# 2048小游戏

**学号：20191824222**

**姓名：张磊**

## 游戏功能介绍

### 1.游戏玩法概述

游戏仿照手机游戏2048制作。玩法即在4\*4方格中随机生成数字2，玩家通过上下左右的移动让所有数字进行位移，并让相同的数字合并加和，同时保证在不断生成新数字时不会使得方格被占满且没有任何可加和的数字。移动次数越多，分数越高，玩家随时可以重新开始游戏。

操作方式：玩家用鼠标向不同方向拖拽方格即可完成数字的拖移。

## 游戏开发概述

1. ***核心运算脚本：GameCore***

游戏核心运算脚本不依赖unity平台，而是利用二维数组模拟游戏进行。

其构造函数实例化了4\*4 map数组模拟地图。



**变量解释：**

map：当前地图。

mergeArray：地图每一行(或列)的所有值的一位数组。

removeZeroArray：值均为0的一维数组，用于储存去0后的mergeArray的值。

emptyLOC：为一个Location类的一维列表，Location为一个包含RIndex和CIndex变量记录行列的struct结构体

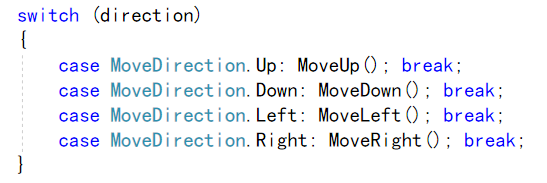
random：随机数，略。

originalMap：变化前地图，用于判断是否有变化。

**核心外观函数：Move(MoveDirection direction)**

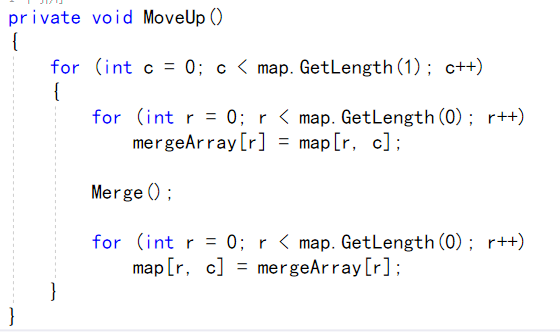
Move函数使用了一个自定义枚举类型MoveDirection，其中包含Up,Down,Left,Right用于表示移动的方向。

通过MoveDirection类型调用不同的数字移动函数。



此外，该函数会储存每次移动前的数组并与移动后数组进行对比，作为布尔变量IsChange储存。

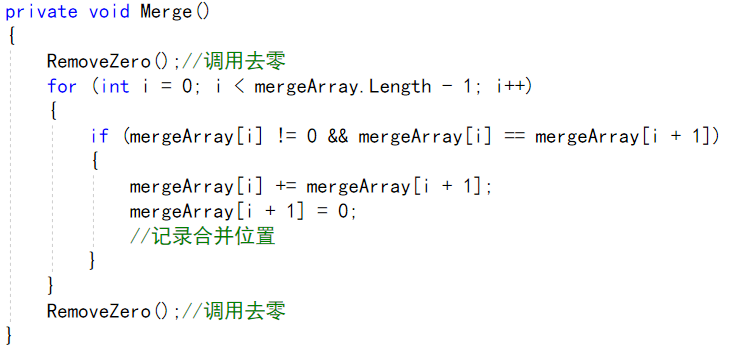
**函数：MoveArrow（这里拿MoveUp举例）**



for循环中，可以看出，mergeArray获取map的一行值以后调用Merge合并函数对mergeArray进行行合并，再将mergeArray返还给map的此行。每行依次进行，完成合并功能的实现。

