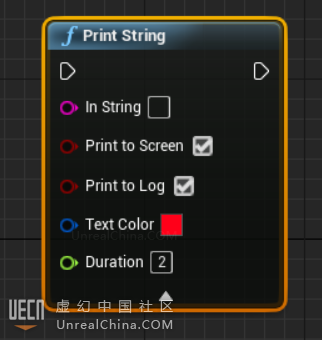
print string节点，位于‘工具’-----‘字符串’  
此节点虽然游戏中用不到，但对于调试项目可以起到化腐朽为神奇的功效  
粉红色的字符串变量用于接受需要打印的字符串  
第一个红色的布尔变量用来选择是否将字符串打印到屏幕上  
第二个红色的布尔变量用于选择是否将此次打印输出到日志里  
蓝色的结构体变量用于调整打印字符串的颜色  
最下面的浮点.变量决定打印的内容将会在屏幕上停留多少秒



add to viewport节点，位于“用户接口”-----“视口”  
它的功能是将你所做的UI呈现到屏幕上，它会根据UI的原始尺寸平铺到屏幕上。  
User Widget类型输入变量需要连接你需要的umg控件蓝图，  
你可以通过添加一个你的umg控件蓝图的变量来使它更加灵活。不过在这之前你必须先用create节点生成它。  
下面的整型变量为它的“优先级”意思就是当多个UI直接重叠时他会覆盖在“优先级”低的上面  
如果你不想让血条出现在游戏菜单里面，请设置这一项~~



Delay节点位于“工具”--“流程控制”  
这个节点非常实用，它的作用是延迟。  
浮点的输入变量用于决定延迟的秒数



事件tick，位于“添加事件”内，每一帧都会执行一次。输出的浮点数则是当前游戏的Ms值  
FPS:帧数越高越好  
MS：网络延迟，越低越好  
我想不明白的是如果是单机的话，网络延迟就没有了啊~~搞不懂  
总之这是一个非常常用的节点，但要限制使用它的次数，因为每一帧都执行计算会较大的耗费电脑的内存



事件beginplay，位于"添加事件"内，游戏开始时它将被执行一次。  
注意它被执行的时间在函数检索之后，如果用在除关卡蓝图或游戏控制器蓝图之外的一般蓝图，需要将其放置在场景中才可以被调用。



possess节点位于“Pawn”之中，用于人物之间的切换。如果在游戏中你死亡的话，你首先需要生成一个‘Pawn尸体’。  
然后使用此节点附身上去，最后销毁自己本来的‘Pawn尸体’实现复活功能。  
上面的游戏控制器变量需要输入你当前的游戏控制器，  
下面的Pawn变量，则是你需要附身的‘Pawn尸体’  
整个过程你可以理解为，将一个木偶的线棒扯下来，缝到另一个没有线棒的空白木偶上  
当然，你可以使用此节点，附身到任何关卡上存在的pawn上。  
它只存在于服务器之中，当然，如果游戏是单机则无需考虑此项==

一般此节点与**Spawn Actor节点一同使用，详情请见**  
**https://docs.unrealengine.com/latest/CHN/Gameplay/HowTo/RespawnPlayer/Blueprints/index.html**



un possess节点，位于“pawn”之中，它的功能是将游戏控制器与“pawn”之间断开连接。  
仅此而已，它同possess节点一样，并不会删掉原有的“pawn”  
就像将人偶的线扯下来丢在那里，如果你没将输入写进游戏控制器里面，而是直接写进了“pawn”里面  
那么此时视角将无法移动，因为游戏控制器已经与此”pawn”断开连接了。  
游戏控制器变量输入当前正在使用的controller，通常用get player controller来获取



DestroyActor节点位于“工具”------“类型转换”之中。  
它用于删除场景中已经存在的Actor，因为pawn是Actor的子类，所以此节点也可以删除人物，可以与possess节点搭配使用。此节点会彻底删除掉场景中的Actor并立即释放内存。  
Actor变量输入用于确定需要删除的对象，也可以使Actor的子类。



Detach from Controller Pending Destroy节点位于“pawn”之中。  
它的功能与un possess节点类似，不同的是unpossess节点的输入值是控制器，  
此节点的输入值是“pawn”前者是控制器主动脱离“pawn”  
，后者是“pawn”主动脱离控制器。  
二者在功能上并没有什么本质上的差别，都是使控制器与所控制的“pawn”断开联系。



分支节点，位于“工具”-----“流程控制”  
它的作用我想在整个编程世界都占有重要位置，它是一个思想。  
比如：这件事儿是真的吗？如果是真的，就走上面这条路，如果不是真的，就走下面那条路。  
布尔类型的输入变量是一个可以控制铁轨变更线路的开关，通常它会提升为变量，以便在其他地方修改这个布尔值，  
在你觉得有必要走那条铁轨的时候，直接控制开关就行了。



is local player controller节点位于“Pawn”。  
它用来判断所输入的游戏控制是否是存储在本地的。  
根据我的实验，在正常模式下它始终返回为真。  
在多人模式下服务器会返回真，并且返回假。而客户端则返回为真。  
我据此推断在多人模式下服务器会单独分成两部分，一部分是纯服务器（看不到），另一部分类似与客户端，尽管它俩全都显示的名字是（服务器）  
这就可以解释为什么会有三个返回值输出  
  
