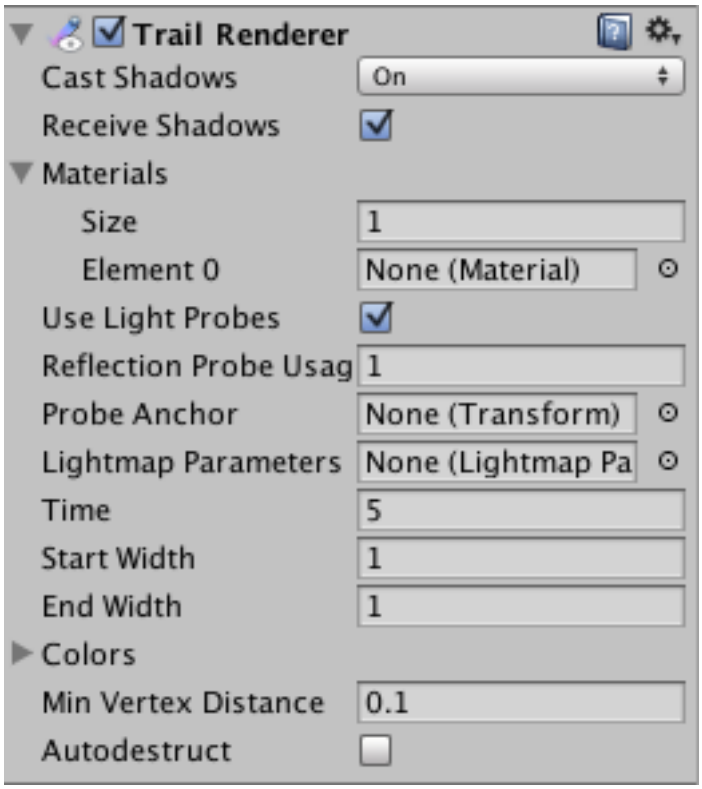
一　拖尾渲染器

通过对象依次单击Component->Effects->Trail Renderer给游戏对象添加拖尾渲染器，然后单击选中对象，就会在检视视图中查看到拖尾渲染器的属性。展看可以看到如下信息。



接下来，我们来看看具体参数的意义，如下已经列举出。

属性	说明
Materials	拖尾的材质
Size	材质总共有多少元素
Element 0	其中的一个元素
Use Light Probes	是否开启灯光探头
Probe Anchor	探头的锚点
Reflection Probe Usage	反射探头使用率
Time	拖尾的长度，以s为单位
Start Width	开始位置的拖尾宽度
End Width	结束位置的拖尾宽度

Color0~4	拖尾的颜色，从初始到结束
Min Vertex Distance	拖尾锚点之间的最小距离
AutoDestruct	当拖尾结束后，是否销毁对象

1.1 案例

此案例需要修改前面学习的车轮碰撞器的案例。

二 寻路技术

Unity自从3.5版本之后，增加了NavMesh寻路的功能。在此之前，Unity用户只能通过第三方插件（如Astar寻路插件）等做寻路功能。有兴趣的同学可以使用下A*寻路插件。不过由于不是自带的功能，所以在设定网格和烘焙的过程难免会出现很多不便。NavMesh作为Unity自带的功能，用法和LightMapping烘焙或者遮挡剔除Occlusion Culling有很多相似之处。(烘焙和遮挡剔除在我们后面的内容补充到)

在Unity编辑器中，我们选择菜单栏Window->Navigation，即可打开Navigation面板，如图2-1所示。这个Objcet的面板是对应当前选择的物体的(前提是该物体带网格)，旁边的Bake面板是对应全局选项的。上面的All、MeshRenderers、Terrains是对Hirarchy面板里面显示的物品选择的一个筛选过滤。

第一个Navigation Static选项是选择该物体是否用做寻路功能的一部分。只有勾选了这个选项，下面的其他选项才可操作。

Generate OffMeshLinks选项是选择该物体是否根据高度、可跳跃宽度等全局的选项自动生成OffMeshLink，这个会在以后的讲解中详细说明，这次就暂时不讨论。

