**我们的游戏开发**

组长：谢志诚

组员：吴昊、王开奇、王禹卿

**写在前面的话：**

在正式开始写这篇论文之前呢，有些话想先写下来。其实在这次实训课之前，我有一直希望能有机会的话接触一下游戏开放方面的软件，代码。这当然主要是个人兴趣，这次实训课正好给了我一个机会，所以当时我就和我的另外两个室友提出我们能不能试着去开发一下游戏，他们对此也有兴趣所以一拍即合。隔壁的小欲（王禹卿）听说之后对此也非常感兴趣，也加入了我们，所以有了我们这个四人的团队。当然我们虽然美其名曰：游戏开发，实际上并没有那么高大上，更多的可以说是四个对游戏开发一窍不通的大学生一路摸爬滚打不断尝试接触游戏开发的一个过程。整个开发历时十周，说是说十周但实际上我们每周也就是花周一下午的时间聚在一起做游戏所以严格来说是十天，在这个摸索的过程中呢，我们学习到了一定的游戏开发基础，最后也是做出了，啊，所谓的“游戏”。但是受限于我们的美工技术和时间以及试用版的种种限制，这个成品的质量真的是不敢恭维，希望老师能做好心理准备哈哈哈。哦对了，在这里也是放出我们github的网页https://github.com/siiiiil/game，老师有兴趣的话可以去那里看到我们每一次更新的时间和更新的内容以及最后的成品。

那么接下去就让我来介绍一下我们整个“游戏”的开发过程吧。

第一周

第一周的时候我们主要做了一下准备工作，首先我们四个组员聚在一起讨论了一下我们究竟要做什么类型的游戏，以及我们使用哪款引擎来编写我们的游戏。根据我们的经验游戏的引擎主要有两大类：3D引擎和2D引擎。现在市面上主流的3D引擎有：虚幻3,4、unity3D等。2D引擎主要有：unity3D的衍生产物unity2D、gamemakerstudio、RPGmaker

等。考虑到这是我们自己制作的游戏，我们不想所有素材都搬运别人的，至少要有一些我们自己制作的角色，而我们又完全不熟悉3D建模，我们最终放弃了做3D游戏的想法。而在这些2D引擎中，RPGmaker完全是傻瓜向的软件，可以说代码的部分捉襟见肘，作为实训课来说未免不太合适。因此我们最初选择unity2d作为我们的引擎来进行开发。游戏类型我们也是准备做比较简单的2d横版的过关类游戏。这样一来工作量不是很大，作为初学者的我们应该也能handle。

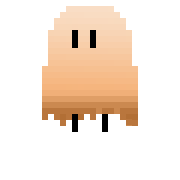
第二周

定好了大致的方向并且在第一周的时候安装好了软件，我们就准备开始正式编写代码了。不过在开始之前有一个问题，那就是我们需要搭建一个属于我们四个人的工作平台，这样大家就可以各自处理自己的部分最后进行整合。这个时候比较熟悉shell语言的小欲就提出了使用github。使用github还有一个非常重要的好处就是可以非常方便地进行版本管理，在之后的开发中回滚版本也是给我们带来了非常大的好处。当然，对于当时的我们，还有一个非常大的坎就是要熟悉git相关的shell语句。在简单地了解了一下github相关的语句之后我们就正式开始使用unity2d进行开发了。但是！开发过程并不顺利，这并不是说unity2d是一款不好的引擎，相反，而是因为它太好了，可以说unity2d作为一款引擎实在是太优秀了，它的核心代码有非常多的参数以及模组可以进行大量参数的改动，可以说unity2d拥有非常大的可能性。但对于我们四只刚出生的雏鸟来说，大量的数据和自由的编辑，以及它背后大量的看不懂的英文词汇反而让我们不知道从哪里入手，开发一下子就进入了瓶颈。虽然网上有一些教学视频，但是这些教学视频少则100来集，多则500多集。大量我们不需要使用的参数和资源的介绍让我们无力看完所有的教学。再者，因为unity2d实际上是unity3d的衍生产物，所以它本质上还是一款3d的引擎，只不过它隐藏了一个z轴，所以如果对unity3d不熟悉，在开发2d游戏的时候也是非常困难。于是，在苦苦挣扎了一周以后……

