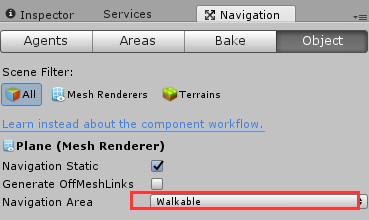
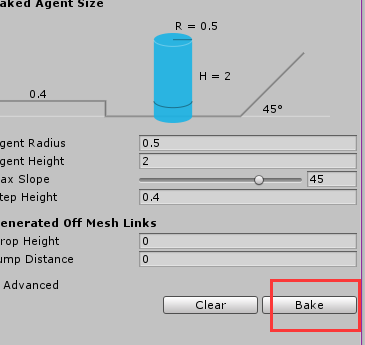
一、navigation创建，烘焙和使用

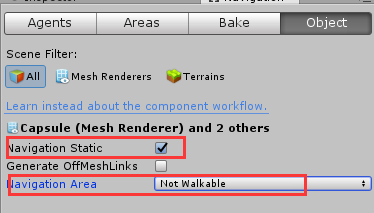
在windows窗口下打开navigation之后，把要生成nav网格的物体的static里面的navigation static勾选上。



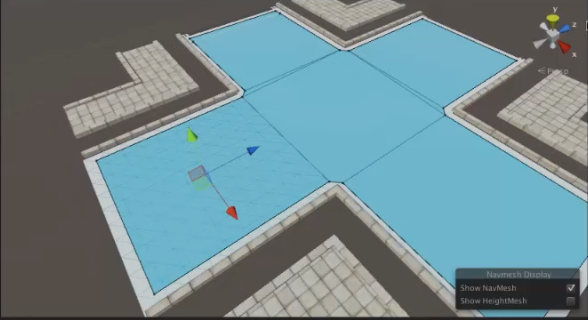
然后先保存一下场景再bake.



如果有阻挡物，把阻挡物设置成

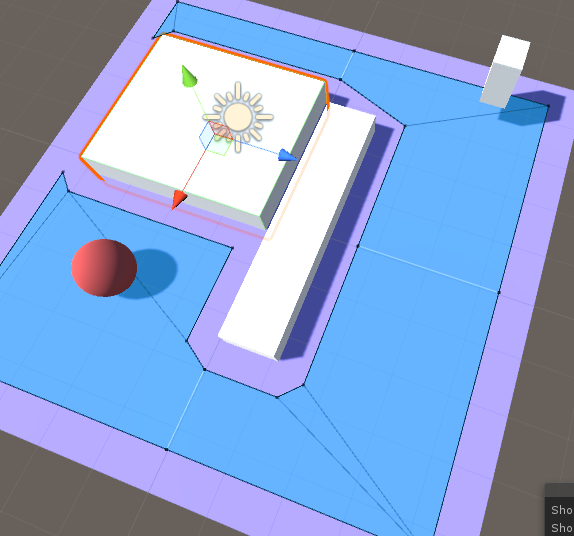
然后再点击bake里面的bake.

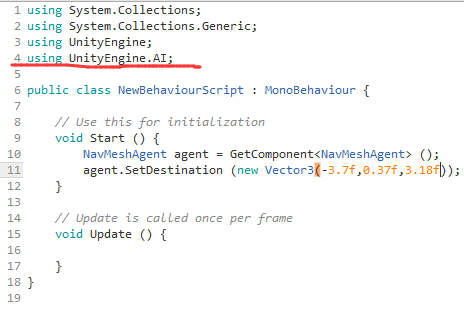
如果一个比如说平面，上面有的放可以行走，有的地方不能行走，那先把整个平面的inspector里面的static里面的navigation static取消勾选，然后在可行走的地方加几个plane，把plane的navigation static勾选上再烘焙。再把这些plane取消勾选。



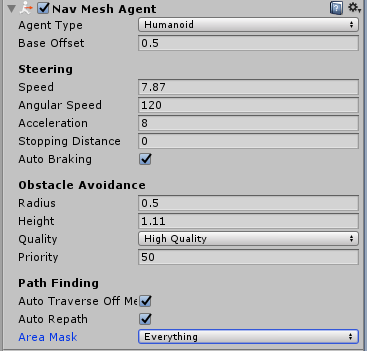
**如果物体a需要在地形中走动，在a身上加一个组件nav mesh agent;**

