# 作业8-VR项目尝试

## 前言

本次作业中，我在之前自定义着色器的类minecraft方块放置项目基础上，使用unity中的Meta XR SDK对quest2平台上的VR软件开发进行了尝试。由于时间有限，我没能实现作业要求中的“游戏开始和结束条件“，”游戏计分机制“以及像分数统计、开始界面等的用户界面。与之相对地，我将重点放在对这个SDK中的各个功能的分析和应用上，尤其是对手势的部分。在这个报告中，我将阐述在演示视频中所展示的，对其中部分功能研究出的一些结果和看法，以及具体应用的效果。

在VR平台上的开发使用的SDK有多种。目前主流的为OpenXR，它是一个为了简化 XR 开发而制定的一套标准，相当于在编写的 XR 应用程序和不同的硬件设备间新增了一个中间层，最上层是 XR 应用程序，最底层是硬件设备，而这两个层的中间夹了一个 OpenXR 层，它向上提供一套统一的开发 API，不同的开发引擎对该 API 进行二次封装，提供给 XR 应用开发者使用。而不同的设备厂商为硬件编写支持 OpenXR 标准的接口，使得中间层能够向下兼容对应的硬件设备。如今，越来越多的 XR 设备开始支持 OpenXR 标准，这意味着选用 OpenXR 进行 XR 设备的开发可以成为一种通用的方案。

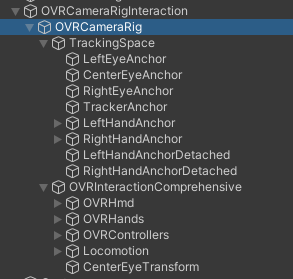
*SDK介绍部分参考于*[*https://blog.csdn.net/qq\_46044366/article/details/133926318*](https://blog.csdn.net/qq_46044366/article/details/133926318)*——Unity Meta Quest 开发（一）导论：开发现状与主流 SDK。*



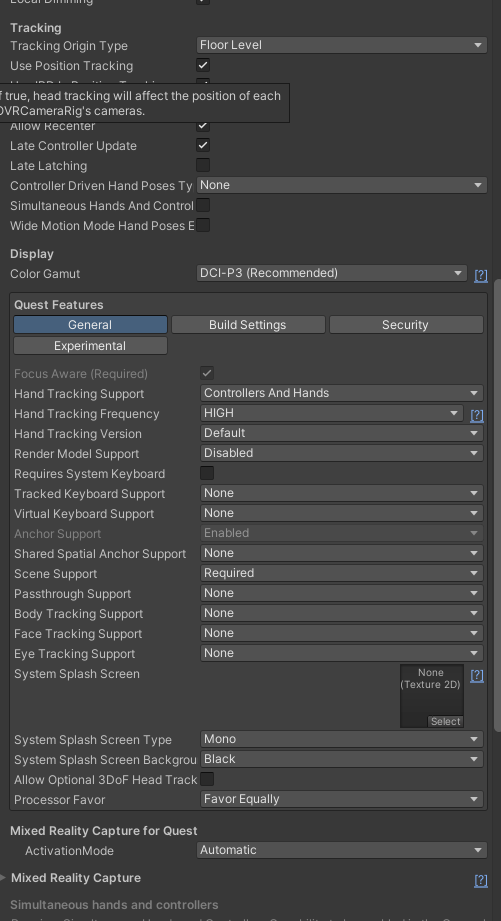
这是unity版本的SDK。其中oculus在facebook改名后变为了meta，也就是本次我所使用的metaXR SDK。也正由于官方文档较为复杂，本次所有结论和推测均为根据SDK带有的Sample包中的物体进行学习而得，故不一定完全正确。接下来我将分部分介绍作业实现的过程和运用到的相关功能，并在最后附上自行实现的一些额外效果。另外，环境安装和调整部分将不进行展开。

## VR显示和信息获取

在Meta包中，将VR显示的摄像机、手部、手柄等输入输出部件整合为了名为OVRCameraRig的物体。其结构如下：



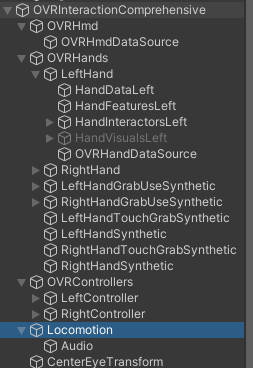
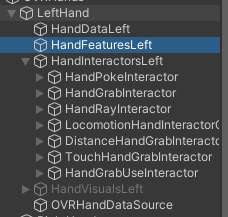
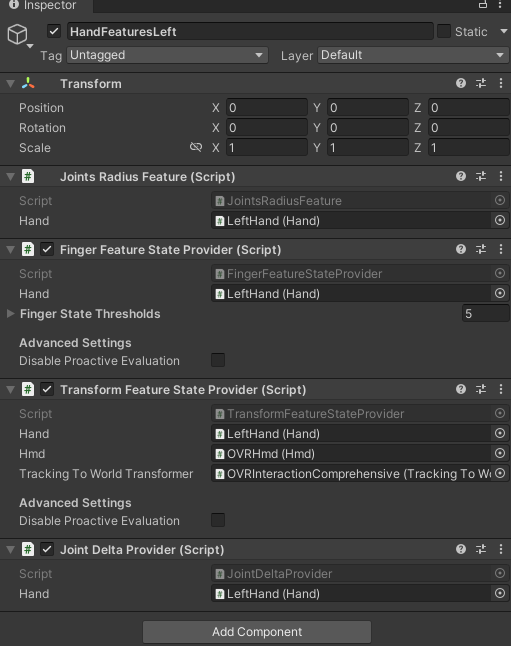
其中CameraRig涵盖了各种设置，包括摄像机设置、画面设置、后期设置、追踪精度等设置。



