# 1各种自带窗口属性介绍汇集

## 1 选择和移动node

1

主要用到这些菜单功能



按下这个按钮：

点击鼠标按钮 然后鼠标左键 选中物体 就选中了一个node

按下shift+鼠标左键 可以选中多个

按下ctrl+a 就可以选中所有的node

选中物体 然后按下ctrl+d 就产品copy了一个 在同一个位置

选中物体 然后按下 detete键盘键 就可以删除node



按下这按钮 ：然后选中node 就可以旋转 缩放 移动



Center pivot 是用来切换选择对象的枢轴点,控制选择对象的运动或者旋转 是根据这个枢轴点来进行的。 当我们同事选择多个对象的时候 center 表示的是枢轴点喂这些对象的3d世界的中心处，当我们选择pivot时候 此时枢轴点为 最后选择的对象的3d中心处。

当然当我们就操作一个对象的时候 这两个是一个意思 没有区别。



用来指定 选择对象的旋转或者移动的枢轴点 是依旧什么坐标系来实现的。 世界坐标系 或者局部坐标系 或者父对象的局部坐标系



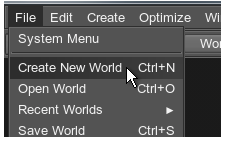
按照一定的角度来旋转选择的node。 选中这个按钮 然后设置angle

然后选中node 旋转就会按照这个单位来旋转 同理对



还不懂！！！

## 2 File菜单项功能介绍

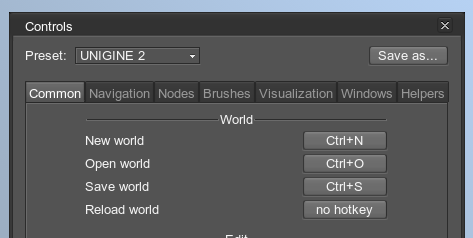


这里是控制 场景的创建 删除 保存

当创建了一个场景以后 就会出现两个文件

newworld.world :包含场景的渲染设定等东西

newworld.cpp：世界脚本文件



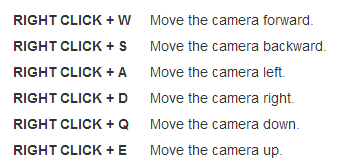
按下windows-》control 就是打开了如上的窗口，这是unigine预定义好的 快捷键

也可以在这里添加删除 修改

## 3 当前摄像机数据位置显示控制



也就是右下角的这块 如图，可以设置camera的移动速度 现实当前摄像机的位置。



控制摄像机前后左右上下运动，但是没有旋转它

按下右键+拖（上下拖动 或者左右拖动） 可以旋转摄像机的视野



F 摄像机到旋转的对象上面

按下鼠标滚条 就可以改变设计的 焦度。

在世界中可以有任意多个摄像机,把他们放在不同的位置 不同的角度,然后再camera面板就可以切换摄像机 ,这些摄像机虽然切换 但是上述的导航窗口控制什么的都适用。

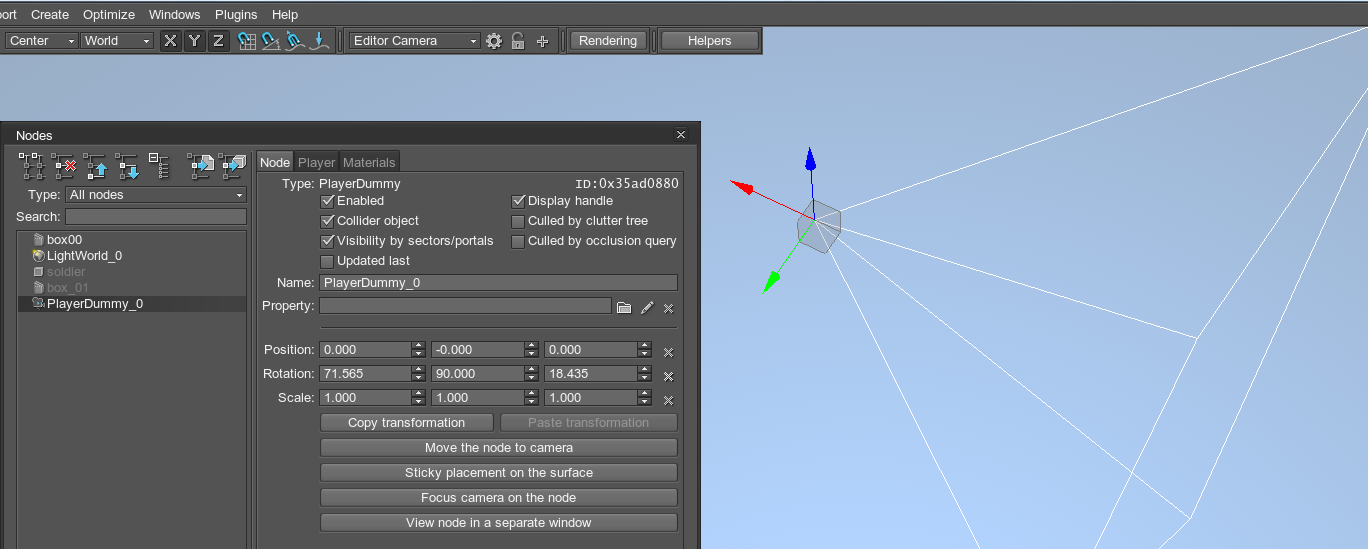
## 4 摄像机控制



系统默认会自带两个摄像机 game camera和editor camera摄像机两个。Game 摄像机是在世界脚本中创建的 可以被删除。它总是跟随着player

Editor摄像机是editor脚本创建的 它不能被删除 总是存在的。。。

我们在unigine中是不可能看到那两个默认摄像机的视域体,只能看到我们自己创建的自定义的摄像机,因为这些自定义的摄像机会座位node 出现在node窗口中。如图



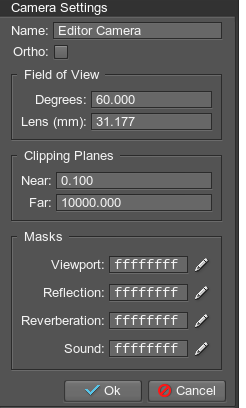
我们能在editor或者game摄像机下 看到新创建的摄像机的视域体。



分别是： 设置摄像机 锁定摄像机 新建摄像机。

新建摄像机 或被座位一个node放入node window下。也作为一个对象

然后我们就可以设置新建摄像机的参数了 点击

然后

Othro ：是正交试图还是 透视视图。

Masks： viewport 指定那些对象可以渲染在视口中

Reflection：指定那些对象的反射可以渲染在视口中

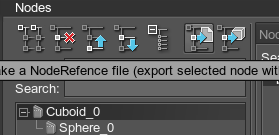
**Reverberation：回响 mask，和声音源和 回响区域的遮罩要匹配才可以听得到回响**