这个节点涉及到网络，感谢amd58919的提醒，在此只需了解即可，据我所知自定义事件对网络来说非常重要。网络对新手来说·还是相对复杂的。以后如果涉及网络的我还会加一些我个人的见解（可能是错误的求纠正~~）



add to viewport节点，位于“用户接口”-----“视口”  
它的功能是将你所做的UI呈现到屏幕上，它会根据UI的原始尺寸平铺到屏幕上。  
User Widget类型输入变量需要连接你需要的umg控件蓝图，  
你可以通过添加一个你的umg控件蓝图的变量来使它更加灵活。不过在这之前你必须先用create节点生成它。  
下面的整型变量为它的“优先级”意思就是当多个UI直接重叠时他会覆盖在“优先级”低的上面  
如果你不想让血条出现在游戏菜单里面，请设置这一项~~



Is Animation Playing节点位于“用户接口”------“动画”  
它的作用就是判断UI的控件蓝图里面你自己制作的ui动画是不是处于播放状态，如果在播放UI动画，它会返回真。  
上面的User widget引用输入，用来确定需要判断的是哪个控件蓝图。  
下面的In Animation引用输入，用来确定需要判断是否播放的UI动画。  
如果想得到上面的引用，请先使用create节点生成目标控件蓝图，如果没有事先生成，你将无法得到上面的变量。(如果你将此节点存放在控件蓝图内则保持默认即可，同时在其他蓝图里设置生成这个控件蓝图。）  
如果想得到下面的引用，请直接将此节点存放在控件蓝图内，这样可以直接引用UI动画的变量。（因为我发现UI动画似乎无法被其他蓝图引用？这只是我的一个猜测）  
如果你想创建一个UI动画，请在控件蓝图里找动画窗口，并创建动画。



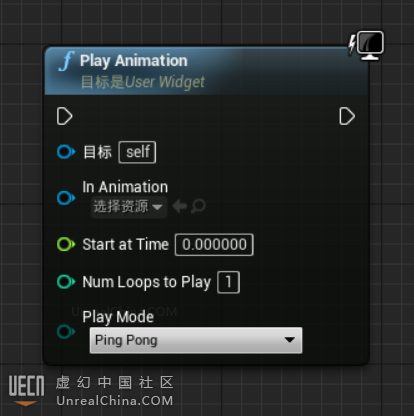
Is Playing Animation节点位于“用户接口”----“动画”  
它的功能与Is Animation Playing节点相似，这两个节点存放的位置也相同。  
你可以使用它来判断所输入的”控件蓝图变量”是否正在播放UI动画，如果在播放，就返回真。  
与Is Animation Playing节点不同的是，此节点判断的是控件蓝图内所有的UI动画，如果你的UI动画  
很多的话建议使用Is Animation Playing节点。但此节点更具灵活，打破了UI动画变量无法在其他蓝图引用的缺点。  
因此，此节点可以再控件蓝图以外的地方使用，并输出结果与Is Animation Playing节点相同。~~  
同样的是你需要事先生成这个“控件蓝图”（用create节点）~~



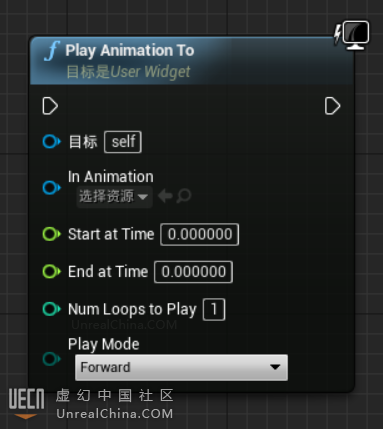
Pause Animation节点位于“用户接口”-----“动画”  
它的作用是在播放UI动画的过程中另其暂停。  
上面的User widget引用输入，用来确定需要判断的是哪个控件蓝图。  
下面的In Animation引用输入，用来确定需要暂停播放的UI动画。  
右边的浮点输出值为：暂停前已播放的UI动画的时间（秒）。  
同样的是你需要事先生成这个“控件蓝图”（用create节点）~~



Play Animation节点位于“用户接口”--------“动画”  
它的作用是播放指定的UI动画。  
User widget引用输入，用来确定需要判断的是哪个控件蓝图。（在这里重复的我就直接复制了~~）  
同样的是你需要事先生成这个“控件蓝图”（用create节点）~~（在这里重复的我就直接复制了~~）  
In Animation引用输入，用来确定需要播放的UI动画。  
Start at Time浮点类型的输入变量，用于输入你想在第几点几秒以后播放UI动画（默认为0）  
Num Loops to Play整型的输入变量，用于输入你想播放几遍相同的UI动画，当值为0时为循环播放。  
Play Mode枚举类型的输入变量，下面是“下拉列表中的翻译”  
--------------------------------------Forward(正方向播放)  
--------------------------------------Reverse(反方向播放)  
--------------------------------------Ping Pong(先正再反)  
  