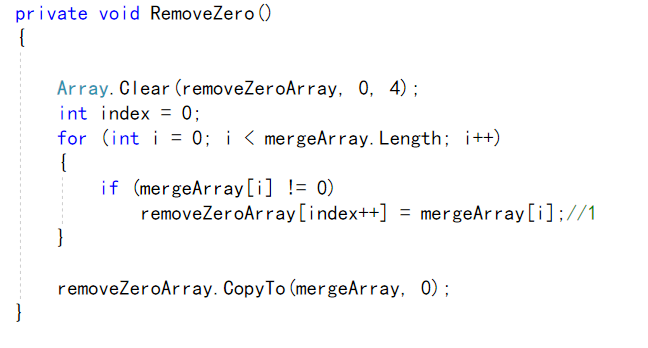
如果需要更改方向，只需要改变c，r的初值并自减即可，不再赘述。

**核心函数：Merge()**



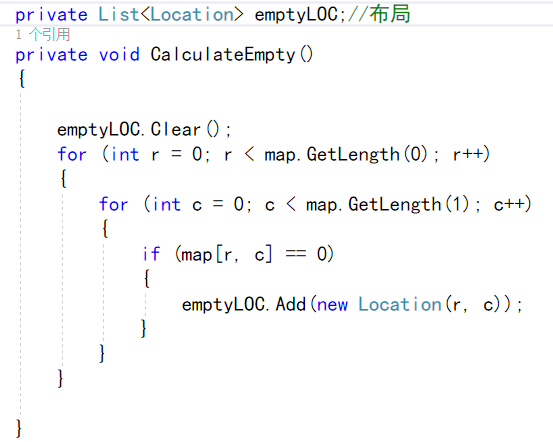
Merge()是对mergeArray一维数组的合并处理。在将数组中所包含的0去除后（将0的位置推向另一侧），检索第i个与第i+1个数的值，相同则合并并记录合并位置。

**函数：RemoveZero()**



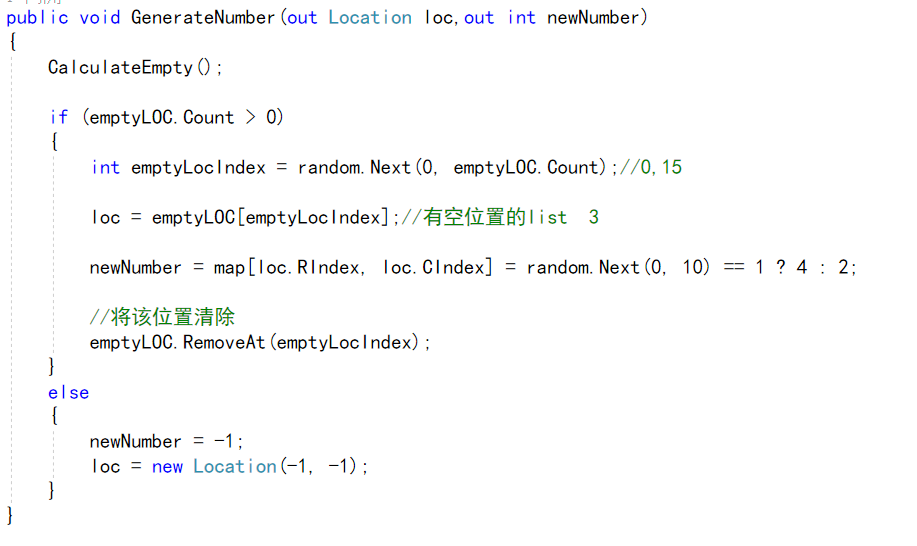
在for循环中对mergeArray[i]是否为0进行判断，并把不为0的值存入（类似append）removeZeroArray数组，剩余用0填充，实现mergeArray数组中0的去除。

**函数：**CalculateEmpty()

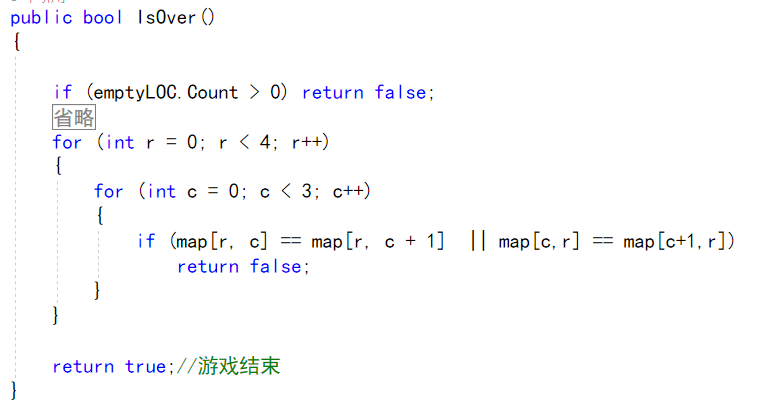


获取所有的空的位置并储存在EmptyLoc一维列表（Location，存储(r,c)）

**函数：GenerateNumber(out Location loc,out int newNumber)**

  
 当空位置数量大于0时，会在EmptyLoc一维列表中随机抽取一个位置，获取该位置的行与列。按照9/10概率为2,1/10概率为4来填充该位置。如果空位置数量小于等于0，则无效。

函数：IsOver()



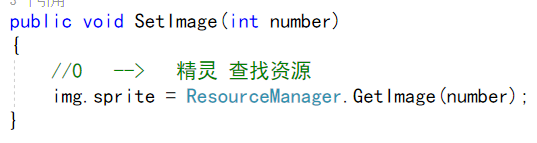
判断游戏是否结束。结束条件：剩余空位小于等于0，（移动后）无任何相邻且相同的数字。

