# 现代操作系统应用开发实验报告

**学号：**13332024　　**班级　：**晚上班

**姓名：**喻乐　　**实验名称：**hw4

## 一．参考资料

http://sourceforge.net/projects/sqlitebrowser/（DB Browser for SQLite）

http://blog.csdn.net/songcf_faith/article/details/45366015（cocos2d 中使用 sqlite）

http://goldlion.blog.51cto.com/4127613/772518（cocos2d 学习笔记之 SQLite 基本使用）

http://www.cnblogs.com/andyque/archive/2011/04/11/2012852.html（如何使用 cocos2d 制作基于 tile 地图的教程）

http://tieba.baidu.com/p/3376577042（cocos2d 的 tilemap 教程）

http://www.cocoachina.com/bbs/read.php?tid=199793( cocos2d 模板容器详解）

http://www.cocoachina.com/bbs/read.php?tid=219259

（TMXTileMap::getContentSize 和 getMapSize*getTileSize 的异同）

http://blog.csdn.net/zhangdell/article/details/21083265(cocos2d::Map<K,V> 详解）

## 二．实验步骤

首先就是要用 Tilemap 制作背景，在制作背景的时候首先你要导入图块，在这里要求既要有 png 文件，又要有 plist 文件，才能够正确导入，导入之后你要给每一个对象定义一个图层，把相应的对象加入到图层当中，这样就可以触发一些事件，写完了这些以后，选择对齐网格，然后就可以分别在不同对象层上发挥自己的想象来构思一个有趣的地图了。接下来就是根据路径导入瓦片地图，设置好相应的位置、锚点、z-order 等。

完成了 tilemap 之后，我们就需要实现 pushbox 里面的函数来完成这个游戏，首先我们需要自己写一个 createscene 函数来初始化一些变量，你要自己初始化 scene、layer 等相关信息，同时利用 `TMXTiledMap* mytmx = TMXTiledMap::create("Pushbox/map.tmx");` 来导入自己制作好的 tilemap ,这个时候你就可以将背景显示在屏幕上，接下来你需要在头文件中声明几个 vector 来管理每个精灵相关信息（尤其是位置信息），接下来利用 `TMXObjectGroup* objects = mytmx->getObjectGroup("wall");` 来从对象层中获取对象数组，并且遍历 ValueVector 来将每个对象添加到屏幕上。

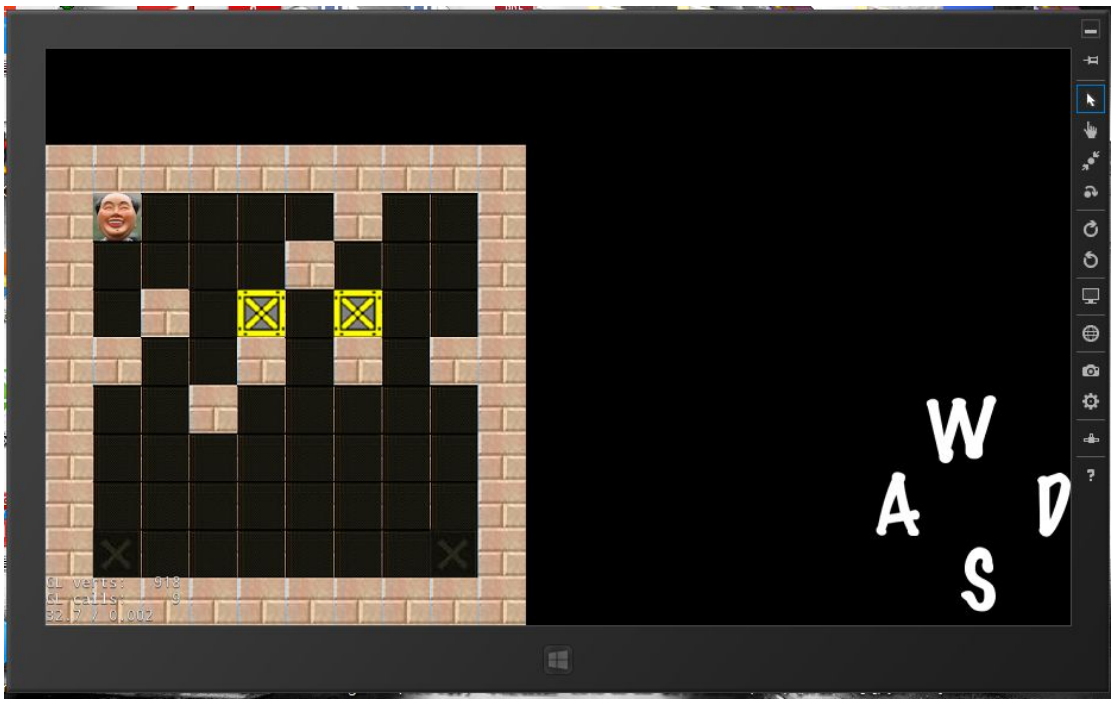
接下来你需要初始化数据库，通过数据库路径来创建数据库，针对这次作业我们只需要创建一张只有一个属性的表就行了，用来记录每次的步数，保存在数据库中，同时我们向表中插入几个比较大的记录，用来和实际走的步数进行对比，最后不要忘记关闭数据库，以免内存泄漏。