还有就是枚举型变量我始终不知道有什么用~~有人能告诉我吗？~~



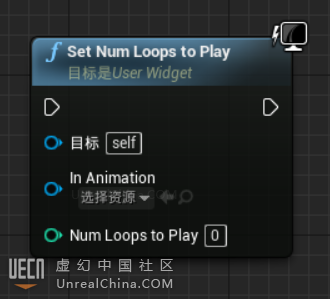
Play Animation To节点位于“用户接口”--------“动画”  
它与Play Animation节点的功能极其相似。其功能请参照Play Animation节点。  
与之不同的是此节点多了一个浮点类型的输入变量，（End at Time）。  
我觉得是Play Animation节点的升级版，它可以限定播放UI动画的范围，而不是直接播放全部，尽管Play Animation节点  
已经拥有限定开始时间的功能了。  
  
我个人理解：节点功能多，所需要执行的效率就越低。所以推荐使用Play Animation节点，而不是此节点。~~



Reverse Animation节点位于“用户接口”--------“动画”  
它的功能是反方向播放所选定的UI动画。  
（我认为这个节点非常的鸡肋，因为Play Animation节点已经拥有相同且更为高级的功能了）  
不过我认为它的效率一定要高于Play Animation节点~~  
上面的User widget引用输入，用来确定需要反向播放的是哪个控件蓝图。  
下面的In Animation引用输入，用来确定需要反向播放的UI动画。  
同样的是你需要事先生成这个“控件蓝图”（用create节点）~~  
（在这里重复的我就直接复制了~~）



Set Num Loops to Play节点位于“用户接口”--------“动画”  
它的功能是给播放的UI动画设置播放次数，当整形变量为零时循环播放。  
此节点更像Play Animation节点功能的一个分支。它的功能与其完全一样。  
  
上面的User widget引用输入，用来确定需要播放的是哪个控件蓝图。  
下面的In Animation引用输入，用来确定需要播放的UI动画。  
同样的是你需要事先生成这个“控件蓝图”（用create节点）~~  
（在这里重复的我就直接复制了~~）



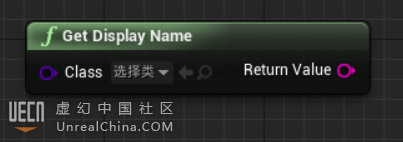
Stop Animation节点位于“用户接口”--------“动画”  
它的作用是停止播放的UI动画。  
它与Pause Animation节点的功能类似，不过他并没有输出变量。  
  
上面的User widget引用输入，用来确定需要停止播放的是哪个控件蓝图。  
下面的In Animation引用输入，用来确定需要停止播放的UI动画。  
同样的是你需要事先生成这个“控件蓝图”（用create节点）~~  
（在这里重复的我就直接复制了~~）  
  
让我疑惑的是“暂停”与“停止”他们应用时内存会有那些细微的变化，难道“暂停”功能不会释放所占用的内存吗？



Get Display Name节点位于“工具”------“按键”  
它的功能用于显示所输入的Object因为在虚幻里Object是所有蓝图的基类（头）。  
所以它可以用于输出任何蓝图的名称并转换成字符串。你可以用Print String节点将它显示出来。  
这个功能通常用于调试项目，比如你想知道是谁被碰撞了，谁被捡起了。  
在游戏中基本用不上~~  
另外如果需要应用此节点必须应用所输入的蓝图。如果是“Actor”你必须将其拖拽到关卡中，以便此节点调用。  
或者利用个别节点的输出充当此节点的输入。（这句话有点绕啊~）  
  
我暂时还没有一个系统的方法获取各类蓝图的引用，因为经我实验直接添加一个变量的方法，个别蓝图会报错。  
原因就是该蓝图没有被调用，你也可以通过cast to的方式来引用，不过这仅限于它的子蓝图。否则同样不会成功。  
我记得这部分貌似叫做蓝图的通讯吧~~



Get Display Name节点位于“工具”------“按键”  
它的功能同样是显示蓝图类的名字，你会发现它与上一个节点只有输入变成了紫色。Calss类蓝图为紫色，表示是在目录里存在的而不用必须应用它，  
你可以直接通过下拉列表选择你需要输出名字的“类”  
你会发现此节点可以选择比另一个更多的东西。而前面的仅限“资源”  
具体有什么样的差别，可以通过两个节点的下拉列表更直观的看到~~



append节点位于“工具”----“数组”。  
可能你找到的是append array。  
它的作用是将两个数组合并，意思就是把下面的数组复制一份，按顺序填加进上面数组的末尾。  
同时下面的数组保持不变~  
首先，你要懂得什么是数组，我会在未来的视频中详细讲解。  
数组是很多相同变量的组合，比如玩家列表就是一堆字符串的组合。它记录了很多玩家的名字  
（这是编程基础哦~~）  
当然，你不能将两个不同的数组放入这个节点，尽管它会自动转换，但我觉得你不会希望这样做的~  
当你将数组添加进去的时候，你会发现发变成其他颜色，如果你还不了解数组，就请点击一下变量右侧的小魔方，然后开启你的奇妙的编程之旅吧~~



shuffle节点位于“工具”----“数组”。  
它的功能是打乱数组的顺序，此节点并不会对数组内的变量做出改变。  
数组内的变量们也是有自己的顺序和编号的；比如  
-----------数组 = 【“第一个”，“第二个”，“第三个”】  
-----------其中，“第一个”编号为0，“第二个”编号为1，“第三个”编号为2  
此节点会打乱这个顺序~~  
数组打乱顺序也是一个编程基础的功能，这个是永久打乱，有没有临时打乱的呢~~



clear节点位于“工具”----“数组”。  
它的功能是删除掉数组内的所有变量，使其长度变为零。  
目前我还不太清楚有什么具体的必要应用这些功能。  
不过这同样也是编程基础哦~~  
  