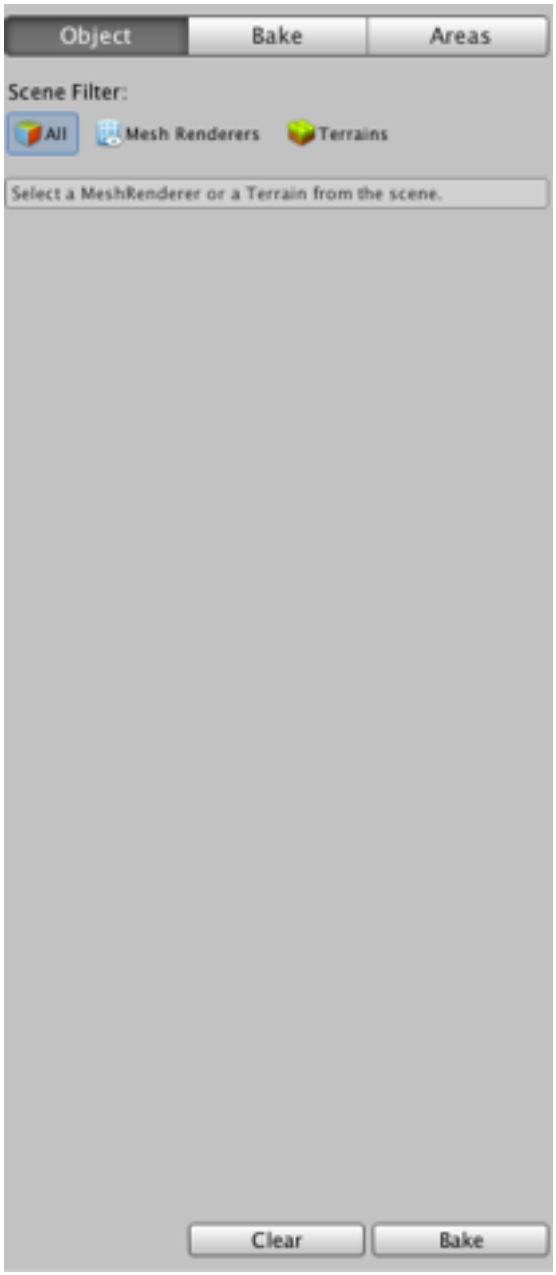


图2-1

Navigation Area(旧版叫做Navigation Layer)是对参与寻路功能的地图物体的一个分类，用层来分类，默认有三个层可以选择，当然也可以自己添加层。寻路层就是在2-1的图中的Areas中来设置。

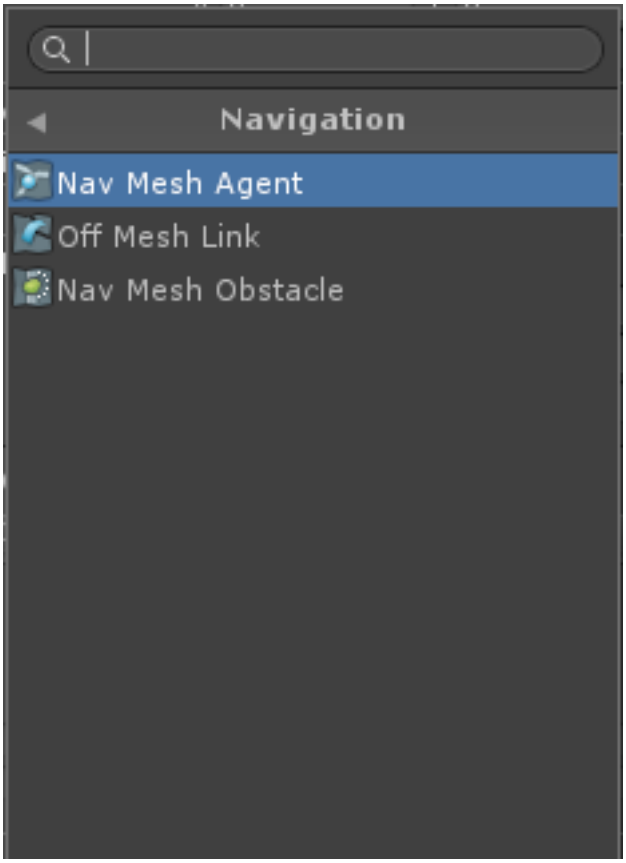
下面我们就来具体看看。

自带寻路Navmesh的三大组件：

1.Nav Mesh Agent：主要挂在寻路物体上

2.Off Mesh Link：实现区域转移功能(例如，有时不一定只是在地面上进行寻路，可能有些高高的平台，平台与地面是不相连的，使用该组件可以跳到平台上)

