第 1 章　从外部控制 UE5 界面

起初，与 UE 远程通信的想法只是某些 UE 设计人员在工作时，为了实现能在几个不同代码版本间快速地来回切换而想到的点子，于是他们开始在 UE 中引入插件，通过使用插件的方式让 UE 可以接收基于 HTTP、DMX、OSC、UDP 等协议的数据。慢慢地，人们发现，这种做法不仅可以给开发、比对设计等工作需求带来一定的便利，还可以成为终端用户与虚拟数字内容进行交互的途径。例如，UE5 作品完成设计后投放在大屏幕上观看，此时如果觉得屏幕上内容色调有点灰暗或者场景中的某些物件位置需要挪动少许，以前工作人员都需要从大屏幕前跑到计算机前完成修改调整的操作，再返回到大屏幕前再次观看调整的结果。如果能利用手机或平板直接遥控 UE5 的界面操作，工作效率会显著提高。如果可以利用 UE5 所在计算机之外的其他设备与 UE5 互动，将可以实现很多有趣味的创意互动。

要想实现从外部控制 UE5，就需要外部的软硬件能与 UE5 进行通信，这就要使用相关的通信协议。不同类型的软硬件所使用的通信协议各有不同，如灯光器材设备通常使用 DMX 协议，而音乐器材设备则多使用 MIDI 协议。掌握了如何在 UE5 中使用各类不同的数据通信协议，会让今后的互动创意之路变得更加广阔而自由。

用手机来遥控 UE5 的场景看起来最为常见和易用。本章首先从这个场景入手，为读者介绍 SocketIOClient 这款插件的具体使用方法。利用这个插件可以让手机通过网页与 UE5 实时通信，部署起来相对容易，而且不仅可以实现一对一的通信，也能实现一对多的通信。而 Remote Control API 则是为了方便用户遥控 UE5 而特意提供的一套 API 接口，利用这些接口可以很容易地访问 UE5 中各类对象的属性信息并通知它们执行各类动作指令。只不过调用 Remote Control API 需要用户自己编写网页程序并调试代码，而且还需要搭建 Web 服务器，技术门槛会略高一些。

除了利用 Web 页面来遥控 UE5，还可以使用一些专业软件来远程控制 UE5。例如，控制灯光器材的控台软件 grandMA、dot2 onPC 等都可以基于 DMX 通信协议，对 UE5 中的虚拟灯光系统进行非常细密的远程控制。DMX 通信协议支持的数据量更大、数据变化更灵活，所以对 DMX 通信协议的学习可以进一步拓宽读者对远程控制的认识。DMX 不仅可以用于灯光秀的操作，也可以用于控制 UE5 中的其他互动流程。TouchOSC 软件是基于 OSC 通信协议进而与 UE5 通信的，它能够提供非常易于定制的操作界面供用户使用，在触摸屏设备上可以非常丝滑地使用 TouchOSC 进行对 UE5 的控制，使用 TouchOSC 可以大大提高设计控制界面的效率。而 REAPER 软件是通过 MIDI 通信协议与 UE5 通信的，MIDI 是音乐器材里得到广泛支持的一种通信协议，学习 MIDI 后可以使用各类 MIDI 控制器来操控 UE5，让 UE5 可以与各

类电子乐器充分融合。

掌握了第 1 章的知识，读者就具备了使用既有的专业软件远程控制 UE5 的能力，并且能够自行定制开发用于遥控 UE5 的操作界面了。

本章重点

- 在 UE5 中使用 Socket.IO 实现服务器与客户端之间的通信
- 利用 Remote Control API 发送 HTTP 请求直接访问 UE5 对象
- 专业软件借助不同的通信协议，如 DMX、OSC、MIDI 等与 UE5 通信

1.1 借助 SocketIO 实现用移动端网页控制 UE5

1.1.1 安装SocketIOClient插件并设置WebSocket

WebSocket 是 HTML5 新增的一种通信协议，其特点是服务端可以主动向客户端推送信息，客户端也可以主动向服务端发送信息，是真正的双向平等对话，属于 Web 服务器推送技术的一种。

Socket.IO 是 Web 前端技术人员熟知的实现 WebSocket 功能的一种 JS 框架，它支持及时、双向、基于事件的交流，可以在不同平台、浏览器、设备上工作，性能可靠而且速度稳定。

如果能让 UE5 借助 Socket.IO 的力量，就可以轻松实现利用移动设备远程控制 UE5 的画面内容，这会让 UE5 项目变得更加有趣、更加便捷。在虚拟制片工作现场，当导演看到由 UE5 投射到演员身后大屏上的虚拟背景需要调整时，他只需轻点手机就可以对大屏画面的各个细节任意调整，还可以依据剧情的发展随时遥控大屏里的物体，让它们配合剧情运动起来，这无疑为导演们从事虚拟制片提供了全新的思路。

让 UE5 借助 Socket.IO 还能构建更多有意思的互动场景，如在 UE5 程序界面上单击虚拟物品，随即真实空间里的某些硬件设备就出现了相应的反应和变化，也就是说可以通过操作 UE5 向其他设备发送信号。

最近更新的 SocketIOClient 插件支持 UE5，让开发人员可以很轻松地把基于 Node.js 的 Socket.IO 框架与 UE5 衔接起来，实现实时的数据交换，为 UE5 插上了远程互动的翅膀。下面讲解一下具体的做法，一步一步地让 UE5 基于 SocketIOClient 插件与 Web 页面实现通信。

首先，打开 UE5 程序（本项目采用 UE5.1 版本），建立一个 GAMES 模板下的 Blank（空白）蓝图项目，取名为 SocketIODEMO，如图 1-1 所示。

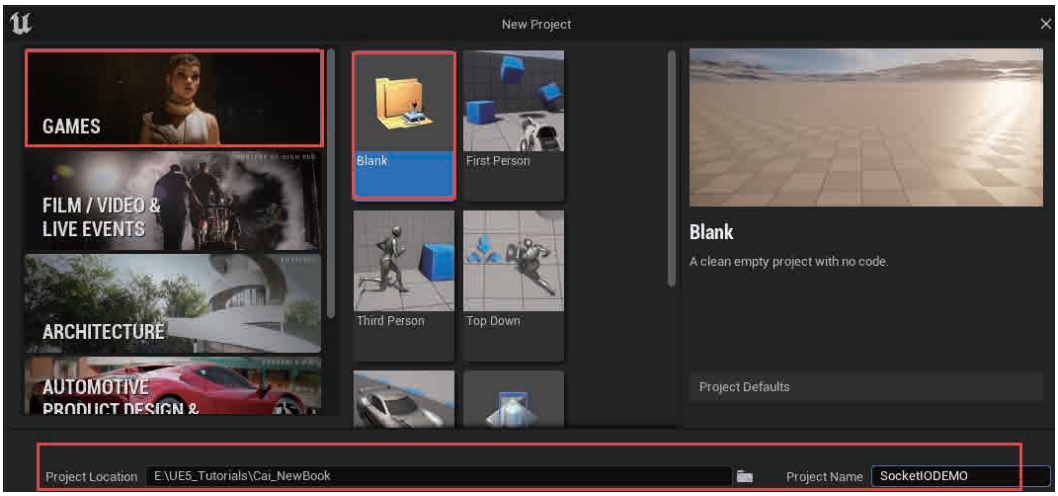

图 1-1　UE5 创建空白新项目

接下来从 Github 网站上找到关于 SocketIOClient 插件的网页，下载插件，如图 1-2 所示。

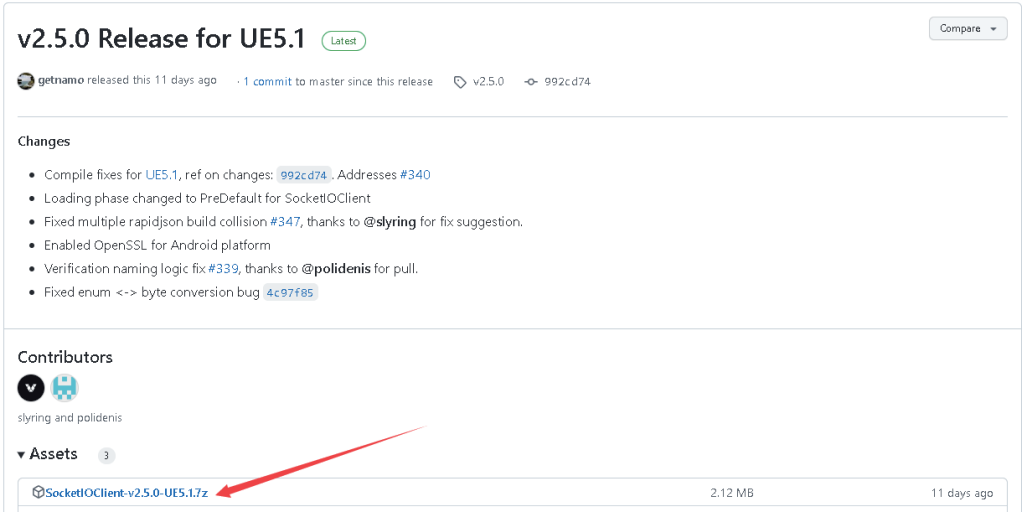

图 1-2　下载 SocketIOClient 插件安装包

这时需要在刚才新建的 UE5 项目文件夹里手动建立一个名为 Plugins 的文件夹，把图 1-2 中这个插件压缩包下载后，解压到 Plugins 文件夹里，结果如图 1-3 所示。

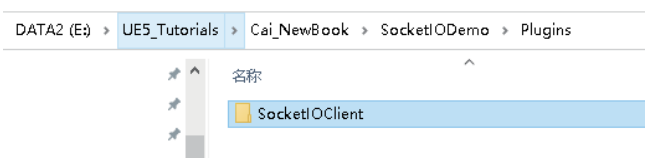

图 1-3　SocketIOClient 插件解压到 Plugins 路径

在 UE5 的主菜单栏依次选择 Edit → Plugins 命令，会看到如图 1-4 所示的界面，找到名为 Socket.IO Client 的插件，在前面的复选框中打勾。

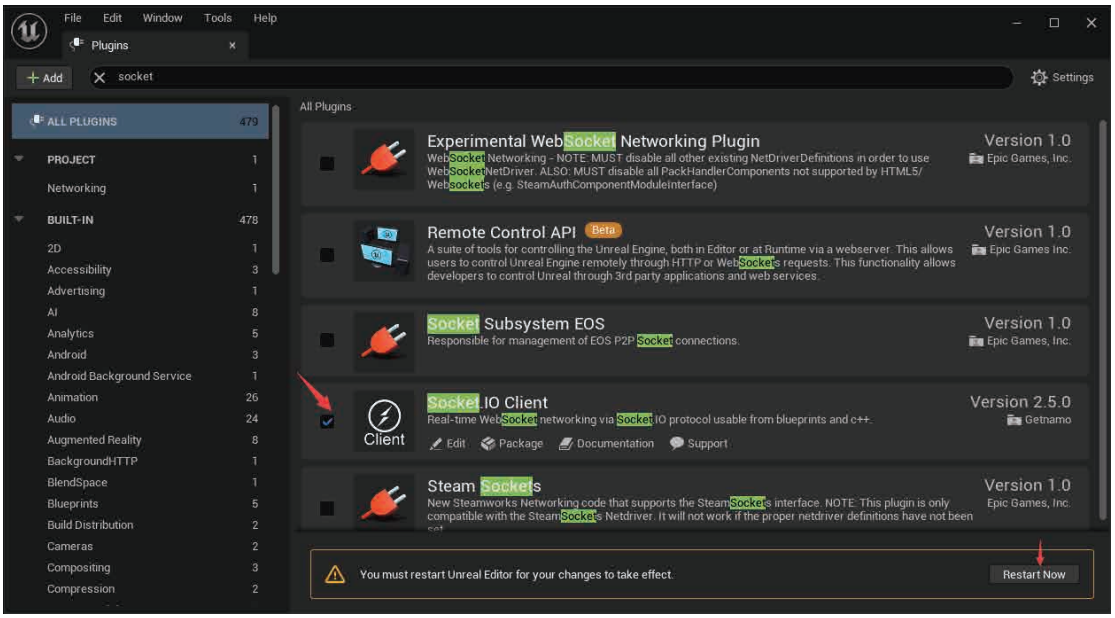

图 1-4　UE5 启用 SocketIOClient 插件

然后单击 Restart Now 按钮，重启 UE5 项目。接下来在 UE5 的 Content Browser 中新建一个 Actor 对象，取名为 SocketActor，双击这个 SocketActor，打开它，在它的 Components（组件面板）里单击“Add”按钮，添加一个 SocketIOClient 对象，如图 1-5 所示。