Sound: 和声音源的mask匹配就可以听到声音。没位是一个声音channel。

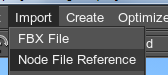
当你点击 按钮以后 ,就可以锁定以后 对这个摄像机设置的改变就不会保存。解锁后 就可以保存了。

## 5导入导出node和nodereference

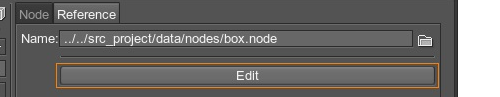
Nodepreference：当在场景中 有很多重复的node的时候 就要用到nodereference,这样就可以统一管理node 。相当于unity3d内部的 预设。 可以将一个node设置成 node reference 或者很多node设置成node reference。但是很多对象的时候 必须具有父子关系，我们将父亲node做成reference 子node 自然就也到同一个node文件里面去了。可以有同一个node的多个reference

点击  就会出现保存node reference的文件操作框架，然后就可以 做成reference。

导入node reference 就

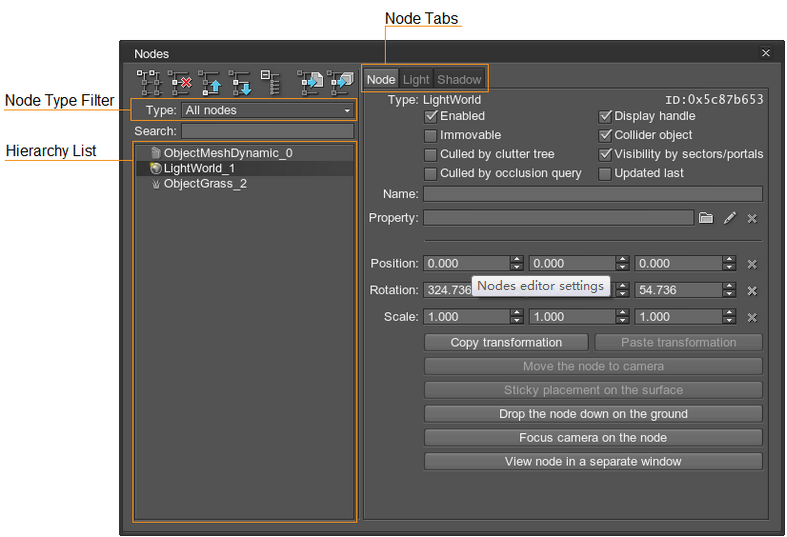
 选中前面那个node

在node windows窗口中 也可以编辑reference对应的源node 一变全都改变。



## 6 node 标签属性

按下n 打开node window窗口,能改变node的集成关系和 node的相关数据。



Node标签是所有的node都有的标签, 其他tab 依据不同的node类型,有不同的标签。

按下alt 选中一个node 拖动到另外一个 实现继承, 按下alt 拖动一个node到左边就解除继承。

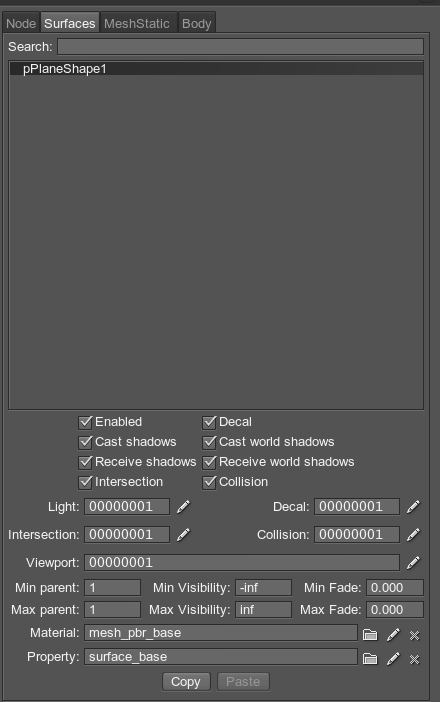
Type:结点类型 id ：结点id

Enable: 是否激活node

选中一组node 按下**ctrl+g**  就会自动变成一组。生成一个node dummy。

通常来说：第一级别的node都处于世界坐标系上，他们的孩子node 都位于父亲node的局部坐标系上，这样就很方便通过控制父亲node来控制整个继承列表。世界坐标系的起点都位于场景中心,局部坐标系的起点 位于父亲node的pivot point上。

## 7 surface 标签属性



每个node 都关联的对象。每个对象都有mesh,而一个mesh就可以有多个surface对应。比如人 分为 上身 外套 背包 下身之类的。每部分都是一个surface 。这样就在surface标签下 列出来《我们可以为每个surface设置属性。下面来讲解下 下面的 每个属性是什么作用！！

Min Visibility：与摄像机最小多少距离 能被看见 比如为a

Max Visibility：与摄像机最大多少距离 能被看见 比如为b

MinFade: 最小距离的缓冲距离 逐渐完全显现 比如c 则a-c 到a 逐渐完全显示

MaxFade:最大距离的火冲距离 逐渐完全消失 比如d 则b到b+d 逐渐完全消失

Min visibility 到max visibility是可以看见这个surface的距离。但是这个距离是这个surface和camera的位置计算而来的。 有时候会有问题,比如一个房子 从某个方向靠近的时候,有的surface是high lod 而有的是low lod。最终的效果还是low lod。所以我们需要设置

Min parent 和max parent。   
Min parent：如果为0 的话就是 本surface参与计算距离

Max preant: 如果为1的话就是这个surface的父亲surface 或者这个surface所属于的对象。

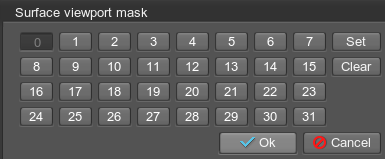
所以用默认就可以了。都设置成11.

Property：设置这个surface的属性

Material：设置这个sufface的材质

Light 或者decal intersection等都是 32位的mask。

# 2 Bit Masking



Bitmasking是一个32bit的数,额、每个位可以设置成0 或者设置成1.

Bitmasking的比较是通过 两个masking的每个位进行and运算。只有32位中有一个位and 后为1 那就是匹配成功了！！！

使用案例：

## Object 被渲染到viewport

Camera的 viewport mask 应该匹配

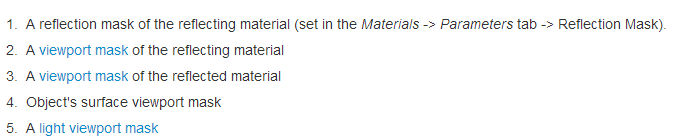
1：对象的surceface的 viewport mask

2：surface的material的view poatmask

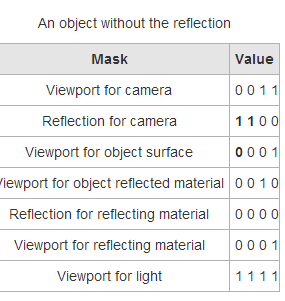
3：light的 viewport mask 否则场景是黑色的

默认情况下 所有的对象都可以渲染到viewport 内。

## Object reflection 被渲染到viewport



默认情况下 所有的反射都能渲染到view port中

理解这就可以。

## Light mask 和decal mask

# 3物理模块

## 1 物理总结

**Dynamic object 与static object**的区别就是：

1 拥有physical body （可以用来施加力）

2 拥有collision shape （碰撞探测）

而且 physical body能够被joint 来链接起来。

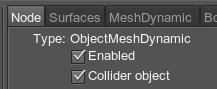
静态object 没有physical body,**不能给静态物体施加力, 所以是不动的 但是也可以防止别的对象穿过。**

