**一、前言**

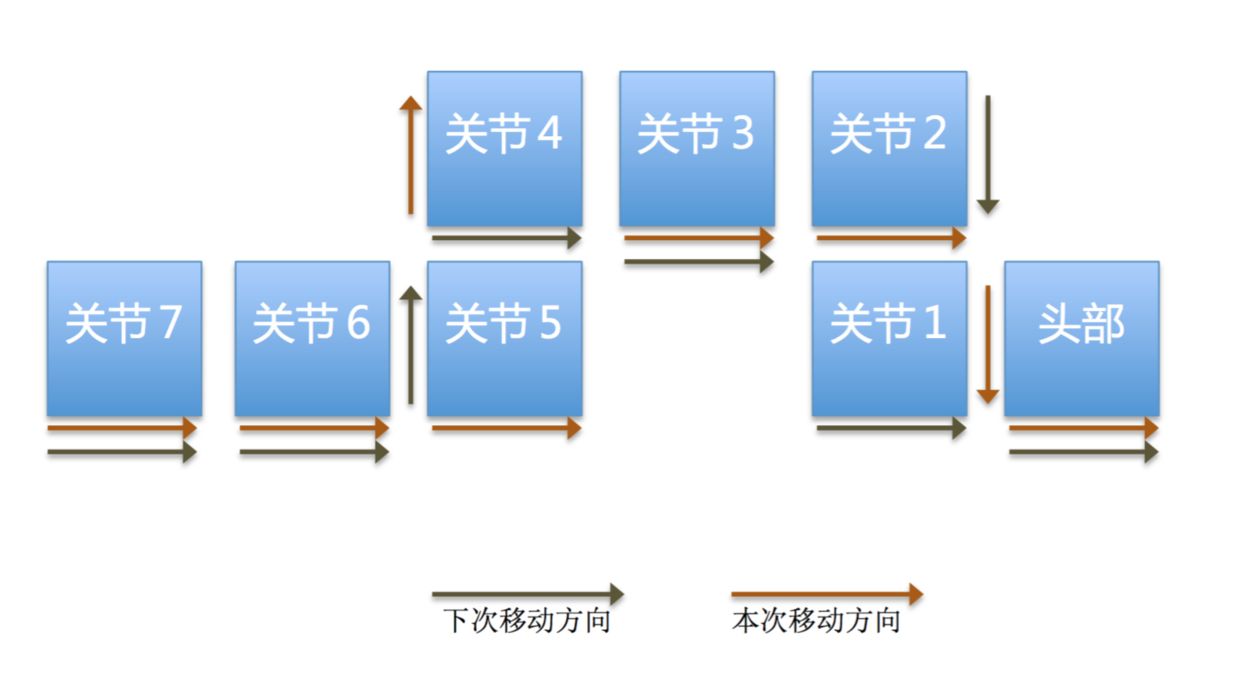
相信贪吃蛇大家都玩儿过，我对贪吃蛇的印象就是在电子词典上，一只像素蛇在屏幕游走，饥渴难耐，看着豆子就要去吃，吃到豆子就会长一节，当蛇的身体越来越长的时候，它才发现这个世界变了，每走一步，都是寸步难行。当它的蛇头触碰到任意的物体，无论是屏幕边界还是自己的身体，游戏都将结束。这款游戏应该是比较经典的一个童年记忆。刚接触游戏开发的人可能比较喜欢以这款游戏入手，因为贪吃蛇包含了很多游戏开发中的原理，并且难度也不大，而我刚好在学习Cocos CVP的课程，学习在一个中间阶段，我也来拿这个练练手，以下就把我做贪吃蛇的过程分享出来。