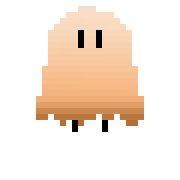
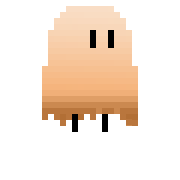
第三周

这一周我们做出了一个艰难的决定：我们要更换引擎。是的，不得不承认，unity2d是我们四个菜鸟暂时无法掌握的可怕对手，与其说是在这款引擎上苦苦挣扎不如抓紧时间找别的出路，毕竟我们只有十几周的时间。于是我们开始各自物色更适合我们的游戏引擎，这个适合这款引擎映入了我们的眼帘，那就是gamemakerstudio，简称GMS。这款GMS也是开发过著名的spelunky和hyper light drifter的游戏引擎，并且简单的看了一下它的界面，非常的简洁，代码部分也是难易度适中，它有面对傻瓜和开发者的两个版本，傻瓜版本代码部分较少，大多为模块的拖动和判定的添加。而开发者版本代码大部分需要我们自己进行编写，可以说非常适合作为学生的我们，就这样我们终于确定下来使用GMS进行开发。

第四周



经过了三周的摸爬滚打，我们总算是可以安心下来进行游戏开发的作业了，真是的不容易。首先，我们对如何导入素材进行了研究。GMS这款引擎的核心是sprite和object的联动，sprite是指精灵，导入时导入某个物体的外貌，动作等等。Object则是指具体的某个角色或物体。Sprite好比是一个外壳，一张皮，而object则是内部的核心，具体的代码都在object里。因此，再编写角色的代码之前，我们先要给它准备一个sprite。于是我们因为没有美工，不得不自己设计了一个角色作为主角，为它做了简单的站立动画**spr\_hero\_std**，并且将它导入到了游戏之中。我们还为主角对应的obj\_hero添加了移动的判定。在step函数中加入了如下代码：



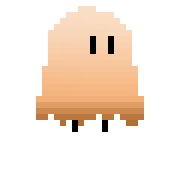
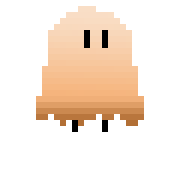
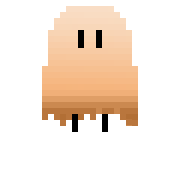
if(keyboard\_check(ord('A')))

{

x = x - 4;

if(keyboard\_check(ord('W')))

{



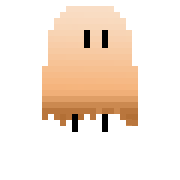
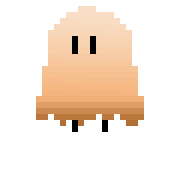
y = y - 4;

}

else if(keyboard\_check(ord('S')))

{

y = y + 4;



}

}

else if(keyboard\_check(ord('D')))

{

x = x + 4;

if(keyboard\_check(ord('W')))

{

y = y - 4;

}

else if(keyboard\_check(ord('S')))

{

y = y + 4;

}

}

else if(keyboard\_check(ord('W')))

{

y = y - 4;

}

else if(keyboard\_check(ord('S')))

{

y = y + 4;

}

这样就完成了简单的八方向的移动。

第五周

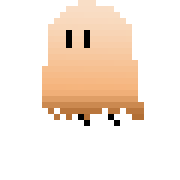
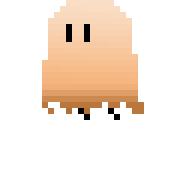
在添加了简单的移动代码之后我们的人物就可以进行移动了，但是却没有对应的移动动画，因此第五周我们加入了两个新的sprite分别是spr\_hero\_mov\_side和spr\_hero\_mov\_back，并且在完善了移动的代码，使得在移动的时候人物会播放对应的移动的动画。同时我们还启用了主角的物理碰撞熟悉，这样一来主角就可以与场景中的物体进行互动了，那么对应的x要变成phy\_position\_x。改变后的step代码如下

if keyboard\_check(ord("A"))