其下分为TrackingSpace和OVRInteraction两个子物体。顾名思义，两个物体分别负责追踪和互动功能。

追踪中有不同眼睛的摄像机（若在设置后会两个眼睛使用不同摄像机），以及不同手部的锚点位置。摄像机可以设置不同的遮罩或后处理效果等来获得不同的双眼显示效果，并且这需要在设置中将渲染管线设为多管线。这样会对每个摄像机进行一次渲染，使用更大的开销。若设置为单管线则只会渲染一遍，但是需要在着色器中使用unity的宏函数来输出stereoPosition供vr渲染。

互动部分则定义了不同工件的互动功能。包括头盔，手部，控制器和位移。比如后面需要在某些部分使用某个互动的效果，手柄的控制器或者手部的抓取点击等，就需要引用这里的部件。由于本次作业以手部互动为主，所以对其他部分没做过多调整。例如在手部里，就有数据、特征、互动等信息。

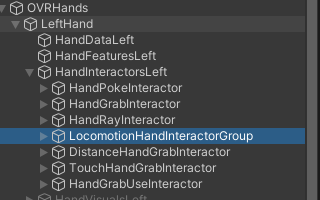
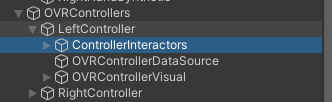
  

若只有最简单功能的话，这个CameraRig的架构会简单很多。但假如当我需要手能抓东西，能戳东西，就需要将对应功能的Interactor传入HandInteractors里。假如当我需要知道手势的动作，甚至精细到指关节的时候，就需要在手部特征里加上额外获取关节和变形等的脚本。

也就是说，这个OVRCameraRig就相当于编程语言里的一个大的库函数，除了直接实现了底层的基本架构之外，还#include了各种接口和实现，供具体功能部分使用。因此我将在接下来的部分对其中一些功能的应用进行展开。

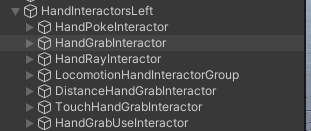
## 移动、基本互动与应用

移动可以根据手柄的上摇杆或者手势实现。摇杆是通过ControllerInteractors获取得到射线和输入，手势则通过HandInteractors中的手势组获取输入指令。

## 

接下来的部分将以手势追踪为主。正如之前所说，HandsInteractors会将所有需要用到的Interactor组件Include进来，供在需要的时候使用。下图是一个比较全的HandsInteractors的组成：

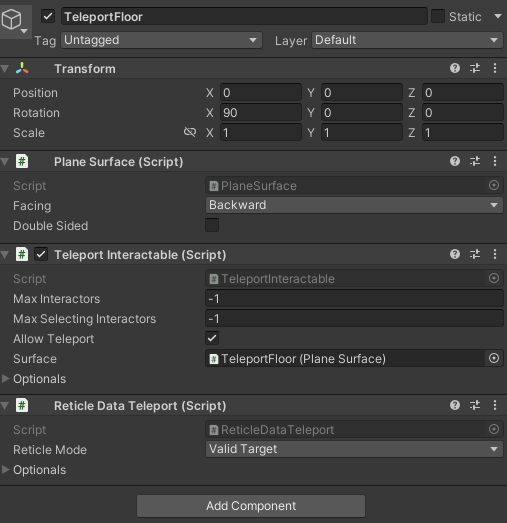
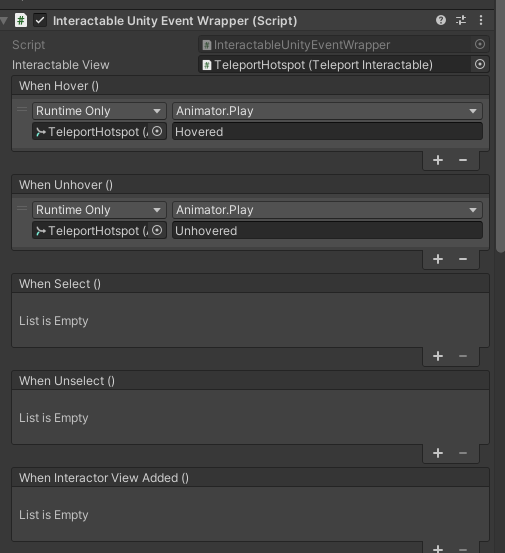


其中依次为：戳，抓取，射线，移动，远距离抓取，触碰抓取(?)和抓取使用。这些功能可以在不同的地方使用。例如戳可以戳ui面板和按钮，抓取可以抓取物体，射线可以操控远程面板和互动，移动则是控制移动和转向，远距离抓取可以进行远程互动，抓取使用则可以用于使用特定物体。

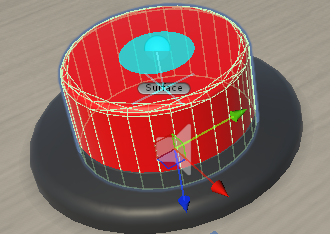
首先是移动部分。手势被LocomotionHandInteractor解释为移动或转向指令，再通过Locomotion部分获取摄像机位置和头部位置之后，就能获得传送移动的效果，并且发射出一条抛物线来标记落点。落点只会落在地面水平面上。

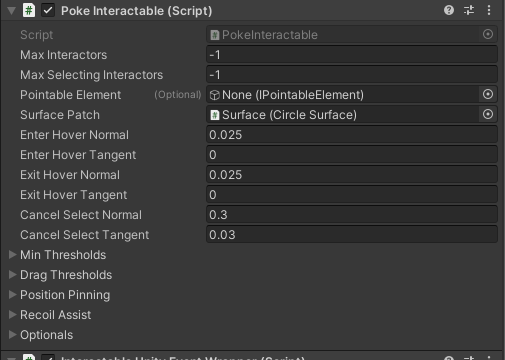
但除了得到落点之外，还需要额外定义可传送的区域，并且有需要的话还可以添加几个固定设置的传送热点。这部分通过LocomotionHotspots实现。TeleportFloor决定了各种可传送区域的表面和限制条件，TeleportHotspot则定义了热点，包括位置和互动时的事件逻辑和动画效果。具体实现的话，大概是通过碰撞来实现的落点检测。通过将传送操作时抛物线与地面的表面的落点作为碰撞点，便可以拿来和不同的限制条件或者热点的碰撞体作检测了。