**二、游戏分析**

**1.身体关节**

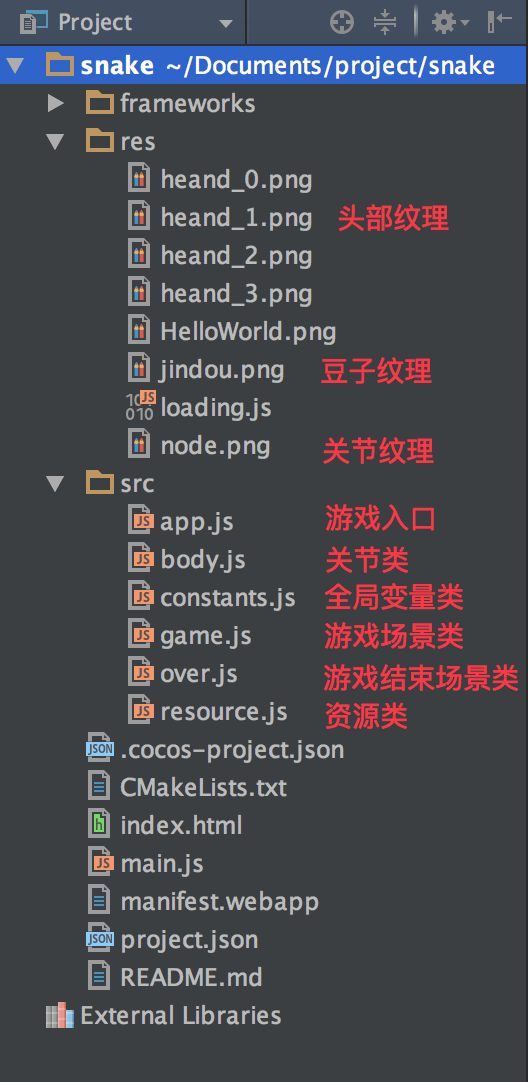
贪吃蛇的实现可能有多种方法，但今天，我想用面向对象的思想来对游戏进行设计，到今天，任何的程序开发都离不开面向对象的思想，通过面向对象的思想我们能把很多抽象的问题具象化，方便我们解决很多问题。而在贪吃蛇中，面向对象的思想依然实用。   
在贪吃蛇中，我们可以把一条游走的蛇的每个关节当做是一个对象，而蛇本身是由多个关节组成的整体，当每个关节在移动时，我们就能看到整个蛇的移动，每个关节的位置以及移动方向都跟它的上一个关节息息相关，那么我们就可以把关节与它的上一个关节关联起来，实现如下结构：   


**2.移动方向**

如上文所说，按照上面的关节关系来实现，那么蛇的移动方向就是与父节点的移动方向相关联，每一个关节应该有一个当前移动方向和下次移动方向，每一步的移动，都是跟着当前移动方向走的，而父关节的当前移动的方向即为子关节的下次移动方向，这样，只需要调整蛇头关节的下次移动方向，整条蛇就能顺着各自的父关节方向移动，蛇的移动方向图如下：   


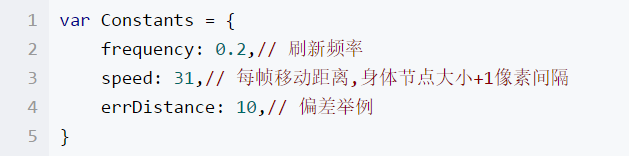
**二、开发设计**

**1.项目结构**

按照前面的设计，我们可以大致可以划分出游戏场景类和关节类，进入游戏场景类，再将一些全局的变量单独存在一个类中，项目结构可划分如下图：   


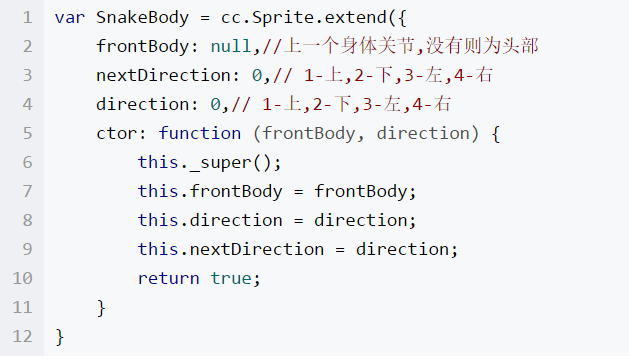
**2.全局变量类**

游戏中的全局变量，提炼为一个全局变量类，其中参数可以根据需求灵活变动配置



**3.关节类**

按照以上设计的结构，每一个关节对象都应该包含蛇的当前方向、下次方向和蛇的父节点三个属性，代码如下：



上面说的，关节的位置跟它的福关节的位置是息息相关的，那么初始化的时候，我们就需要根据父关节的移动方向来进行次关节的位置设置，这部分代码，我们可以放在onEnter方法中，代码如下：





• 有了这三个属性，每一个节点还应该有最重要的游戏逻辑——move方法，每一个关节分别调用move方法，从游戏场景中就能看到整条蛇按照预定方向进行移动，而整条蛇的运动方向就是跟着头部关节的方向走，头部关节的方向则通过点击屏幕区域控制。