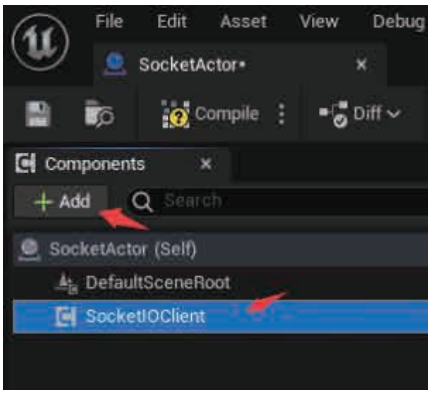

图 1-5　蓝图中加入 SocketIOClient 组件

在这个 SocketIOClient 对象的 Details（细节面板）里，单击 Events 左侧的下拉按钮，在下拉列表里选择 On Connected 事件，单击其右侧的加号按钮，如图 1-6 所示。

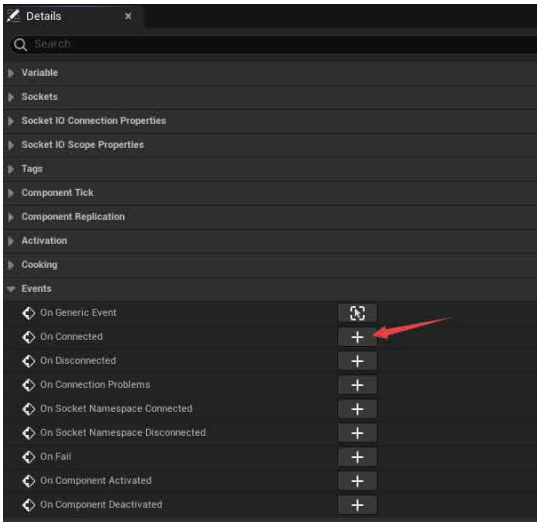

图 1-6　SocketIOClient 组件添加事件

在 Event Graph 里可以简单地添加以下蓝图内容，意思是在 SocketIOClient 对象连接服务器成功后打印输出 connected! 字样，如图 1-7 所示。

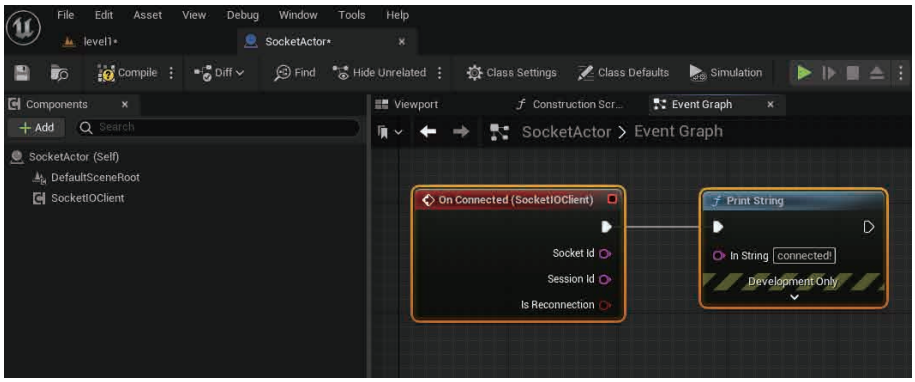

图 1-7　SocketIOClient 组件蓝图打印连接信息

修改蓝图后单击 Compile 按钮，编译并保存蓝图。

接下来需要进一步构建基于 Node.js 的本地 Socket 服务器。Node.js 是一种可以让 JavaScript 运行在服务器端的程序。如果计算机上还没有安装 Node.js，请从 Node.js 官网选择下载与自己计算机系统相匹配的安装包执行安装。笔者所用的计算机为 Windows 10 操作系统，安装完毕 Node.js 后，笔者在计算机的 E:\UE5_Tutorials\Cai_NewBook\SocketIODemo\SocketIO_server 路径下部署 Socket.IO。部署的方法是通过 CMD 命令行进入 SocketIO_server 文件夹，运行 npm install socket.io 指令，安装 Socket.IO 组件，具体操作如图 1-8 所示。接着通过运行 npm install express 指令安装 Express 组件。

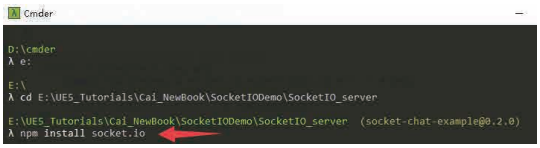

图 1-8　安装 Socket.IO

笔者在 SocketIO_server 文件夹里编写了一个 server.js 文件，可以通过记事本直接编辑这个 JavaScript 脚本文件，其完整的内容如图 1-9 所示。笔者在代码里加入了详细的中文注解，方便读者理解。

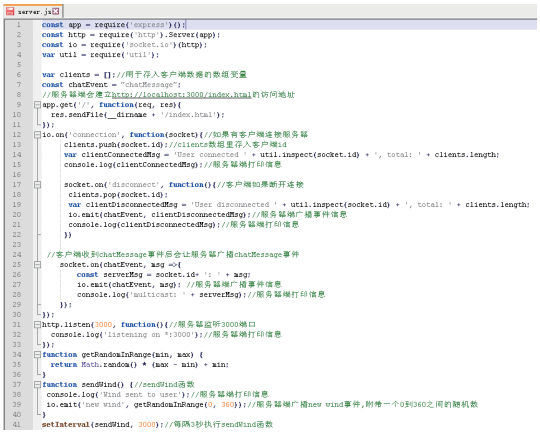

图 1-9　server.js 的全部代码和中文注释

在 CMD 命令行中输入 node server.js 来运行这个服务器端脚本文件，如果看到命令行下方出现了 listening on *:3000 的字样，就表示已经开始运行 server.js 里的逻辑代码了，也就是说 Socket 服务器已经运行起来了，如图 1-10 所示。

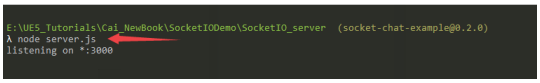

图 1-10　启动 WebSocket 服务器

1.1.2　使用JavaScript与蓝图通信

上面我们已经看到了运行在本地服务器端的 server.js 文件，这个 JS 脚本文件实现了基础的服务器与客户端之间的通信逻辑，如图 1-11 所示。

```
server.js
1  const app = require('express')();
2  const http = require('http').Server(app);
3  const io = require('socket.io')(http);
4  var util = require('util');
5
6  var clients = [];//用于存入客户端数据的数组变量
7  const chatEvent = "chatMessage";
8  //服务器端会建立http://localhost:3000/index.html的访问地址
9  app.get('/', function(req, res){
10   res.sendFile(__dirname + '/index.html');
11 });
12 io.on('connection', function(socket){//如果有客户端连接服务器
13     clients.push(socket.id);//clients数组里存入客户端id
14     var clientConnectedMsg = 'User connected ' + util.inspect(socket.id) +
       ', total: ' + clients.length;
15     console.log(clientConnectedMsg);//服务器端打印信息
16
17     socket.on('disconnect', function(){//客户端如果断开连接
18      clients.pop(socket.id);
19      var clientDisconnectedMsg = 'User disconnected ' + util.inspect(socket.
        id) + ', total: ' + clients.length;
20      io.emit(chatEvent, clientDisconnectedMsg);//服务器端广播事件信息
21      console.log(clientDisconnectedMsg);//服务器端打印信息
22     })
23
```

图 1-11　server.js 局部

从这段脚本代码可以看出，如果服务器发现有用户连接上来，则会在服务器端打印字符信息，信息的内容是：'User connected '+ 客户端 id+', total: '+ 现有客户端数量，接下来利用 UE5 蓝图实际验证一下。先从 Content Browser 里拖曳一个刚创建的 SocketActor，放到视口中来（这里提示一下，本书中所说的视口在其他 UE5 的书籍中还有关卡、舞台、场景等不同称谓，但实际都是同一个含义），如图 1-12 所示。

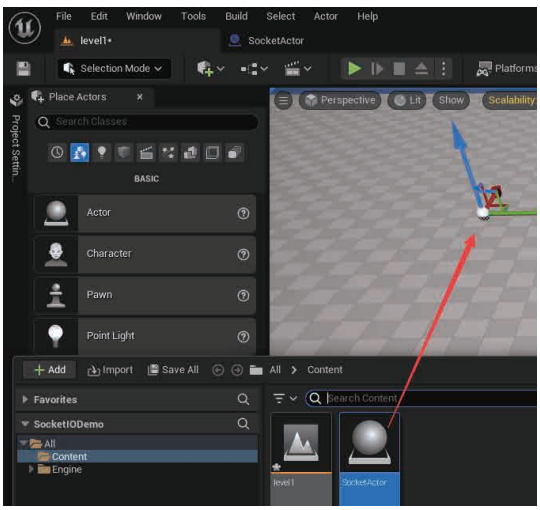

图 1-12　把 SocketActor 对象拖入视口中

运行 UE5，也就是单击绿色的三角形按钮，如图 1-13 所示。

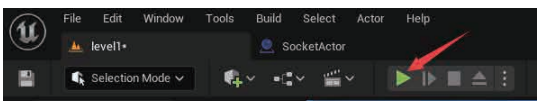

图 1-13　单击绿色三角形按钮运行 UE5

此时会在视口左上角看到一串字符 connected!，这个区域就是 UE5 里输出打印内容（Print String）的地方，如图 1-14 所示。

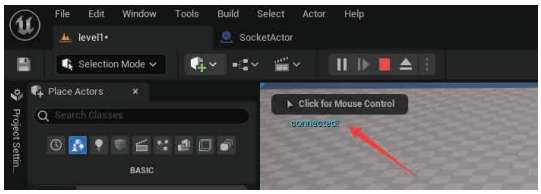

图 1-14　UE5 视口左上角显示打印输出信息

同时，在 CMD 命令行窗口中也可以看到服务器端的反馈，如图 1-15 所示。

```
E:\UE5_Tutorials\Cai_NewBook\SocketIODemo\SocketIO_server  (socket-chat-example@0.2.0)
A node server.js
listening on *:3000
User connected 'DNk7VDo4yQ8YQdBFAAAB', total: 1
```

图 1-15　命令行窗口显示服务器端信息

接下来，可以更进一步利用蓝图读取来自服务器的具体信息。通过分析 server.js 文件的最后一行代码 setInterval(sendWind, 3000) 可以明白，服务器每隔 3 秒会向所有客户端广播一次事件名为 new wind 的事件通知，而且通知里还附带了一个 0~360 的随机数信息。

于是，从组件面板里将 SocketIOClient 组件拖入蓝图编辑区域，使用 Bind Event to Generic Event 节点为 SocketIOClient 组件绑定 new wind 事件，如图 1-16 所示。

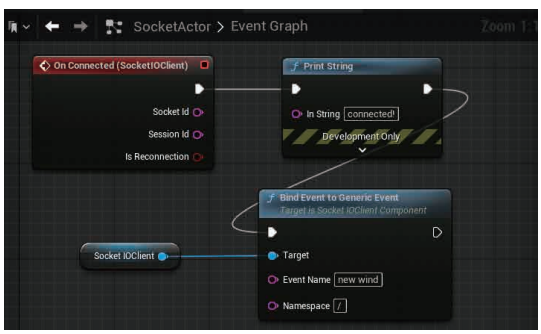

图 1-16　UE5 蓝图绑定服务器事件

然后单击组件 SocketIOClient，在 Details（细节）面板里单击 Events 左侧的下拉按钮，在下拉列表里选择 On Generic Event，

单击其右侧的加号按钮。这时蓝图中会出现 On Generic Event 节点，将这个节点下的 Event Name 节点拖出来，在搜索框中输入 equal，搜索到 Equal Exactly (String)，如图 1-17 所示。

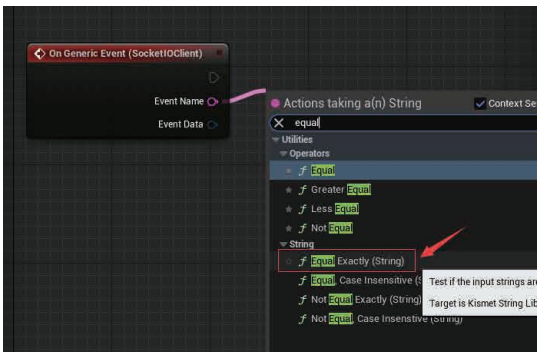

图 1-17　UE5 蓝图判断服务器事件名称

完成的蓝图内容如图 1-18 所示。这样就可以判断服务器发来的事件名称是否是 new wind，如果是，就把 Event Data（事件附带的数据）进行 JSON 编码，并 Print String（打印输出）。

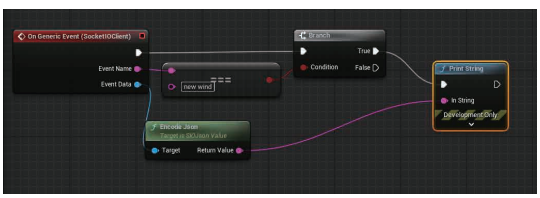

图 1-18　UE5 蓝图打印服务器事件数据

运行 UE5，就可以从视口的左上角区域看到打印输出的随机数了，如 321.742924。

这里也可以使用 Bind Event to Delegate 节点为 SocketIOClient 组件绑定事件，绑定 Event Name（事件名称）为 chatMessage 的事件，在收到 Socket 服务器发来的数据后打印输出事件数据。蓝图内容如图 1-19 所示。