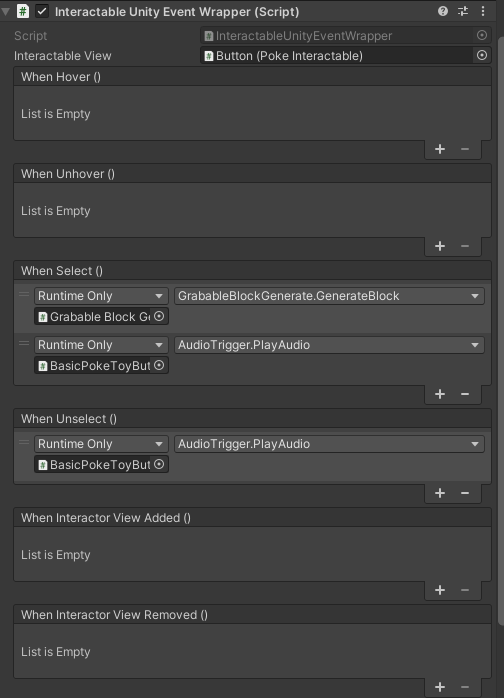
 

再是戳击和抓取部分。这两部分实现大致类似，但也略微有些不同。若是需要某个物体能对这两个操作产生反应，例如按钮被戳动和方块被抓取，则需要在对应物体上附加相应功能的脚本。先从戳开始。戳需要在物体上使用PokeInteractable脚本，并且需要定义一个Surface表面来决定被允许的戳的方向和路径。

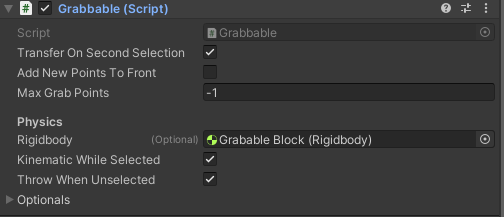


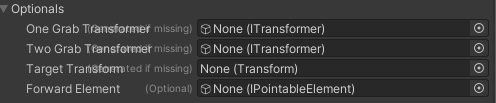


在PokeInteractable里选择表面后，就可以详细定义具体的悬浮、选择和取消的阈值。悬浮指的是离物体还有一定距离的时候，选择就是戳下去的时候。之后，还可以使用事件来定义不同情况下的详细的逻辑，例如函数执行和声音播放。

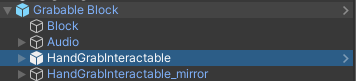


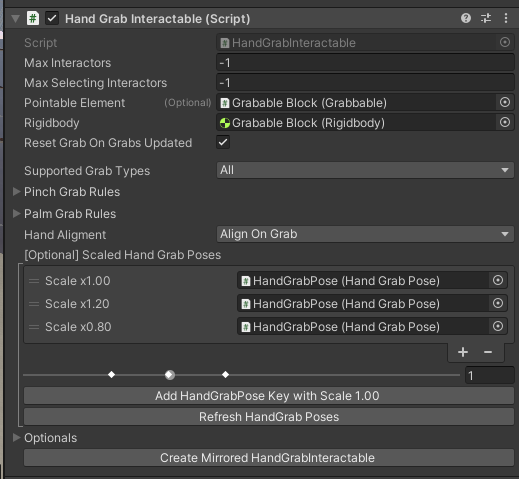
而Grab和Poke所在的不同之处在于，Grab还需要定义更多的信息。首先，可以被抓取的物体需要一个Grabble来标记它是可抓取的。这样之后就会根据这个物体的刚体来通过碰撞检测什么时候可以抓取。Grabable里还可以设置其他物理参数，例如原本是静止的还是运动的，以及进一步抓取时触发的事件逻辑。

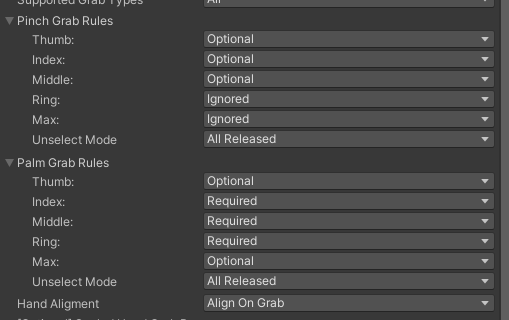




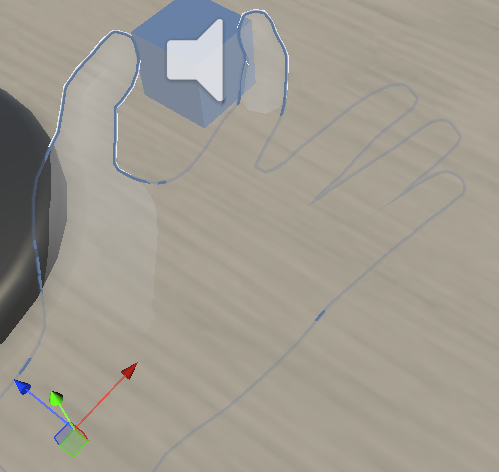
另外，抓取物还需要定义抓取时的手。这需要另外两个子物体，每个子物体上也还需要HandGrabInteractable组件来定义其他需要的信息，包括允许的抓取条件，抓取时的动画显示等。







通过在Pinch和Palm中对不同手指进行设置，来决定抓取或者释放这个物体需要的条件和后续的动画效果。例如小方块就可以只用大拇指和食指夹住，但大的东西就需要整只手握住。像图中就是小方块的定义，可以用大拇指、食指或中指来夹住，但要是想握住就必须需要中间三根指头。并且还需要调节手的Transform，来保证抓取时手的位置和物体的位置协调。