在move方法中，我们需要做以下事情：

1.按照关节的下次移动方向移动本身长度的像素的距离

2.如果是头部关节，需要改变关节纹理

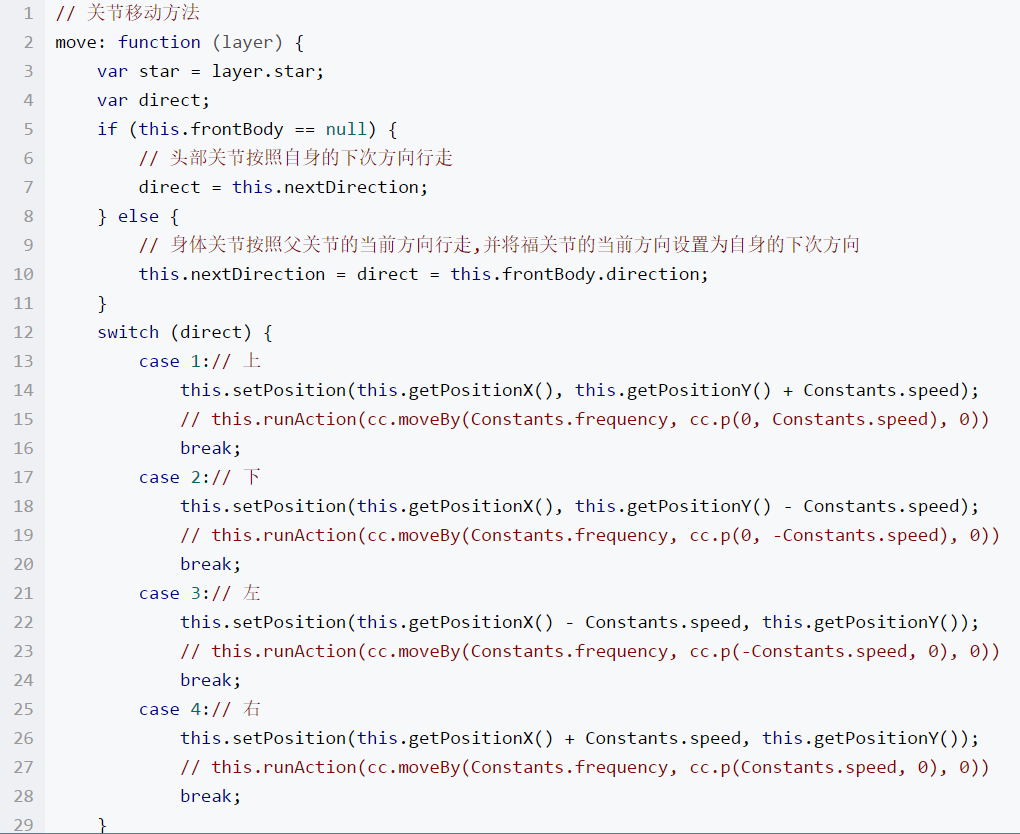
同时，如果是头部关节，我们还需要判断以下三个临界条件：

1.头部关节是否触碰到屏幕边界

2.头部关节是否吃到屏幕中的豆子

3.头部关节是否触碰到自身关节

其中1、3条件达成，则判定游戏结束，2条件达成，则能增加游戏分数，并且游戏继续。   
move方法代码如下：





![C:\Users\Dell\AppData\Roaming\Tencent\Users\44767925\QQ\WinTemp\RichOle\~C(AR2$H4}NP(UW](@@1W2M.png](data:image/png;base64,)

**4.游戏场景类**

在游戏场景中我们需要以下几个变量：

1. 贪吃蛇数组：用于存储贪吃蛇所有的关节节点

2. 贪吃蛇尾部：每添加一个关节节点 ，都将此变量指向这个新加的节点，以便下次继续再尾部节点添加

3. 吃的星星：屏幕中随机产生的星星，用于判断头部关节是否与它产生碰撞

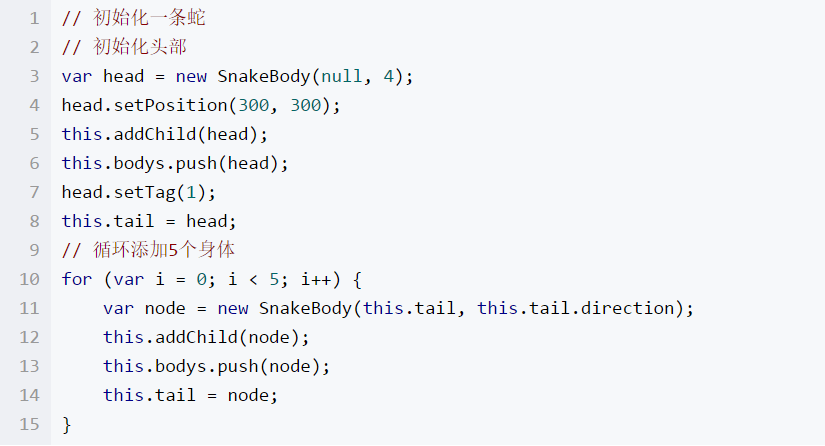
4. 是否添加节点：如果在定时任务中判断到吃到星星，那么可以次变量为1，代表可以添加一个节点