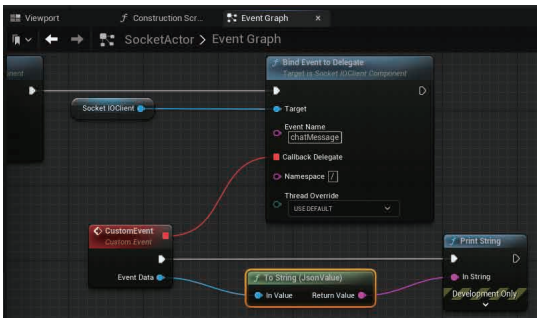

图 1-19　UE5 绑定服务器事件的更多方法

在蓝图区域的空白处右击，从弹窗里搜索 keyboard m，可以找到键盘事件里按 M 键所对应的事件，意思是当用户在 UE5 里按下键盘上的 M 键时所触发的事件，如图 1-20 所示。

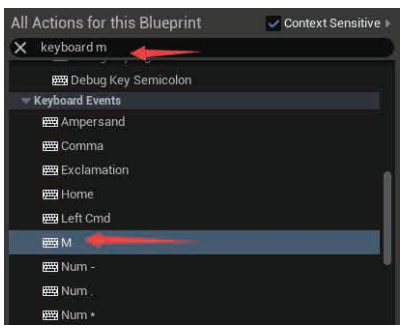

图 1-20　UE5 绑定键盘按键的方法

图 1-21 所呈现的蓝图内容表示在按下 M 键后，会让 SocketIOClient 组件向 Socket 服务器发送名为 chatMessage 的事件，事件里附带的 Message（消息）为 I am ok。

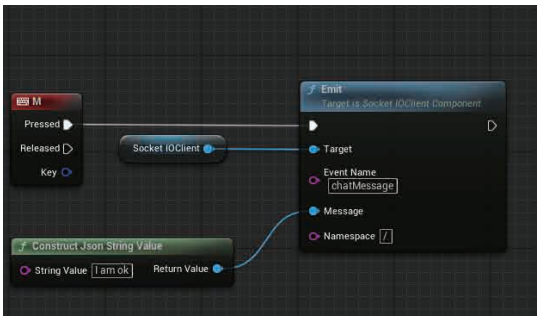

图 1-21　UE5 绑定用户按下 M 键后发送事件到服务器

这里需要注意的是，如果要让 UE5 运

行时可以接收到用户敲击键盘的事件，需要在蓝图里写入如图 1-22 所示的这部分内容，让 Player Controller（玩家控制器）接受用户输入。按下 M 键后 UES 和服务器各自的显示如图 1-23 所示。

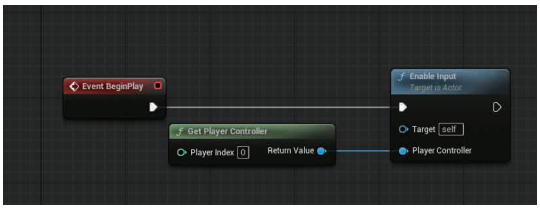

图 1-22　UE5 开启接受键盘输入

提示：运行 UE5 后，需要先单击一下视口区域，确保场景获得用户输入的焦点，然后再按 M 键测试才会有效。

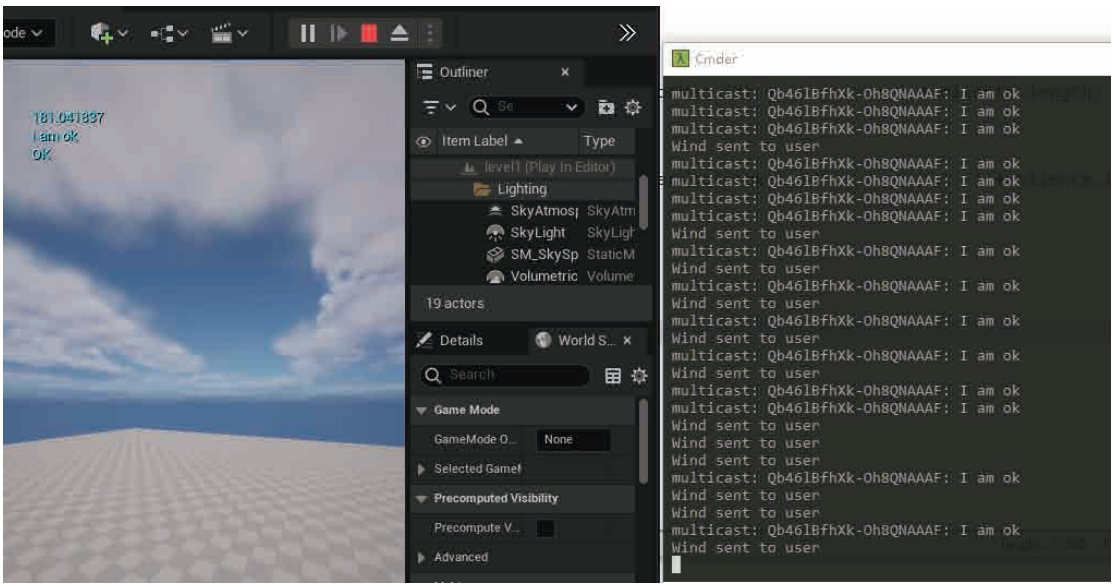

图 1-23　按下 M 键后 UE5 和服务器各自的显示

此时按下 M 键可以向服务器发送一个名为 chatMessage 的事件，并附带字符信息 I am ok。如果要向服务器发送 JSON 格式的数据，可以把蓝图作如图 1-24 所示的修改。

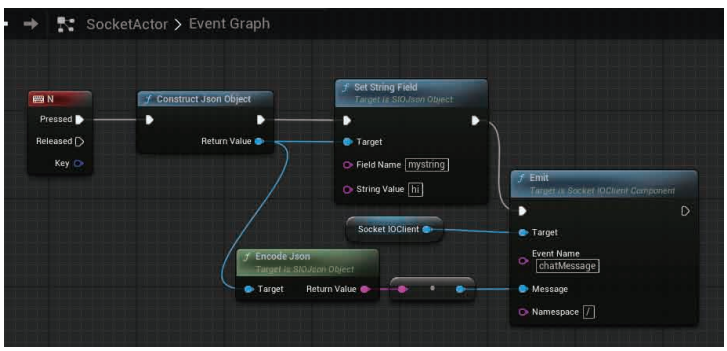

图 1-24　按下 M 键后向服务器发送 JSON 数据

以上蓝图发送了 chatMessage 事件给服务器并附带了一个内容为 {"mystring":"hi"} 的 JSON 格式的数据。按键发送数据后可以观察 UE5 视口的打印输出和 Windows 命令行中的显示，如图 1-25 所示。

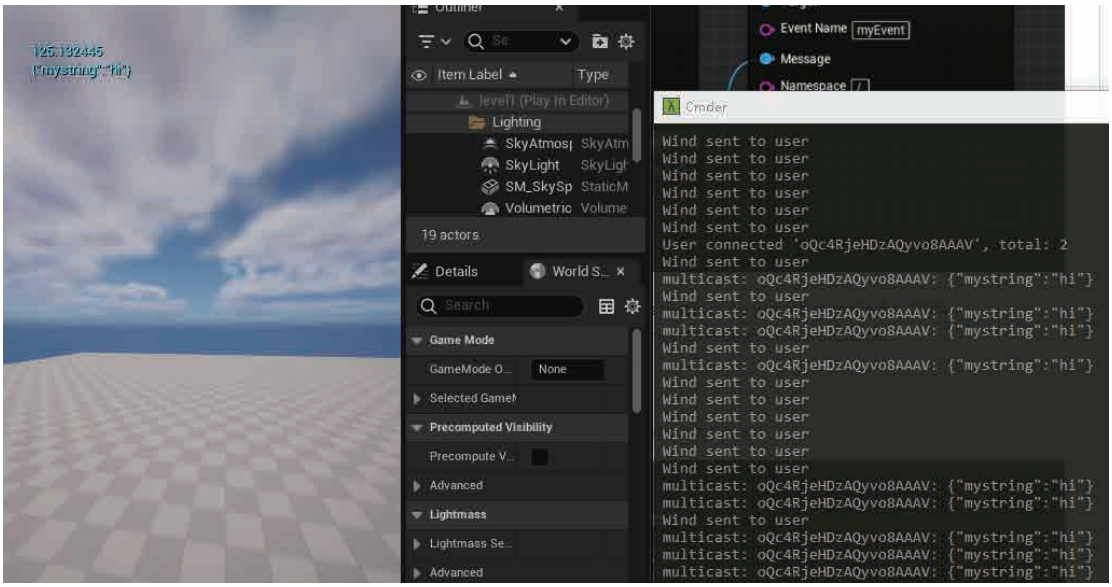

图 1-25　按下 M 键后 UE5 和服务器分别显示 JSON 数据

留意到 server.js 中有下面这样一段 JS 代码，我们可以详细理解一下它的用途。

```
app.get('/', function(req, res){
  res.sendFile(__dirname+'/index.html');
});
```

这段代码的意思是在服务器开启后，用户通过访问服务器 IP 地址（如 3000/index.html 这样的网址就能访问到服务器上的 index.html 网页文件。笔者在文件路径 E:\UE5_Tutorials\Cai_NewBook\SocketIODemo\SocketIO_server 下放置 index.html 文件，该文件详细的代码内容如图 1-26 所示。

```
32      <input type=button value="Action A" Class="btn" id="btnA"/>
33      <input type=button value="Action B" Class="btn" id="btnB"/>
34      <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.0.0/socket.io.js"
        ></script>
35      <script src="http://code.jquery.com/jquery-1.11.1.js"></script>
36      <script>
37          var socket = io();//连接socket.io服务器
38
39          $('#btnA').on('click', function(msg){//如果点击按钮A
40              socket.emit('chatMessage', 'A');
                //客户端向服务器发送chatMessage事件并附带字符A信息
41          });
42          $('#btnB').on('click', function(msg){//如果点击按钮B
43              socket.emit('chatMessage', 'B');
                //客户端向服务器发送chatMessage事件并附带字符B信息
44          });
```

图 1-26　index.html 文件局部内容

该 HTML 文件在页面上摆放了两个按钮（button），通过 JavaScript 为两个按钮绑定了单击事件（on click），单击按钮后会分别向服务器发送字符 A 和字符 B 的数据信息。在计算机上打开 Chrome 浏览器并输入网址 http://127.0.0.1:3000/index.html 便可以访问到该页面。单击页面中名为 Action A 的按钮后，可以看到 UE5 输出了字符 A。如果单击页面中名

为 Action B 的按钮，则会看到 UE5 输出字符 B。这样，通过 UE5 外部的浏览器就能够实现向 UE5 发送字符消息了，如图 1-27 所示。

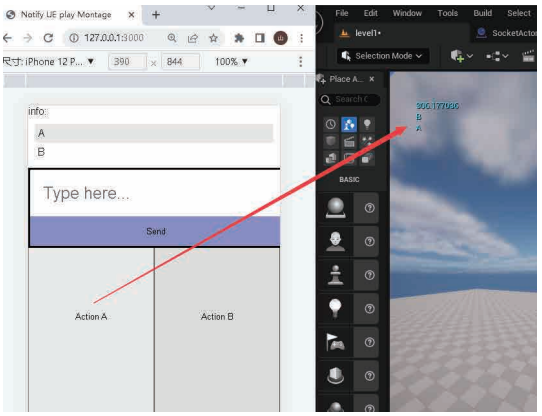

图 1-27　单击 index.htm 页面上的按钮看 UE5 的打印输出

1.1.3　与UE5中的物体进行互动的两种方法

目前，通过外部的浏览器页面与 UE5 通信的功能就已经实践成功了。如果要借助这些外来的数据信息进一步控制 UE5 中的物体对象或其他细节元素，而不仅是打印输出，应该怎么做呢？

接收到外部数据后需要通过蓝图向 UE5 中的对象发送指令，这个过程通常需要在 UE5 中实现对象与对象之间的信息传递。两种常用的在 UE5 对象之间发送消息的机制介绍如下。

1. 使用事件分发器（Event Dispatcher）

首先，笔者演示一下在 UE5 里使用事件分发器来实现不同对象之间传递信息的过程。在 SocketActor 蓝图中添加一个事件分发器用于发送事件信息，取名为 SocketEventDispatcher。如果看不到 My Blueprint 面板，则可以从菜单 Window 中选择 My Blueprint。需要加以区分和注意的是，这个事件分发器与之前提及的 WebSocket 服务器毫无关联，而是 UE5 内部对象之间传递信息的一种机制。添加的方法如图 1-28 所示。