接下来有大量的工作是围绕着 W、A、S、D 四个按键按下后的相关处理函数，虽然需要判断的条件比较多，但是写出了一个就相当于写出了四个。首先你需要判断走的方向下一个位置是不是墙（通过遍历墙的 Vector），如果是直接返回，如果不是你需要判断是不是箱子（通过遍历箱子的 Vector），如果不是的话，直接移动 player 就行了，

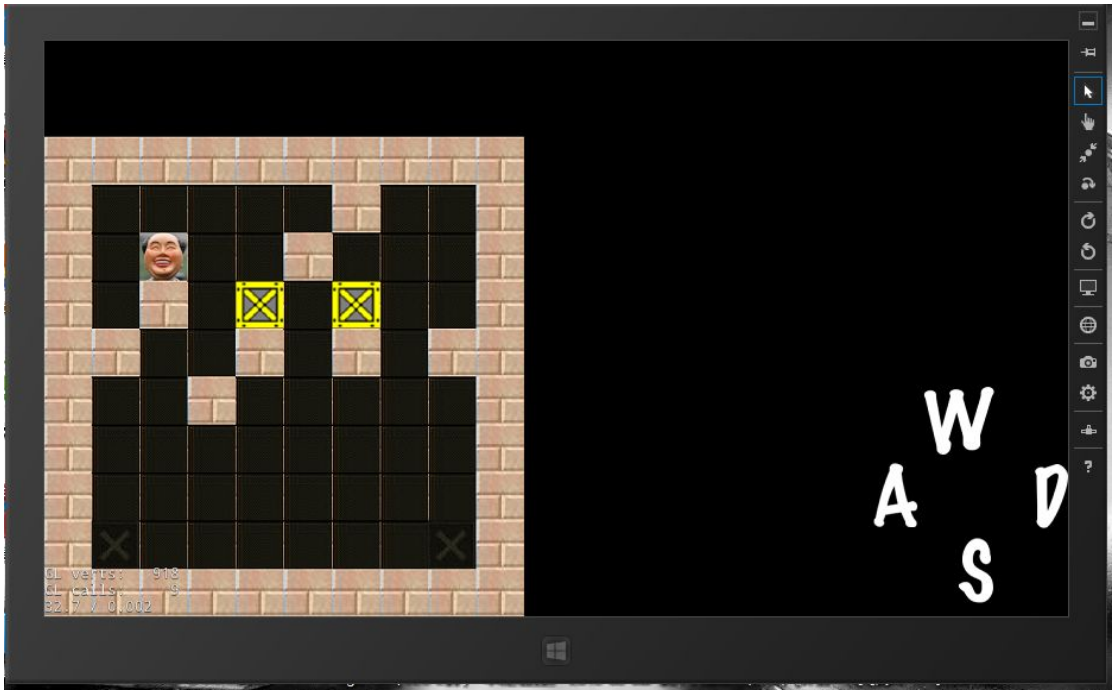
如果是箱子的话，判断一下下一个位置是不是墙，如果是墙直接返回，如果不是判断是不是箱子，循环往复，在把每个对象位置做相应更改之后，我们接下来要修改对应 count（用来记录步数）的值，最后我们要判断当前状态游戏是否结束（两个箱子位置是否和两个目的地位置相同），如果不是，继续下一次按键判断，如果是，我们要进行如下操作：

首先要把这次的 count 加入数据库中，接下来从数据库中找到最小的记录，最后在屏幕上显示“You Win !”、最少步数、本次所用的步数等等，最后为了让用户可以一直玩下去，我们添加了一个 Restart 按钮，点击之后可以重开一局。