{

phy\_position\_x += -8; //phy\_position\_x在打开room physics时使用

sprite\_index=spr\_hero\_mov\_side; //sprite



image\_xscale=1; //sprite方向

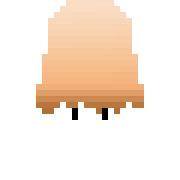
if keyboard\_check(ord("W"))

{

phy\_position\_y += -8;

sprite\_index=spr\_hero\_mov\_back;

}



if keyboard\_check(ord("S"))

{

phy\_position\_y += 8;

}

}

else if keyboard\_check(ord("D"))

{

phy\_position\_x += 8;

sprite\_index=spr\_hero\_mov\_side;

image\_xscale=-1;

if keyboard\_check(ord("W"))

{

phy\_position\_y += -8;

sprite\_index=spr\_hero\_mov\_back;

}

if keyboard\_check(ord("S"))

{

phy\_position\_y += 8;

sprite\_index=spr\_hero\_mov\_side;

}

}

else if keyboard\_check(ord("W"))

{

//sprite\_index=;

phy\_position\_y += -8;

sprite\_index=spr\_hero\_mov\_back;

}

else if keyboard\_check(ord("S"))

{

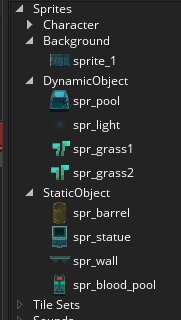
//sprite\_index=;

phy\_position\_y += 8;

sprite\_index=spr\_hero\_std;

}

第六周

这一周我们的主要任务是做主角和其他物品的物理碰撞互动，并且创建一个场景来测试这些互动，由于我们美术功底有限，背景和物体的素材要我们自己画实在是有点强人所难，所以这里就借用了一下网上的素材，从网上我们找到了8个物体和1个背景供我们测试。创建了他们对应的sprite：spr\_pool、spr\_light、spr\_grass1、spr\_grass2、spr\_barrel、spr\_statue、 spr\_wall、spr\_blood\_pool

以及object（这样我们才能做出他们的物理碰撞）：obj\_pool、obj\_light、obj\_grass1、obj\_grass2、obj\_barrel、obj\_statue、obj\_wall、obj\_blood\_pool

为这些obj添加了物理属性，并创建了一个测试空间rm\_test之后，我们发现他们可以进行碰撞了，但同时也带来了3个问题：

1. 遮蔽问题，物体与物体之间的遮蔽非常有问题，主角明明走到了桶的前面但是桶却依旧在上层
2. 翻车问题，主角在与桶发生碰撞之后会很搞笑的进行旋转，就像翻车一样，这是我们不想看到的
3. 固定位置问题，有些物体，比如说一个柱子，我们希望它能固定在一个位置，而不是被主角推着到处走。

为了解决这些问题，我们在网上找了不少资料，最终一一解决了他们。

对应第一个问题，解决方法是：在每一个有物理属性的obj的step函数中添加以下代码：

depth = -y; //即深度值越小，离相机的距离越近

第二个问题的解决方法是，在每一个有物理属性的obj的create函数中添加以下代码：

phy\_fixed\_rotation=1; //即锁定翻转角度

第三个问题的解决方法是，将这些需要固定位置的物体的create函数中添加以下代码：

density=0; //即重量为无穷

这样一来主角与这些场景中物体的互动就基本做好了，在测试中我们可以控制主角推着一个木桶到处跑，同时撞到墙上或是柱子上时会被拦住。

第七周~第九周

场景互动基本完成之后，我们就开始着手添加敌人了，那几周正好遇到连续的放假，我就要求每个组员一人设计一个敌人，并且负责制作它的动画以及AI。再经过几周的设计之后呢，大家设计的怪物也都基本成型了，让他们自己来介绍一下吧。