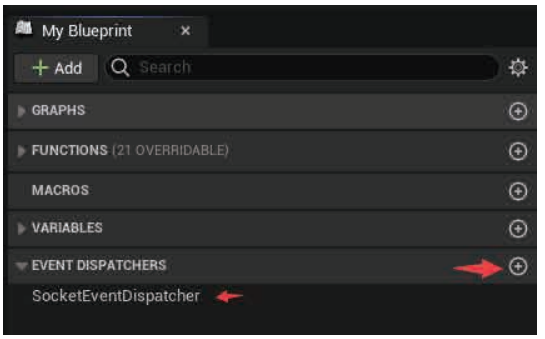

图 1-28　在蓝图中添加事件分发器

接着在 My Blueprint 面板中单击 VARIABLES 右侧的加号按钮，添加一个字符串类型（String）的变量（Variable），取名为 StringFromSocket，用于存储 Socket 服务器发来的字符数据，以便其他 UE5 对象后续读取，如图 1-29 所示。

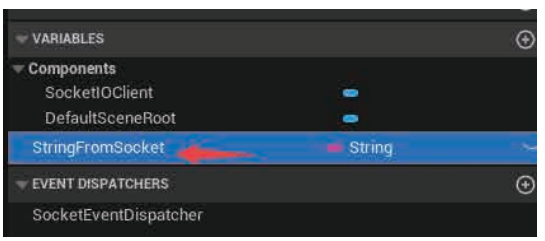

图 1-29　在蓝图中添加字符类型变量

进一步完善蓝图，在接收到 chatMessage 事件后把收到的数据（EventData）写入变量 StringFromSocket 中，同时在 UE5 内部调用（Call）这个名为 SocketEventDispatcher 的事件分发器，如图 1-30 所示。

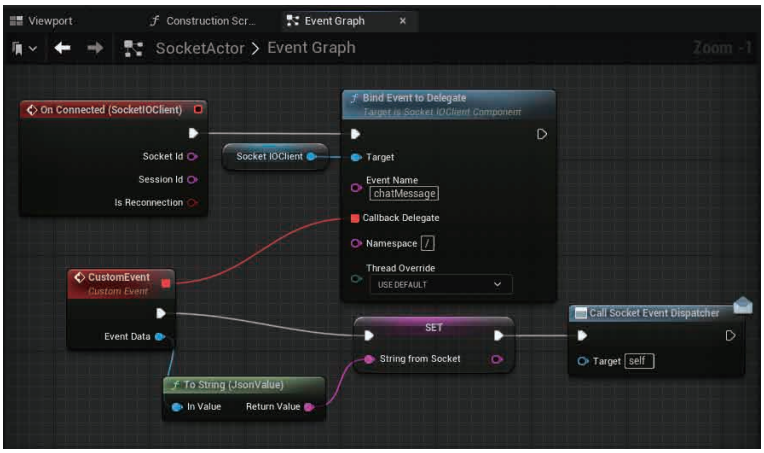

图 1-30　把收到的事件数据写入变量并调用事件分发器

这样就能让 SocketActor 对象在收到服务器通过 chatMessage 事件传来的信息时，将信息存入变量 StringFromSocket 中，然后调用（也就是触发）SocketEventDispatcher 这个 UE5 事件分发器。借助这个事件分发器可以让 SocketActor 通知其他的 UE5 对象。要完成整个过程，需要继续在场景（对场景的称谓有多种，也可以称为视口，或叫作关卡、舞台等）中选中 SocketActor 对象，如图 1-31 所示。

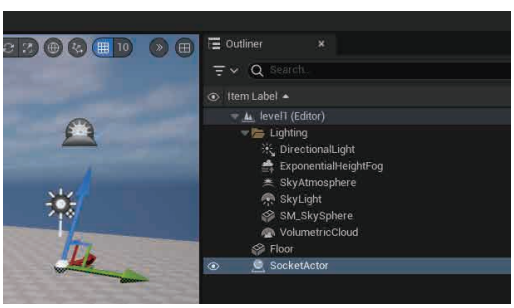

图 1-31　在视口选中 SocketActor

然后单击 Open Level Blueprint，打开关卡蓝图，具体操作如图 1-32 所示。

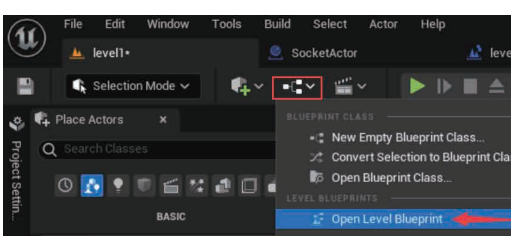

图 1-32　UE5 打开关卡蓝图

在关卡蓝图的空白处右击，可以从弹出菜单中获得对刚才在场景中所选中的 SocketActor 对象的引用，如图 1-33 所示。

图 1-33　关卡蓝图中获取对所选对象的引用

如果看不到 UE5 主界面上的 Place Actors 面板，可以依次选择 UE5 顶部菜单命令 Window → Place Actors，然后在 Place Actors 面板里找到 Cube 拖曳到视口里，这样就在场景中添加了一个立方体。拖曳的具体操作如图 1-34 所示。

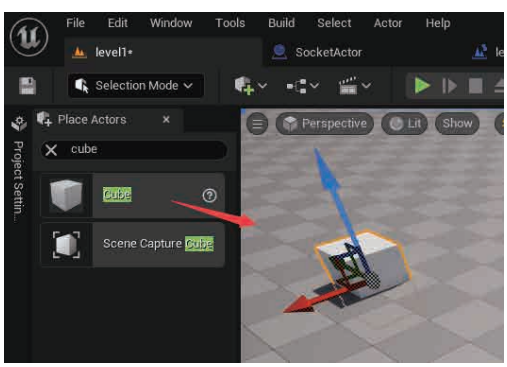

图 1-34　在关卡中直接拖曳添加一个立方体

选中场景中的立方体，在它的 Details（细节）面板里设置它的 Mobility（移动性）为 Movable（可移动的），这样才能在后期通过程序来移动它，否则它是固定的、无法移动的。设置方法如图 1-35 所示。

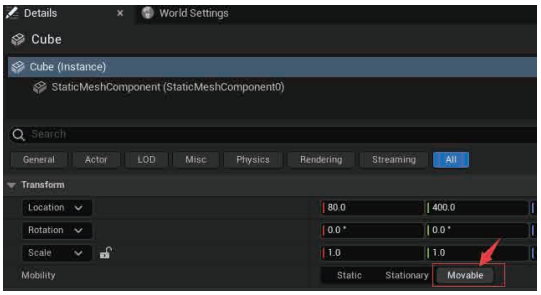

图 1-35　设置立方体可移动

让场景中的立方体继续处于被选中的状态，然后进入关卡蓝图中，在蓝图空白处右击，可以获得对立方体的引用，如图 1-36 所示。

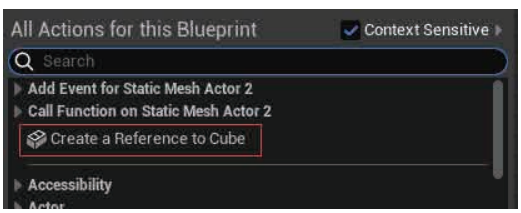

图 1-36　在关卡蓝图里右击获取对立方体的引用

接下来可以把关卡蓝图中的节点内容设置为如图 1-37 所示的样子[①]。

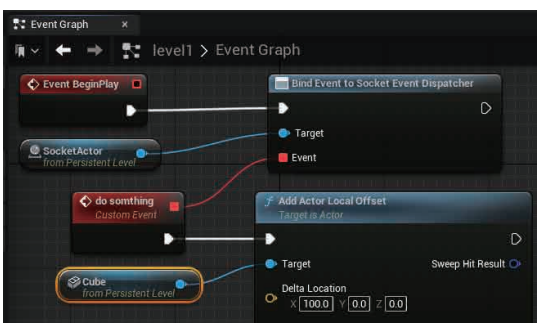

图 1-37　设置关卡蓝图绑定事件

蓝图内容的含义是当关卡开始运行时，会把 SocketActor 与 SocketEventDispatcher 事件分发器绑定，一旦事件分发器被调用，就会执行自定义的事件 dosomething。也就是说如果 SocketActor 内部发起 SocketEvent Dispatcher 事件时，就会执行这里的 dosomething 事件，而 dosomething 事件所做的事情是让立方体相对其现有位置为 X 轴坐标增加 100 个单位。

如果此时单击浏览器页面上的按钮 A 或按钮 B，那么立方体都会沿着 X 轴的方向移动 100 个单位的距离。如果希望按钮 A 或按钮 B 能分别控制立方体向不同的方向移动，可以参照图 1-38 中呈现的内容来布置蓝图，用 Switch on String 节点来判断 SocketActor 里的变量 StringfromSocket，是 A 的时候给 X 坐标值加 100，是 B 的时候减 100。

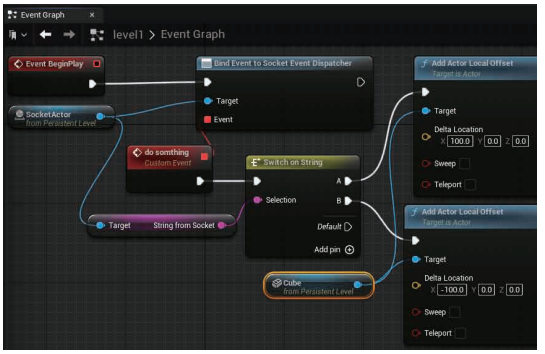

图 1-38　关卡蓝图里对 A 和 B 区分对待

如果需要用手机浏览器访问这个网页页面，则需要确保手机与这台计算机都连了同一个 Wi-Fi，而且要知道这台计算机在局域网内的 IP 地址。在命令行里输入 ipconfig 并按回车键，就能看出本机的局域网内的 IP 地址，显示结果如图 1-39 所示。

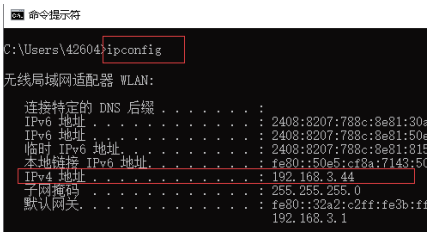

图 1-39　在命令行输入 ipconfig 获取本机 IP 地址

① 图中somthing拼写有误，应该是something。但因为是截图，故保留了原始界面的状态。

接下来手机通过浏览器访问网址 http://192.168.3.44:3000 就可以打开页面来操作了。

注意：192.168.3.44 应当替换为读者自己所用计算机的 IP 地址。

2. 利用变量的 Instance Editable 属性

采用另外一种方法也可以让 Socket Actor 对象访问到场景中的其他物体。在 SocketActor 蓝图里建立一个名称为 TargetObject、类型为 Static Mesh Actor（静态网格体对象）的变量，并把它的眼睛图标打开，如图 1-40 所示。

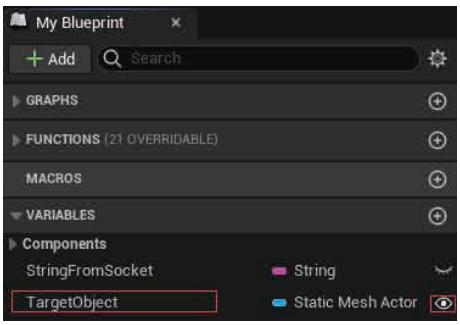

图 1-40　在 SocketActor 里创建 TargetObject 变量

点开这个变量的眼睛图标实际上就是将这个变量的 Instance Editable 属性设为 True 了，也就是这个属性会被打勾，如图 1-41 所示。

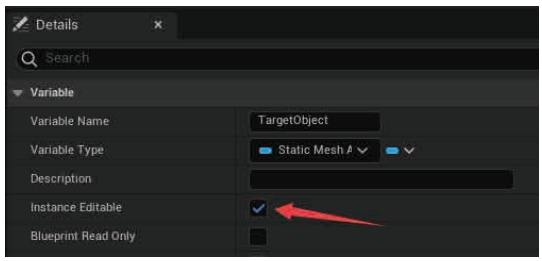

图 1-41　设置 Instance Editable 属性为 True

这样在 SocketActor 对象被放置在视口中（这个过程称为被实例化）后，还可以直接在视口中修改 TargetObject 变量的值。在场景里选中 SocketActor，在它的 Details 面板里找到 TargetObject 属性，在其右侧会看到一个吸管工具图标，单击吸管工具然后点选场景中的立方体，就把变量 TargetObject 指向了舞台中的 Cube（立方体）对象。操作过程如图 1-42 所示。

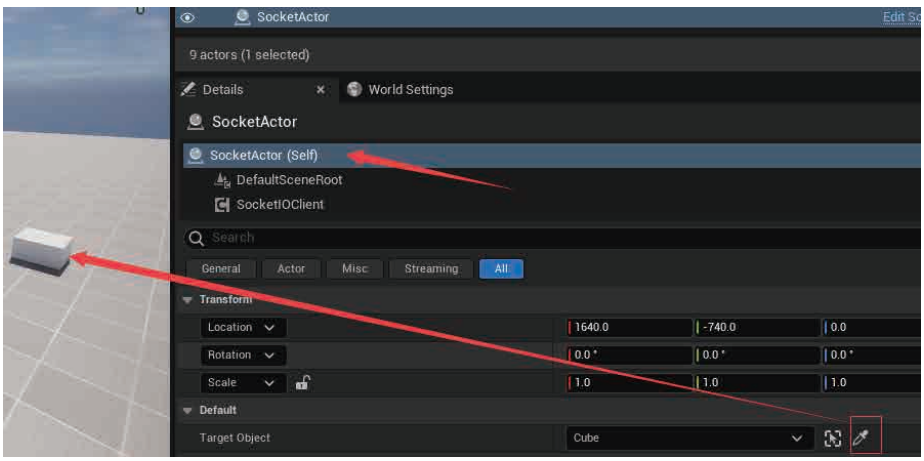

图 1-42　设置变量 TargetObject 指向场景中的立方体

这样，SocketActor 里的蓝图在访问 TargetObject 变量时，就等于访问到了视口中的立方体对象。也就是说，通过这个方法，SocketActor 内的蓝图可以方便地访问 SocketActor 外的任何对象。详细的蓝图内容如图 1-43 所示。