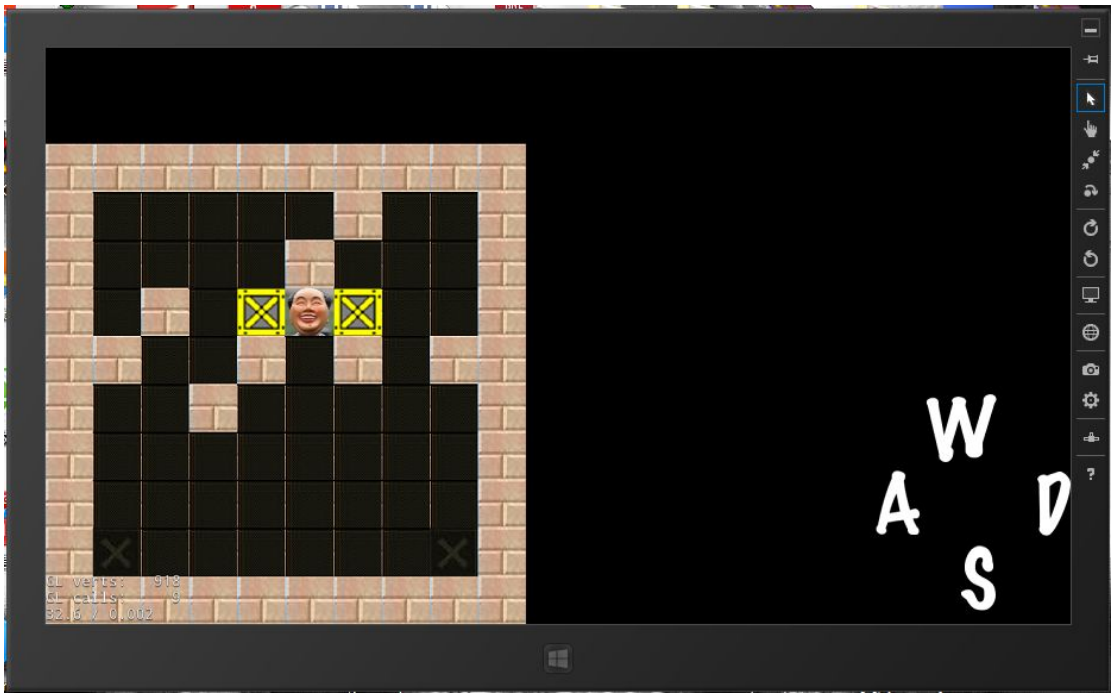
## 三．实验结果截图




打开程序一运行就可以看到如图所示的界面，左上角是我们的 player，中间有两个箱子，左右下角是我们的目的地，中间和四周都有墙，一旦碰到墙就不能继续往前走了，遇到箱子可以推着箱子一起走，多个箱子也是一样的，一起推着走，直到两个箱子都被推到目的地，游戏结束。

