**红绿灯设计报告**

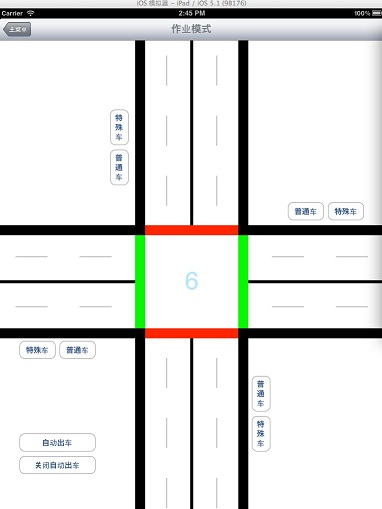
1152745 邱峰

1. **使用说明**

本程序分为游戏模式和作业模式，游戏模式是在作业模式的前提下，对程序稍作修改，制作成一个小游戏，启动程序，出现如下界面，可以选择相应模式



选择作业模式后出现如下界面



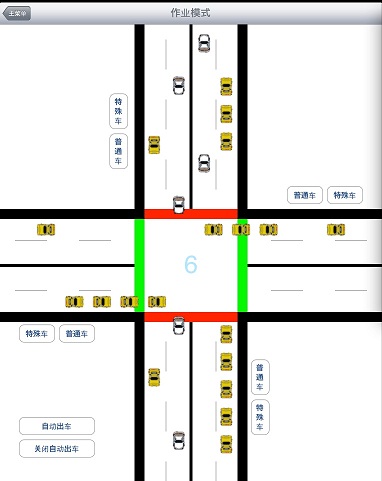
其中，正中央的数字为红绿灯的倒计时。

点击各个按钮可以为各个车道增加车辆（点击以后是先加入到一个队列中，相当于进程处于等待创建的状态，当有一定车距时，车辆才会出来）。由于点击按钮后是加入到等待创建的进程队列中，所以有时候一直点一个按钮，只会出来几辆车，然后不点它了，车子又会一直出来，这是因为在之前点按钮时，任务被加入到了等待创建的队列中。

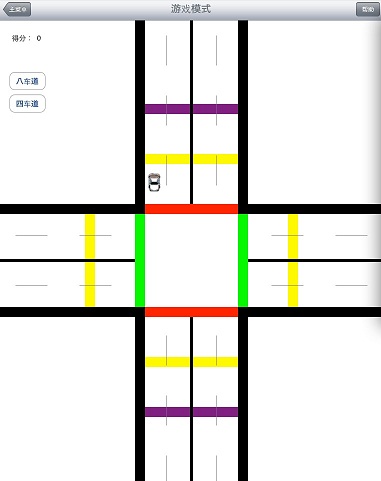
自动出车按钮会根据目前车辆的多少调节车辆，相当于如果有某个方向有很多车正在等待创建状态，自动出车不会出这个方向的车辆，点击了自动出车以后，仍然可以手动添加车辆，与车道上的创建车按钮不冲突。

白车为特殊车，可以闯红灯（下图上面那辆车会无视那个红灯过去），车距也相对较大；黄车为普通车，会在红灯停车（下图右下角的黄车在红灯时停车）。黄车的车道在对应车道的外侧，白车车道在内测。