碰撞总结：

没有body 也就没有shape标签。 所以可分：如果node 带有body标签的就是dynamic物体,如果么有带有body标签的就是 static 物体。

**静态物体碰撞**：**1首先 此时物体是静态的,不会动**

**2其次 此时物体如果也需要碰撞效果（本身不动 阻止别的object穿过）, 是属于碰撞对象的 。在node标签下勾选collider object。**



**3 最后 此时物体的碰撞是通过surface的碰撞来实现，此时要设置surface标签下面的 collision标志,以及collision mask 来剔除不碰撞的对象**

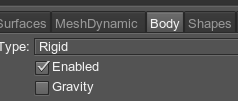
 

**动态物体碰撞：1首先：物体是运动的**

**2其次：此时物体也需要碰撞效果（被反弹或者被阻止穿过某个对象）,此时物体也是属于碰撞对象。在node标签下面勾选collider object。**

**3最后, 动态物体的碰撞是由碰撞器来实现的,也就是给对象添加的shape.此时surface的colider 设定不设定没有关系 无所谓的。只要给shape标签下面的碰撞器设定就可以。**

 当然body标签下面要先打开 **重力开关**

 勾选gravity才能利用碰撞器shape来碰撞。。

## 2 Collision mask

就是某些object 屏蔽某些对象的碰撞 就可以用mask 只要不匹配就可以了。

先决条件 对象先要有 physicalbody +shape。

如果要和某些对象碰撞 就要mask 有一比特匹配。如果不匹配，就会彼此穿过。

## 3 Intersection Mask

先决条件 对象先要有 physicalbody +shape

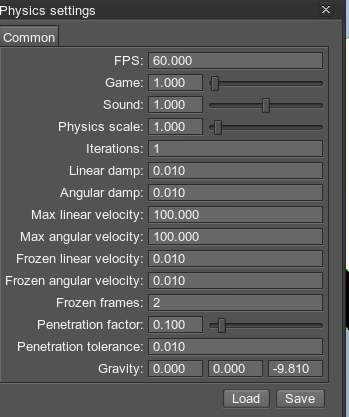
其实是 intersection ray 射线。能够减轻计算量。只要有一bit匹配。

## 4 Physical Mask

用来过滤对象 与一些物理node的碰撞 **比如一个物体是否受 force,wind 或者water的影响**

比如 ：如果一个对象放在水上 physical mask 没有匹配mask 就会沉下去。

## 5系统物理windows参数介绍



一个对象在场景中 只有 liner速度和angular 速度。

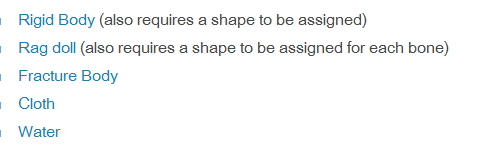
Liner angular damp：指定这两个速度的阻碍

Max liner veloticy ：指定这场景内的最大线性速度

Frozen liner velocity：指定场景内 一个物体静止时候 速度时要小雨这个值，并且时间要大于2帧 才算是静止了。Frozen famres 为2 就是这意思。

## 6 body类型

**Body 是用来模拟各种物理效果的。不同类型body 有不同物理特性**

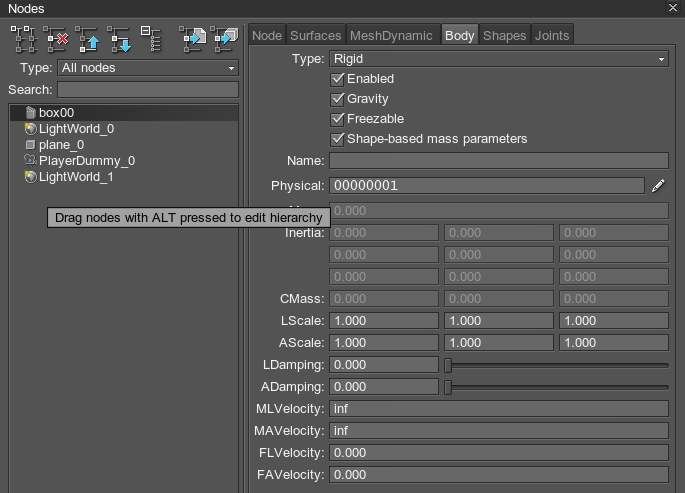


**Rigidbody ,rag doll类型的body必须含有shape。**

**Rigidbody dynamic：刚体动态学原理,这个适用于以下几种body：**

**Rigibody ,rag doll， Fracture body**

### Rigid body（刚体 身体）



如图：我给box00 添加了body类型为rigid body！

然后说说这个rigid body的属性

勾选 shape-》**shape-mass paramertes** 就是用默认的body属性 不勾选就会激活下面的灰色框框 要自己手动填写数值。

**Shape\_mass parameter**：勾选的话 就是根据shape（shape 有质量，因为他又体积 也有密度，body 只是说明一一些物理属性）。 所以我们勾选的话, 就会调用shape的物理效果来模拟body的物理效果。 如果我们不勾选的话, 那么就会激活下面的 inertia等栏目, 这时候body的物理效果等东西 就是根据body自己的属性来设置 和body无光