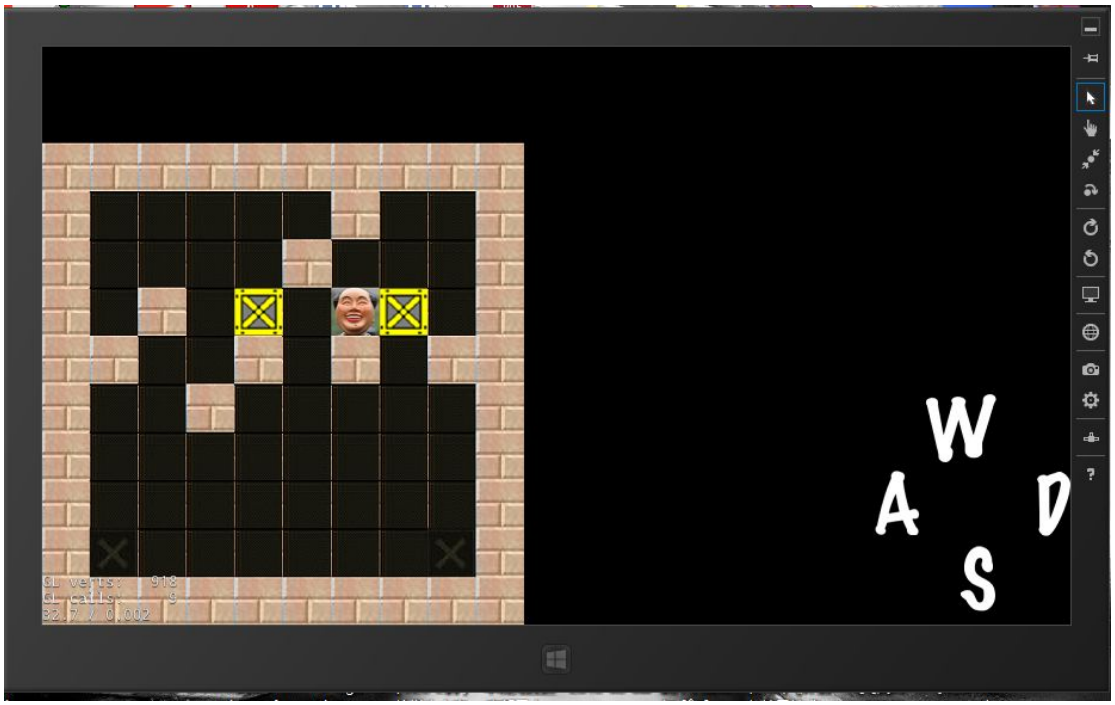



往下走的时候遇到了墙，不能继续往下走了，只能往上或者往右走。

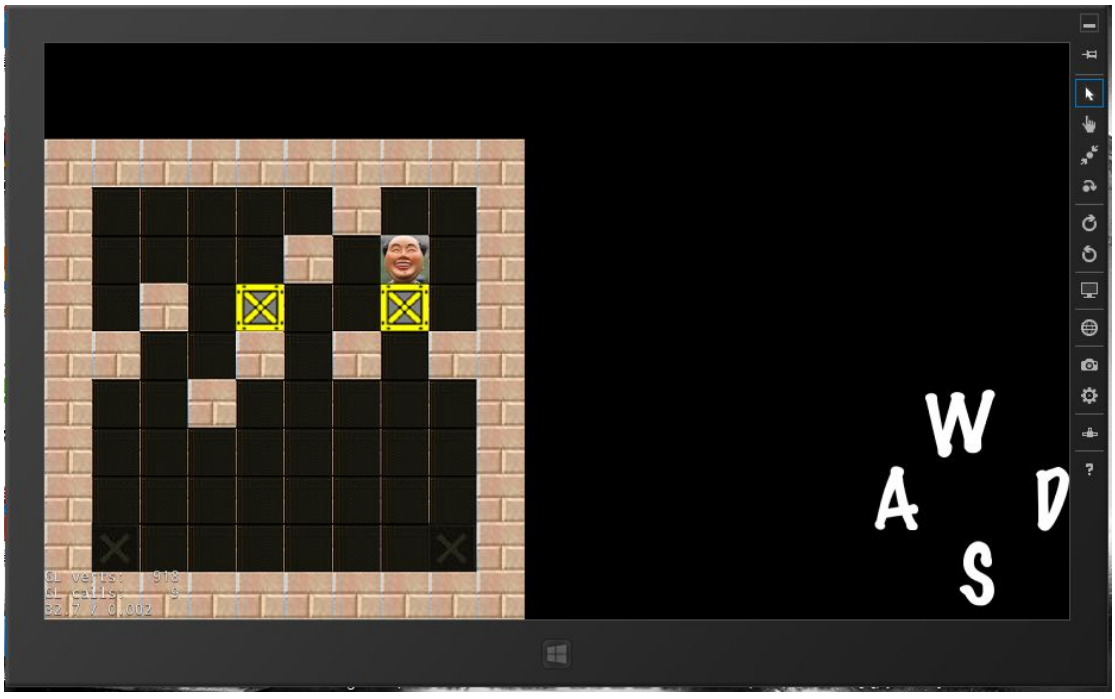



这个时候不管向左还是向右走，都可以推着箱子一起走， 除非箱子前面是墙。

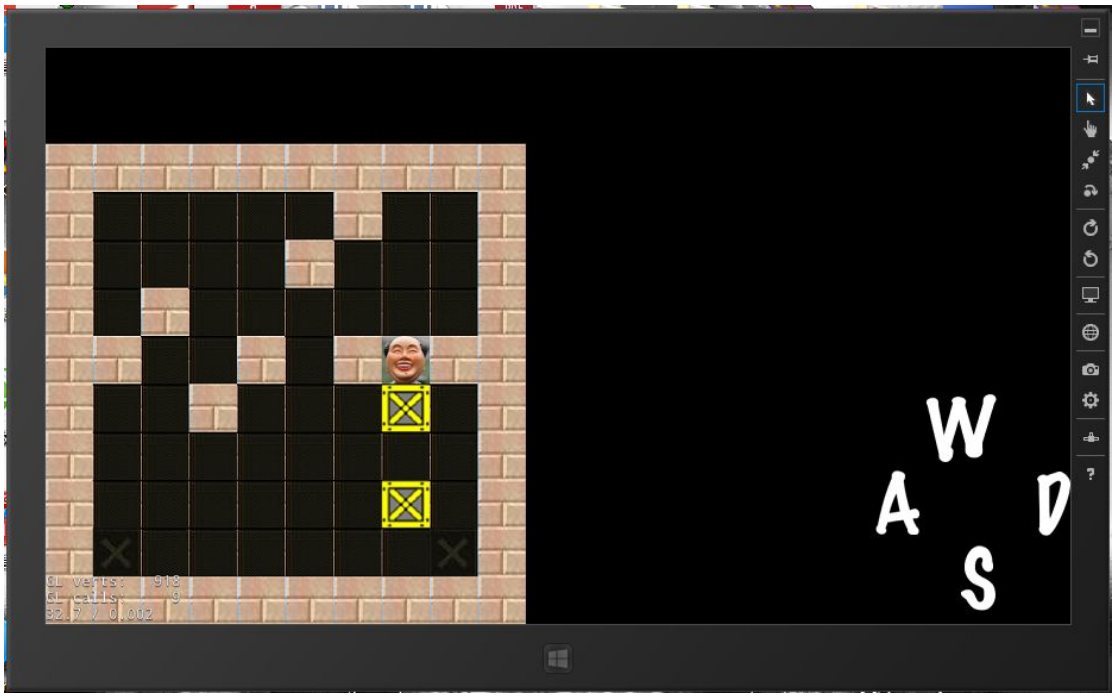
PS：这个游戏想要通关必须要先把右边箱子推到对应目的地，再把左边箱子从右边出口推出去才行，否则会卡在左边的出口处。