你可以将数组输入魔方拖拽出来，你会发现“情景关联”里面变得少了许多。也就是需要用到数组功能的全部节点了。



resize节点位于“工具”----“数组”。  
它的功能是设置数组的长度。  
节点采用多退少补的原则，数组输入为需要设置的数组，不限种类。  
下面的整形变量为需要设置数组的长度。（大家不要以为这个节点只能整型数组用哦~）  
当数组内部的变量个数（这里用元素个数代替），小于所设置的整形个数，则在数组的最末尾补足所设置的整型数。  
（自动补充的元素的值全部默认为零，需要重新设置；如果是布尔数组。默认为假）  
当数组元素个数，大于所设置的整形个数，则节点将自动删掉多余的元素。

你也可以将整型值设置为负数，那样不会有任何变化。  
此节点通常用于在不知道将要有多少数据使用时，首先设置一个峰值，以免溢出~~



last index节点位于“工具”----“数组”。  
它用来判断所选数组一共有几个元素（变量）。  
因为计算机数数是从“0”开始的，所以右侧的整型输出值会比正常个数少一个。  
如果你想知道一个房间里到底有几个玩家，就可以用到这个节点。  
数组这类节点如果用到好可以做到化腐朽为神奇的功能。  
其次，数组也分一维，二维，三维，（由于个别情况使用方便给数组弄出等级）  
就像家谱一样，和类的继承有点相似，如果你要记录棋盘上的棋子。这时一维数组（正常数组）就显得捉襟见肘了，因为你不知到一个棋子它具体的编号（因为棋子是一个平面），如果用一维数组可能  
棋子=58  
如果用二维数组就可以~  
棋子=【6】【4】（第六行第四个）  
这样你就可以更直观的直接知道你选定的棋子的数组编号了。  
我认为多维数组仅仅帮助编写者分清楚而已，他们本质是一样的。  
同样，三维数组，可以用来标记你家是第几栋第几层，第几号  
通常只用的到三维，再往上就没什么必要了~~  
（不知道我的见解对不对~~请纠正~~）



contains节点位于“工具”----“数组”。  
它也可能叫做contains ltem  
它的作用是检查数组里有没有这个元素（变量）。  
上面的是需要检查的数组。  
下面是输入需要检查的元素。  
如果数组里面有这个元素，就输出真，如果没有就输出假。  
如果你把它连上一个数组，你会发现下面的值已经可以直接手动输入了。  
当然，你也可以引入一个变量来充当所需要判断的元素。  
~~



find节点位于“工具”----“数组”。  
它也可能叫做find ltem  
它的作用是查找所选元素存在于数组的第几个位置（当然也是减一的~）  
上面的输入用于检测需要判断的数组。  
下面的输入用于输入需要判断的元素。  
如果输入的元素不存在于数组之中，那么返回-1（这个功能和contains节点重复了~）  
如果输入的元素在数组里有多个则按照数组元素的顺序0,1,2,3，。。。来  
首先被检测到的元素返回编号。直接停止向下面的检测~~  
也就是说如果  
数组 = 【“哈哈”，“嘻嘻”，“霍霍”，“呵呵”，“哈哈”】他们的编号分别是0,1,2,3，4  
如果我检测“哈哈”在这个数组里第几个位置，它只会返回0，然后停止，不会再返回4~~  
如果我检测“呜呜”则直接返回-1~~



remove index节点位于“工具”----“数组”。  
它的作用是指定编号，删除数组里的元素。  
上面的输入用于检测需要删除元素的数组。  
下面的整型输入用于输入需要删除的元素编号。  
如果被删除的元素不是最后一个元素，则将后面的元素统一向前移动一个单位。  
使其顺序不变，元素值不变。  
相当于把一张牌从一摞牌中抽走，剩下的牌会自动覆盖在原来的牌之上。  
数组节点有的是输入数组的元素编号，有的是直接输入元素值。使用前请注意~~



remove节点位于“工具”-----“数组”。  
它也可能叫做remove ltem  
它的功能是，在数组内删除指定的元素。  
上面的输入用于检测需要删除元素的数组。  
下面的输入用于输入需要删除的元素。  
同样，它也会检测这个元素在没在这个数组里面，  
表现形式为：如果删除成功布尔值将会返回真，删除失败（数组里没有这个元素）则返回假。  
当你指定的元素被发现在数组里存在多个相同的元素时，也一并删除~~



length节点位于“工具”-----“数组”。  
它的作用是输出所选数组的长度。  
此节点与last index节点完全一样。  
让人哭笑不得的是此节点输出的内容是从1开始计算的，  
而last index节点是从0开始计算的，所以其结果正好比此节点少一。  
此节点完全鸡肋，鉴定完毕~~