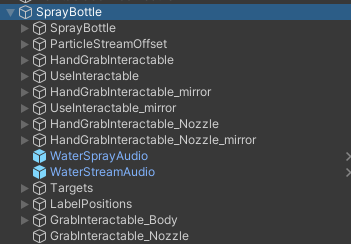


另外，可以看到上述三个操作中Interactable脚本的第一行都是锁定的，并且指向的就是HandInteractors中的对应组件。

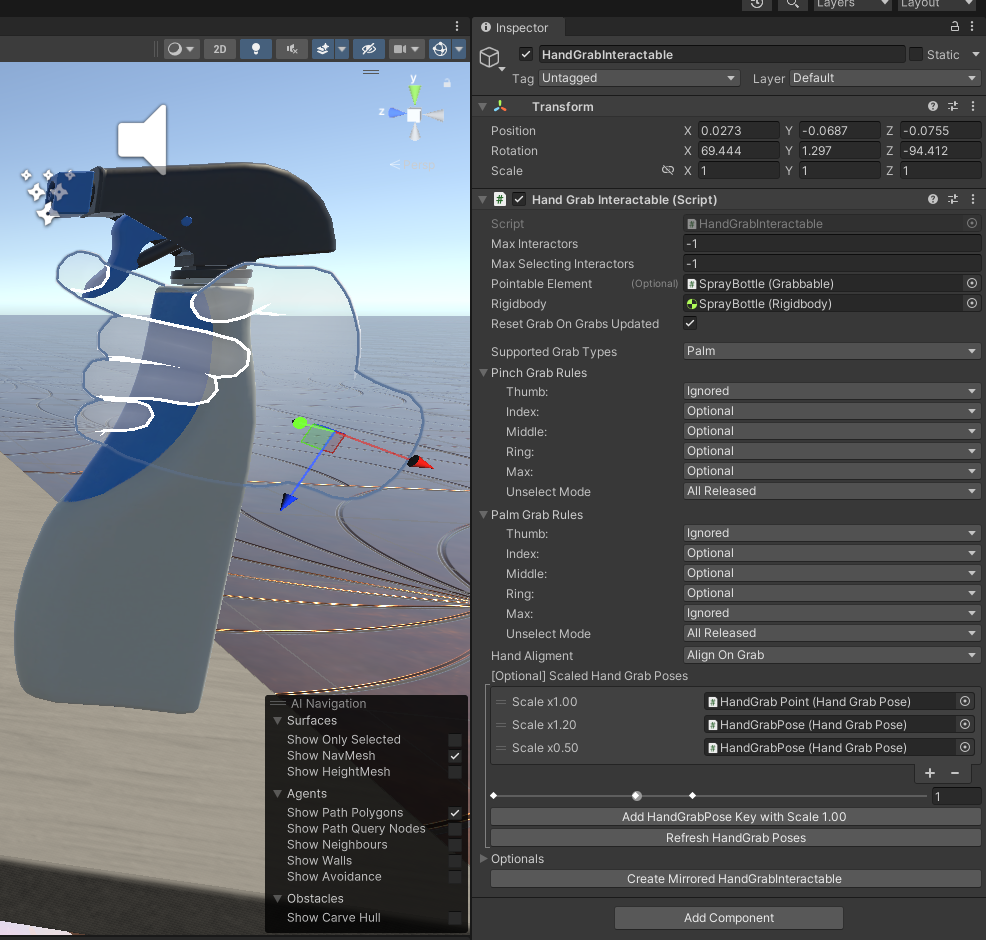


## 精确互动与应用

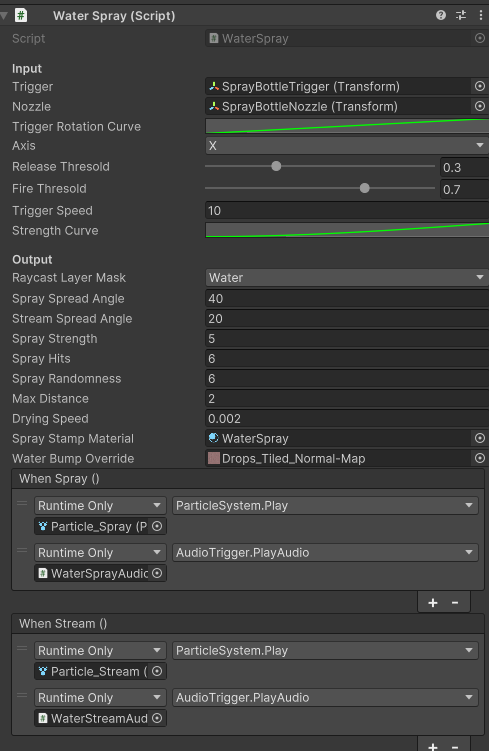
除开基本的移动、戳和抓取之外，还可以详细定义对某个物体进行使用时需要的手部条件和效果。下以项目中使用自Meta提供的样例的洒水器为例。



该物体包含了多个子物体，有粒子系统，网格体，音频系统和互动逻辑。下面仅对互动逻辑进行展开。首先是和上一部分提到的一样的GrabInteractable组件。它定义抓取这个喷水器时的手部需求和效果。