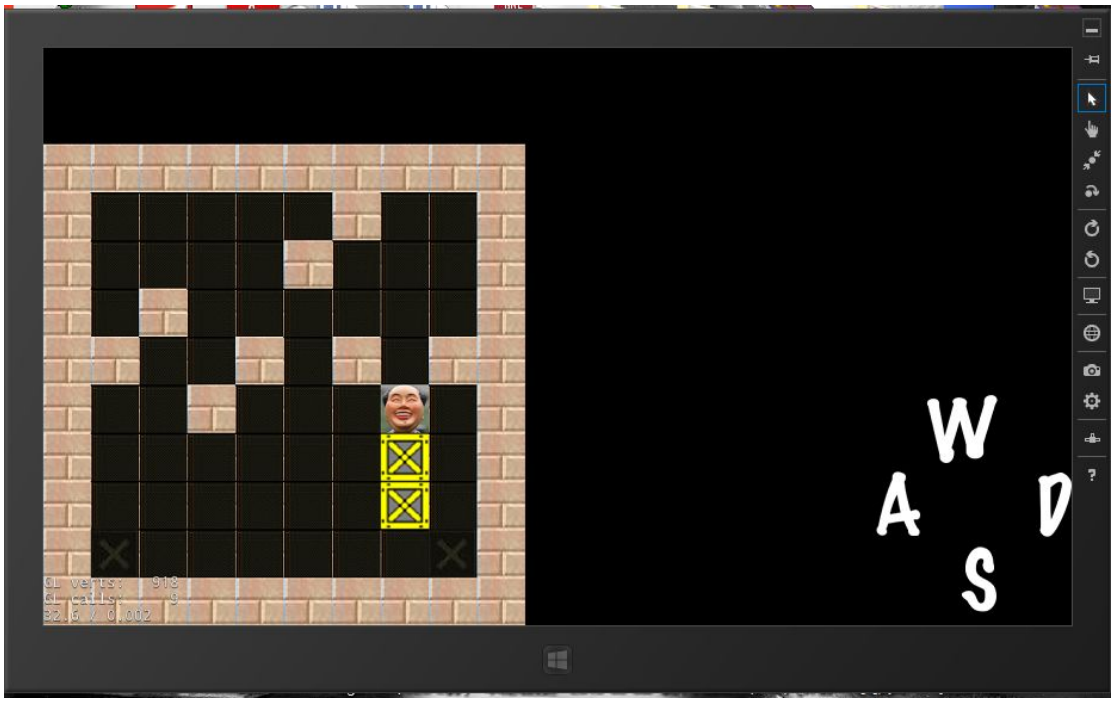

如图，不断地把右边的箱子往出口处推



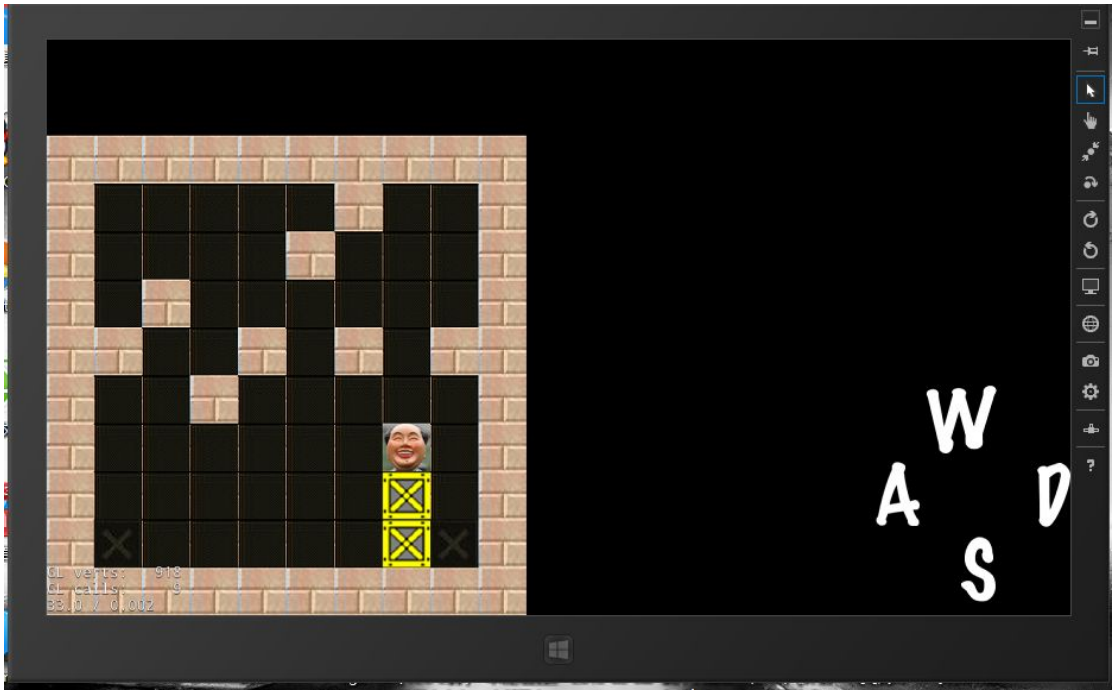

这样，我们右边的箱子已经被推了出来，如果想要步数最少，这个时候可以再去把左边的箱子推出来，然后两个箱子一起推，这样可以节省步数（见下图）



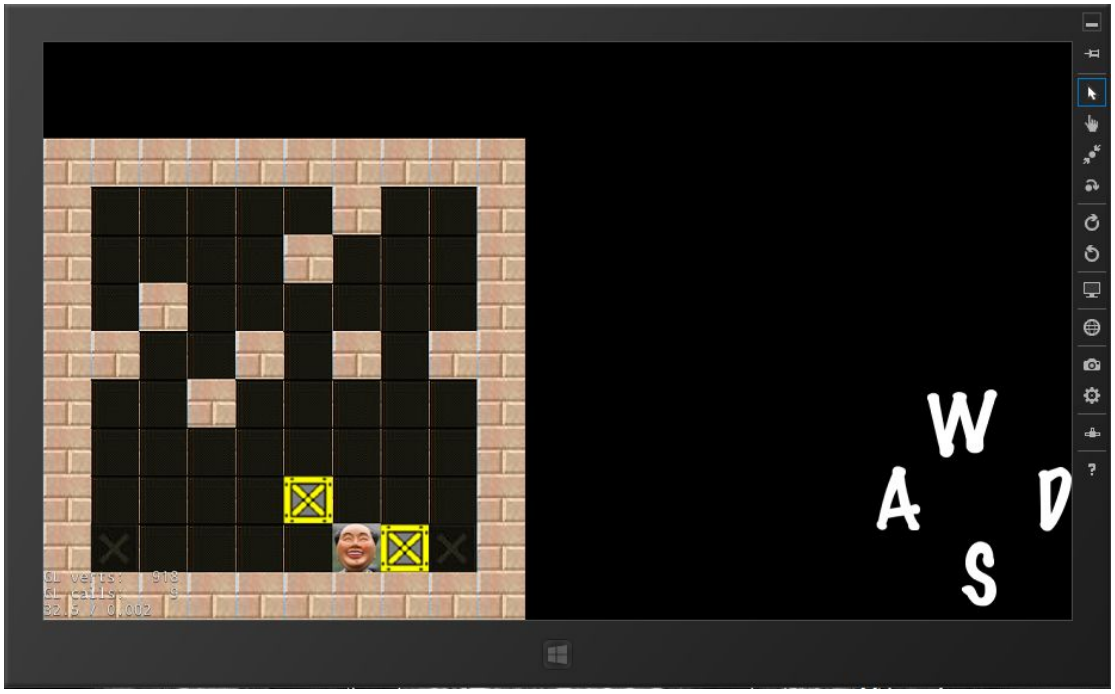

如图，我们将两个箱子都推出来了，然后我们来看同时推两个箱子的情况。

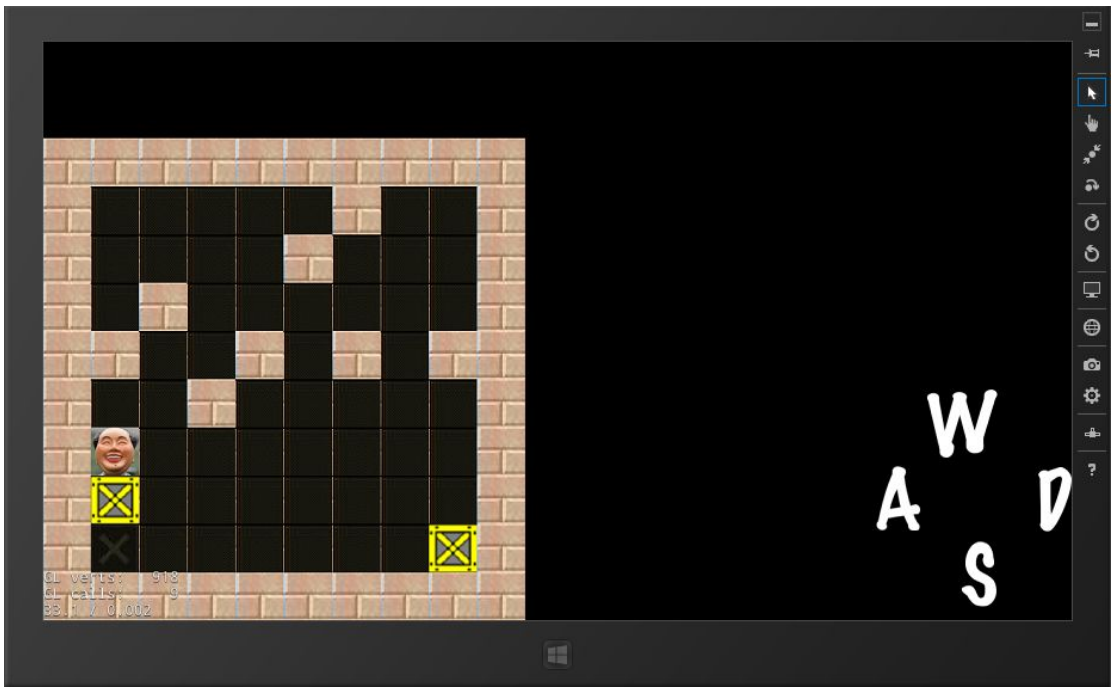



我们同时推两个箱子 ,两个箱子都会对应的移动 ,接下来就是将它们分别送到对应的目的地。

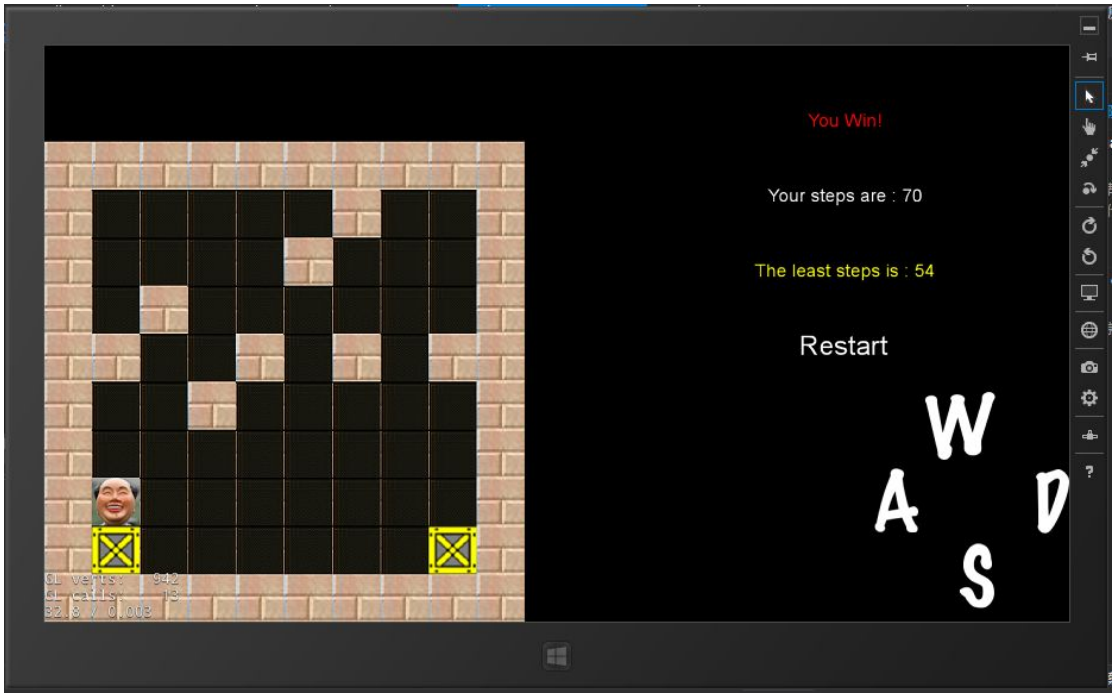



怎样做到最少步数完成这个游戏是一门艺术。

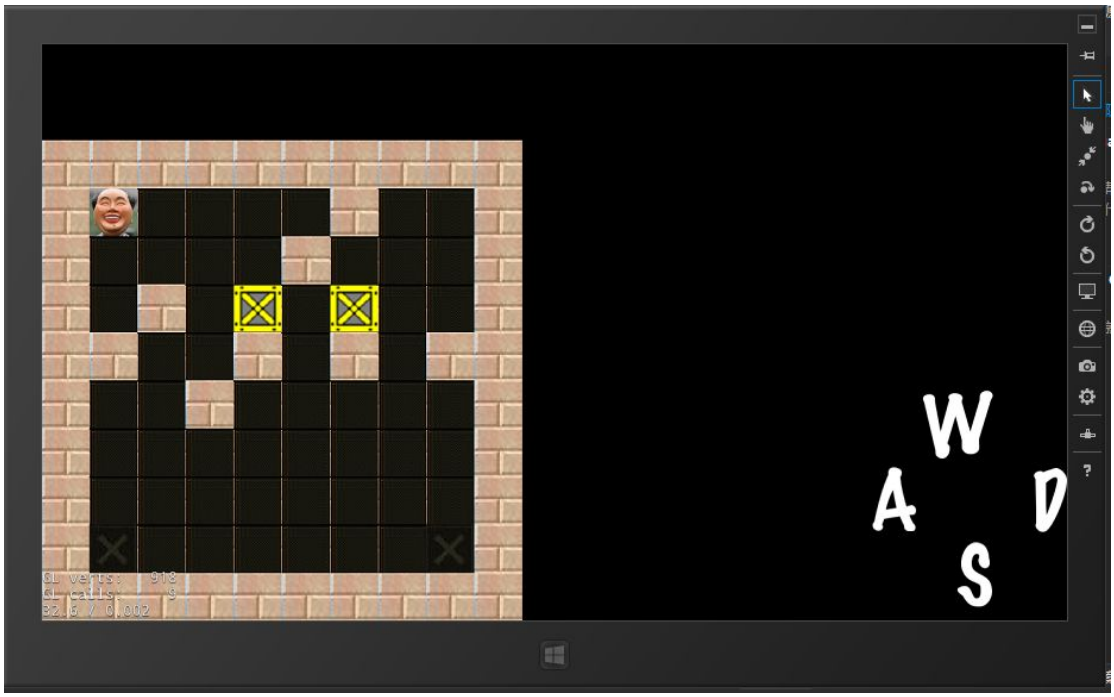