add节点位于“工具”-----“数组”。  
它的功能是在所选数组的最后添加一个元素。  
上面的输入用于检测需要添加元素的数组。  
下面的输入用于输入需要添加的元素。  
右侧的整型输出的是所添加的元素所在的编号，（也就是说数组的元素个数啦~~）  
如果有一个玩家上线了，就把玩家列表里的元素添加一个吧，么么哒<(￣3￣)>   
突然感觉会看着很累的样子啊~~，还是多加点比喻和例子吧~~



insert节点位于“工具”----“数组”。  
它的功能是根据所给的编号位置，为数组添加一个自定义的元素（变量）  
你想这个新元素存在什么位置都行，但经过我实验，你必须输入原有最大编号以内的数值（超出这个数会没有任何反应，我也不知道为啥）  
第一个输入想要添加变量的数组。  
第二个输入想要添加的元素变量。  
第三个输入想要添加的位置。  
这样说可能形象一点：  
数组 = 【“a”，“b”，“c”，“d”，“e”】  
它的编号为 0，1，2，3，4  
这时我添加一个编号为2的元素“g”  
所以原本的2,3,4就向后搓了一位变成了  
数组 = 【“a”“b”“g”，“c”，“d”，“e”】



add unique节点位于“工具”----“数组”。  
它的作用是添加所输入的元素（变量）到数值。  
不过他用了一个更为特别的方法，此节点会检测数组里存不存在所输入的元素变量。  
如果有就直接退出不添加新元素；  
如果没有，就在数组的末尾添加这个元素（变量）~~  
另外，如果添加成功的话则输出此元素的编号~

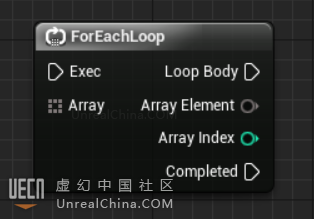


get节点节点位于“工具”----“数组”。  
它的功能是根据编号获取到所选数组里的元素。  
整型输入为希望获取到的元素的编号  
（如果你不知道你想获取的元素是第几号，就请不要用这个节点）  
  
蓝图节点里面也有很多以“get”开头的节点，他们的作用都是获取东西~~  
他们的输出值可以像毛细血管似得，将蓝图直接连接的更为紧密

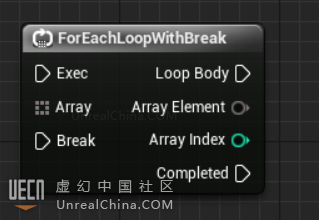


ForEachLoop节点位于“工具”----“数组”。  
它的作用是将数组的元素，按照编号的顺序，一项一项的不间断输出，俗称“遍历”  
众所周知，数组是由很多元素组成的，但是，你没有办法将很多的元素同时列出。  
计算机只会按照顺序将数组里的所有元素依次输出，这个过程很快，你可能无法感觉到，不过他们的确是有顺序的（这里忽略时间）。  
  
我想把玩家列表全部打印到屏幕上面，首先此节点会找到第一个玩家输出，然后回去再寻找第二个玩家，再输出。  
并依次进行，直到最后一个元素被输出才结束（这叫循环）（你甚至可以弄个上百万元素的数组，这对计算机来说就是几秒的事儿）  
  
左边输入需要（遍历）循环输出的数组。  
右边的LoopBoby（不管这个节点工作完不完成，直接开启下一个）  
completed（在遍历结束时调用）  
Array element（依次输出遍历出来的元素）  
Array index（依次输出上面所输出来的变量的编号）

~~知道你肯定不懂啦，到时候视频见~~



ForEachLoopWithBreak节点位于“工具”----“数组”。  
它的作用是ForEachLoop节点的升级版，  
你会发现它比ForEachLoop节点多了一条输入。  
这条输入可以随时停止遍历，不过我觉得没么用，  
其一，你不知道是遍历数组快，还是连接这个输入的节点快（没准你想停下的时候人家早就遍历完了）  
其二，除非你遍历非常多的数据，造成节点卡住了，否则我认为没什么必要非得让他停下来



创建数组节点位于“工具”----“数组”。  
它的功能是通过输入的变量组合出一个数组来。  
你可以通过点击右下角的加号来添加一个输入，（你还可以删掉多余的输入，鼠标悬浮在输入变量上面右键，可以删除）  
------------------------------------------------------------感谢天才小熊猫的大力支持，嘿嘿你要不说我还真没发现这个~~  
当然，你也可以仅通过一个变量就生成一个数组。  
生成后数组的顺序也与之前左侧的输入顺序一致~~  
  
另外添加一个小知识：如果你正在调试，可以将变量引脚处点击右键，“查看值”可以轻松查看变量的值，方便调试



Set Array Elem节点位于“工具”----“数组”。  
它的作用有点特别。  
首先你向数组里添加了一个元素，（按照所输入的编号位置）  
那么此位置上原有的元素将被删除，替换成你所给定的元素。  
其次，他还有一个功能，你注意到了那个红色的布尔了吗？当你将它点为“真”的时候。  
如果你填写的编号在超出了数组的长度，则依然添加这个元素，然后节点会自动帮你补齐中间的编号，（这些补齐的元素为默认值）  
（如果你没有使这个布尔为真，则不会有自动填充的功能，如果超出了数组的长度则没有反应）  
另外填写负值也是没有反应的~~  
  