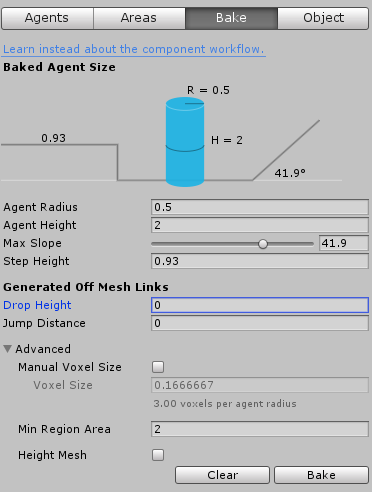
A物体挂在了这个组件之后，。要记得在这个组件里面设置速度，默认的速度为0，那么不会移动。如果在地形烘焙之后新添加了障碍物，要重新烘焙

如果加了新的障碍物，重新烘焙之前需要做一些操作。把新的障碍物的inspector里面的static里面的navigation static勾选上，再在navigation面板上的object选项里面的walkable选为not walkable,再点击bake按钮，就烘焙出新的地形。不重新烘焙的话，小球会穿过障碍物直接到目标点。

把这个代码挂载带a上面，如果想让A物体的一头朝向这个目标的话，由于unity默认是z轴朝向，只需把该物体的那一头旋转到与z轴对齐即可。选中a物体，出现，勾选agent display前三个选项，运行的时候在scene模式下可以看到a的运动路径经过的网格。

就算有collider，想要产生碰撞前提是要加rigidbody

。A物体的这个组件，angular speed是转弯速度，auto braking选项是它是否在目标点刹车，不勾选的话他是不会停下来的。stopping distance是离目标点多少距离停下来的意思，radius,height是这个物体navigation的框框的大小，area mask是表示他可以经过哪些地形。Auto depath，当地形动态时，需不需要重新规划路径。



Agent radius在网格上运动物体的网格半径

Agent height在网格上运动物体的网格高度，

Max slope物体最大爬坡角度

Step height物体最大跨越高度

Height mesh勾选上的话，在爬楼梯的时候，物体是在楼梯上而不是在斜着的网格上运动。

二、从一个地形到另一个地形的offMeshLink

在两个没有联通的地形边缘添加两个物体，这两个物体表示联通点，就是物体要从一个地形到另一个地形必须经过的地方，把其中一个添加组件offMeshLink（另一个不用添加这个组件），把这个组件里面

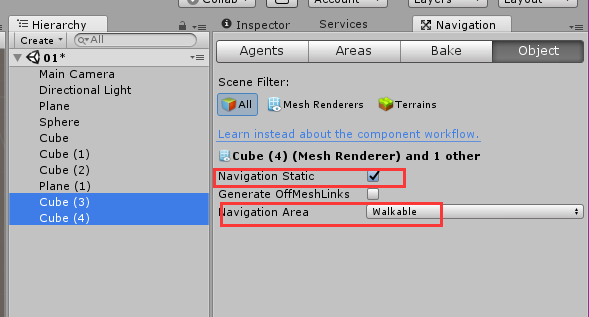
这两个属性用着两个连接点赋值了。

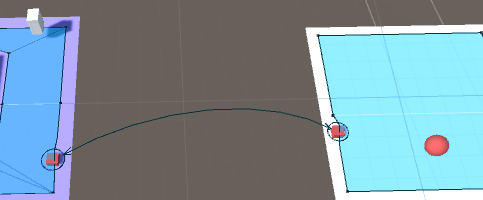
bi directional勾选的意思是起点和终点是联通并且双向的。

Activated是动态控制传送点是否是激活状态。

auto update positions勾选上的话就是桥根据这个点动态变化而变化

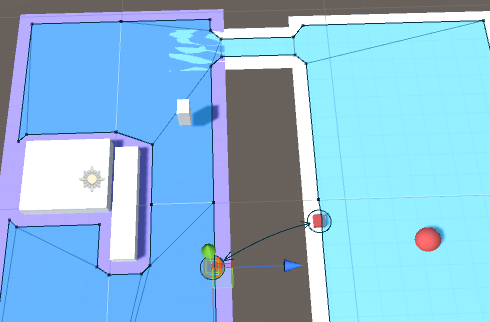
这个组件下面的设置。

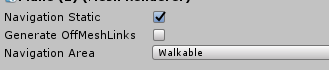
这两个连接点选中之后在navigation里面设置成这样再bake,

烘焙完成之后这两个点就是一对了。

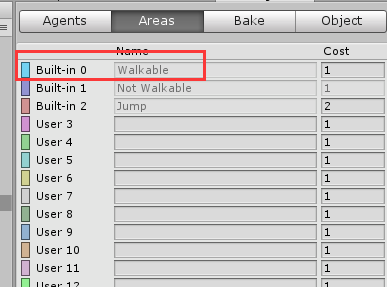
**调整了场景之后记得重新烘焙，不然要出问题。**

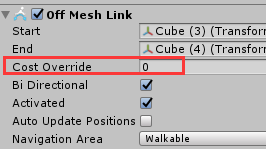
在场景中间连一个桥：



这个桥，要选中它，在它的inspector面板上面把static里面的navigation static勾选上，再在它的navigation面板中object里面，再烘焙，就可以形成路了。