Mass:body 的质量。

Inertia 是惯性。

 xx 直的是 绕x轴旋转时候的x轴上的惯性

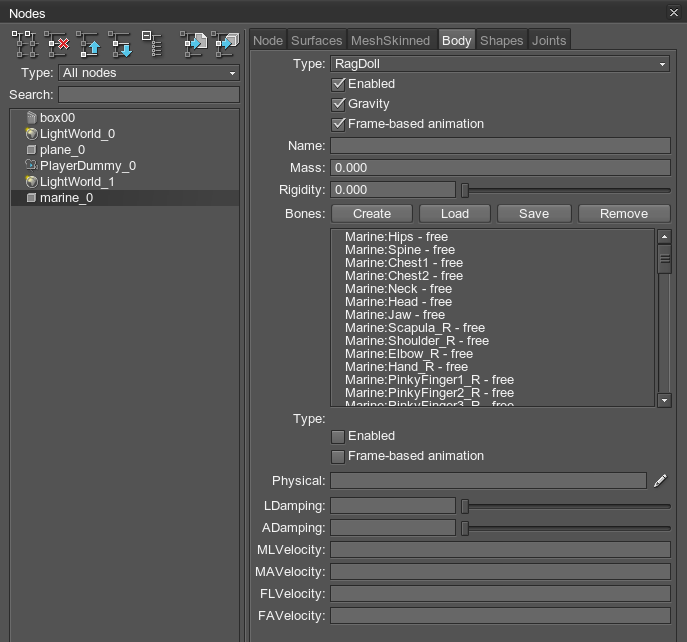
Xy 直的是 绕x轴旋转时候的y轴上的惯性

其他参数和pahyscial setting的设置是一样的。

Cmess：质心的在 对象的局部坐标系下的位置。

**Freezable：最好勾选 表示用户在操作这个body的时候 会有一定的延时 效果明显很多**

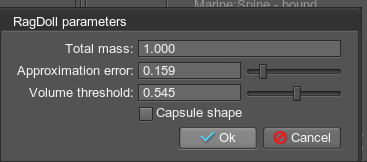
### RAG DOLL BODY（布娃娃 身体）



如图：我给marine 对象 添加了body类型为 ragdoll的body。

Ragdoll 只能用在 skinned mesh的对象上。

然后我们就需要给每个bone 骨头,添加一个rigidbody, 和一个shape（胶囊体或者 convex hull） 。可以点加create 自动长生这些东西。出现如下界面



分别为： 总质量 近似错误 胶囊体或者convex多靠近bones

Volume threadhold ：体积阀值 当小雨这个体积的bones 将不合并到靠近的大的bones里面。

### Fracture body（破碎 身体）

只能用在 dynamic mesh 对象下。

### Cloth body（衣服 身体）

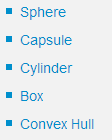
只能用在dynamic mesh对象下。用来模拟红旗 衣服等软的body效果

### Water body（水 身体）

可以用在 dynmic mesh和water mesh 下。

### Dummy body（虚拟 身体）

## 7 shapes 类型

有

不同类型的body是用来模拟不同类型的物理效果的,shape就是用来

表示 rigidbody在空间中占用的体积，、需要物理模拟的对象,总是带有一个body 和数量不等的shape 用来处理碰撞。在重力环境下,带有shape的body 会自由下落。如果一个body没有被赋予shape,就好像是一个dummy body 不碰撞 飘浮在空气中。我们建立一个node 这个node关联一个对象 比如说是一个mesh,然后再suface tab 标签下 就可以看到这个对象所有的表面 可以修改表面的属性 另外我们可以给这个对象 添加一个body 用来模拟物理效果 但是物理效果的模拟如果直接通过mesh本身来计算的话 效率太低 所以我们需要一个shape 就相当于这个对象的一个外部简单网格, 碰撞什么的都是通过这个shape来计算的 方便计算 提高效率。

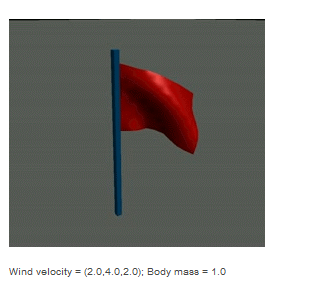
## 8 physicals 物理对象

### Physical wind

是一个cube形状的盒子对象,放入其中的对象 会受风的影响。

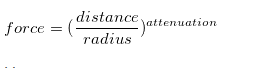
这个对象必须表姐在wind盒子的内部,而且必须被分配了 rigid body（必须有shape） 或者cloth body。

PhysicalWind 类



### Physical force

是一个球形形状的node，在球的中心是有一个力,这个力逐渐减小,



Distance: 对象和球体中心的距离

Fadius:球体半径

Attention: 力的atten 参数

只有在半径内的对象或者粒子对象才可以受力。对象必须含有:rigidbody(shape)，cloth body

以及rope body。

类：physical force

### Physical noise

### Physical water

是 一个cube形状的node，在这里面可以模拟水的效果。

### Physical trigger

总共有四种类型的 trigger,（球体，胶囊体,甜甜圈，盒子）物理或者非物理的对象进入trigger，就会促使 trigger对象调用回调函数。

物理对象： 必须含有 physical body（physical mask 匹配 trigger的physical mask）,

必须含有 shape（collision mask 匹配trigger的collision mask）

非物理对象： 必须在node的标签下 勾选 collider标志

必须在surface标签下 勾选 clooision标志

必须在sufface标签下 构造property属性的collison标志

类

PhysicalTrigger

# 4 Node 类型

Node是所有对象的基类,

基类： Node

派生类 nodedummy nodetrigger … objects player

Objects又派生出 objectstaticmesh objectmeshskinned….

Players 有派生出 playeractor playerpersecutor

## base Node 类型

**Node 分为两类。** Base node 和 related-object Node.

BaseNode ：**是看不见的的一些node**,比如 DummyNode 和layer pivot trigger 这四种。

Related-object Node：任何被添加到场景总的对象,就成对象相关的node.

### Dummy node

Dummy node：是一个没有大小的 看不见的node,用来组织其他node变成一个 hierarchy关系。当你按下ctrl+g 时候 就会自动生成一个dummy node 形成一个group。

或者通过  创建一个dummy node

### Layer node

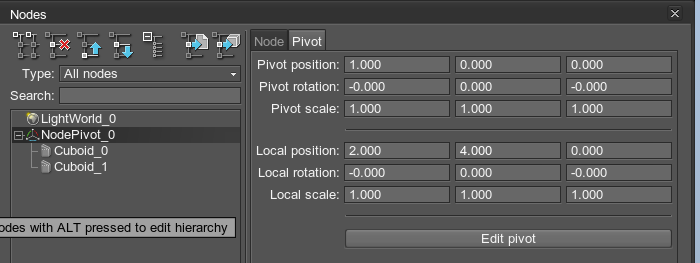
也是一个没有大小 看不见的node 用来组织其他node 变成一个hierarchy关系 而且同一个层内的数据是存放在同一个 .node文件内部。

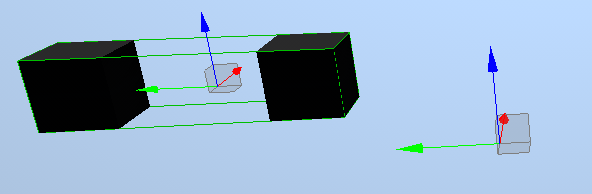


### Pivot node

枢轴node 主要是给他的子node 提供一个枢轴点pivot,当你需要给一组node设置一个特殊的povot的时候,就需要添加pivot node。

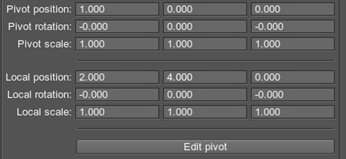
注意区别 pivot point 和pivot node。 这是不一样的。





Node tab：用来控制pivot node

Pivot tab：用来pivot point



Pivot position：用来调整枢轴点和枢轴对象的距离

Local positon：用来调整子对象整体相对于枢轴点坐标系下的 位置关系

### [Trigger Node](https://developer.unigine.com/en/docs/objects/nodes/node_trigger/#add_trigger)

出发node，是一个看不见 没有大小的node，当他enbale或者disable 或者transformation改变的时候,回调函数被调用,一般来说 trigger node都作为另外一个node 的子node,以便当他的双亲结点enable 或者disable 或者transformation 改变时候,调用回调函数。