注意这是替换，不是添加，它与添加节点不一样。  
如果你勾选布尔值变量就会拥有resize节点的功能。~~



Filter Array节点位于“工具”----“数组”。  
它的功能就是过滤，左侧的输入引脚，是过滤前的数组。  
下面的紫色的引脚输入你想查找的Actor。  
（注意，你输入的就是过滤之后剩下的，而不是将紫色输入的过滤掉）  
你可能疑惑，为什么紫色的明明是蓝图类，为啥可以被当成Atcor来筛选呢？  
你会发现在下拉列表中，不符合Actor类的蓝图全部都不见了。  
我也是弄了挺长时间才弄明白这个节点的意思。  
你的输入数组的类型也变成了Actor，作为输出的数组，它已经被删去了多余的元素，只剩下你所选的那个紫色的Acyor元素。  
  
我想不明白这样做的意义何在，既然已经知道了元素的名字（作为输入项），那么获取它应该不是问题。为啥还要过滤呢。  
何况我认为这个节点只能过滤出一个Actor来。



Enable Input节点位于“输入”  
它的功能是给予Actor使用按键的权限。  
通常，能用按键触发的蓝图只有关卡蓝图和控制器蓝图。在其他蓝图你用按键是没有任何反应的。~~当然，你还是必须事先将其丢进场景里面。  
  
此节点的上面输入需要按键功能的Actor蓝图，通常用“self”自己。（哪里用放哪里，就无需管它）  
下面是控制器（我觉得是需要获取控制器使用按键的权限吧~~）  
一般也用get player controller来连接。  
  
关于“self”在类的概念里，所有变量在生成时就已经复制一份进入“self”里面了，   
所以“self”可以代替改类，也就是该蓝图~~



Deactivate节点位于“组件”------“激活”。  
这是一个很有意思的节点；它可以将Actor组件停止使用，这并不意味着它会删除掉这个组件。  
像是暂停键一样，不同的组件连接它所显示的效果各不相同。  
我试过将Mesh(模板里的那个小人)连接此节点，他显示的效果是暂停动画，  
小人保持一个动作像一个小丑~~，因为你的动画蓝图是在Mesh里面设置的。  
其实应该还有很多效果没有显示出来。只要是所连接组件的功能全部被停止了，  
如果你连接摄像机组件，你会发现摄像机失效了。等等。。。  
总之，它的功能就是暂停组件，如果你希望游戏世界的时间停止可以用此节点来代替~~



类型转换为节点位于“工具”-----“类型转换”。  
你会发现非常非常多的此节点，他们负责将类型转换到名字上面写到的蓝图类上。  
首先，为什么要使用这个功能；有些功能是个别蓝图特有的，比如关卡蓝图与控制器蓝图就具有输入按键的权利。  
  
其实你也可以使用Enable Input节点来使其他蓝图类也拥有这个功能，不过这不是这节的重点。  
当你需要其他蓝图的“功能”，“变量”或者是其他蓝图自身的引用时。你可以用此节点来实现。  
（另外说一句，变量通常只能在本蓝图生效，引用只有在Actor放置到场景中才能被其他蓝图调用。）  
当然你得实现将变量设置为共有才行（点开小眼睛）  
类型转换只能是转换到所连接的输入端的子类才能成功，如果你需要这个蓝图类里面的变量，你可以通过Get节点  
获取的它的父类引用。  
这个过程叫做蓝图的通讯，稍后我还会介绍另外几种通讯方法，他们是蓝图的毛细血管。  
右侧有一个cast failed，意思是转换失败时调用，这也意味着此节点不会一直成功，这取决于  
连接的输入端是不是他的父类，如果你想知道某节点的输出到底是什么（比如碰撞检测，碰到Actor与Pawn的输出绝对不是一样的）  
我一发子弹打到了石头上，将石头这个输出的引用连接到此节点的输入上（比如你想给怪物减血）那么就会转换失败，  
因为石头类不是怪物类的父类。（这是模糊的一个使用方法）  
（明确的使用方法就是直接用Get节点获取到它的父类，这会一直成功，而不用管cast failed输出了。）  
右下的引用输出在成功转换后会开始输出，这是就可以为所欲为的调用此节点代表的蓝图类里面的一切了~~



VectorLength节点位于“数学”-----“向量”。  
大家知道金黄色变量代表向量型数据，你可以把向量理解成在空间里一条带箭头的直线。  
最原始的时候这条线的起点就是0,0,0（也就是你点击一下Actor出现的坐标轴的那个点）  
当你创建一个向量类型的变量后，你可以改变这个坐标，  
于是从原点，到达你输入的点，二者连线，之间的长度就是此节点的输出值。  
  
向量型变量还可以将它显示3D控件，在场景中生成一个小菱形体以便确定位置，你可以把值调为原点来仔细揣摩一下上述的意思。  
（当然前提是你需要点一下小眼睛把这个变量暴露出来）。  
  
这个长度就叫做向量的（膜长），高中的数学知识。它可以用在游戏里测量距离，在官方实例Example Content中的Blueprints\_Advanced关卡里有一个会变长的篱笆，就是应用了向量的知识。



round节点，位于“数学”----“浮点型”。  
它的功能是输入一个浮点型，将其四舍五入变成一个整型。  
这个也在上述的实例中以一种非常巧妙的方式呈现。总之不要忽视它的用途，，其实这个节点非常有用。~~