3.Nav Mesh Obstacle：主要挂在障碍物上。如下所示。



2.1 基础

1.选中静态对象，勾选Navigation Static

2.Window/Navigation，弹出Navigation视图，点击右下角的Bake按钮生成导航网格

3.在Bake选项卡中调整参数

4.新建一个胶囊体，Component/Navigation/Nav Mesh Agent，添加导航组件

5.为胶囊体添加脚本

```
using UnityEngine;
using System.Collections;
```

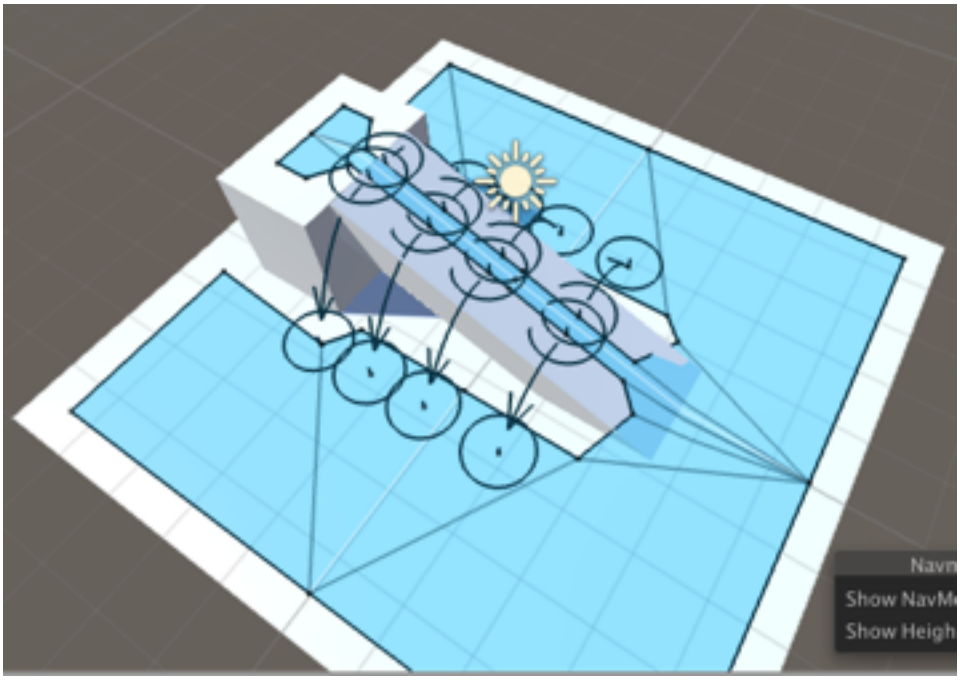
```
public class FindObj : MonoBehaviour {
    public Transform targetObj;
    void Start ()
    {
        GetComponent().destination = targetObj.position;
    }
}
```