但是这些回调函数必须是写在世界脚本当中的。如果你直接在编辑器中指定函数，那么这个函数只能接受一个或者零个 参数。如果完全是代码来控制就可以接受更多个参数。

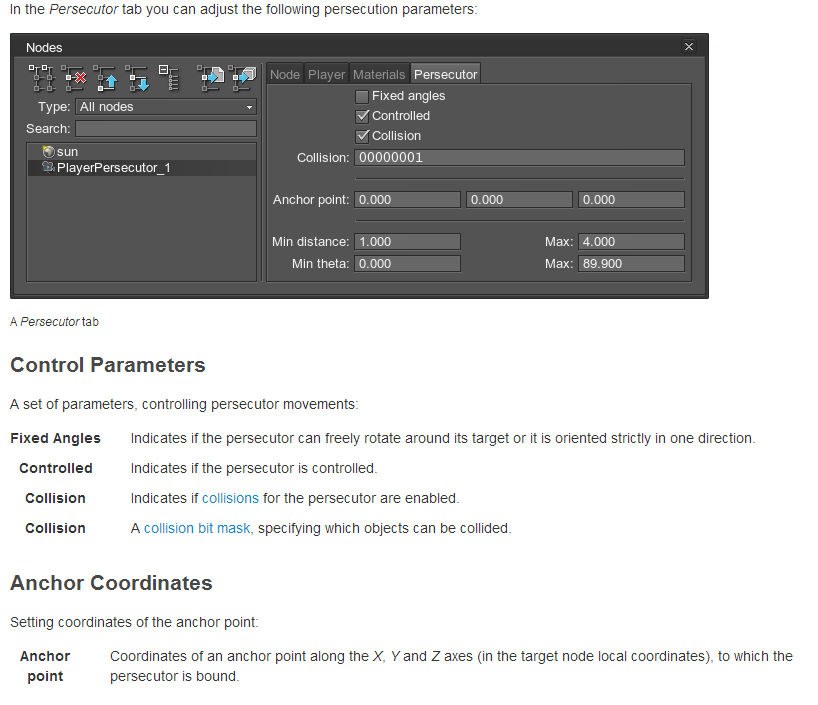
<https://developer.unigine.com/en/docs/scripting/library/nodes/class.nodetrigger>

例子： 比如打闪电的时候 要播放声音，就可以试用 trigger node。

脚本中：

NodeTrigger

## 对象相关Node类型



### Mesh cluster

成片成片的 房屋

### Mesh clutter

一定形状的草 等

### Node和NodeReference

除了那四种base node之外,在场景中的其他对象都是对象相关node。 可以被导出 或者导入或者制作成一个node reference。

**导出node:**



**导入node：**



**导入node reference：**

首先到处node 到node文件中然后



脚本中:

Node类

NodeReference类。

### Dummy object

**与dummy node不一样**。 这个对象可以分享对应的path body rigidbdy body 给他。他通常被用作一个带有物理属性的,看不见的对象。

脚本中：

ObjectDummy

### Static mesh

是三角形顶点的集合,他可以被移动 选装 缩放 但是不能被修改。通常用来作为 建筑物。静态网格是由很多的几何体构成。每个几何体被叫做surface。每个surface都有自己的属性和材质。一般都是在3dmax 软件中制作 staticmesh 导出为.mesh格式的文件。

脚本中：

ObjectMeshStatic:

### Skinned mesh

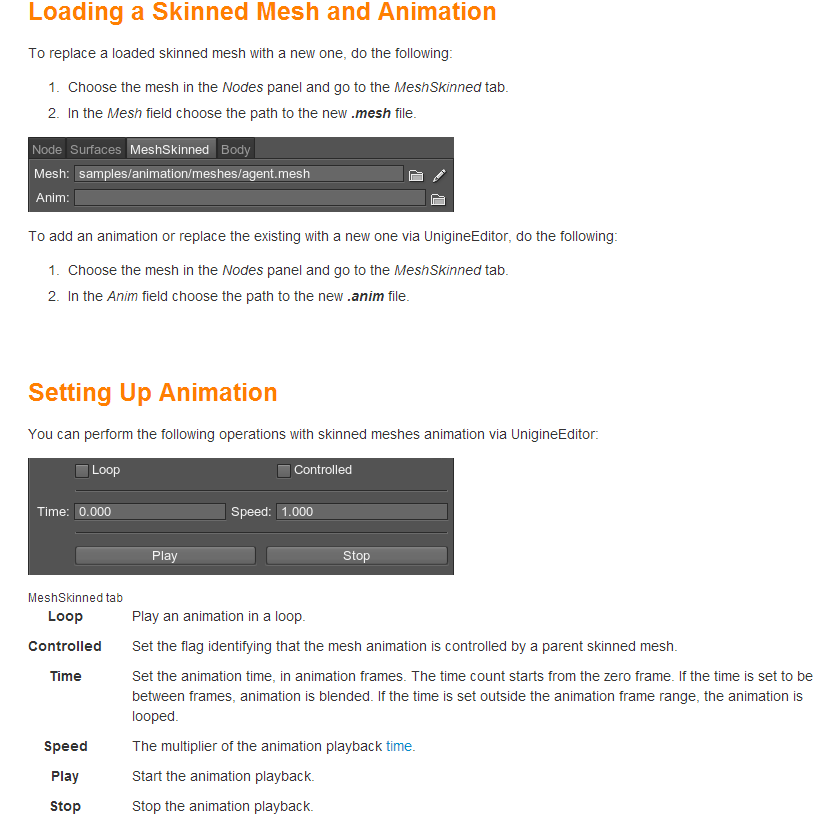
一个skinned mesh是一个object,包含mesh 几何体和 动画文件。动画文件和mesh文件可以包含在一个mesh文件内，或者分开 一个在mesh文件 一个anim文件。

当一个角色有很多动画片段时候 就分开mesh和anim文件

当一个较色动画片段不多时候 就和在一起 一个mesh文件 就可以了

脚本中：

ObjectMeshSkinned：



### Dynamic mesh

动态网格 主要可以修改导入的mesh。如衣服 红球 水面

### player

Players 就是创建世界视口的摄像机。包裹

Player actor：带有刚体body 只能在地面行走

Player dummy ：没有物理属性 不能喝其他对象碰撞

player [Persecutor](https://developer.unigine.com/en/docs/objects/players/persecutor/) ：是一个自由飞行的摄像机,没有物理body, 允许跟随某个目标对象处于一定的距离《 能和其他对象碰撞 但是不能相互作用

[Player Spectator](https://developer.unigine.com/en/docs/objects/players/spectator/) ：是一个自由飞翔的摄像机,没有物理body,用来长生一个观察者mode,可以和其他对象碰撞 但是不能相互作用。