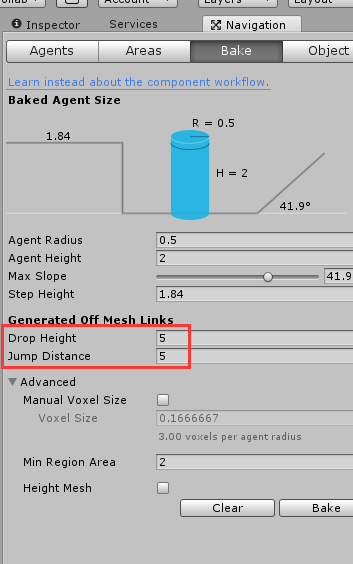
但是对于球来说，到底是走两个传送点还是走桥？

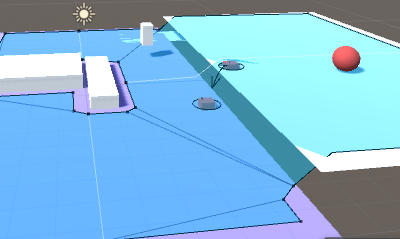
由于桥是walkable层，就是说走桥的话每走一个桥网格代价是1， 假如走传送点，在传送点上每一步代价是0，

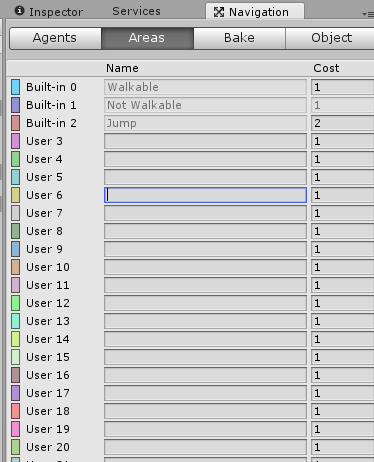


这时候，球会计算从它的位置到目标点，走那种方法代价最小。并不一定是走上面显示的代价数值最小的路径，（并不一定会选择传送点）。它的选择是根据当前位置到目标点，每条路的代价总和哪个最小定的，球的位置不一样，结果不一样。因为起始点和重点之间有很多网格，每条路网格个数不一样，其中网格的代价也不一样，要全部一起算了才知道。

Unity’可以在两个断裂面自动把navigation网格连在一起，形成可跳跃的网格。

这两个是物体可跳跃的高度和距离，如果这个距离涵盖了两个断裂面的高度差和距离差，烘焙之后可以形成这样的网格：

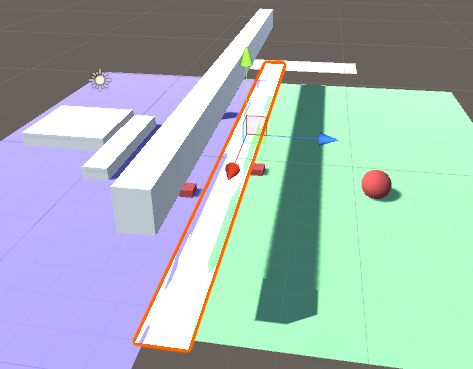
这样小球到现在这个面边缘的时候可以垂直跳下来到下个面，再去目标点，这是双向跳跃的网格。

其中前三个不能改变名字，not walkable其实是无限大的数值，这些代价都不能小于1.

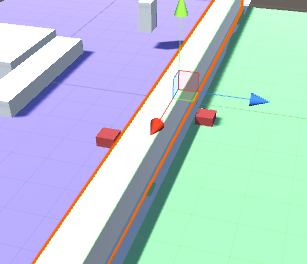


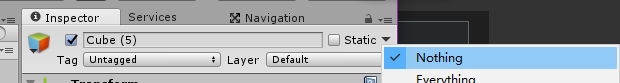
Offmeshlink组件上面cost override属性如果是-1，就是说选择walkable代价，如果改成其他值，就是使用这个值的代价。