就差一步就完成这个游戏了，一旦游戏完成，屏幕上会显示游戏的相关信息。




如图，屏幕上会出现“You Win !”这个 label，同时还会显示你这次所用的步数，同时可以通过数据库 sqlite 查询来找到历史最少的步数并且显示出来，最下方会有一个按钮“Restart”，点击这个按钮，你就可以重新开始一盘游戏（见下图）。




**四．实验过程遇到的问题**

首先遇到的第一个问题就是 tilemap 的问题，发现无法向 tilemap 中导入图片，后来经过查询才知道必须要有一张总的 png 和 plist 才能顺利导入进去。接下来的一个问题就是运行之后发现屏幕上只有一张背景，对象不见了，经过询问才知道 cocos2d 对 tilemap 只会显示背景层，对象层要自己通过提取对象数组方法添加到屏幕的对应位置。在这里顺带提一下后面可能会出现的问题，比如箱子推到对应目的地之后就消失不见了，或者出现穿箱子的情况，这些都是由 addchild 的第二个参数 z-order 决定的，z-order 越大，绘画的优先级就越高，通过设置好 z-order 就可以解决这个问题。

之后的一个问题就是按下 WASD 之后进行处理的函数，有可能出现各种 bug，要通过不断玩、发现 bug，解决 bug。最有意思的一个 bug 就是箱子乱动的现象，最后发现是将判断 position 的地方改成了判断 positionx ，这样如果多个箱子的 x 位置相同，就会同时动，所以好的代码是调试出来的。

最后的一个问题就是数据库的问题，第一个问题就是发现 `SELECT * FROM step ORDER BY ID DASC;` 在 sqlite 中不能正常运行，后来无奈只能换成 `select * from steps` 来导出整张表，然后进行全表扫描，万幸的是记录比较少，根本不会有什么影响。然而 sqlite 并没有提供获取每条记录中对应属性值的函数，所以只能自己计算每条记录的相对偏移值，第 i 条记录第 j 个属性的位置为 i*col+j，每条记录的长度就是 col，加上表头之后一共要乘以 i，然后加上 j 之后就得到对应的位置，直接读出来就行了。

## 五．思考与总结

这次实验的代码量相比较上两次增加了很多，突然觉得有一个好的基础还是挺重要的，由于四个 WASD 的判断函数大致相同，所以可以把很多相同的部分提取出来重新生成一个函数，这样不仅可以减少代码的冗余量，一旦有 bug 改起来也容易很多，在判断的时候如果用上迭代会事半功倍，因为电脑会自动帮你生成迭代调用的 stack，所以代码很多地方其实是相同的。

尤其是在 sqlite 中，想到了虽然数据库的表是二维的，但是存储在计算机上还是以一维二进制的方式存储的，所以可以利用偏移值来获取对应属性，尤其是在这次试验中使用 vector、map，虽然有些地方和 STL 中 vector 不同，但是大体上还是相同的，如果利用 STL 中的经验来操作 cocos2d 中 vector，势必会事半功倍。

所以其实代码量是一点一点积累出来的，没有一个好的积累，很多工作都会很困难，因为它们是相通的，所以学习就是要一步一个脚印，只有之前好的积累，才能走好之后的代码之路。