Set Material位于“渲染”------“材质”。  
作用：设置某StaticMesh（静态网格体）的某个材质通道的材质  
目标：要设置的StaticMesh（静态网格体）  
Element Index：目标如果是多维材质，指定要设置哪个通道  
Material：可以是基本材质，也可以是材质实例  
另外Set Material里面的Element Index是通道ID，一个物体具有多个材质的时候（多维材质），你需要指定要设置哪个ID的材质



FlipFlop节点位于“工具”------“流程控制”。  
作用：第一次输入则输出A，第二次输入则输出B。  
Is A：如果输出的是A则返回真，否则返回假。  
通常用于“开关”。



Play Animation（人物）节点位于“Components”-----“Animation”  
  
作用：给骨架网格物体播放动画。  
目标：要设置的骨架网格物体（人物模型）  
New Anim to Play：设置动画序列（可以是动画蒙太奇）（可以是混合空间）。  
Looping:是否循环播放动画。  
人物动画主要还是动画蓝图，个别用的少的动作可以用此节点。  
动画蒙太奇是动画加声音或特效，做技能用。  
混合空间用于多个动画混合，可以让动画不是那么生硬。



get actor transform节点位于“工具”----“变换”。  
作用：获取Actor的3D变换（Transform）一个橙色的变量。  
这个橙色的变量是一个“结构体”，它包含了Actor的位置，旋转，缩放。  
结构体就是许多变量结合到一块，与数组不同的是结构体通常用的不是一种变量，而是多种。  
如果你想单独修改其中一个值请先将其用break节点打开，修改后再用make节点装好。~~  
我搞不明白的是获取位置或旋转的节点有很多，他们往往输入的“目标”不同，但效果却相同~~  
Actor目标，与某某组件为目标有什么区别呢~~



set looping节点位于“组件”----“时间轴”。  
此节点一共有五个一样的，分别为不同目标使用。  
作用：将时间轴引用设置为循环。  
（你要记住Looping是循环的意思）  
时间轴一般用于静态网格物体，如果需要此节点的功能，我觉得是风吹在旗子上，旗子会一直飘动。（因为旗子不会停下来，所以需要此节点设置一下）。  
其实在时间轴节点可以直接设置此项，不过此节点更为灵活。~~



set play rate节点位于“组件”----“时间轴”。  
同样，他也有很多名字相同的节点。  
功能：设置时间轴的播放速度。  
new rate：播放速度的倍数，值越大，播放的越快。



stop节点位于“组件”----“时间轴”。  
作用：使时间轴立即停止（不是返回原点）  
》》》最近正在学编程，等我学好以后就做一个虚幻教程的网站，然后山寨官网也弄一个中文版的~~



play节点位于“组件”----“时间轴”。  
作用：让处于停止状态的时间轴恢复播放（可以与stop节点搭配使用~）。  
停止和恢复时间轴的搭配使用是一个很有意思的效果。



reverse节点位于“组件”----“时间轴”。  
作用：反向播放时间轴。  
（从调用那一刻开始立即反向播放，而不是跳到最后再反向播放）  
尽管时间轴节点上已经存在了此功能，但不论是效率，还是美观程度，显得此类重复的节点也挺有必要的~~



reverse from end节点位于“组件”----“时间轴”。  
功能：从被调用的那一刻起，让时间轴跳到最末尾，然后反向播放它。



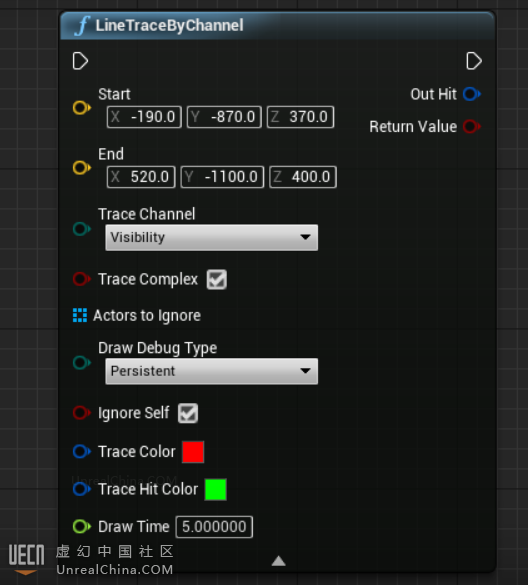
play from start节点位于“组件”----“时间轴”。  
功能：从头开始播放时间轴。  
不过我认为这个功能并不是很常用~~



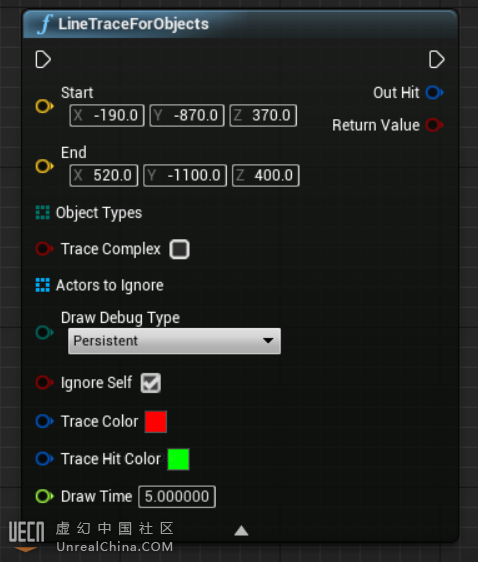
set new time节点位于“组件”----“时间轴”。  
作用：设置时间轴从第几秒开始播放，必须在时间轴未播放状态下调用，否则会报错的~