1. ***数字精灵图像脚本：NumberSprite***

挂载于每个map网格上，提供根据传入数字设置网格图片方法。

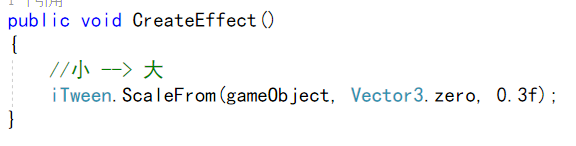
**变量：Image img，初始化函数获取Image组件**

**函数：SetImage(int number)**



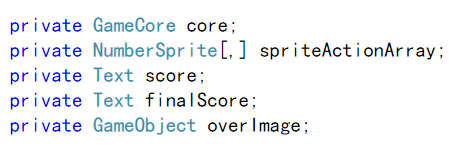
根据传入的数字获取数字图像并存储到img.sprite，

**函数：CreateEffect()**



数字精灵生成缓边特效。这里使用了iTween动画插件。

1. ***游戏管理器脚本：GameController***



**变量解释：**

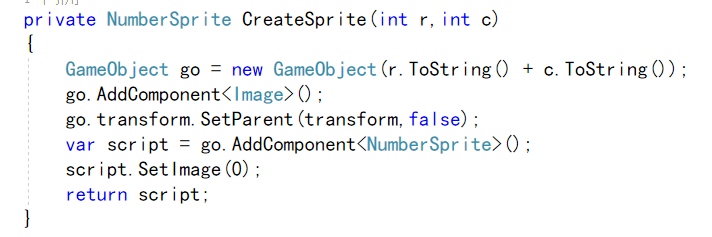
core：游戏核心

spriteActionArray：NumberSprite数组，初始化后用于储存16个方格的NumberSprite组件。

Score/finalScore：UI分数

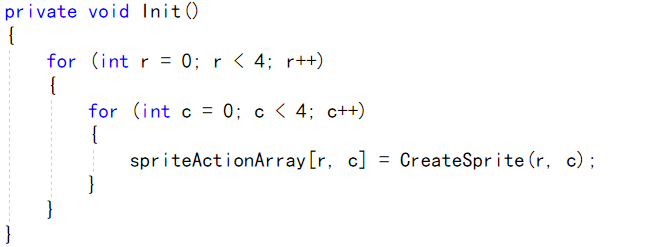
OverImage：游戏结束UI图像

**函数：CreateSprite(int r,int c)**



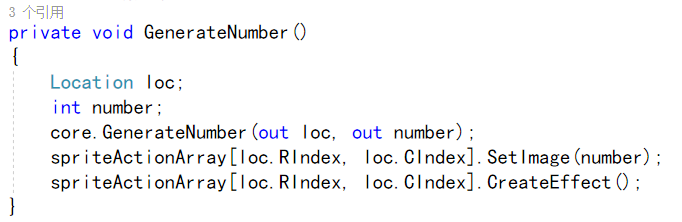
在给入r，c参数后，该函数会实例化一个新的带有Image和NumberSprite组件的GameObject物体（方格），并归于UICanvas子物体，同时，其NumberSprite将被返回（存储到spriteActionArray）。

