# 《Super Uncle》日志、需求、bug及解决方案：

2017年7月25日

## bug与解决方案：

###### 1.操作对象还是对象中数据成员

有的时候，移动的是character的坐标,有的时候，移动的是character中的man精灵的坐标，是操作角色类本身的对象还是操作角色类中的精灵属性？（**解决：操作精灵**）

###### 2.太空步问题

太空步：角色朝向与角色行走放下不符(**解决**) => 将转向与移动写在了一起Move函数

###### 3.BGM的loop卡顿

重复播放音乐=>第二遍衔接时，导致卡顿（**解决：真机不卡**）

Cocos2dx引擎对win32支持问题，只针对手机端进行了优化

###### 4.角色初始化的旋转动画

用于角色初始化的构造函数中，需要执行旋转动画改变3D动画的x,y,z轴。部分机型生成角色时会导致角色x,y,z轴错误。（**解决**）：设定旋转动作，操纵Sprite => 不能设置为0.f部分机型不执行

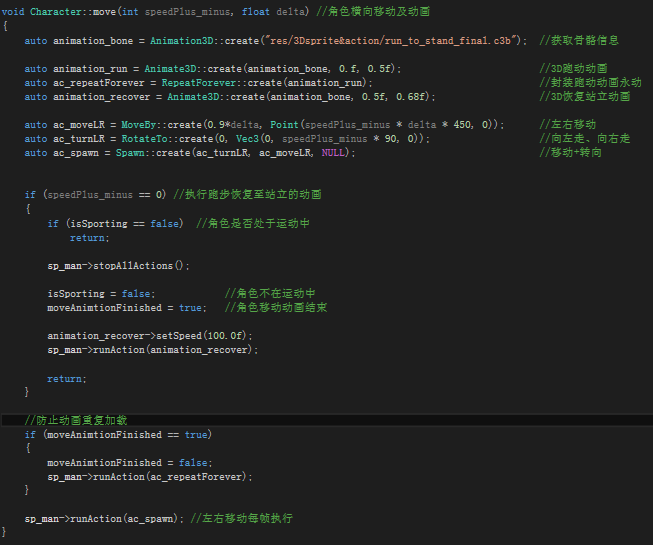
auto ac\_RotateTo = RotateTo::create(0.001f, Vec3(0, 90, 0));

###### 角色移动控制相关

手机与电脑要分别进行处理（**解决**）（~~画横线的内容为上一个版本~~）

**需求：1.**首先要明确一点是：角色行走是一个动画的前一部分，这个动画的后一部分是行走停止后的收尾（该收尾动画只在每次行走后被执行1次），所以行走执行动画前一部分，收尾执行动画的后一部分。**2.**需要达成的状态：**①**按←左：角色转向左，且向左移动； **②**按→右：角色转向右，且向右移动；**③**左右同时按住或者抬起时：角色方向不变，也不发生移动=>执行一次**④**如果此时松开左键或者右键，角色会向相应方向（转向+）移动。**⑤**不会发生太空步

**解决方案：**1.按键：左、右箭头键对应***map\_keyPressed***容器<string, bool>2.设定键盘监听以及回调函数=>每次按下一个键时，回调**onKeyPressed**函数，将对应的map的键的value值设置为true。每次抬起左、右键时，与对应的键的value设置为false。3.定义一个按键控制器函数**keyBoardControler（**放置在帧定时器当中）=> 根据键盘对应的map的value值的true/false对角色进行进一步的操作。这样能解决③④的问题：如果左右同时为true，或者同时为false，则角色需要执行跑->恢复站立的动画，从而恢复至站立，但是游戏一开始左右键都是为false，会自动执行该动画导致动画附加，需要一个开关来控制，初始化该开关“运动状态”为false => **bool isSporting = false**; ，只有当运动时，为true，只有true时，且满足左右键同为true或者false时，执行恢复动画，执行恢复动画后，“运动状态为false”，~~且移动动画回调的开关也为true因为静止后，默认移动动画也停止了，所以也算是直接完成了回调~~**~~moveAnimtionCallback()~~**~~，故把回调当中的开关~~**~~moveAnimtionFinished~~**~~打开，可以随时开始play动画。~~如果左对应的bool变量为true就一直向左；如果右对应的bool为true就一直向右。4.行走运动可以是每帧都回调的，但是动画播放不能每帧都进行=>一个角色精灵的动画不能重复不断地附加在精灵上~~。 =>当动画执行完成后，执行一个回调函数~~**~~moveAnimtionCallback()~~**~~，告知我们动画播放完了，~~**最新版本用永动动画，开关为false则不重复施加动画，省去了回调函数。**只有这个条件**moveAnimtionFinished**为true时，才能在下次调用的时候播放动画，这就达成了动画的连续播放的衔接效果，而不是同时附加播放。而移动则是每帧都执行的。



