以先创建一个用于加载图片的文件，且加载图片的任务应该在项目初始化的时候就要完成，这样在后续的程序使用到图片的时候，才能保证程序不会报错。我们先来看一下这个文件应该怎样书写：

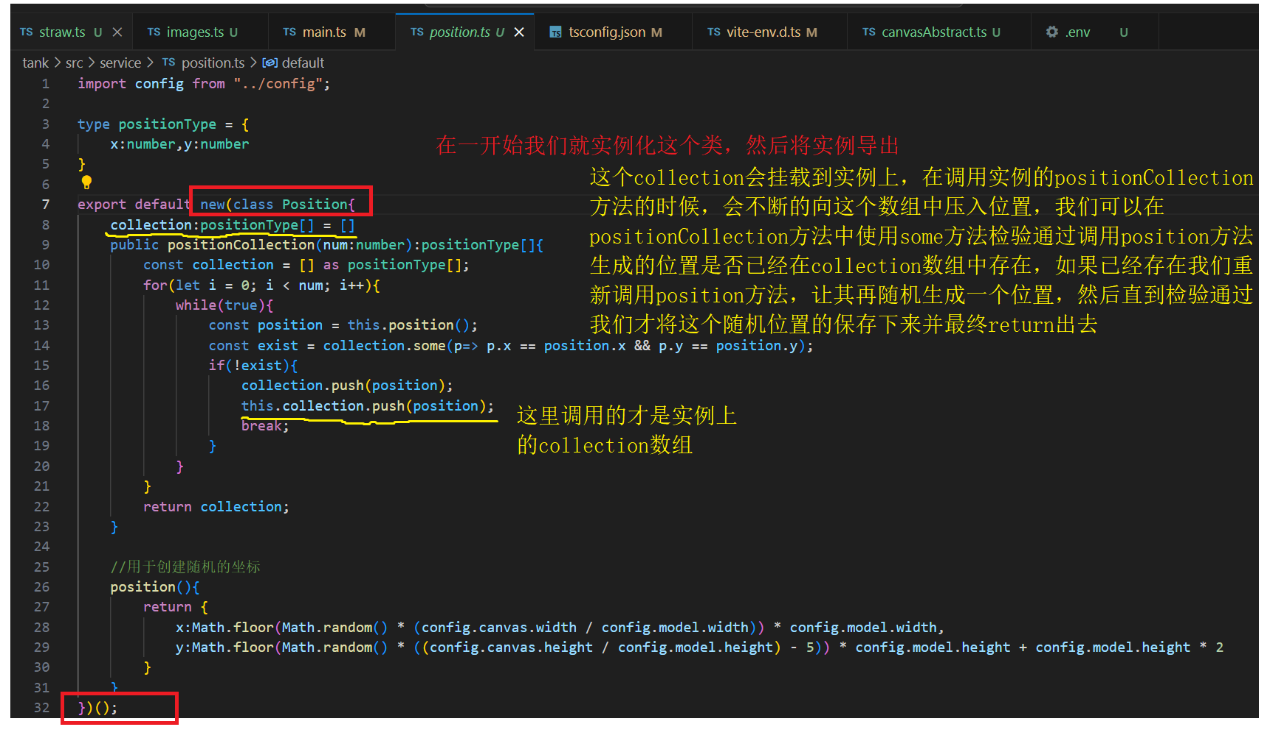
我们可以用一个map类型的数据来保存好加载好的img对象。利用promise创建的微任务来实现图片的异步加载。我们来看一下具体的文件：