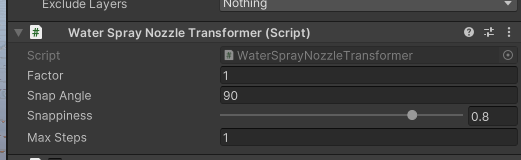
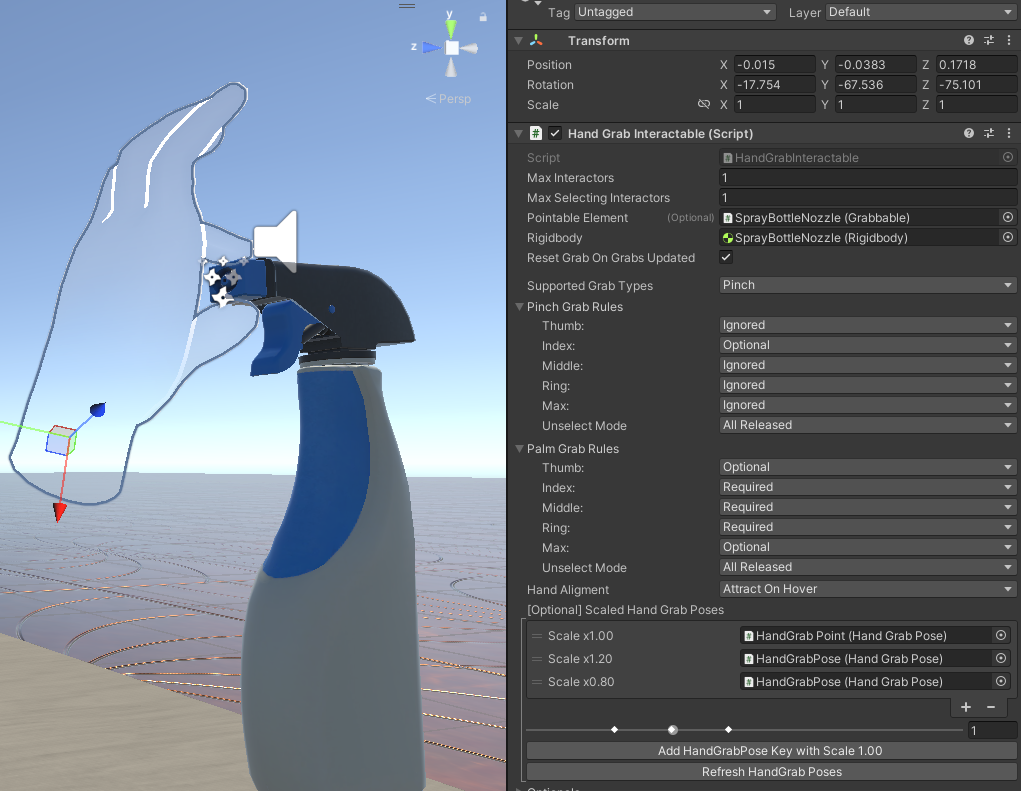
初次之外，它还在Pointable Element项设置了脚本。在指向的目标脚本中，定义了详细的使用逻辑，如下图所示。



它定义了Trigger相关的输入和输出逻辑。在输入中，包括了扳机模型旋转的角度速度，Fire和Release的阈值，力度需求的曲线等。由于脚本的关键代码未列出，推测这是通过抓取后对特定手指或手势的动作进行转换后得到的动作逻辑。这里的识别关键应该是Axis项的X，它在代码里指向一个SnapAxis类的X轴偏移。



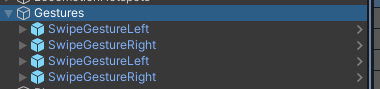
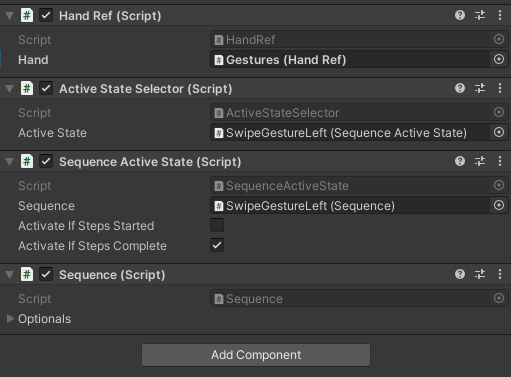
与之类似的还有同一模型的nozzle功能。



这里是通过角度Snap Angle而不是位移来决定动作效果。

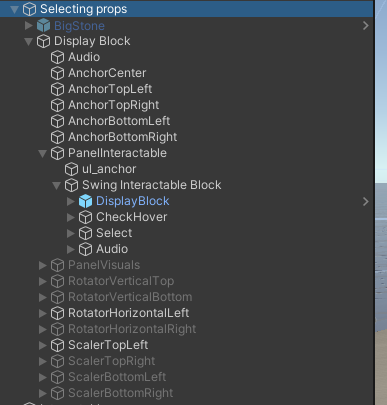
## 手势互动与应用

除去基本的操作之外，meta sdk还提供了手势识别的效果。这需要一个Gestures物体。

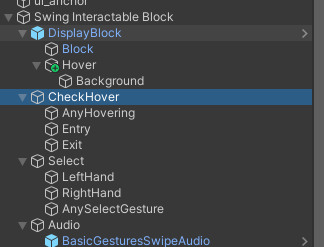
它定义了识别手势需要的数据等信息，这些信息则是通过之前在HandFeatures中额外添加的脚本获取的。

