接下来我们将认识另外一个非常重要的插件-VRTK，同时在场景中实现更多的功能。

打开上一节内容所用到的项目，在它的基础上进一步完善。

首先切换到Asset Store,搜索VRTK

然后打开第一个VRTK-SteamVR Unity Toolkit，下载并导入到项目中。

当然，对于这个插件，更推荐直接在Github上获取。

<https://github.com/thestonefox/VRTK>

只要你想得到的HTC Vive交互，这个插件里面都有具体的示例，和可以直接使用的各种脚本。而且在github上的更新极其活跃。

如果不习惯命令行，那么推荐大家安装SourceTree（不推荐用Github官方的客户端，坑太多，经常崩溃），链接如下：

<https://www.sourcetreeapp.com/>

Sourcetree是一个非常强大且优雅友好的Git管理工具。

关于Sourcetree的具体使用这里就不多说了，会用一篇单独的文章来讲。在github的页面中找到VRTK的相关地址，然后复制git的链接。

在Sourcetree中点击“克隆/新建”，然后在科隆仓库选项卡的源路径处粘贴这个链接，点击克隆，就可以获取到插件了。

如果是用这种方式获取的插件，那么只需要把相关的文件夹拖到项目的Project下的Assets里面就行。

需要特别强调的是，要使用VRTK插件，必须先把SteamVR的官方插件导入到项目中，否则会报错。

在项目中找到VRTK文件夹，会发现其中提供了大量的示例，Prefab和脚本。

其中Examples目录下提供了多达39个（这个数字还会不断增加）的示例场景。每个场景都简洁但又十分清晰的展示了如何使用HTC Vive进行互动。

限于篇幅，这里不一一赘述。但是强烈童鞋们把每个示例场景都亲自体验下，然后自己来复现一下场景，或根据自己的需要将不同的交互融合在一起。

当然，作为第三方的插件，VRTK并非完美。最容易遇到的坑就是在多个交互同时进行时，可能会出现某个或某几个交互失效的情况。不过好在Scripts中有所有的代码，有经验的童鞋可以根据自己的需要来更改其中的代码，也可以为这个开源项目提供自己的力量。

接下来我们利用VRTK插件在场景中实现最常用的漫游功能。

选中CameraRig，然后在Inspector中点击Add Component，搜索VRTK\_BasicTeleport，添加该脚本。

这里面有若干个参数设置，其中Blink Transition Speed是漫游传送时的闪动过渡，其它几个参数里面，比较重要是Ignore Target With Tag or Class，通过设置该参数可以在传送时不至于跳到水里面或者传送到屋顶。Nav Mesh Limit Distance这个参数可以用来设置传送的最大距离，默认情况下是0，也就是不受限制。

当然，光有这个是不够的，如果现在点运行，基本上没啥反应。

接下来我们要在两个手柄上分别添加事件响应的脚本。

点开CameraRig的下三角，选中Controller(left)，点击Add Component，搜索vrtk\_，添加一个VRTK \_ControllerEvents脚本。

这一点非常重要，因为手柄上的所有交互都必须依赖于这个脚本。

在VRTK\_ControllerEvents脚本中有好几个参数设置，这里我们暂且使用默认的。

然后继续添加组件，这一次我们可以添加一个VRTK\_SimplePointer脚本。它的作用是产生一道类似激光的指示线。里面的参数设置我们先保持默认。

为了方便测试，我们在Controller(right)上面同样添加这两个脚本。

此时点击运行，按下手柄上的圆盘区域，就会看到手柄的顶端发射出一道激光一样的射线，而且射线的终点在哪里，我们所在的区域就会传送到哪里。

当然，这种设计对于VR的新用户来说极其不友好。我们需要做一个小小的调整。

在Hierarchy中选中Controller(left)，在VRTK\_SimplePointer的脚本设置中找到Pointer Visibility这个参数，将其从默认的On\_When\_Active修改为Always\_On。把Controller(right)上面加挂的脚本做同样的参数修改。

再次运行，在默认情况下也能看到手柄顶端的激光。

好了，这部分的学习暂且到这里。

在下一节的内容中，我们将会对HTC Vive当前所使用的交互系统及其原理做一个详细的介绍。这样在学习更多的交互事件时，会有更加深刻的了解。

今天的福利是。。。