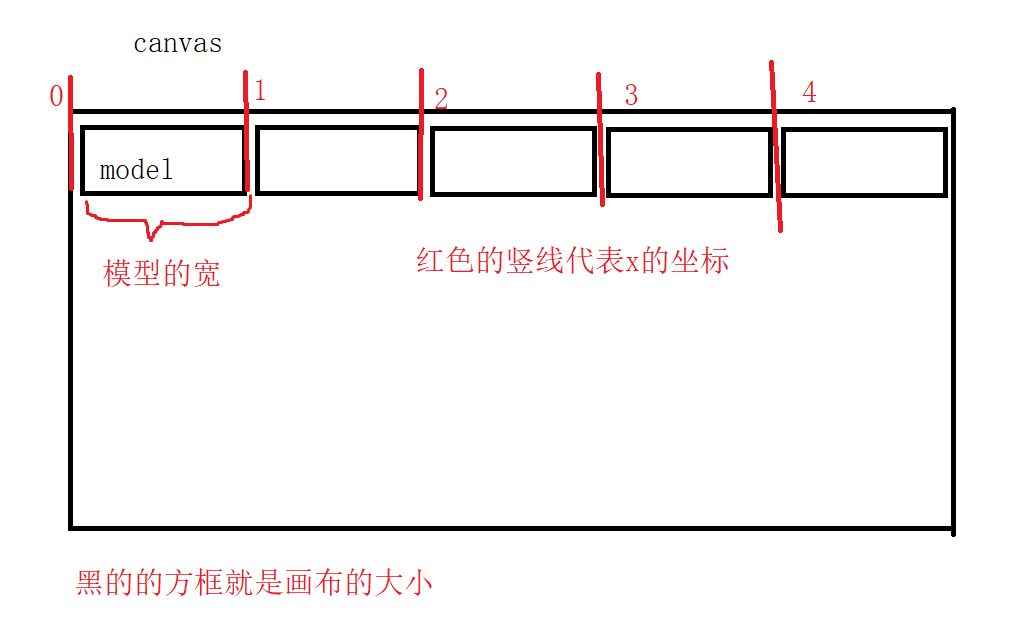
但是在真正加载图片的时候，我们还需要用到Promise给我们提供的all方法，只有所有的img.onload方法都执行完的时候，整个Promise.all()才会返回一个promise结果的有序数组。同时我们利用await 函数，将其异步阻塞掉，保证它在整个程序开始之前执行完成。



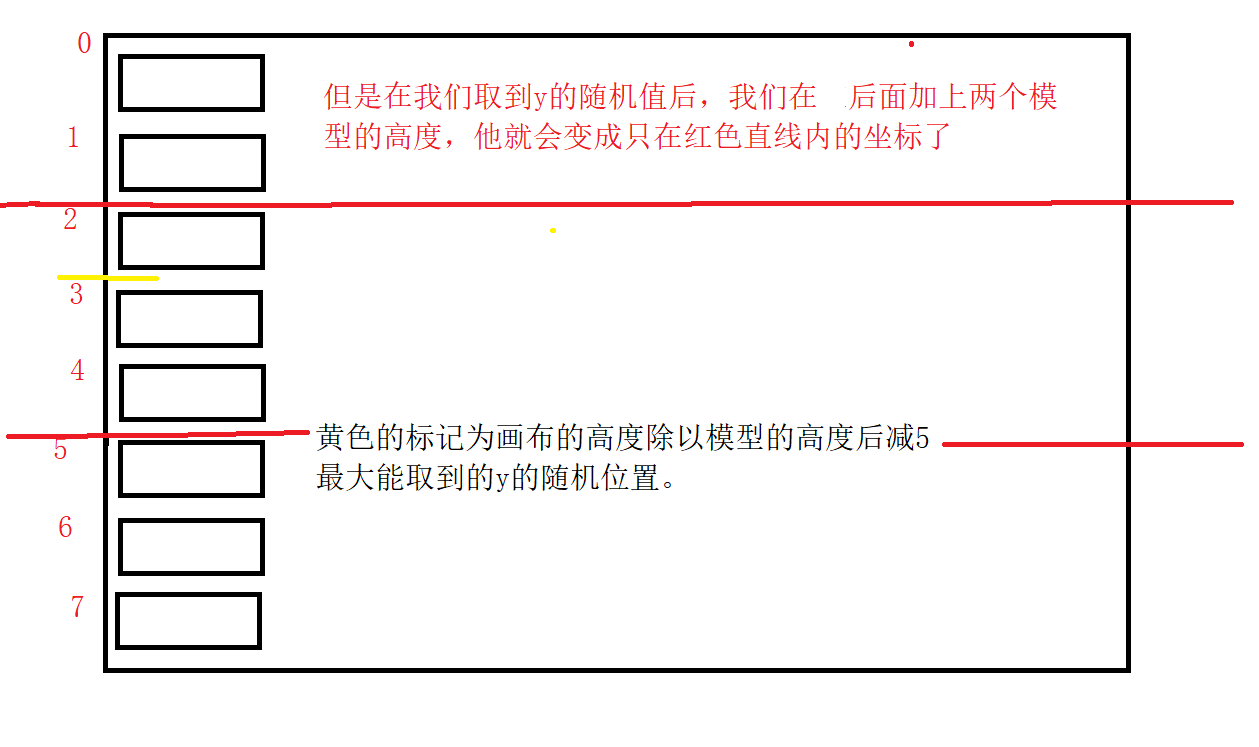
现在我们讨论模型的位置应该如何生成，模型的位置应该是随机的，且其位置不能重复产生（一个元素覆盖到另一个元素上）。而且由于后续的几乎所有元素的位置都是随机生成的，所以我们还是单独写一个提供位置服务的文件好了，通过不断的调用这个文件给我们提供的生成随机位置的方法，来实现在画布的任意位置渲染模型图片的效果。这个文件我们分两部分写，第一部分是用来保证生产的随机位置不会重复，另一部分则需要考虑的是如何生成随机位置。