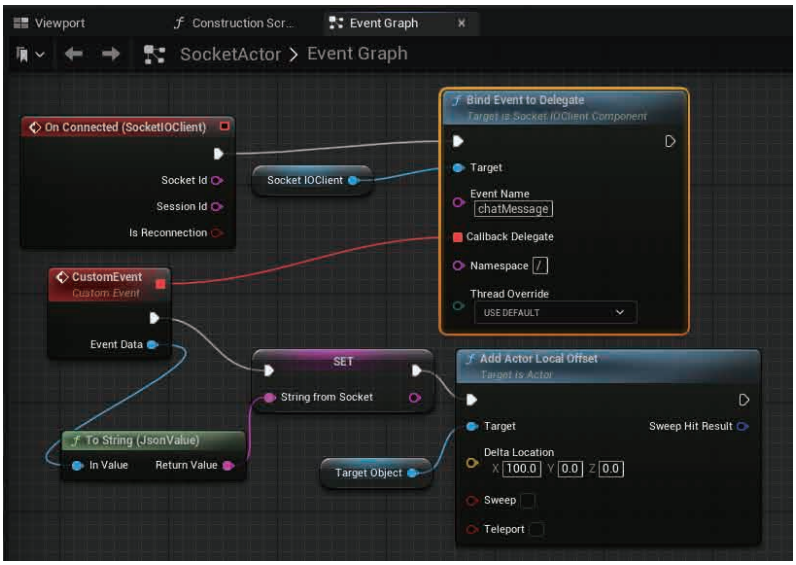

图 1-43　在蓝图里绑定 Socket 事件

此时运行 UE5，预期的效果也同样实现了。

1.1.4　实例：手机遥控点亮UE5屏幕上的烟火

实例的讲解既可以巩固前面所学的基础知识，又可以让我们在已经掌握的基础知识之上有所提升。本书的第一个实例就是利用手机来控制 UE5 中画面内容的变化，具体来说是通过触摸手机上的网页界面来控制 UE5 里焰火的启停。针对本实例，笔者新建了一个 UE5 项目，名为 SocketIODemo_live，项目路径为 E:\UE5_Tutorials\Cai_NewBook\SocketIODemo_live。打开项目文件后，从虚幻商城找到免费的 Realistic Starter VFX Pack 添加到工程里。这个文件包里含有本实例将要用到的焰火粒子系统素材。具体操作如图 1-44 所示。

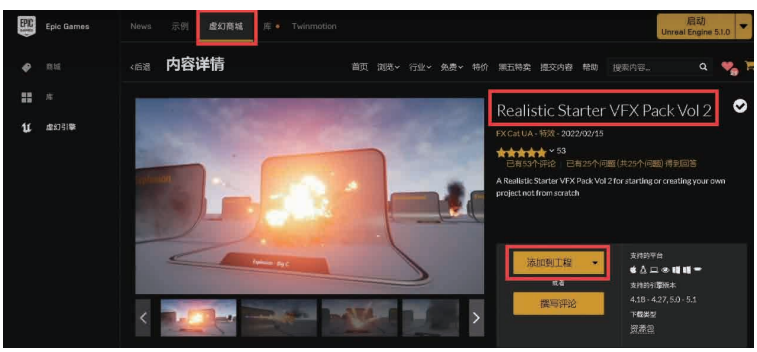

图 1-44　从虚幻商城下载 Realistic Starter VFX Pack Vol2

这样 UE5 项目的 Content 文件夹里就会出现一个 Realistic_Starter_VFX_Pack_Vol2 文件夹，里面的 Maps 文件夹里有 Overview_Map_Day 关卡文件，打开这个关卡可以看到里面有很多粒子系统的展示，如图 1-45 所示。

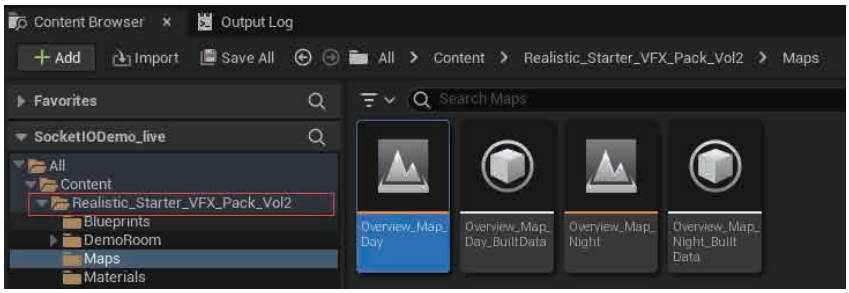

图 1-45　Content 文件夹里出现了下载的素材文件夹

双击打开这个关卡，从它的 Outliner（大纲）面板里找到名为 B_Sparks_G 的对象，选中后按 Ctrl+C 组合键复制它。然后新建一个关卡，取名为 level1，在视口里按 Ctrl+V 组合键，将刚复制的 B_Sparks_G 粘贴到 level1 里。这样在 level1 里也就能使用 B_Sparks_G 这个对象了，如图 1-46 所示。

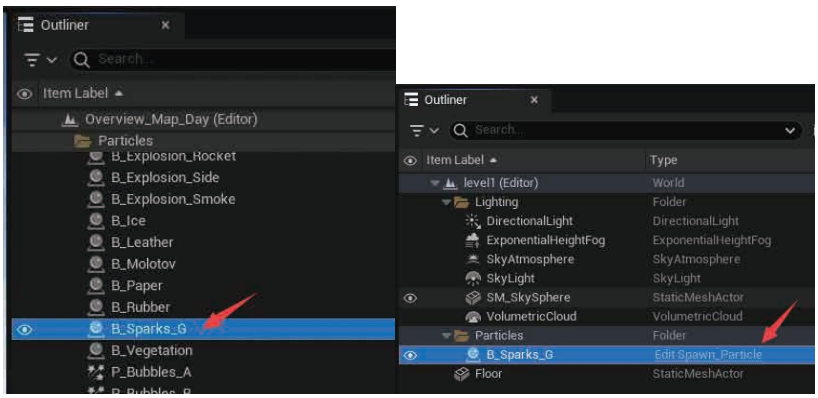

图 1-46　复制对象到 Level1 中然后编辑

接着可以单击 B_Sparks_G 右侧的 Edit Spawn_Particle 来编辑它的蓝图。进入蓝图后，把里面的组件面板中的 VFX 组件复制一份，取名为 VFX1，再添加一个 SocketIOClient 组件。VFX 组件是一个粒子系统组件，复制 VFX 得到的 VFX1，设置其属性如图 1-47 所示，让它实际调用了 P_Sparks_A 粒子系统。这样 Spawn_Particle 里就有两个不同的粒子系统了，如图 1-47 所示。

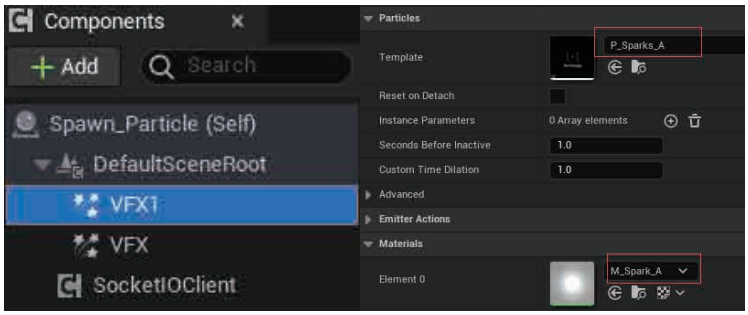

图 1-47　复制 VFX 组件得到 VFX1 后修改属性

值得注意的是，SocketIOClient 组件的细节面板里 Address and Port 属性默认就是

http://localhost:3000，如果 Socket 服务器的部署不在本地计算机上，就需要把 localhost 改为服务器 IP 地址。修改的位置如图 1-48 所示。

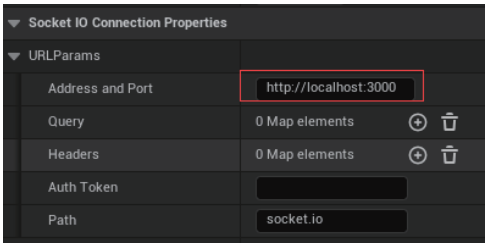

图 1-48　设置 SocketIOClient 组件的地址和端口属性

关于 Socket 服务器的设置还是和上一节的讲述完全一样，在这里的 Spawn_Particle 蓝图中，可以将蓝图节点进行如图 1-49 所示的布置，从而实现单击网页上的不同按钮便可以控制 UE5 运行不同粒子系统的功能。

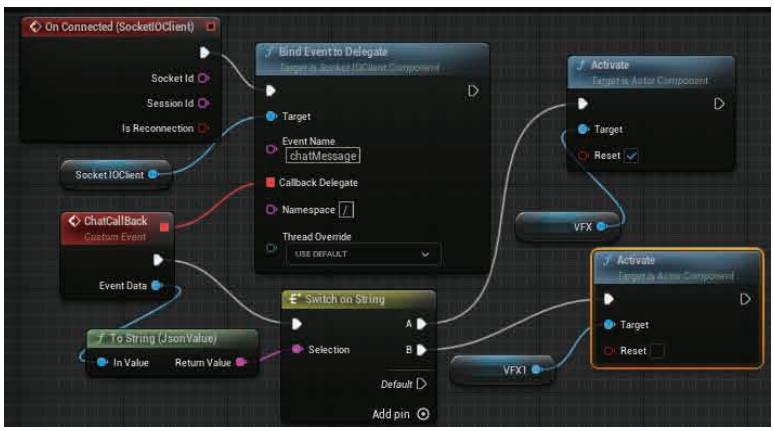

图 1-49　Spawn_Particle 的蓝图内容

这样，当通过浏览器访问 http://localhost:3000 并单击页面上的第一个按钮时，UE5 会播放一个烟花爆炸风格的粒子特效。如果单击网页上的第二个按钮，则会播放另外一个 VFX1 对应的龙卷风类型的粒子特效，如图 1-50 所示。

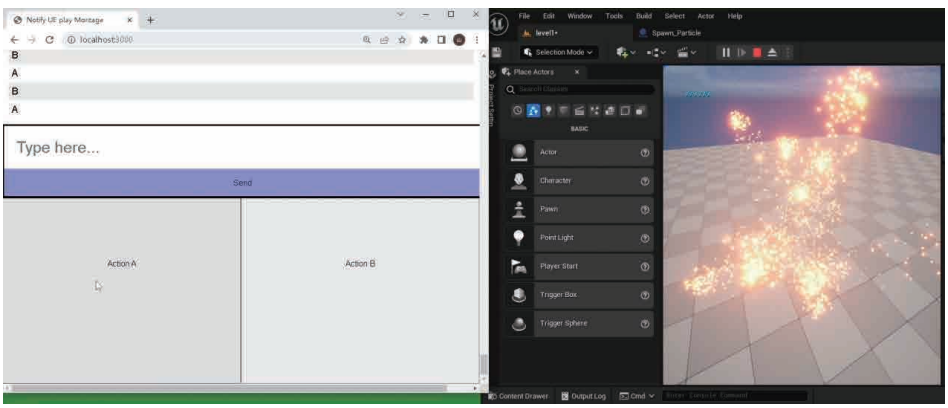

图 1-50　单击 ActionA 按钮激发烟花爆炸粒子特效

用手机打开网址 http://192.168.3.44:3000（注意实践时要将 IP 地址替换为自己计算机的局域网 IP），在手机上用手指触击按钮 A 可以播放烟火粒子，而触击按钮 B 则可以激发龙卷风粒子特效，如图 1-51 所示。