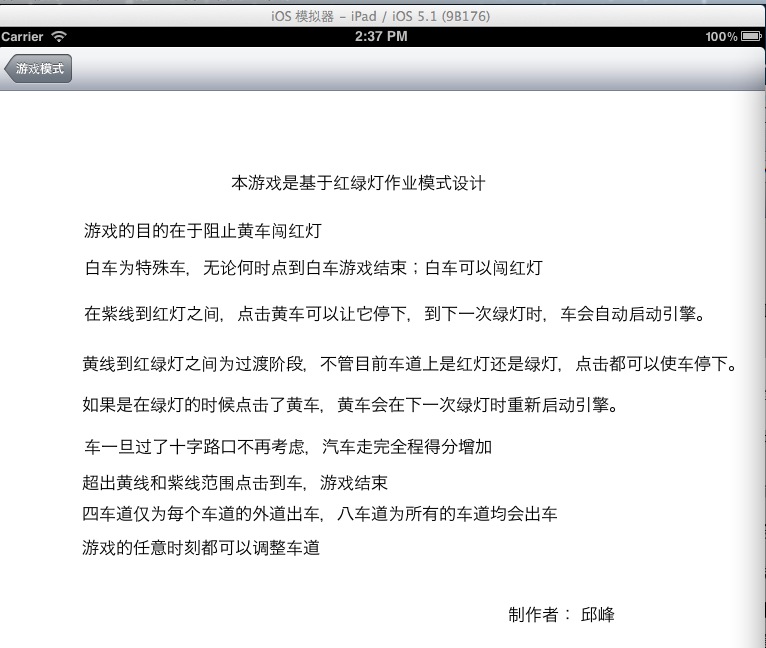
左上角的主菜单可以返回到首页



选择游戏模式后，会出现如下界面



点击右上角的帮助，显示了如下信息

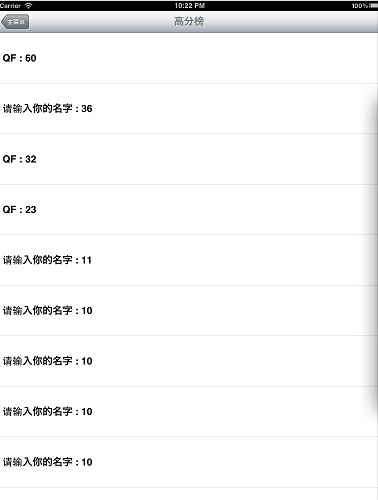


一旦游戏结束跳转到下面的画面，点击屏幕任意一个地方回到主菜单。



或者

点击游戏高分榜可以得到游戏历史的高分榜

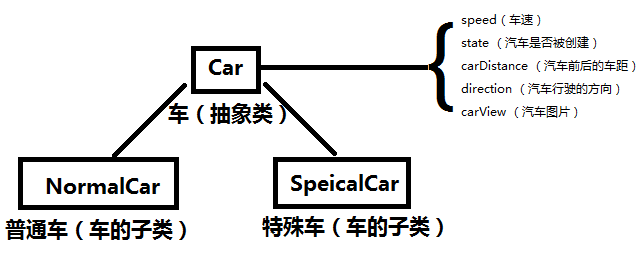


我最高打到60分。。游戏每10分一个档次，随着分数的增加，会缩短黄线、紫线的距离，车辆会增多，红绿灯切换时间变短。

1. **程序设计图**

这里，我想先说明一点，由于游戏模式是在完成了作业模式以后，突发奇想

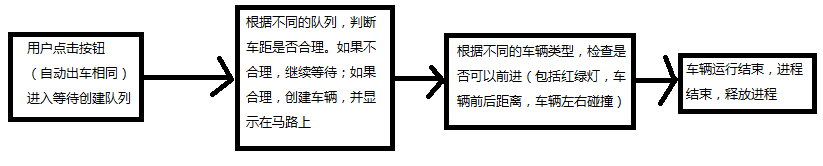
添加的，因此，当初设计时，并没有把游戏模式包含在内，而游戏模式很多代码都是从作业模式直接copy过来的。而最后两个模式融合到一起的时候，只是对所有的类加了homework和game的前缀。