我们先来讨论第一部分：  
假设文件中我们已经有了一个用于生成随机位置（x,y）的方法，只要我调用它，它就能返回给我一个随机的坐标。那我们就可以使用类属性的方式来保存这个生成的随机位置，用于检验位置是否重复的方法也放到这个类中，只实例化一个实例并导出这个实例，在其它文件中通过导入这个实例，然后调用其中的方法的方式来获取永不会重复的位置。我们来看一下具体的代码：  


现在我们考虑应该如何生成随机位置，我们知道模型的宽高是固定的，我们可以将画布的宽高按照模型的宽高等分，然后利用Math.floor()方法取0~1之间的小数（不包含1）乘以等分的份数后再乘以模型的宽度得到x的随机位置。我画个图方便理解：



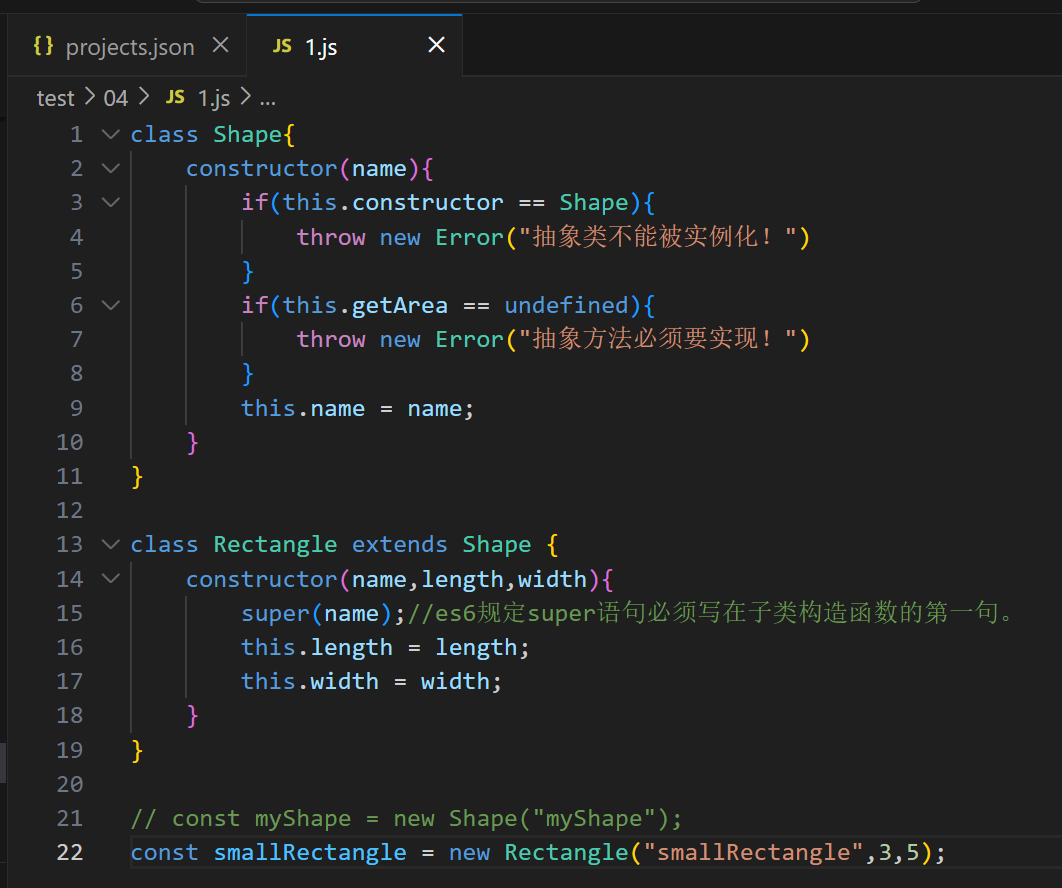
“y”的坐标同理，但是我们还需要考虑的是，要将画布的上面留一部分生成敌方坦克，下面留一部分绘制我方boos，上面留两个模型的高度，下面留三个模型的高度即可。