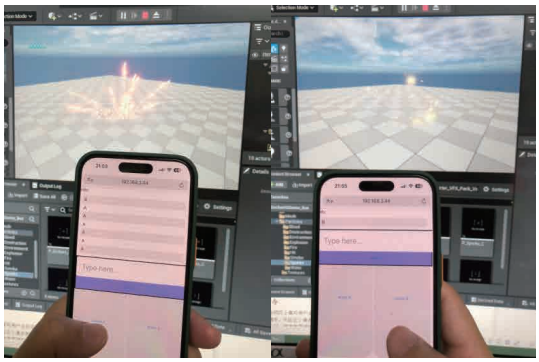

图 1-51　手机上触击不同按钮
遥控 UE5 播放不同粒子特效

本实例详细的操作步骤可以通过扫描下方二维码来观看。

1.2　用 Remote Control API 远程控制 UE5 界面

1.2.1　初步认识Insomnia调试指令

大家对 HTTP Request（HTTP 请求）应该都不陌生，在网页上填写用户名和密码，然后单击提交按钮，其实就是向网站服务器发送了一个 HTTP 请求，并且这个请求附带了用户填写的数据信息。Web 前端工程师在测试自己的页面是否功能健全时，就常常会模拟发送类似这样的 HTTP 请求到自己开发的页面地址，通过模拟测试工具获得的返回值来判断有无Bug。例如，Postman 就是业内知名度很高的一款模拟调试工具。而笔者要给读者介绍的是一款名为 Insomnia 的相对轻便并且免费、开源、跨平台的调试工具，如图 1-52 所示。

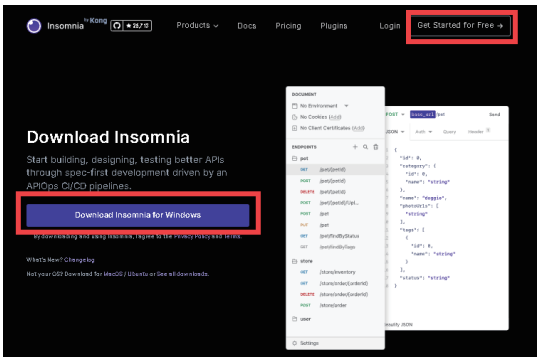

图 1-52　免费下载 Insomnia

在本节里笔者会基于 Insomnia 这款调试工具模拟 HTTP 请求并发送给 UE5，通过这种方式来操控 UE5 场景中的对象，同时获取来自 UE5 的信息反馈。

从 Insomnia 官网下载免费版的 Windows 安装包，安装好程序后打开 Insomnia。从程序界面上可以看到一个加号图标，如图 1-53 所示，单击它可以构建一个 HTTP 请求。

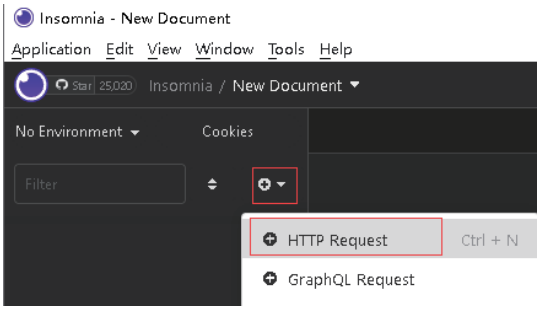

图 1-53　Insomnia 程序界面构建 HTTP 请求

1. HTTP 请求的详细设置

一个完整的 HTTP 请求包含请求方式、请求的网址和请求的数据。笔者接下来带着大家使用 Insomnia 具体设置一下 HTTP

请求的请求类型以及请求的数据格式和请求的网址。

通过单击图 1-54 所示的白色下拉箭头可以将 HTTP 请求方式设置为 PUT 类型。

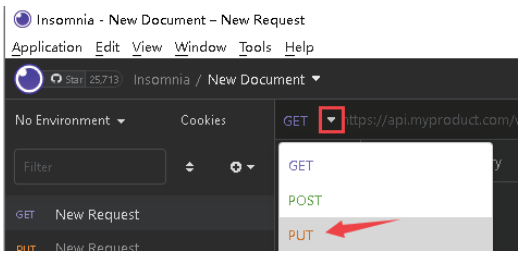

图 1-54　设置请求方式为 PUT

如果对 Web 开发知识有所了解，会知道 HTTP 请求里最常见的请求类型是 GET 和 POST 这两种。关于请求类型，网上有非常多的介绍资料。本节着重就 PUT 这种请求方式为读者作一个简要的介绍。PUT 请求是一个幂等的 HTTP 请求方式。幂等指的是发出同样的请求时，被执行一次与连续执行多次的效果是一样的，服务器的状态也是一样的。换句话说就是，幂等方法对数据不应该具有副作用。在正确实现的条件下，GET、PUT 和 DELETE 等方法都是幂等的，而 POST 方法不是幂等的，具体详见表 1-1。

表 1-1　HTTP 请求的多种方式的对比

HTTP 请求方式	请求地址 URL	效用
GET	http://...../tickets	获取 ticket 列表信息
POST	http://...../tickets	新建一条 ticket 记录
PUT	http://...../tickets/1	更新 ticket 编号为 1 的记录
DELETE	http://...../tickets/1	删除 ticekt 编号为 1 的记录

①多次调用 GET /tickets 和第一次调用 GET /tickets 对数据库数据的影响是一致的。

②多次调用 PUT /tickets/1 和第一次调用 PUT /tickets/1 对数据库数据的影响是一致的。

③多次调用 DELETE /tickets/1 和第一次调用 DELETE /tickets/1 对数据库数据的影响是一致的。

④多次调用 POST /tickets 都将产生新的数据（对数据库中的数据的影响是不一致的）。

综上所述，可以认识到，GET、PUT 和 DELETE 方法都是幂等的，而 POST 方法不是幂等的。理解了 PUT，再为 HTTP 请求输入访问的 URL 网址，然后再将 HTTP 请求对服务器所发送数据 Body 的类型设为 JSON 类型。如图 1-55 所示，输入的网址是 http://localhost:30010/remote/object/property。

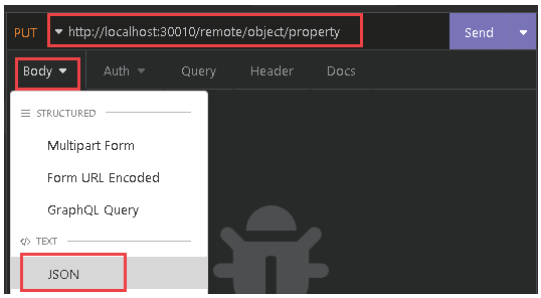

图 1-55　输入请求网址
并设置请求数据格式为 JSON 格式

到这里，就在 Insomnia 软件里设置好了一个 HTTP 请求，就已经做好了向服务器发送数据的准备！只需要单击图 1-55 中右上角的 Send（发送）按钮，即可将 HTTP 请求发送出去。

2. 启动 UE5 内置的 WebControl 服务器

接下来笔者要带大家使用 Insomnia 所发送出来的 HTTP 请求来实际控制 UE5 项目中的对象，同时也包括在 Insomnia 软件里查看 UE5 反馈的数据信息。但实现这些的前提是需要先在 UE5 项目中开启 UE5 内置的 WebControl 服务器。下面先讲一下如

何开启这个 WebControl 服务器。笔者新建一个 UE5 项目，取名为 RemoteCtrlAPIDemo，在项目的 Plugins 菜单里需要开启 Remote Control API 插件（插件打勾后需要重启 UE5 项目），如图 1-56 所示。

图 1-56　开启 Remote Control API 插件

在 UE5 的 Project settings（项目设置）里可以看到 Plugins 下 Remote Control 项的相关设置，默认设置为 Remote Control HTTP Sever 会监听本机端口 30010，也就是说访问 http:// 本机 IP:30010 的 HTTP 请求都会被 UE5 内置服务器接收到（这里看到的 30000 端口将在 1.2.3 节里出现），如图 1-57 所示。

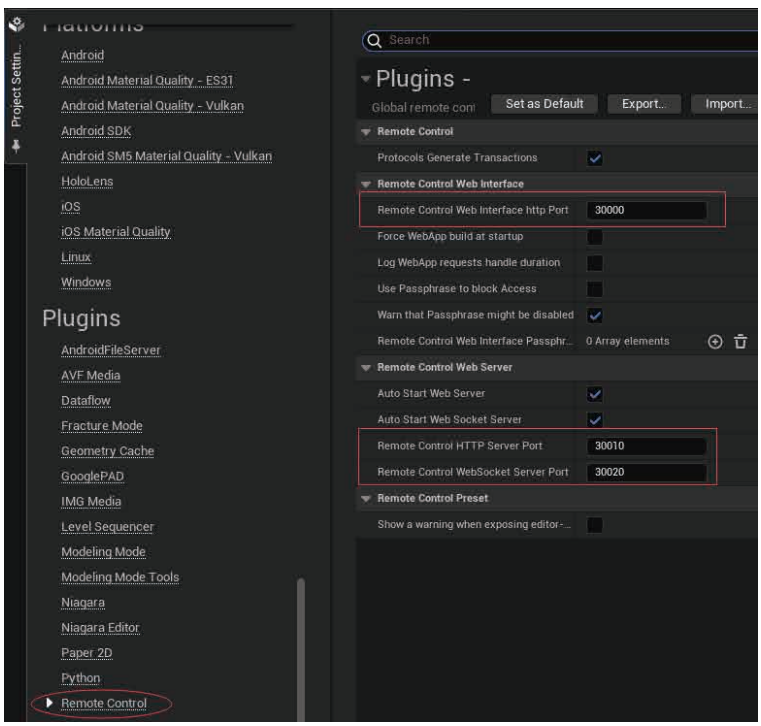

图 1-57　项目设置里 Remote Control HTTP Server Port 的默认值

在 UE5 的界面底部能看到如图 1-58 所示的 Cmd 输入框，如果在这个输入框里输入 WebControl.StartServer，然后按回车键就可以启动 UE5 内置的 WebControl 服务器了。

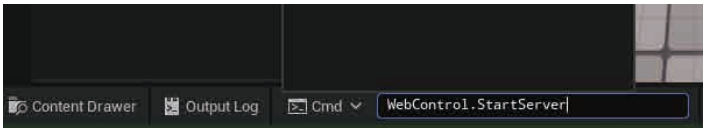

图 1-58　项目设置里 Remote Control HTTP Server Port 的默认值

在 UE5 顶部的 Window 菜单里勾选 Output Log，就能在 UE5 底部看到 Output Log 标签，单击此标签能看到 UE5 的所有 Log 信息记录。如果输入如图 1-58 所示的指令并按回车键后，能看到 Output Log 里显示出 Cmd:WebControl.StartServer 字样，就表示 UE5 内部的 WebControl 服务器启动成功了，如图 1-59 所示。

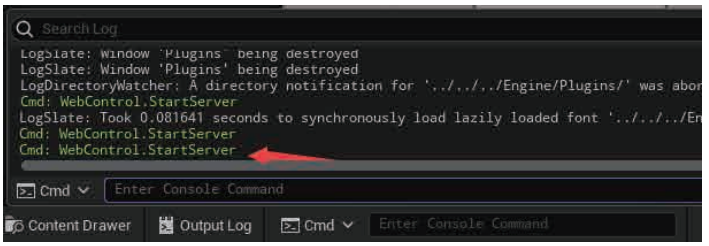

图 1-59　UE5 查看 Output Log 的输出

3. 使用 HTTP 请求访问和修改 UE5 对象属性

启动了 UE5 内置的 WebControl 服务器后，就可以正式使用 Insomnia 发送 HTTP 请求给 UE5，以访问 UE5 对象的属性或修改 UE5 对象的属性了。在本节里笔者带大家详细实践一下具体的做法。

笔者在 UE5 视口中放置一个立方体对象，然后从大纲面板（Outliner）里选中这个立方体（Cube）并右击它。从弹出的右键菜单里选择 Copy path 复制路径，可以获取这个立方体物体在视口中对应的远程控制地址信息。这种类似于网址格式的地址信息是后续通过 HTTP 请求访问 UE5 中物体的入口地址。具体操作如图 1-60 所示。

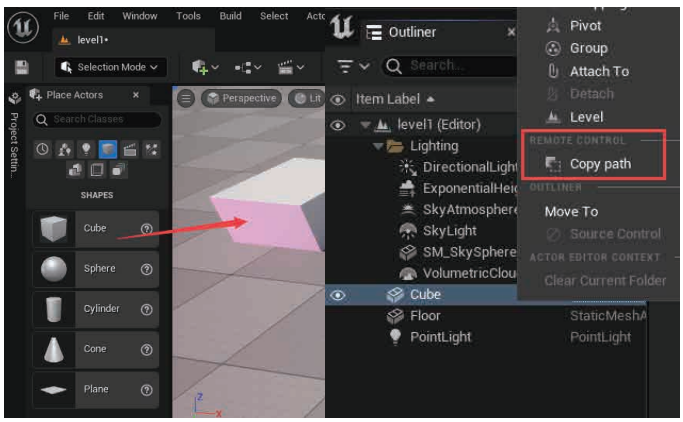

图 1-60　右击选择 Copy path 获取物体的远程控制路径地址

如果把复制到的信息粘贴到记事本里可以看到：/Game/level1.level1:PersistentLevel.StaticMeshActor_0。

这里需要将这个路径信息填写到 Insomnia 的数据区域来构建一个 JSON 格式的数据。

```
{"objectPath":"/Game/level1.level1:PersistentLevel.StaticMeshActor_0" ,
"propertyName":"bHidden"}
```