## Player Actor

Player actor 就是一带有rigidbody的主角（**注意 不是camera 不能成像 只是一个带有刚体的bode**）。可以想象成一个带有物理属性的胶囊体。不像其他主角可以任意方向运动，actor 只能在地面上行走。可以用来做 第一人称角色

C++ 类： Unigine::PlayerActor 类。

## Player persecutor（第三人称视角摄像机）

是一个可以绕着目标物体 随意选装**的摄像机,**与**目标物体**保持一定的距离,**但是么有pyhsical body**。摄像机跟随的目标对象的那个点 叫做 anchor. Player 可以绕着目标旋转 或者固定在一个方向上。 这个player **可以想象成一个球体**,允许和其他物体碰撞 但是不能作用。

## Player spectator（）

是一个没有physical body的能自由飞行的摄像机。可**以想象成一个球**,能和其他对象碰撞《但是不能作用其他对象m

C++类：[Unigine::PlayerSpectator](https://developer.unigine.com/en/docs/cpp_api/reference/api_playerspectator_class)

## 20 Schemer

Schemer 是一个可视化的脚本编辑系统 ,不需要手动编写脚本。

**Schemer流程图包含以下几个实体：**

**Blocks:** 真是schemer的主要实体，用来描述 （ 事件 操作 或者不同类型的变量）。已经有一些已经定义好的blocks,用来展示各种各种的操作,或者打印信息。

**Nodes:**一个node就是一个已经被添加到 graph canvas （画布）中的一个bolck。

**Joints:** 用来连接不同的node,有两种类型的 joint, Paths和Links.

Paths:为白色,当schemer脚本运行的时候,他们控制执行node的流程。

Links:为绿色,他们只是简单的连接node,存放值 并不代表其他东西。

**Anchor:** nodes之间的连接点称为 anchor,有不同的形状, 圆形对应output 正方形对应input

注意：schemer editor 是一个画布,可以在它上面绘制流程图，用 unigine::widgets::graph类绘制

**使用方法：**

为了创建流程图，需要拖动blocks到右边的编辑器画布上。

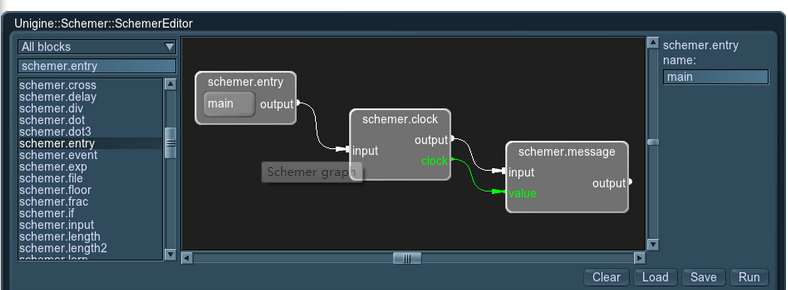
1第一个node,其实node,必须是 schemer.entry，而且必须舍子这个node的name为 main

2 继续拖动一些需要的block到画布上

3鼠标点击node的output 点上 拖动一个线到下一个node的 input点上。这些从input到output的连线控制了一个执行流程。（线是白色的）

4在一个node内部,这个**node操作相关的变量 ,**都**是绿色显示的。**

如果这个变量位于node的左边 那么就是本node操作需要的输入值,如果这个变量位于node的右边,那么就是笨node操作结束的输出值。 如果前一个node的输出值在下一个node中要被用到,那么久需要添加连接线, **此时连接线还是绿色。**



5在流程图创建好以后 连接线也定好以后，那么我们可以点击run按钮，一个script将被长生, 编译和运行。

也可以点加save按钮,讲这个脚本保存为.scripts的脚本， 为日后编辑和执行。

## 21 主角控制

## Node控制

直接获取node的 变换矩阵 然后这两个物体也没有body,此时就是依靠surface表面来碰撞,改变其中一个node的变换矩阵, 这样就会重合。后来我给绿色的添加了body shape 还是会重合。说明： 直接改变node的变换矩阵 是不会管碰撞什么的 直接到那个位置。

## 通过player来控制

Actor 胶囊体

## Unigne 创新点 特色点

Kinect2.0 体感输入设备

<https://developer.unigine.com/en/docs/start/sdk/addons/kinect2/>

multi monitor render

<https://developer.unigine.com/en/docs/principles/render/multi_monitor/>

## 23 c++ 代码总结

## 检测某个node 修改tab标签下的属性

//根据名字获取 nodeptr 然后根据ptr 有获取对应的node指针后设置现有的（node标签）的属性

pNode=Editor::get()->getNodeByName("Cuboid\_0");

cout<<"node id :"<<pNode.get()->getID()<<endl;

## 检测某个node 修改其他标签下的属性

Node分为两类, base node,一个是objcet-related node。

我们可以在运行的时候

直接创建base node,比如 node dummy 就会获得node的指针 ,就可以获取修改node标签的属性。目前没有办法去修改其他标签的属性。

直接创建 meshstatic这样的object(自动生成node) 此刻我们会获得objcet的指针,此刻就可以修改除了node至外地属性……

## 新建一个node 然后添加到editor下

//克隆一个node

pCloneNode=pNode.get()->clone();

//设置克隆node的位置

dmat4temperMatrix=pCloneNode.get()->getWorldTransform();

temperMatrix.setTranslate(dvec3(1,0,0));

pCloneNode.get()->setWorldTransform(temperMatrix);

cout<<"node id :"<<pCloneNode.get()->getID()<<endl;

//设置克隆对象的名字

pCloneNode.get()->setName("123");

Editor::get()->addNode(pCloneNode);//如果不用这句 editor类就检测不到新建的node。。。

cout<<"当前node数目:"<<Editor::get()->getNumNodes()<<endl;//

**总结：**我们在editor中 只能看到设计时候 手动添加好的游戏node,在运行时候 我们新建什么node 只会在游戏场景中看到。在editor中是看不到的。当然新建的场景如果没有利用editor：：addNode 方法的话 就不会被检测到,虽然在游戏场景中看的到。所有我们动态创建一个游戏对象后 都要addnode。

## 怎么从 PlayerDummyPtr 转化成PlayerPtr

PlayerDummyPtrppd;

Ppd->getPlayer() 就获取了对应的PlayerPtr

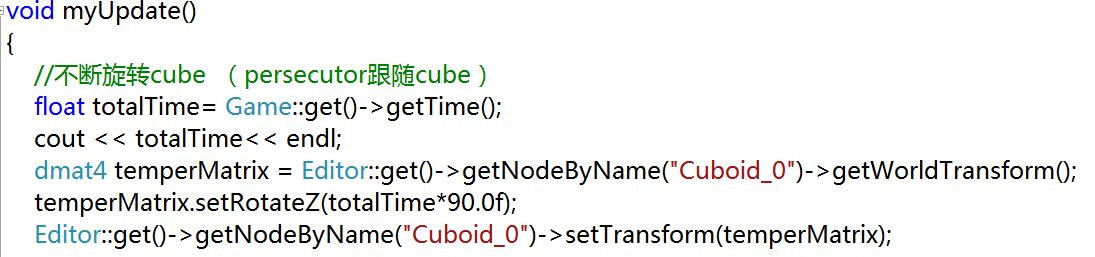
其他 object->node 或者具体object->object基类转化

从子类的ptr 转化成基类的ptr

## 设置游戏摄像机为一个自定义的 Persecutor camera



然后在myupdate里面不断改变cube的位置 这样跟随就会有效果



总结：设置了一个自定义的摄像机的话, 鼠标控制摄像机的视野的旋转还是可以的。就是跟随的target 目标是不受控制的, 现在接下来就是 做target的控制。Persecutor是一个第三人称的摄像机。