三、使用导航网格的层动态控制门开关

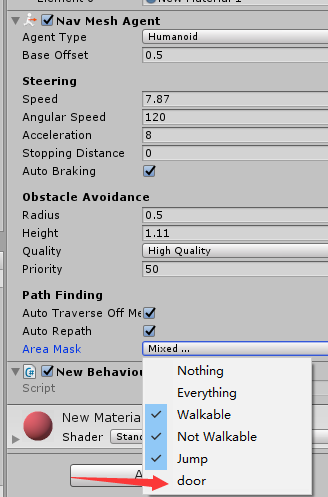


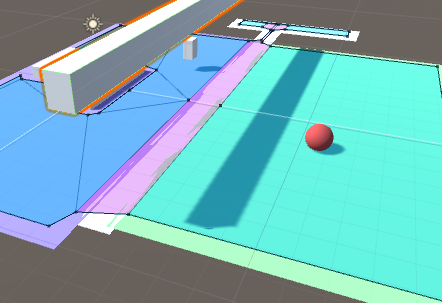
要使用悬空的cube作为控制开关的门，（这个cube只是阻挡小球前进的一个形象而已，真正起作用的是它下面的plane）.现在它下面放一个plane,把这个plane在navigation里面的object下面的navigation static勾选上，再把它的层改变成一个新增叫door的层。



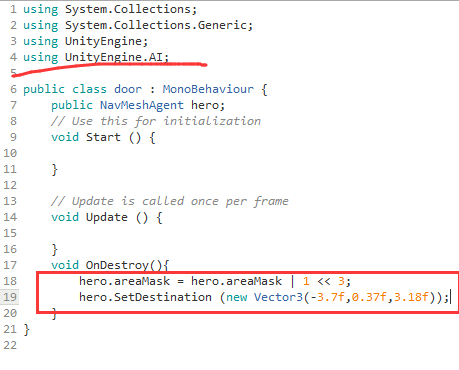


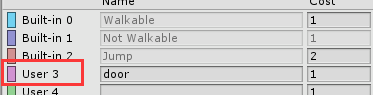
把这个plane上面放的cube的static设置成nothing.表示不可以在这个cube上面寻路。

把主角身上的可寻路层取消door的勾选，表示不可以在它上面寻路。

烘焙之后，小球到不了另一个地形，只能在它现在的地形靠近door边界停下来。

这种情况可以是通关的场景，就是说主角把它现在所在的一遍的怪打完了之后，这个门才打开，它才能去另一边，把门打开，把主角的area mask勾选上door层就可以了。

创建一个名字为door的代码，把这个代码挂载到门的cube上面，再把主角赋值给hero红框里面第一行代码的意思是在这个代码挂载的cube被删掉之后，主角可寻路的层中加入第三层，因为door层在层的定义中为第三层。

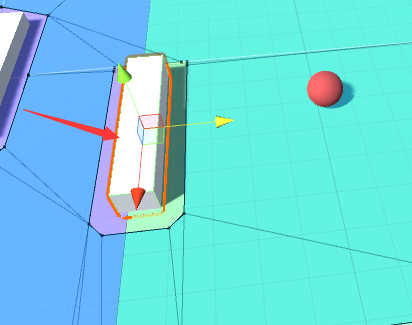


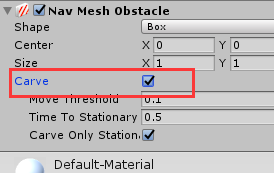
第二行代码是，在这个cube被删掉之后，继续寻路。因为unity这个寻路是在代码运行的时候算好的，在中途改变场景里面的东西，不会继续寻路，所以要再设置目标，就是

SetDestination.

在运行的时候，等小球到了边界不动了之后，删掉cube这个时候小球会继续到目标点。点击停止运行，cube会又回来。在运行中操作不会最终改变场景。

四、导航网格阻挡，动态改变导航网格

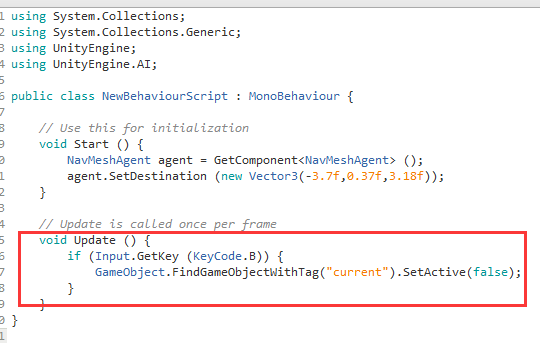


给这个物体添加Nav Mesh Obstacle组件，渲染之后勾选这是移动cube,渲染的网格也会跟着它移动，

要想让cube和它身边的网格一起移动前提是不能勾选cube的inspector上面的static里面的navigation static

这个carve下面的参数意思是：在移动距离不超过这个数值的情况下不改变网格位置。这样节省了计算量。

由于作为门的cube身上挂载有Nav Mesh Obstacle组件，就是可以动态的改变场景的网格，在小球身上的代码里面写到：

当小球走到门被门挡住了之后按下B键，门消失，动态改变了网格，可以联通，小球就继续走向目标点。

记住给门加上current的tag

把navmesh导出来有点麻烦，要用到插件