现在我们来考虑应该如何绘制模型。

为什么要使用ts给我们提供的抽象类。

Ts给我们提供的抽象类，可以在其中写抽象方法和抽象属性，也可以写普通方法和普通属性。继承了抽象类的类其中必须要有在抽象类中定义的抽象方法和抽象属性，且可以使用抽象类的构造函数生成实例，也可以通过这个生成的实例直接调用抽象类中的普通方法和属性，同时你也可以在子类中复写抽象类的方法。但是请你注意，Js本身并没有给我们提供抽象类这个写法，想要像Java一样有抽象类，需要我们自己手动封装。我们这里只是借用ts给我们封装好的代码而已，它实际上还是Js的普通类。为了防止你迷惑，我们还是来看看在Js中是如何实现抽象类这个类的。



如果直接运行上图的代码，它会报“Error: 抽象类不能被实例化！”的错误。

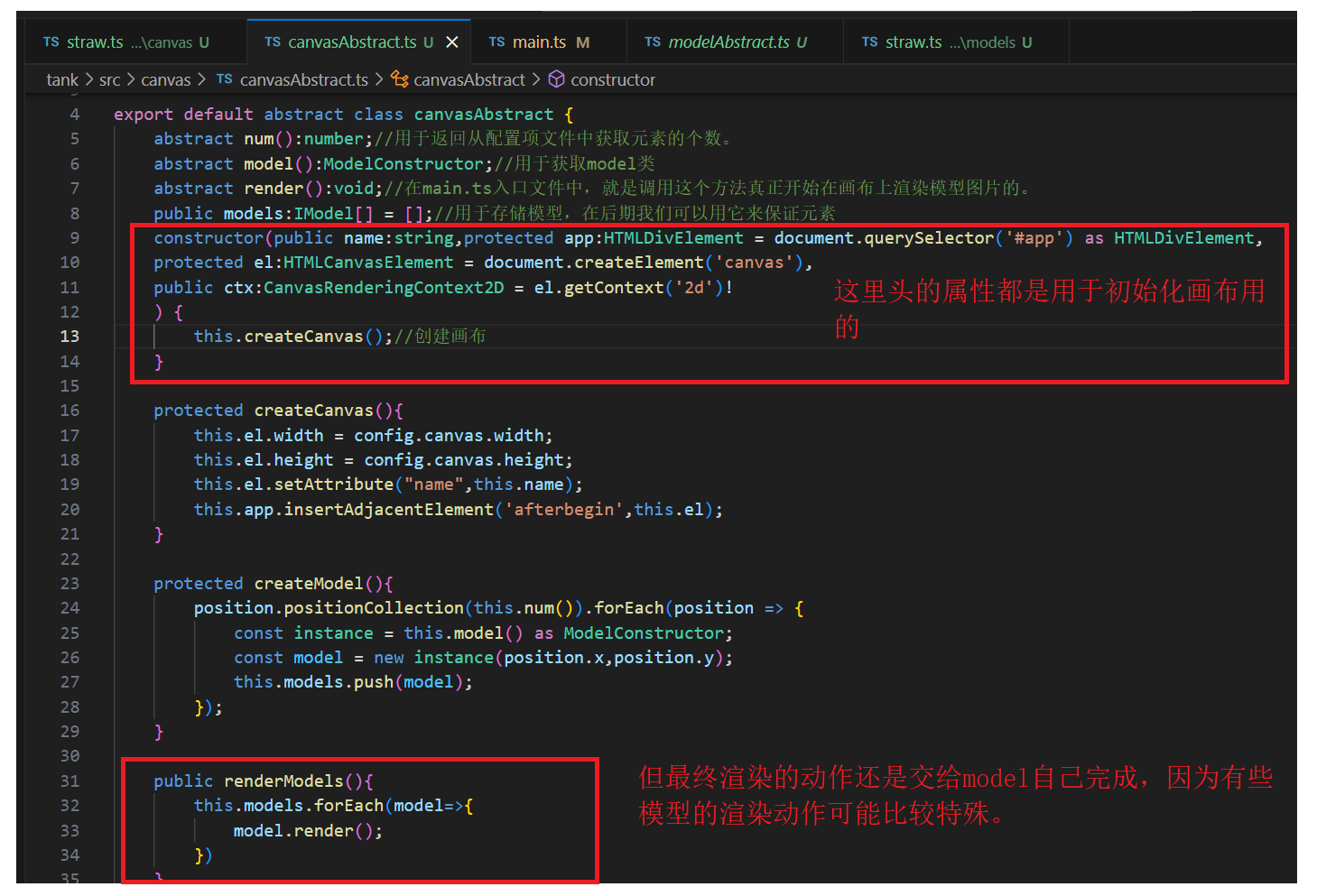