## 新建一个node

1 调用相关类的 构造函数

2 添加到edior内

3 设置材质

4 设置属性

5 设置位置

Ok！！记住一定要设置材质 设置属性 要不会看不见的。

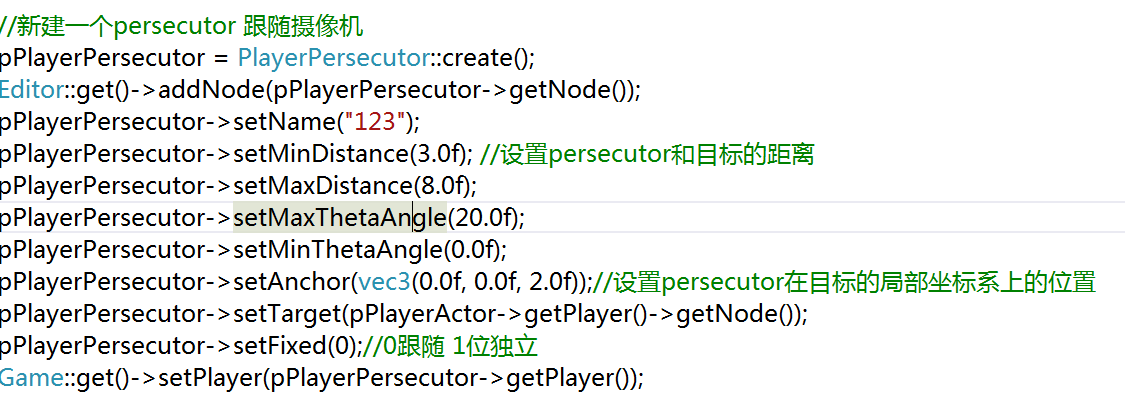
注意：任何新建的node 默认位置都是在（0，0，0）上。初始的局部坐标系都是和世界坐标系一样方向一样。但是当你旋转后,那肯定就不一样了。

## ObjectMeshSkinned 动画控制

### 新建一个 meskinned

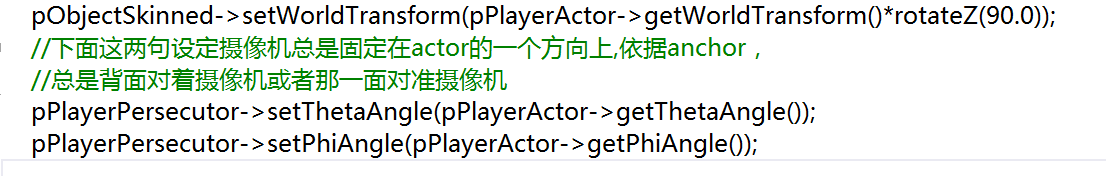


### 新家一个persecutor跟苏actor



### 让这个meshskinned 跟随actor运动

在while（）中添加如下代码

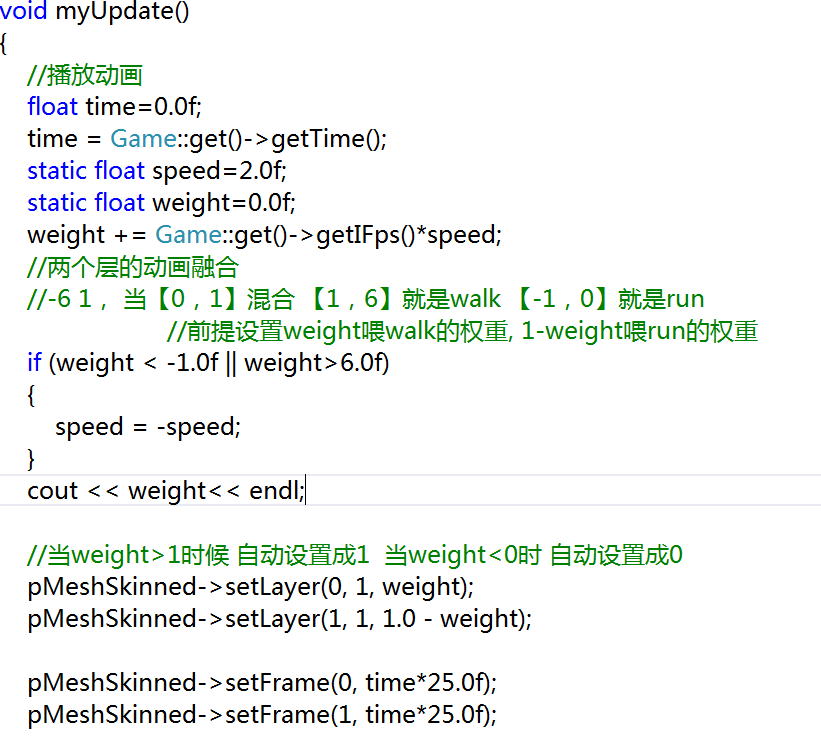


这样子 控制actor的运动 就会带着mesh运动。系统已经为我们做好了actor的控制，

Wasd 控制actor在自身坐标系下,沿着x轴正向 ,x轴负向 ,y轴正向 y轴负向移动。方向键盘 <- ->就是控制跟随actor的 persecutor的旋转。。我们在旋转摄像机的时候 actor也会自动跟随着旋转….

### 根据actor的运动情况,我们来设定meshskinned的动画播放

#### 两个动画层的融合



注意:

权值都是通过 ifps来改变

setFrame：都是通过 time来改变

默认情况下,系统只哟一个层,这个层是激活的 而且权重为1.我们如果设置了多个层,那么必须设置这些多的层的 激活状态 以及权重（缺一不可）。 Setlayer(层号,激活,权重)

## 24 unigineScript 代码总结

## 1数据类型

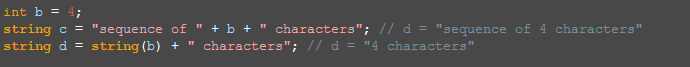
**1 基本数据类型**

int long float double string

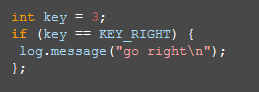
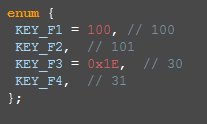
如果整数不写l 就默认为int类型