5. 分数：存储游戏中累加的分数

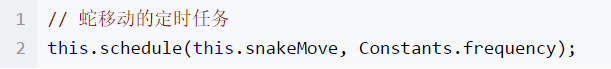
在cc.Layer的构造函数中对以上变量进行初始化，代码如下：



之后，我们首先需要在场景中绘制出一条蛇，初始化定义为1个头部关节，5个身体关节，由于我们对关节类做了很好的封装，所以初始化一条蛇的代码很简单，我们在onEnter方法中进行初始化，如下所示：



初始化完了之后蛇是不会动的，如何让它动起来呢，我们就要用到在关节类中封装的move方法了，我们每隔一个时间，对所有的关节类执行一次move方法，就能实现蛇的移动，首先在onEnter中添加定时任务：



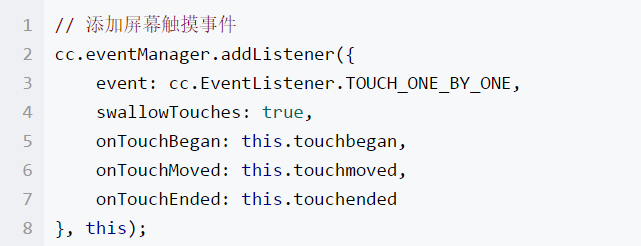
在这个snakeMove定时调用的方法中，我们要写出所有关节移动的逻辑，在这个方法中，我们需要完成以下几件事：

1. 遍历蛇的所有关节，每个关节执行一遍move方法，并在move完了之后，将下次移动方法变为本次移动方向

2. 如果需要新增关节，在遍历完成之后，新增一个关节类，并将其父节点指向之前的蛇尾节点，并把蛇尾指向新加的这个关节

代码如下：



* 目前为止这条蛇是只会按照我们初始化的方向一直走到碰壁，然后游戏结束的，如何改变蛇的运动轨迹呢？前面说到了，蛇头部节点的下次移动方向的改变，即可对整个蛇的移动轨迹进行改变，这里我们可以通过点击屏幕实现蛇头的下次移动方向的改变。   
  首先在onEnter方法中添加触摸事件监听：
* 

然后在onTouchBegan方法中实现点击事件，我们可以允许点击有一个10像素的误差：



最后我们只差最后一步，就是蛇要吃的星星，我们可以在屏幕中任意位置随机产生一颗星星（又或者叫豆子，这都无所谓），只要这个星星满足以下条件，那么它就可以被绘制出来，否则我们需要重新随机这个星星的位置：

1. 星星在游戏场景的屏幕范围内

2. 星星不能与蛇的身体部分重叠

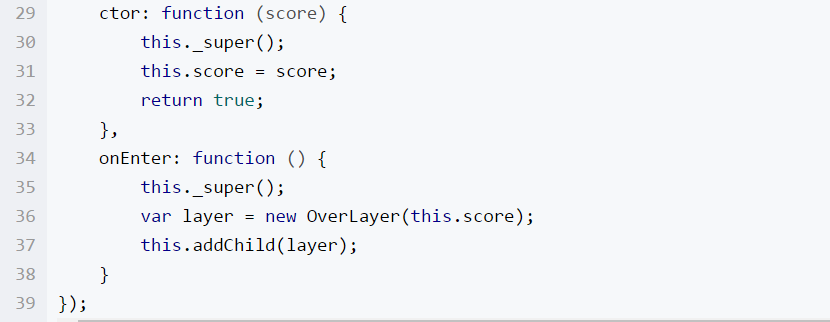
代码如下：



至此，游戏的主要逻辑就大功告成了！贪吃蛇不仅能在屏幕中游走，还能吃星星，并且碰到自身或边缘都会GameOver！

**5.开始/结束场景类**

说到GameOver，那么就必须要有一个Over的场景类了，毕竟有了开始场景，游戏场景和结束场景，才算得上一个完成的游戏流程嘛，结束场景类的实现很简单，只需要把游戏场景中获得的分数传递进来，然后在Label中展示即可，代码如下：



与结束场景一样，开始场景也只需一个Label一个Menu即可，代码如下：



这样，我们就能形成一个完成游戏流程了，游戏加载进入游戏开始场景，点击开始游戏进行如主游戏场景，游戏结束后进入结束场景，结束场景点击“再来一次”又可以回到开始场景。

**四、运行效果**

最后的运行效果如下   