当我们将创建Shape类的实例的语句给注释掉后再运行这段代码，报的就是“Error: 抽象方法必须要实现！”的错误了。

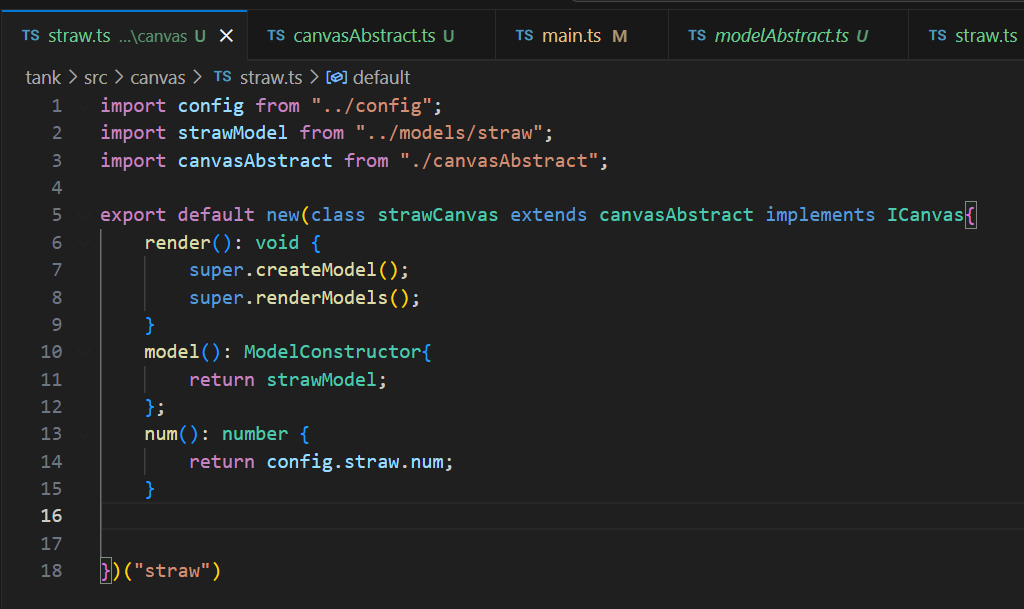


Js类是基于构造函数实现的，先前讨论过如果通过Js的构造函数来实现oop思想。如果您能理解Js的prototype原型链的化，理解这个应该是水到渠成。

在了解了ts给我们提供的抽象类后，因为我们先前讨论过一个canvas画布类用于绘制一类元素，每个画布类都需要有canvas画布元素，画布名称、渲染图形的方式等属性(canvas的contextType，不同的contextType有不同的方法向画布上绘制图形)，还要有向画布上渲染元素的方法，所以这里我们干脆将这些方法都抽取到抽象类中。我们先来看看一个基础的canvas抽象类应该如何定义。