通过手势识别可以实现各种不同的效果。在我的项目里，我使用swipe的手势来切换方块。这其中的互动效果比较复杂，物体关系如下：

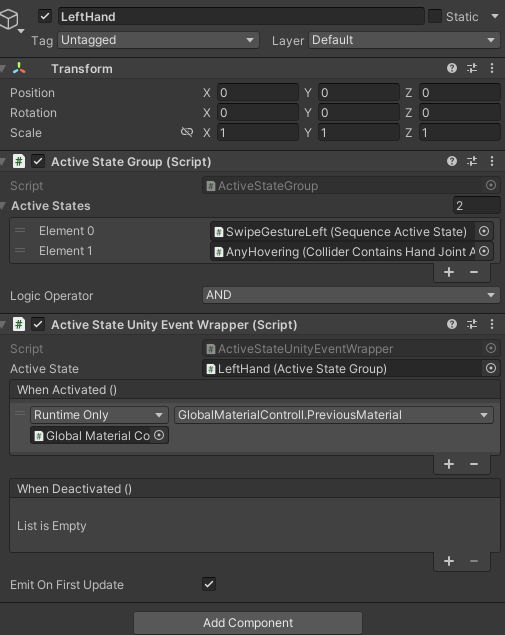


最外层的selecting props通过handref脚本绑定了OVRHands里需要的手。

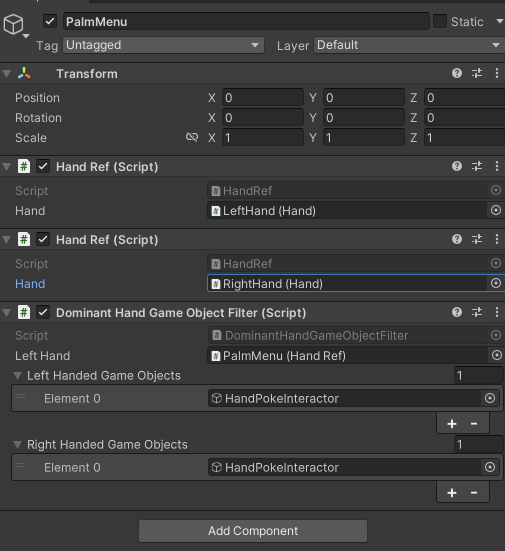
然后其中的第一个DisplayBlock是将Meta SDK另一个3D操作的样例包装后得到的外层，它主要由一系列定义操作空间的锚点，以及一个显示控制ui的操作平面组成。它的功能仅仅为控制Swing Interactable Block这一物体的空间旋转和尺度。而这个物体分为四个部分。



第一部分为显示部分，包括方块的网格体和悬浮选择时显示的白色圆环。第二部分则是悬浮检测。它定义了支持互动的进入和离开范围，以及涉及到的手部或控制器。第三部分就是选择部分的逻辑，定义不同手部时需要使用的手势和触发的事件。最后部分就只是音频部分。



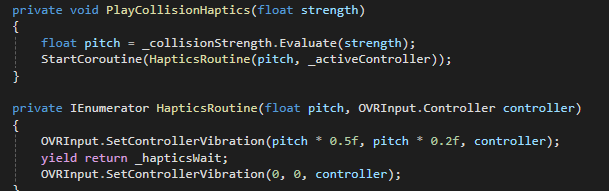
这只是一个使用手势的简单例子。只要在Gestures中添加需要的所有手势并在手部追踪中保证获取到需要的特征，就可以在任意情况下调用手势作为输入并自定义事件效果。例如项目中的源自meta样例的手掌菜单就是通过HandRef的数据来使得张手手势的情况下大拇指和食指相碰来显示菜单。