**（最新版本把部分动画改为了永动机制=>省资源）**

###### 6.瓦片地图滚动有时候抖动（解决）真机问题不明显，win32明显

①可能是因为帧定时器的内容过多，刷新东西过多导致卡顿，或者说，地图滚动放入帧定时器不合适？

②1秒定时器应该只刷新游戏秒数衰减不应该刷新得分等数据，金币，得分等数据要等相应事件后回调再刷新。

尝试解决方案：原本的屏幕移动直接依赖setPosition()函数，现在更改为一个action,进行平滑地移动，动作时间为定时器delta的时间。

**解决方案：**屏幕移动的距离等于屏幕移动的速度乘时间，每帧的delta时间越长，移动的距离就越长，反之越短。总之移动的速度是相同的，只是delta时间不同，故地图不会产生卡顿现象。

**待解决：**UIupdate函数目前被我注释掉了，执行时间过长会导致卡顿，考虑只在得到“加分、加金币”事件后，才进行UIupdate函数的回调=>**观察者模式**（省去每秒都刷新UI的资源占用）

**卡顿的原因目前是出在，角色的位移上，要观察delta，及移速等等，瓦片地图卡是因为角色移动卡而导致的。**

###### 7.项目文件重构：2017-8-29 22:14:32（解决）

几乎把所有.h.cpp重构了一遍，逻辑算法等等方面，游戏目前稳定在60FPS，说起来比较轻松，实则重构了整整一天。改变了原有函数的指针传参，改变了场景的过分冗杂，把多余的代码模块化，并且封装在了Controler控制器中，并且对控制器和角色类进行了单例设计，虽然牺牲了部分封装性，但我觉得代码通俗易懂更重要，今天就写到这里吧。

###### 游戏重力计=>“摇一摇”暂停游戏：2017-8-29 22:16:12（解决）

在scene中，重写父类Layer中的onAcceleration()重力计函数，实现思路就是，记录一段时间内晃动的阈（yu）值，x晃动的平方+y晃动的平方+z晃动的平方大于某个阈值时，视为触发“摇一摇”，而暂停的具体细节就用导演类来实现了。

###### 多点触控实现说明：2017年9月11日14:50:56

多点触控就是把oneByone改为allAtOnce，把回调：ontouchbegan()、ontouchmoved()、ontouchended()改为：ontouchesbegan()、ontouchesmoved()、ontouchesended()。需要注意的是ontouchesbegan()不需要返回值true才驱动ontouchesmoved()。ontouchesbegan()没有返回值。多点触控值得注意的是，在ontouchesbegan()中记录每个能操控角色的touch的ID，在ontouchesmoved()、ontouchesended()中要遍历所有的touches，如果ID吻合才进行这两个函数的回调。

注意：

1. 单点触控每次参数为1个touch，而多点触控函数的参数为一个存放多个touches的容器。开发时，需要遍历出每个touch。
2. Win32模拟器无法模拟出多点触控的效果，需要真机进行测试。
3. 安卓默认就支持多点触控直接能用，iphone则需要**开启多点触控**才行：目前网络中通过修改AppController.mm的方法已经过时，我的方法是，利用xcode配置好游戏后，按住command+shift+f全局搜索**“setMultipleTouchEnabled” => 设置多点触控激活**，将其值设置为YES，从而开启iphone的多点触控。

###### Shader着色器实现暂停后的背景模糊：2017年9月11日15:19:25

游戏摇一摇暂停后，对暂停的背景进行一个模糊化（毛玻璃、高斯模糊）的处理，淡化背景，凸显暂停后的主要内容，菜单。此方法常用于web开发。

Ps中没有透明的毛玻璃特效，故需要在游戏中实现模糊的效果。

思路：

1.为将要进行模糊处理的精灵或图片截图

2.利用官方自带的模糊shader(着色器)对截图进行模糊处理

3.用模糊化后的精灵或图片生成一个普通精灵作为临时背景(假背景)

存在问题：**iphone与部分核显电脑不支持blur着色器（数学运算复杂自动屏蔽）**

参考CSDN：<http://blog.csdn.net/nono_o/article/details/45891757>

###### 屏幕滚动的限制（未解决）

需要把限制条件再加入一条：角色右侧发生碰撞后，屏幕停止滚动。否则可能会发生：角色已经被刚体卡主，而角色位置的x坐标并没有大于或等于屏幕宽度的一半，故屏幕还是会继续滚动的现象。

## Super Uncle开发日志与需求：

1. 需要一个成就系统类似PSN白金奖杯等。
2. 需要Box2D物理引擎模拟系统。
3. 需要互联网组队、对战系统（socket或者http编程，以及服务器编程）。
4. 需要作用于手机屏幕上的虚拟摇杆、按键系统。
5. 游戏进行中的菜单呼出系统要模仿马里奥制造的效果，呼出菜单即暂停，菜单要实现：重新开始本关卡、退出游戏至主界面、音量音效控制（滑块）等功能。
6. 需要屏幕适配功能，特别注意三星GALAXY S8+机型（2960x1440）的适配。
7. 需要把用到CCString的地方改为\_\_String类，这是cocos2dx更新的结果。把有关于字符串转换的方法更改为：\_\_String类中成员函数createWithFormat()。以免在跨平台时因为平台支持限制问题导致崩溃。
8. 需要优化马里奥跳跃，从而实现极其相似的真实物理模拟，包括真实的碰撞和偏移。以及碰撞后的事件处理，以及碰撞发生在角色的上下左右从而产生不同的结果等。