2.2 其他功能

1.设置掉落点

a.选中所有静态对象，在Navigation视图的Object标签页下勾选OffMeshLink Generation

b.在Navigation视图的Bake标签页设置Drop Height，点击右下角的Bake按钮



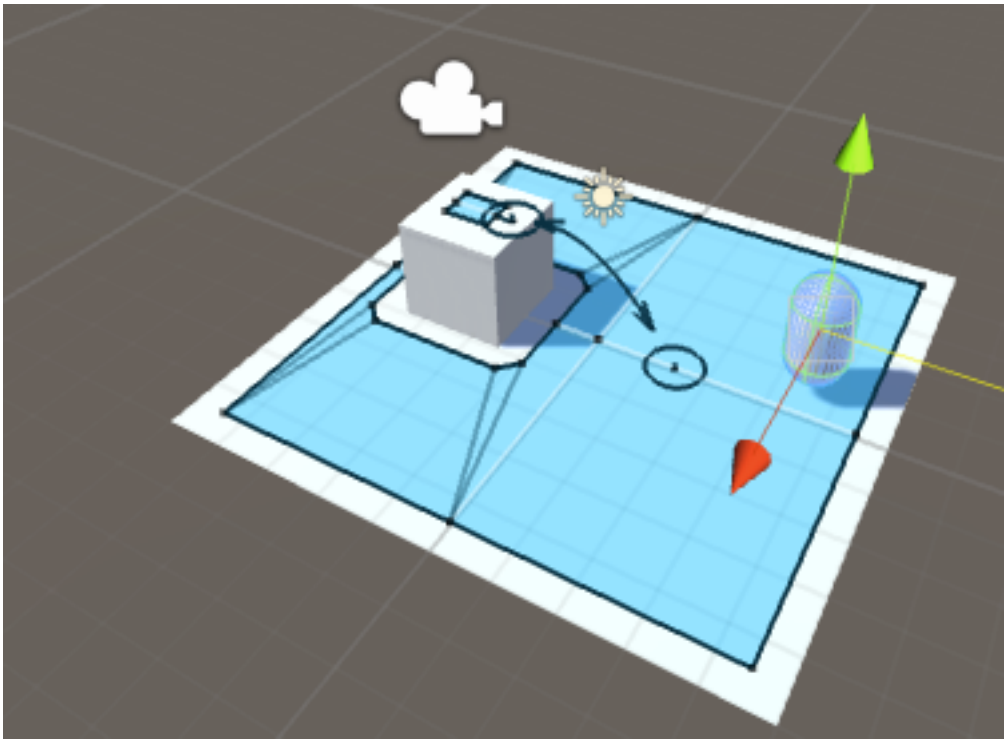
2.跳跃功能

a.添加两个对象，分别为start point和end point

b.对于start point，Component/Navigation/Off Mesh Link，为该组件设置好Start和End

c.Bake后即可看到连接路径，如果看不到可以调整Bake标签页的Step Height

d.新建一个胶囊体，Component/Navigation/Nav Mesh Agent，添加导航组件，添加上面的导航脚本。



如果希望上升过程丰富一些，比如播放一个爬梯或者飞行的动作，那么完全可以通过脚本来自行控制。

首先需要放弃勾选行进物体Nav Mesh Agent组件下的Auto Traverse Off Mesh Link选项，然后编写相应脚本来实现移动过程。在脚本中通过访问NavMeshAgent.isOnOffMeshLink成员来判断是否到达起点或终点，如果到达则访问NavMeshAgent.currentOffMeshLinkData成员来取得起点和终点的信息，最后实现自己的移动过程。完成移动后需要调用NavMeshAgent.CompleteOffMeshLink()来结束手动过渡过程。

3.为网格分层

在Navigation视图下的Areas标签页可以设置层，Object标签页可以为物体指定层，Nav Mesh Agent组件的Area Mask可以指定可行走的层。具有寻路过程中层的过滤功能。

4.动态更改可行进层

关于areaMask

Built-in 0 对应的areaMask为1

Built-in 1 对应的areaMask为2

Built-in 2 对应的areaMask为4

User3对应的 areaMask 为8

User4对应的 areaMask 为16

如此类推，即是areaMask为2的(n-1)次方

当设置areaMask为-1时，表示所有层都能通过

当设置areaMask为0时，表示所有层都不能通过

当设置areaMask为1时，表示只有Built-in 0层能通过

当设置areaMask为2时，表示只有Built-in 1层能通过

当设置areaMask为3时，表示只有Built-in 0和Built-in 1层能通过(3 = 1 + 2)

当设置areaMask为8时，表示只有User 3层能通过

可以通过查看Nav Mesh Agent组件的NavMesh Walkable看到设置areaMask后的结果。通过_agent.areaMask可以打印出寻路代理可行走的层。

5.Nav Mesh Obstacle组件(动态路障)

为路障挂上该组件，当该组件enable为true时，不可通过，否则可以通过。

该方法与设置可行进层的区别：

使用可行进层时，动态物体会在中断处暂停行进而等待新的路径出现后再继续行进，意味着在暂停的时候，物体的加速度为0;

而使用动态路障时，物体将不会暂停，而是一直在运动并试图绕过障碍来向目标点接近，意味着物体保持着一个加速度。

6.防止一群寻路的物体围住目标点

设置Nav Mesh Agent组件中的Obstacle Avoidance Type为None，即可以让寻路物体互相穿过。如下。