如果浮点数不写f 就默认为double

String 字符类型 不要求以’\0’ 结尾



**2枚举类型**



**3 3d相关数据类型**

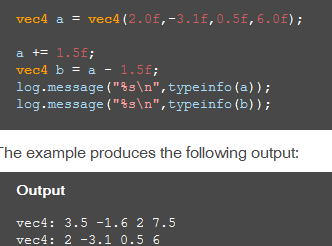
**1 vec3：默认三个float 类型**





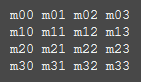
其他类型的 vec3类型类似：dvec3 ivec3

**2 Vec4：默认四个float类型**



**3 mat4：默认4\*4的float 类型**

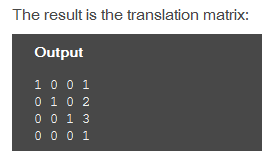
初始化的时候, 默认是单位矩阵。

 矩阵存放在内存的位置关系

**平移矩阵：**

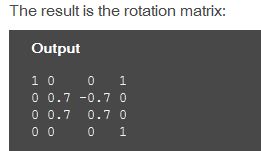
**或者：**



**旋转矩阵：**

**通过四元数：**

**通过向量：**



**4 quat：是四个float类型**

**默认情况下是（0,0,0,1） x,y,z代表旋转轴, w代表旋转角度**

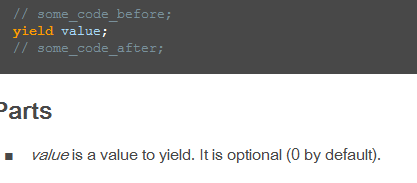
**直接初始化：**

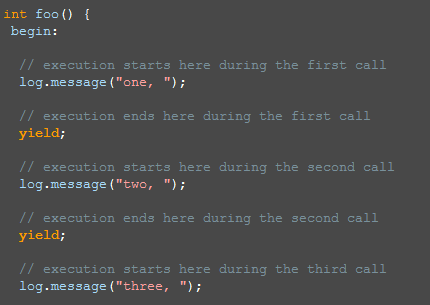
**通过mat4来初始化：**

**通过向量初始化：**

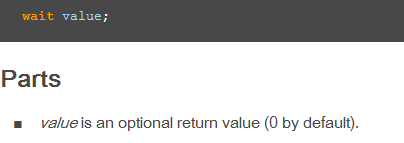
## 2关键字

### yield

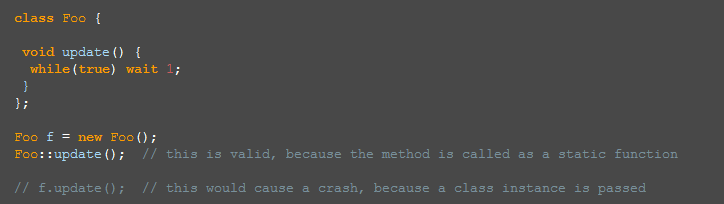




### wait



Wait关键字 **只能被用到线程函数里面！！！**



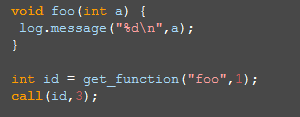
### Call

用来调用一个函数：

1 Call(函数名,函数参数列表)

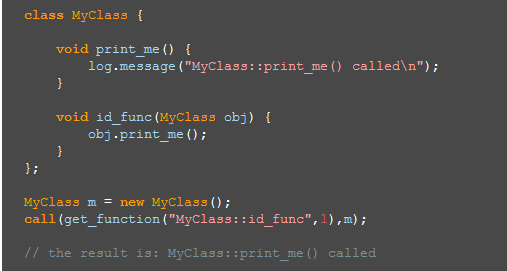


2 call(函数id,函数参数列表)

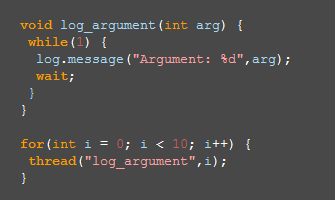


Int id=get\_function(函数名,参数个数)

3 如果你需要**调用给一个类的方法**



### thread



参数一：线程函数名, 参数二：线程函数的参数

新创建的线程会被放入一个FIFO 的线程队列里面,这些函数会在 世界脚本的函数执行完成后才执行, 并且线程函数必须用wait 关键字来释放处理器。

## 3 容器

### Vector

Vector数组的索引从0 开始。

 //预定一个固定大小的vector,此时vector只是一个名字 //可以换掉

/**/定义一个空的容器 必须制定他的大小为0**

//也可以直接在定义个数组的时候 直接初始化 **但是此时必须用( )**

 //添加数据20

 //删除索引处的数据

//返回vector数组的元素个数

//将vector数组内的数据都设置成0

//在vector数组内部 查找value的值 返回索引位置



//你也可以混合不同类型的数据到一个 数组里面。

### Map

# 5. Game Framework

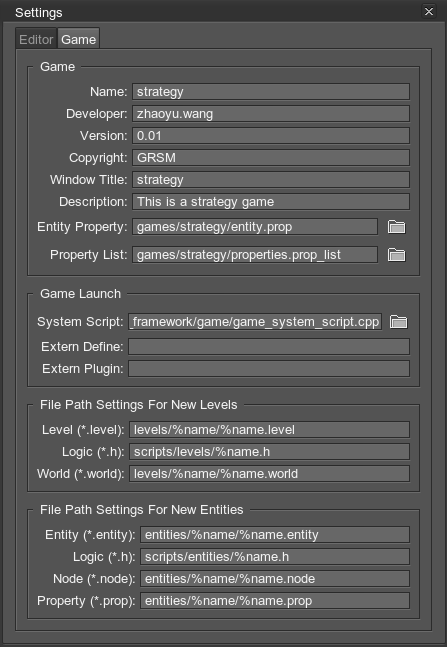
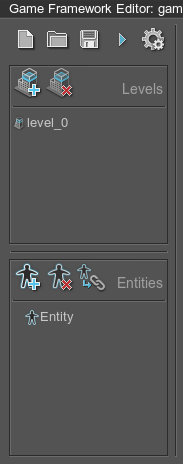
## 1. Script代码入口：

无Game Framework的程序入口：通过EngineArchitecture可猜测，程序的入口并没有暴露出来，而是通过载入脚本文件直接执行；而脚本文件的入口应当在$ProjectName/data/$ProjectName/$ProjectName##.cpp,其中包含了init(), shutdown(), update()；

## Game Framework代码入口：

基于Game Framework的程序，程序入口在add-ons/unigine\_game\_framework/game/game\_system\_script.cpp，实际上此文件中并没有main函数，估计main函数应当是被wrap在exe中，然后调用此脚本的cpp文件；

大体上，程序是由game类做主管，管理对象主要是level类和entity类；game类不仅仅管理level和entity，实际上也是对整个framework架构做了一个完整的wrapper，负责load\_level，设置文件路径等功能，如下图：



主要应当注意的有：GameLaunch/SystemScript， Game/EntityProperty；

level类主要包括world和logic（对应levels目录下的.level文件）；其中LogicClass是引擎自动生成的类，自动pulic inherit自level基类；world似乎主要是用来记录游戏世界环境；