LineTraceByChannel节点位于“碰撞”。  
作用：发射一条射线用于检测碰撞。  
Start：这条射线的起始位置。  
End：这条射线的结束位置。（两点确定一条直线）  
Trace Channel：用于检测的东西是什么类型。  
（这是个枚举值）“visibility”可见的，“camera”相机。~~  
（个人推测camera是不可见的，因为没有试验，所以以后更改）  
Trace Complex：是否检测复杂的碰撞（这个复杂的概念目前还没搞懂）~~  
Actors to Ignore：这个一个数组，用于忽略检测其中的Actor。  
Draw Debug Type：这条射线的状态（也是个枚举）  
“none”不显示射线，“for one frame”只显示一瞬间射线，“for duration”根据最下方的浮点值确定显示射线的时间，“persistent”一直显示这条射线。~~~  
Ignore Self：检测的时候忽略检测自身。  
Trace Color：未检测之前那段的颜色  
Trace Hit Color：检测到之后的那段颜色  
Draw Time：仅仅Draw Debug Type为“for duration”值时可用。  
Out Hit：检测的输出，如果没有则为空，这个结构体包含相当多的东西，以后会介绍。  
Return Value：如果检测到符合要求的就输出真，没有检测到就输出假。  
~~~~其中有几个不明白的不知能否得到解答~~~



LineTraceForObject节点位于“碰撞”。  
它与LineTraceByChannel节点的功能一样，都是射线检测。  
不同的是Trace Channel的枚举变成了Object Types的枚举数组。  
我觉得两者之间的差别在于，所检测的对象不同。你可以看到此节点上的对象已经变成了数组，意思是同时可以检测多种，而前者只能同时检测一种，且二者检测的种类也发生了变化，因为我也不是很明白，所以不能对枚举的每个词汇代表的意思做出正确的解释。~~~  
枚举变量，其实就是简简单单的一组“名字”。与钥匙一样，而大门却在函数里面，具体怎么样因为我刚刚学习编程的缘故还不了解。我所知道的是，你会发现很多的不同的互不兼容的枚举，因为他们都是互不相同的钥匙。  
~~这就可以解释为啥创建一个枚举，只需依次写入名称就行了，不用管其他的东西，因为其他的东西全是在函数里面定义的~~~~~（不知道我说的对不对~~）



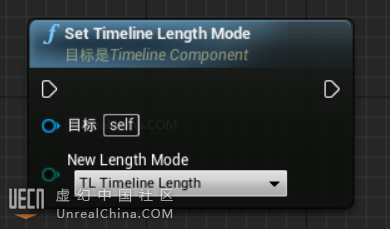
set playback position节点位于“组件”----“时间轴”。  
作用：把时间轴跳的第几秒播放。类似于set new time节点。、  
不过两个布尔值实在不知道干啥用的



set timeline length节点位于“组件”----“时间轴”。  
作用：设置时间轴的长度，（只能在执行时间轴之前使用，另外设置的长度还必须比原本的低）  
总之感觉很鸡肋~~



set timeline length mode节点位于“组件”----“时间轴”。  
作用：设置时间轴定义长度的方法。  
枚举：TL Timeline Length---以时间长度为真实的播放长度。  
         TL Last Key Frame----以0到最后一个关键帧为真实播放长度。  
通常，大家习惯用多少就将两个关键帧设置多少，不过如果你想前后空出几秒空白的长度就需要转换成下面的模式了~~



is playing节点位于“组件”----“时间轴”。  
作用：判断所选时间轴是不是在播放状态，如果是，返回真



is reversing节点位于“组件”----“时间轴”。  
作用：判断所选时间轴是不是在反向播放，如果是，返回真



is looping节点位于“组件”----“时间轴”。  
作用：判断所选时间轴是不是在循环播放，如果是，返回真



get timeline length节点位于“组件”----“时间轴”。  
作用：输出时间轴的长度，返回几秒



get playback position节点位于“组件”----“时间轴”。  
作用：获取时间轴当前播放到第几秒，精确到小数点后六位



MultiGate节点位于‘工具’-----‘流程控制’。  
作用：按一定顺序调用右侧的代码。  
Reset重置，当所有代码被执行后如果没有设置循环执行，则需要此输入设置重置。否则只执行一轮。  
Is Random，是否设置顺序为随机。  
Loop，是否循环调用。  
Start Index，首次执行哪一条。



Set Static Mesh节点位于‘组件’------‘静态网格物体’。  
作用：帮助有Mesh模型组件的蓝图替换模型。  
        Target：需要替换的Mesh组件，一般直接引用蓝图即可，类型会自动转换成Mesh组件。  
        New Mesh：想替换成什么模型。  
        Return Value：如果成功则输出真，一般都会成功。（我没有遇到过输出假的时候）  
成功替换以后“貌似”会失去材质



Random Integer in Range节点位于‘数学’----‘随机’。  
作用：在最小值与最大值之间取随机值，并输出。