**函数：Init()**



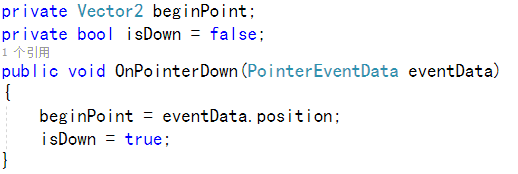
初始化时调用，生成4\*4方格并存储NumberSprite。

**函数：GenerateNumber()**

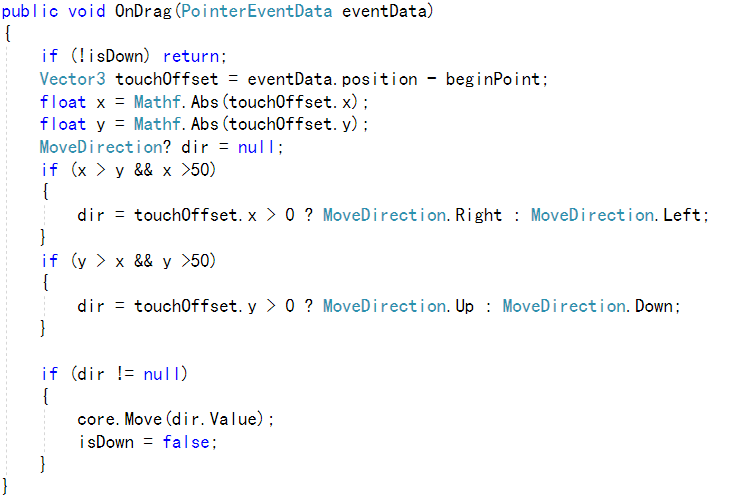


调用core中的数字生成函数，并通过NumberSprite的SetImage方法将方格设置成对应数字的图片。CreateEffect则是生成时的效果动画播放。

**拖拽检测：**

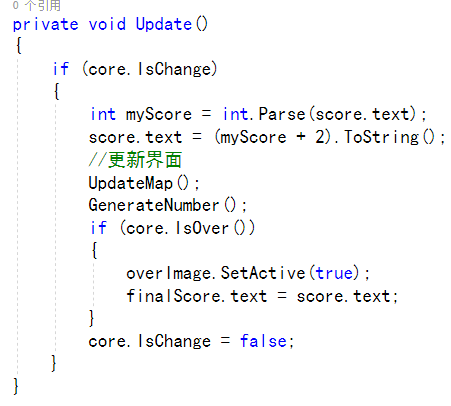


输入检测部分，记录鼠标是否按下及按下的位置。



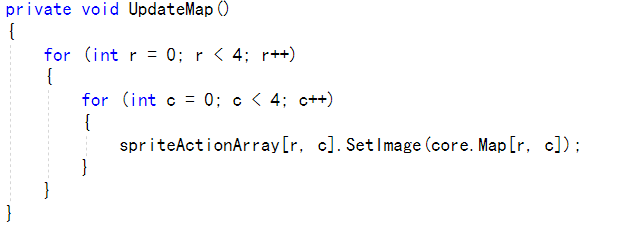
检测，鼠标按下时方法可用。鼠标移动偏移值根据当前鼠标位置和按下的起始位置做差，获得的x，y偏移值取较大者且大于阈值（设定为50）则给MoveDirection赋值，并且调用核心代码的Move函数实现数字移动。当鼠标按下不动或是移动幅度较小时(或者你能做到精准45度鼠标移动)，dir为空，不执行操作。

**Update函数：**



根据core中的IsChange变量判断数组是否有变化，若true，则分数增加，调用UpdateMap更新界面，生成新数（若有空位），判断游戏是否结束（若结束，启用游戏结束UI并展示最终分数）。

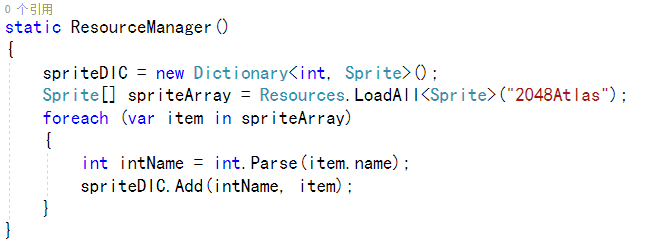
**函数：UpdateMap()**



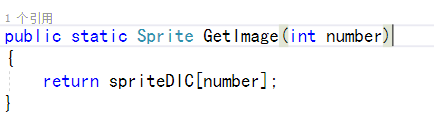
将core中数组值给入NumberSprite的SetImage函数参数，为所有方格绘制数字精灵。

1. ***资源加载脚本：ResourceManager***

**静态函数ResourceManager()：**



构造函数，初始化时直接讲2048Atlas文件夹下的所有图片存储到字典中



使用时直接从字典中按number编号拿取即可。

1. ***重加载脚本：Restart***

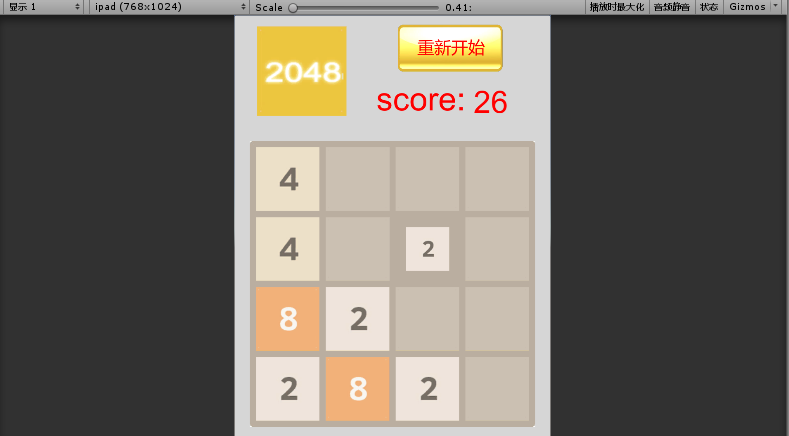
ReLoadScene()重新加载该场景，绑定到按钮事件即可。

## 实机演示

游戏画面：



生成动画：



游戏结束画面：