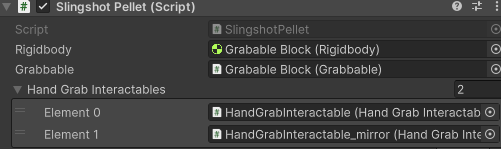
## 弹弓和球拍等的其他实现

这里的弹弓和球拍也同样是源自meta sdk的内容。球拍除了grabable部分外基本只是一个简单的碰撞体，只是额外通过一个脚本来播放声音和控制手柄的震动。

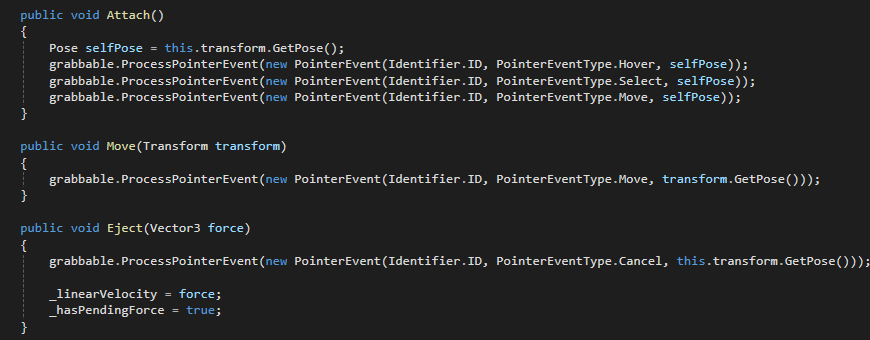




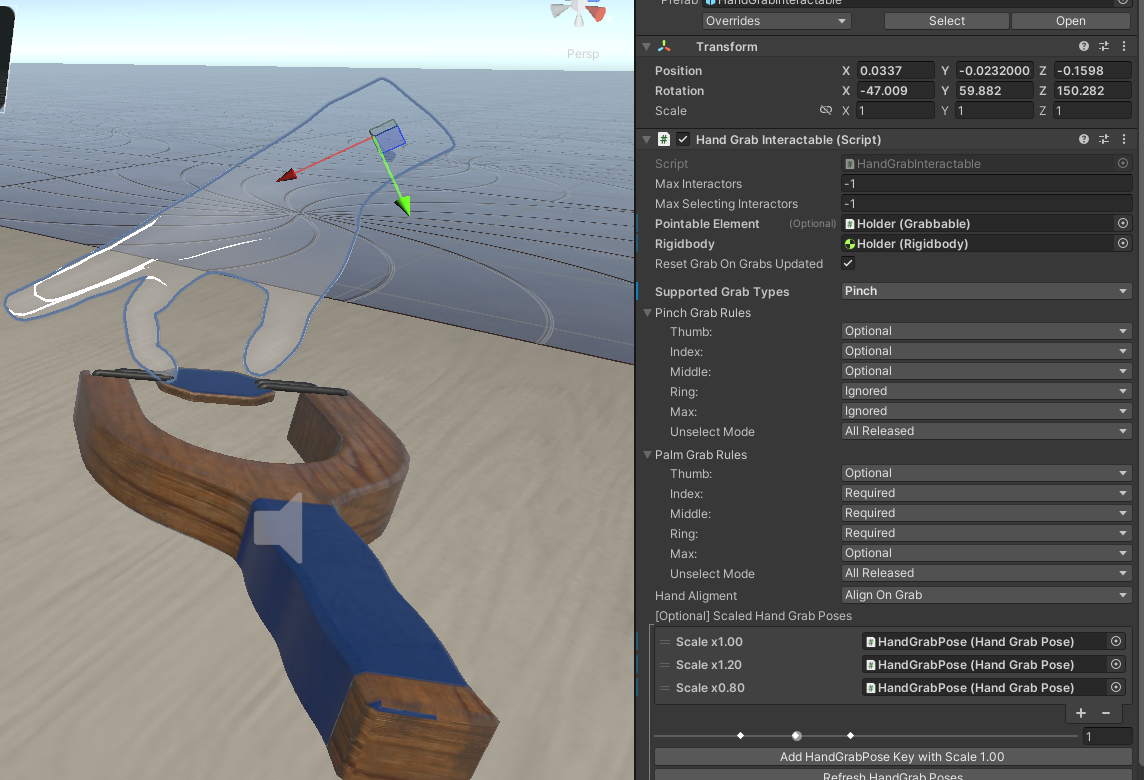
而弹弓部分需要用作子弹的脚本：



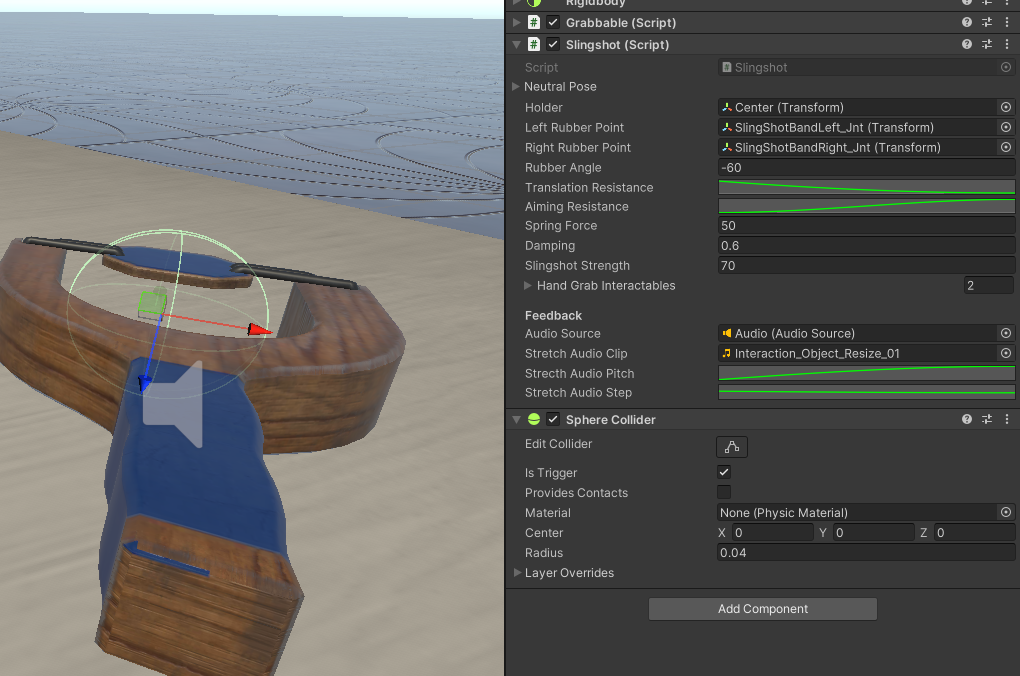
这个脚本控制了子弹在附着到弹弓网上以及弹射出的逻辑。



以及弹弓部分也需要特殊的HandGrabInteractable来处理抓取时的互动逻辑。



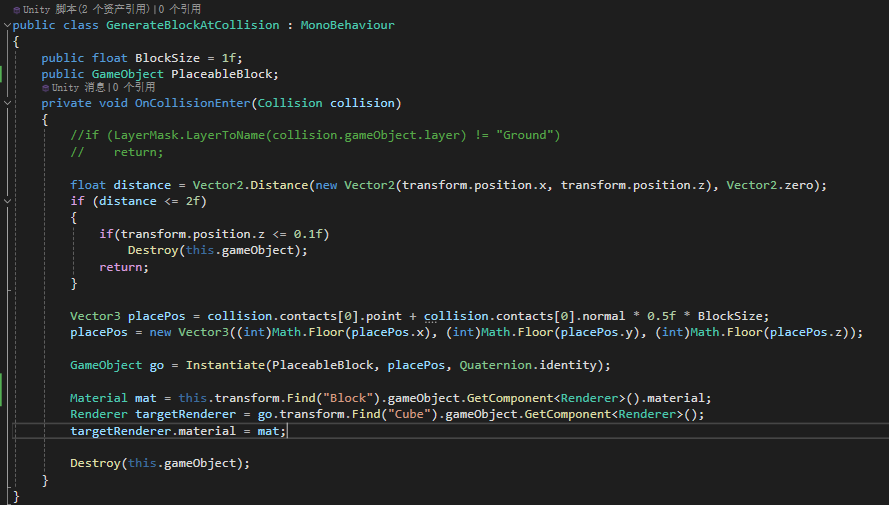
它指向的物体为Holder，也就是弹弓网：



这里详细定义了橡皮筋的锚点、角度和弹性系数、阻尼系数等参数，并且还有一个碰撞体。因此当子弹物体进入到碰撞体后，就视作为装填弹药，开始进行Holder的抓取逻辑。当手势识别判断为松开后，Holder就根据系数来让子弹自己的Eject函数计算发射时的参数。

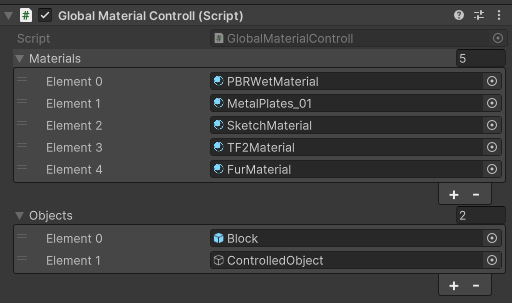
## 自定义逻辑与着色器实现

这整个项目的基础是第四次作业自定义着色器时实现的简易minecraft类方块放置项目。因此在这里，我将这个项目改造为了适合VR的玩法：通过将方块弹射落地来部署方块。相关的方块生成逻辑和之前的方块防止逻辑类似，都是采用了坐标取整的方式来实现全局方块的对齐。