如图 1-61 所示，HTTP 请求构建完毕后就可以单击 Send 按钮向 UE5 发送请求了。

这时在 Insomnia 程序界面的右侧可以看到 Preview 标签下有信息出现，这就是 UE5 内置服务器反馈回来的信息。如果没有报错，并且是 200 OK 状态，就表示请求处理成功了。

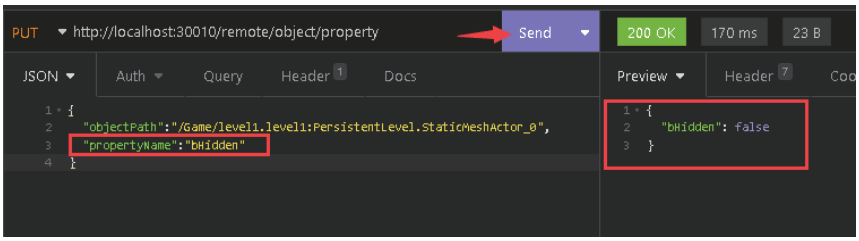

图 1-61　构建 JSON 格式的请求数据

图 1-61 里所构建的 HTTP 请求的含义是针对 UE5 中的立方体，将它的 Hidden 属性设置为隐藏。请求发送完毕后在 UE5 里可以查看视口中立方体的细节面板信息，会发现 Actor Hidden In Game 属性被打勾。这正是 HTTP 请求发送到 UE5 服务器后被正确处理的结果，如图 1-62 所示。

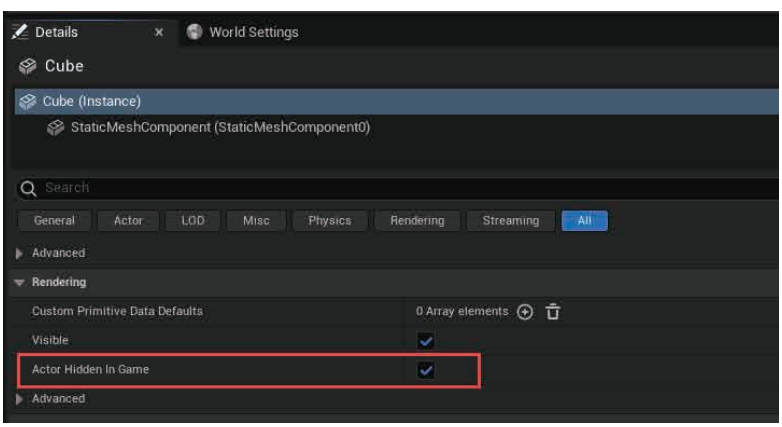

图 1-62　通过 HTTP 请求将立方体隐藏了

如果修改在 Insomnia 里的请求信息，去掉请求数据中的 "propertyName":"bHidden" 部分，那么发送的请求就是查询立方体各项属性的请求，收到的反馈数据将是立方体的各项属性信息。如图 1-63 右侧所示，可以看到查询结果里的各项属性数据。

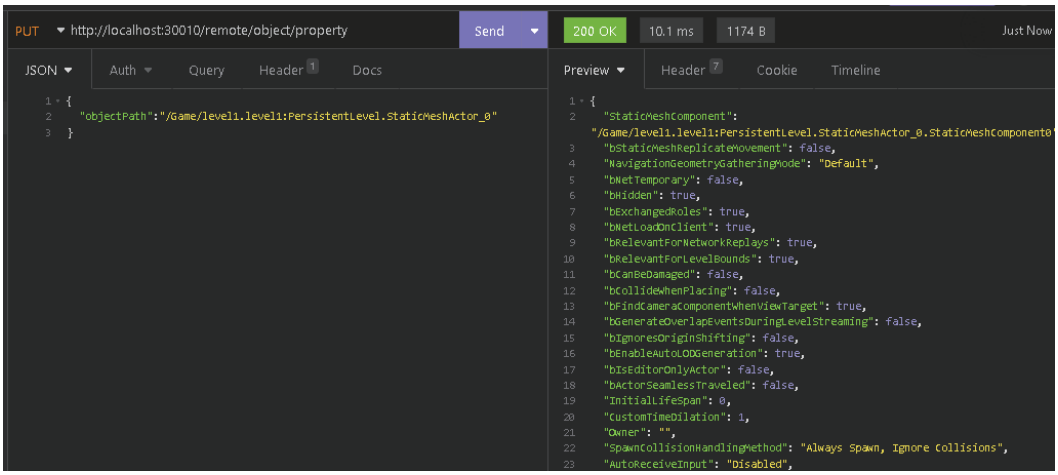

图 1-63　通过 HTTP 请求查询立方体属性信息

如果把请求的网址修改为 http://localhost:30010/Remote/object/call，同时将发送的数据按图 1-64 所示进行修改，就可以实现通过发送 HTTP 请求获取立方体对象的位置坐标。

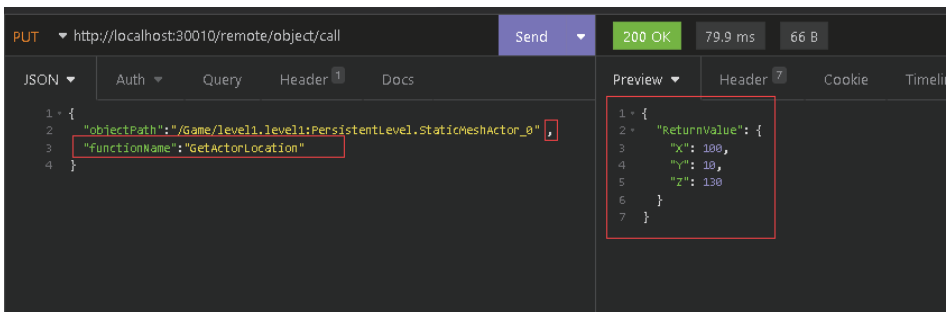

图 1-64　通过 HTTP 请求获得物体位置信息

请求发送完毕后，确实能看到 Insomnia 右侧反馈的信息与立方体在详情面板中的 Location 信息完全一致，如图 1-65 所示。

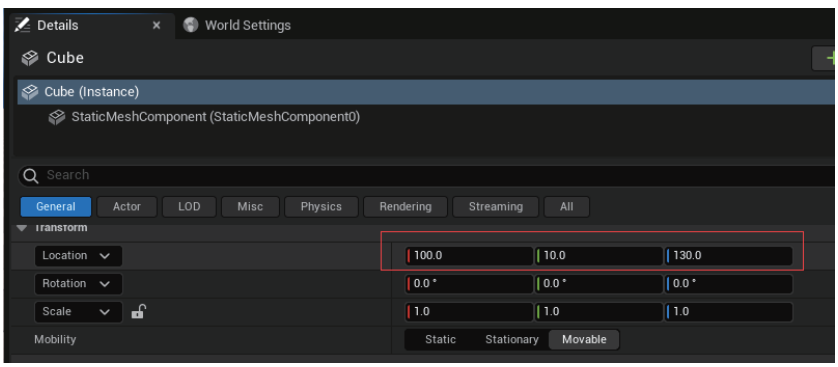

图 1-65　对比 HTTP 请求的反馈信息与 UE5 物体的 Location 数据

接下来可以尝试通过发送 HTTP 请求来修改 UE5 中物体的位置，可以如图 1-66 所示发送 HTTP 请求。

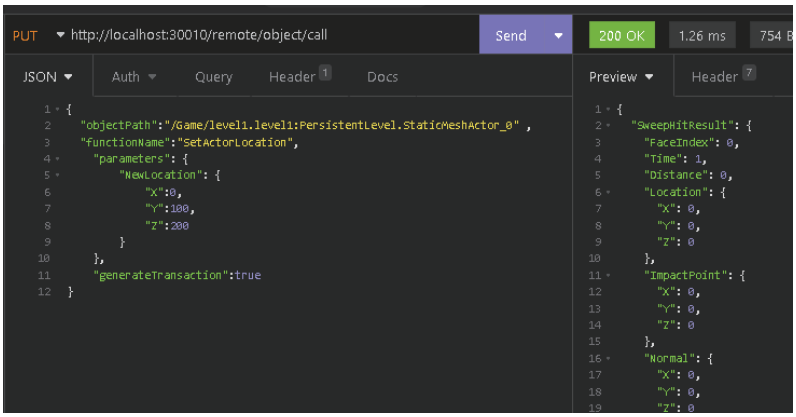

图 1-66　发送 HTTP 请求修改 UE5 物体的 Location 数据

如果想修改物体的朝向角度，也就是旋转物体，可以如图 1-67 所示发送 HTTP 请求数据。具体代码展示如下：

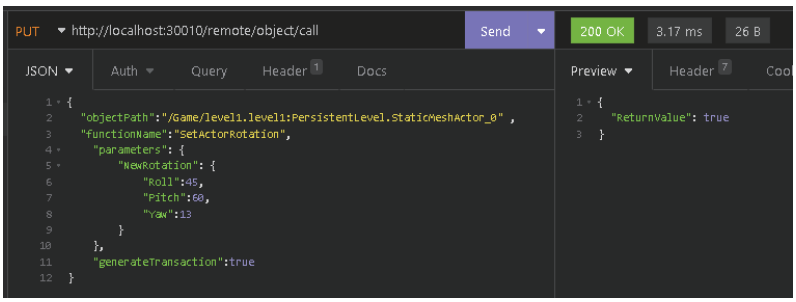

图 1-67　发送 HTTP 请求修改 UE5 物体的 Rotation 数据

```
{
    "objectPath":"/Game/level1.level1:PersistentLevel.StaticMeshActor_0" ,
    "functionName":"SetActorRotation",
    "parameters": { "NewRotation": { "Roll":0, "Pitch":0, "Yaw":0 } },
    "generateTransaction":true
}
```

所发数据的最后 "generateTransaction":true 这句的作用是使这次所请求的操作可以在 UE 里支持撤销，等同于支持 Ctrl+Z 组合键，也就是如图 1-68 所示的操作，当用户想取消本次的修改时，可以单击 Undo 来实现。

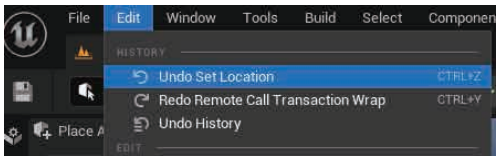

图 1-68　UE5 里的撤销上一步操作

同样地，发送完 HTTP 请求后可以在 UE5 中检查立方体的 Rotation 信息，可以看到 Rotation 属性确实被改变了，如图 1-69 所示。

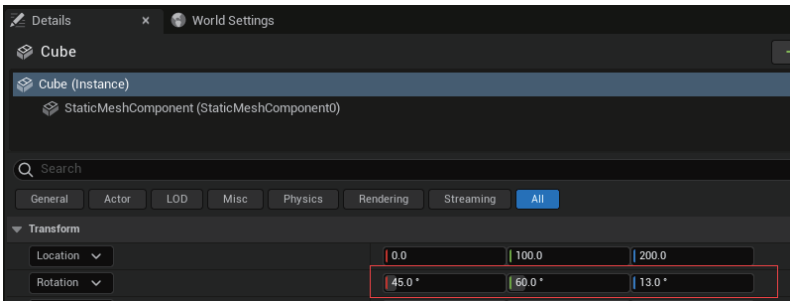

图 1-69　查看 UE5 里立方体的 Rotation 信息

4. 使用 HTTP 请求来调用 UE5 项目中的函数

使用HTTP请求不仅可以访问、修改UE5中对象的属性，还可以直接调用UE5中的函数。大家知道，UE5 中的函数可以顺畅地访问 UE5 的各类细节，而能调用 UE5 中的函数也就意味着可以实现函数所能实现的一切功能。本小节里，笔者带大家具体操作一下如何通过 HTTP 请求来调用 UE5 中存在的函数。

笔者先选中上一节中提到的立方体，通过单击其细节面板里的转化为蓝图按钮将立方体从一个静态网格体对象修改为一个蓝图对象。这个转化为蓝图按钮的所在位置如图 1-70 所示。

图 1-70　单击“转化为蓝图”按钮

从弹出的窗口里，将父类选择为 StaticMeshActor，并将蓝图取名为 Cube_BluePrint。然后单击 Select 按钮，如图 1-71 所示。

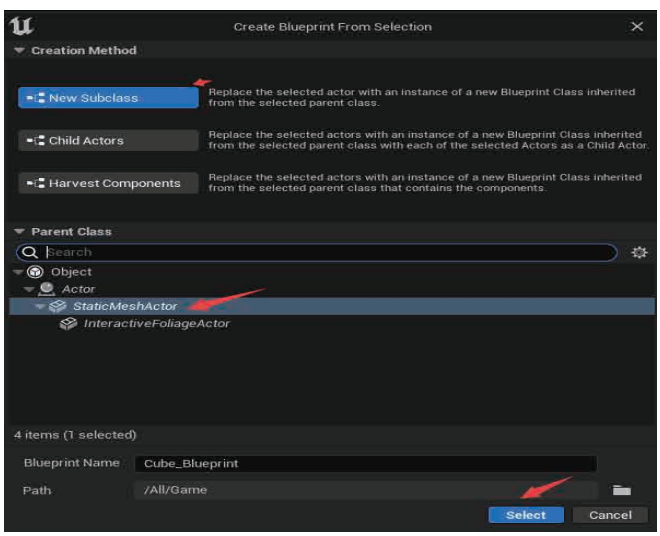

图 1-71　先选择 New Subclass 再选择 StaticMeshActor

在 Content browser 中双击打开这个新建的 Cube_Blueprint 对象，在它的 My Blueprint 面板里增加一个 Function（函数），取名为 Test，如图 1-72 所示。