★在游戏模式中只有car类，并增加了一个属性whoAmI。



汽车状态转移图



**类方法设计图**

**Car类**

checkBeforeAndAfter： 检查前后碰撞

checkLeftRightwithQueue：根据queue信息，检查this与queue中所有车的左右碰撞（十字路口上不同方向的车）

move： 调用车移动函数

**NormalCar类**

addCarAtPoint： 创建一个汽车进程

atView AtDirection inQueue，用户点击按钮，将相应的方向加入到队列中，让进程处于等待创建的状态

canMoveWithLight 根据目前红绿灯状态判断是否可以过去

**SpecialCar类**

与NormalCar函数相同，仅仅没有canMoveWithLight

这里我说明一点，由于这个程序是用objective-c写的，它有一种叫做属性的东东，所以很多get和set的函数都可以不用写，直接声明为属性就好了。

1. **实现思想**

对于红绿灯的设计，是在完成电梯以后去写的。在电梯设计中，崔昊学长给

了我不少帮助，但是这个红绿灯是我一个人独立设计的。

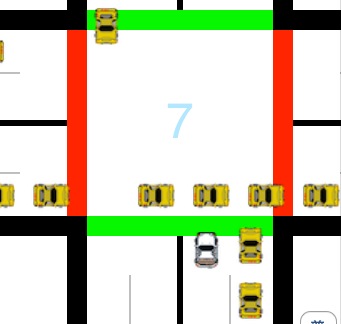
我的思想主要是根据那个状态转移图设计的，主进程中，对所有的队列的每一个进程都不断调用callMove和addCar两个函数。

addCar的任务是，根据同队列中离它最近的一辆车的距离判断这个汽车能否被创建，一旦可以创建，则创建一个新的汽车，加入到图像当中。

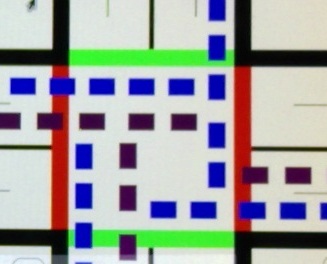
callMove的任务是，检查每一辆车是否能够前进5个单位，如果可以，就让它前进5个单位。对于特殊车和普通车的不同之处是，普通车需要多检查一个红绿灯。

起初在设计checkLeftRightwithQueue（十字路口的交叉碰撞）函数时，我设计思想分了两种情况：

1. 当一辆车出于另外一辆车中间的时候，即（红车向下行走，绿车向左）这种情况。这时候，由于绿车已经占据了位置，所以，只能让红车停下等待绿车走过。
2. 当两辆车都没有处于对方中间，但是如果一起行走就撞到一起时，即这种情况，我的选择是 ↓让←， ←让↑，↑让→，→让↓，之所以这样设计是因为从下图，我们可以看到→方向的车已经马上要通过十字路口了，我希望它能尽快通过这个路口，防止阻塞其他车辆。



但是，上面的这个设计方法却造成了一个隐患，从让车的设计来看，这是一个循环等待，并且每辆车都要求下一个车让出位置，因此可能出现死锁的现象，这也就出现了第一次设计时的缺陷（死锁）。



图为红绿灯的第一个版本，蓝色为普通车，紫色为特殊车。

造成上述的原因第一是满足了死锁发生的条件，第二是因为特殊车闯红灯的原因，由于太多特殊车的加入导致了在红绿灯路口处出现了排队的现象，并且队伍特别长，直接赌住了另外一边绿灯的车辆。然后四边一起出现一堆特殊车，就全部挤在中间了。

解决方法：

后来，我加大了特殊车的车距，理论上仍然有可能出现死锁现象，就是当4辆特殊车同时占据对方的位置时，但是，这个概率变得微乎其微了，我测试了很久，无论我怎么点按钮死锁都出现不了了。

**四、可以改进的地方**

个人觉得死锁的解决方法不是很完美，我是通过加大了很多特殊车之间的车距解决的，也就是在十字路口中间一个方向最多可能出现一辆特殊车。这样，一条道路上最多才能出现3辆特殊车，个人感觉解决方法不是很好，而且，也并没有彻底解决死锁，只是做到了一个预防，把这个死锁的出现降到了最低（测试了好久，基本不会出现了）。

由于游戏模式和作业模式很多地方代码都是相同的，但是，当初设计时，并没有加入游戏模式，所以，这里的类有很多重复的代码。

1. **心得体会**

这个程序，一共经历了我2次较大的改进，第一次是在死锁的问题上。第二

次是加入了游戏模式，由于游戏模式和本课程没什么关系，纯粹是为了娱乐，同时学习下objective-c所写的。

不过，我感觉，如果站在操作系统模拟红绿灯（我觉得这个一个RR制的，轮流让不同方向的进程执行），是不是可以按照车辆先到先走的方法让车辆通过红绿灯呢？

另外，感谢林逸凡帮我找到了第一个版本的死锁现象。