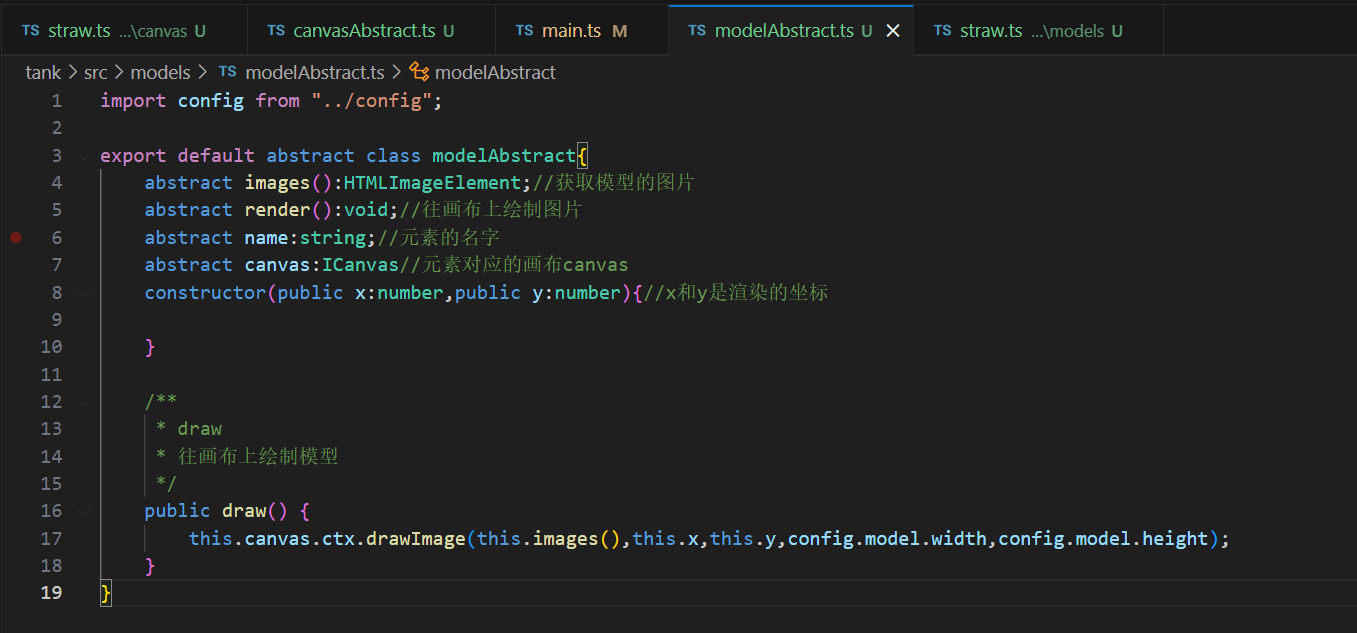
画布的抽象类



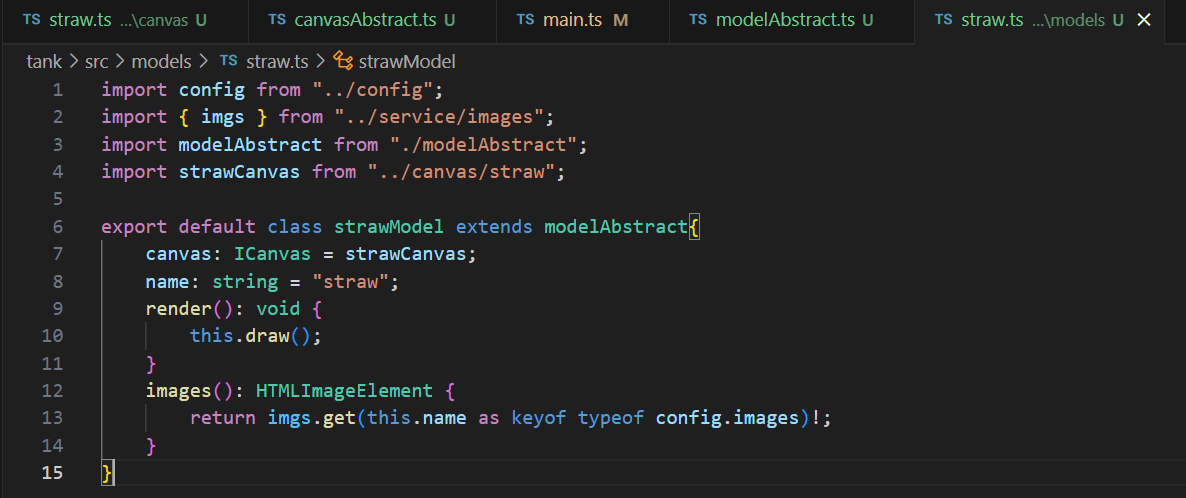
用于渲染草地的strawCanvas类

现在我们来考虑一下模型的抽象类应该如何定义

因为考虑到后续做碰撞检测的原由，应当是每个模型都有它自己的在画布上的坐标。另外因为真正的渲染动作是model类自己做的，所以我们需要将画布导入到每一个模型类的文当中。



模型的抽象类



草地Straw类文件。