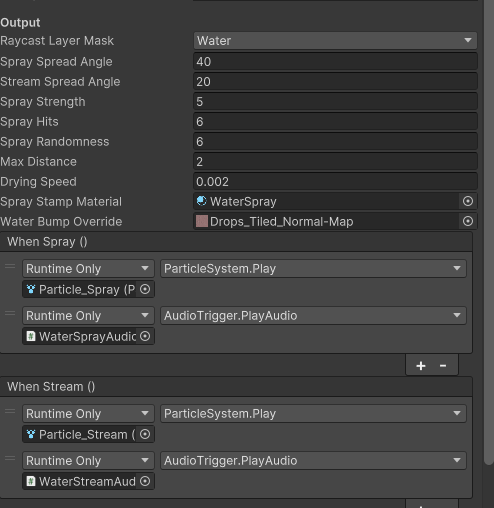
再将各个按钮的Poke动作事件添加上方块生成的函数后，就可以实现生成方块、发射方块、放置方块的完整玩法了。

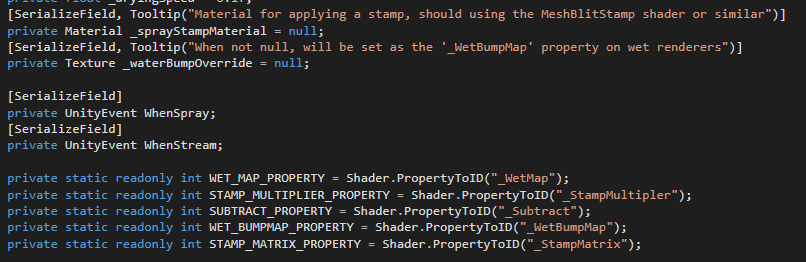
材质方面则是使用了一个全局的材质管理器来控制菜单显示方块的材质。并且这个方块的材质决定了生成方块和其进一步放置下的方块的材质。



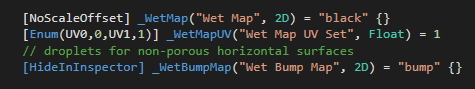
这里的材质基本也基本都是在自定义着色器部分中实现的各种材质的基础上进行了改造实现的。改造的目的是为了在使用喷水器器时材质能够在喷到水的部位体现出不同的视觉效果，比如表面被打湿并产生反光等。这个实现分为两个部分。一个是喷水器喷水时的逻辑实现，另一种是物体着色器被喷到水时的不同显示。

喷水器的脚本中的Output部分就是喷水实现的参数。它的逻辑是检测与Water层对象的射线碰撞情况，并根据各个参数包括力度距离等来得到一个效果系数，通过这个效果系数对目标体着色器中的\_WetMap和\_WetBumpMap进行修改，来向目标的着色器提供不同的输入系数。



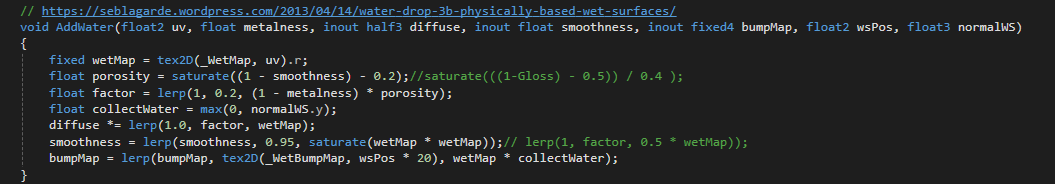
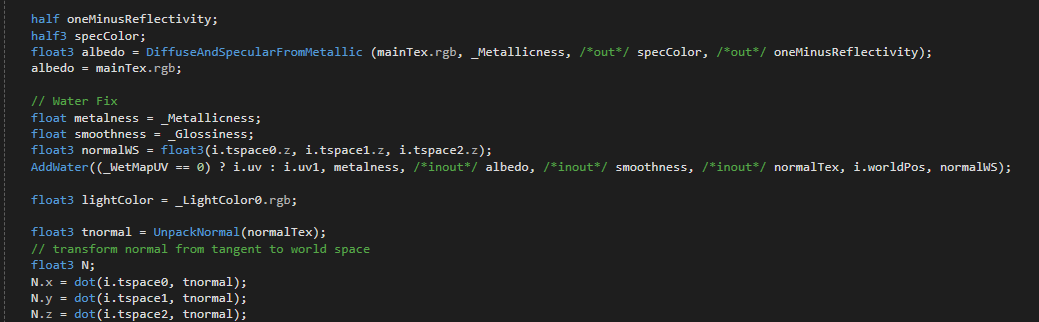


而在目标着色器中，需要在Properties中额外添加这几个贴图。





并且在fragdata和vertdata中也需要额外的uv坐标和texcoord。而在颜色计算的过程中，在获取albedo值和计算法线之前，需要根据这两个贴图来进行额外的处理。



这个addwater函数需要当前计算得到的albedo值，材料的光滑度和当前的法线与坐标值。利用这些参数和wetmap中的贴图值也就是湿度，就可以得到一个材料和湿度结合的系数。再将它和湿度通过lerp来修改当前位置的颜色、光滑度和法线，来得到打湿后的材质。

经过这样，这种拥有WetMap和WetBumpMap着色器的材质就可以正确和喷水器进行互动并产生打湿的效果。

**玩法说明**

使用quest2进行。大部分只需要用手部抓取和戳物体即可。

除外，注视左手手掌并摊开手，在出现菜单按钮后快速拢合大拇指和食指（Snap）以调用菜单，再次操作关闭。

手部平放掌心向上后，除大拇指食指伸直其他三指卷至掌心便是位移操作，食指指向目标为落点，Snap来传送。若需转向保持手势不变靠近自身，在出现转向提示后再Snap即可转向。