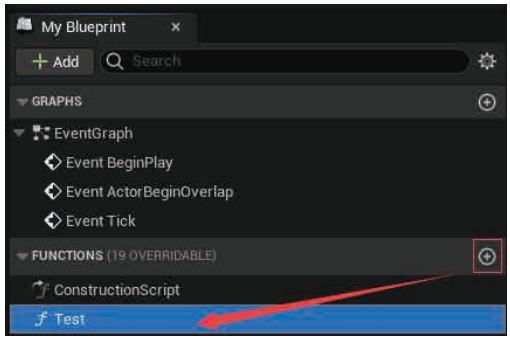

图 1-72　在 My Blueprint 面板里单击加号图标添加函数

双击这个 Test 函数，如图 1-73 所示，设置 Test 函数里的内容，这样调用 Test 函数时就可以调整立方体的世界位置坐标为：x 为 100, y 为 100, z 为 100。注意，原先场景中的立方体现在已经成为 Cube_Blueprint 里的一个静态网格体组件，在组件面板中可以看到它，将它拖入右侧蓝图编辑区域即可加以使用，如图 1-73 所示。

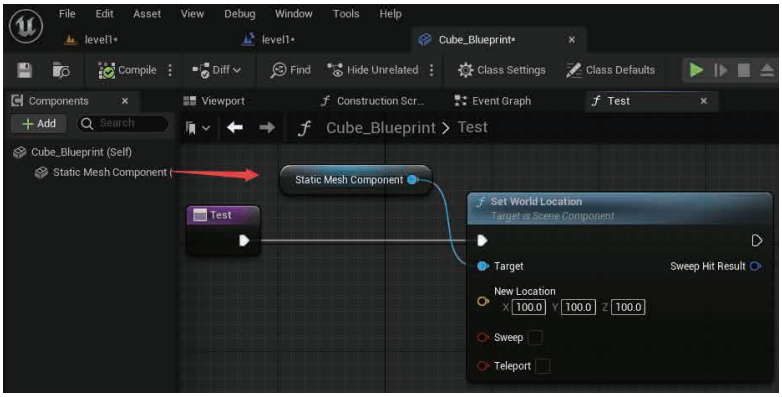

图 1-73　函数 Test 里的蓝图内容

可以继续使用 Insomnia 程序发送 PUT 类型的指令来调用这个蓝图里的函数 Test，配置发送数据，如图 1-74 所示。

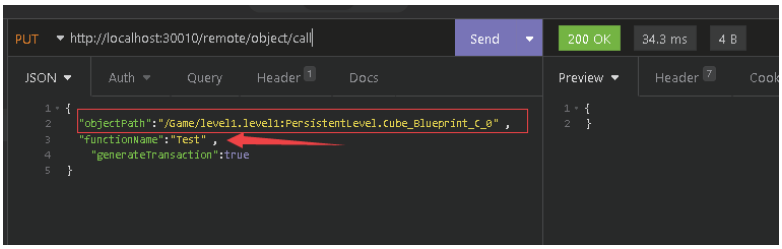

图 1-74　使用 Insomnia 发送 HTTP 请求调用蓝图里的函数

当然 HTTP 请求所需要发送数据里的 objectPath 后的路径地址也需要重新右击 Copy path 来获取最新的，因为立方体网格体已经被转换为一个蓝图对象了，所以它对应的远程

控制路径也发生变化了。

HTTP 请求发送后，在 UE5 视口里查看 Cube_Blueprint 蓝图里的子对象 static Mesh Component（静态网格体组件），也就是原来那个立方体，发现它的 Location（位置）属性确实变为 100、100、100，如图 1-75 所示。

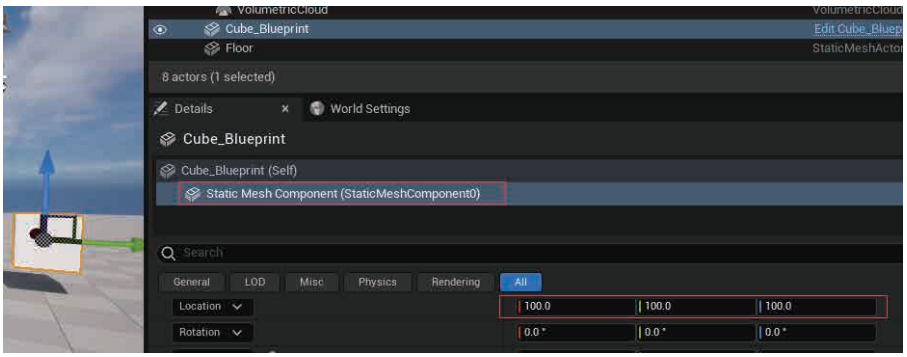

图 1-75　查看 Cube_Blueprint 对象的子对象的 Location 属性

值得注意的是，发送 HTTP 请求并成功执行时，UE5 并没有运行，也就是说即便 UE5 还处于编辑状态，也能够利用 HTTP 请求来改变 UE5 内的设置，这无疑给 UE 设计人员带来了极大的便利。例如，当设计人员有几套将多个物件按不同位置、不同角度摆放到场景中的布景方案时，如果设计人员还希望能够在这几套方案间便捷地来回切换以实现直观的视觉比对，这个功能就能大显身手了！

5. UE5 在运行状态时使用 HTTP 请求

UE5 在编辑状态时能接收 HTTP 请求并对 HTTP 请求作出实时响应。那么在 UE5 处于运行状态时，同样可以发送 HTTP 请求来改动 UE5 中物体的各个属性。但发送 HTTP 请求时有一些注意事项，本小节具体介绍一下这种情况下发送 HTTP 请求在数据内容上的差异。

其实关键点就在于需要在 UE5 运行后再获取物体对应的路径地址，操作方法同样是在视口中选中物体后再右击选择 Copy path 进行获取。但是在 UE5 运行时，视口中看不见鼠标指针，如何用鼠标点选视口里的物体呢？很简单，可以在键盘上按 Shift+F1 组合键，这样就能显示鼠标指针了。然后如图 1-76 所示，单击脱离玩家控制器切换按钮进入模拟状态，就能在 UE5 运行过程中点选视口中的物体了。

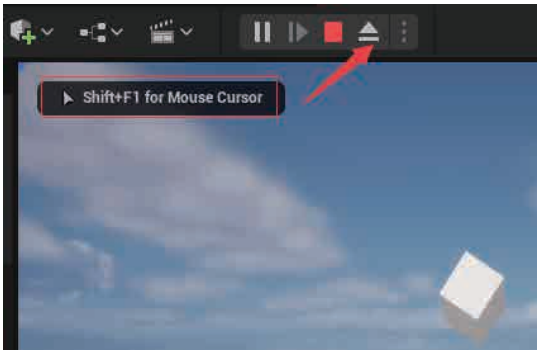

图 1-76　按 Shift+F1 组合键显示鼠标再单击脱离控制器按钮

选中视口中的 Cube_Blueprint 后，可以从大纲面板里看到对应的条目，右击选择 Copy path。也可以直接在视口中选中物体后右击选择 Copy path。

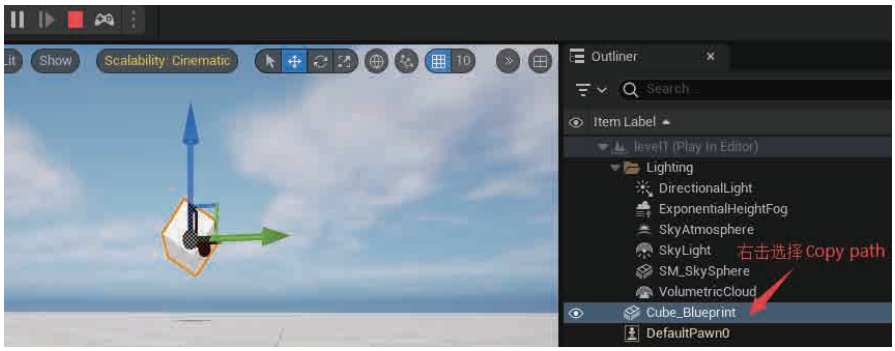

图 1-77　在 Outliner（大纲）面板里选中物体然后右击选择 Copy path

此时得到的路径是/Game/UEDPIE_0_level1.level1:PersistentLevel.Cube_Blueprint_C_0，它与编辑状态时获得的路径信息 /Game/level1.level1:PersistentLevel.Cube_Blueprint_C_0 还是有所不同的。脱离玩家控制器切换按钮单击后会变为附加玩家控制器按钮，再次单击会让 UE5 处于正常运行状态，鼠标指针会再度消失。通过这个过程，读者可以了解到这个切换按钮的用法，如图 1-78 所示。

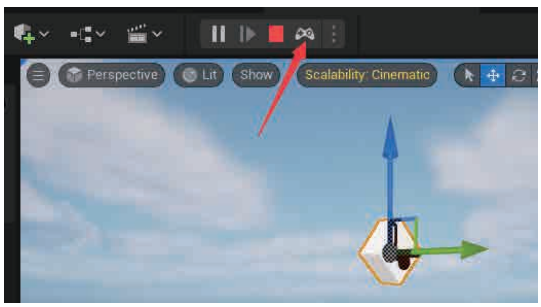

图 1-78　脱离玩家控制器切换按钮单击后会变为附加玩家控制器按钮

将新获取到的路径信息粘贴到 HTTP 请求的发送数据里，然后发送请求，这样在运行 UE5 时也能够通过 HTTP 请求来控制 UE5 视口里的物体，如图 1-79 所示。

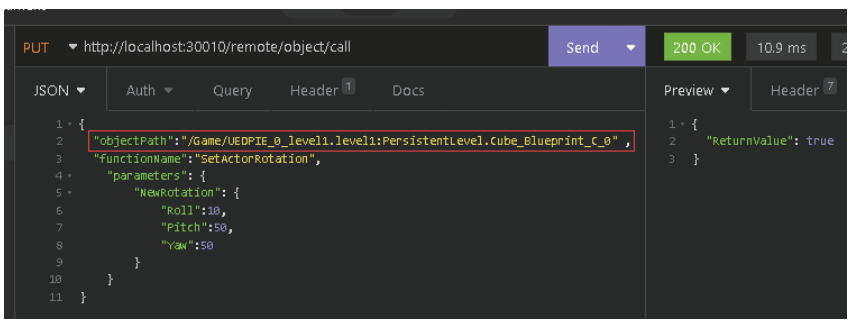

图 1-79　将 UE5 运行时获取的 path 信息粘贴到 HTTP 请求的数据里

Insomnia 工具是一个构建 HTTP 请求的调试工具，本节通过详细的操作步骤让读者了解如何设置 HTTP 请求的数据格式和数据内容、如何填写 HTTP 请求的网址以及如何查看请求是否正确地被 UE5 接收和处理。

1.2.2 采用AJAX+PHP自定义页面发送指令

毕竟 Insomnia 只是一个调试工具，如果能够自己定制网页界面来发送 HTTP 请求指令，那么操作起来肯定会变得更加高效顺畅。下面笔者就在自己的局域网内搭建一个本机 Web 服务器，让手机能打开自己设计的网页，进而控制 UE5 中的物体。

1. 在自己的局域网内搭建一个本机 Web 服务器

首先，从 phpstudy 官方网站下载与自己计算机系统对应的安装包，然后安装 phpstudy。安装好以后，打开 phpstudy，配置一下 Apache 服务器的网站目录，地址为 D:/phpstudy_pro/WWW，如图 1-80 所示。这样本地计算机的这个 WWW 就成为了本地服务器的根目录。

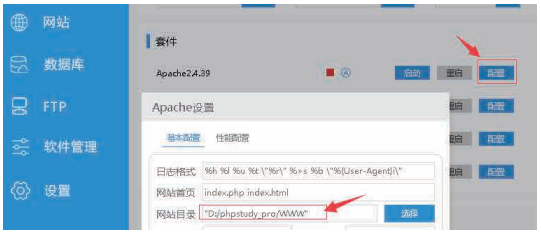

图 1-80 配置本机 Apache 服务器的网站目录路径

设置好以后就可以单击“启动”按钮来启动本机的 Apache 服务器，如图 1-81 所示。

图 1-81 启动本机 Apache 服务器

2. 生成 AJAX+PHP 自定义页面

在网站目录 D:/phpstudy_pro/WWW 下建立一个文件夹 UE5_RemoteControl，并在该文件夹中放入 index.html 和 post.php 两个文件（UE5_RemoteControl 文件夹里的源代码在本书源代码 RemoteCtrlAPIDemo 项目文件夹中也能找到），如图 1-82 所示。

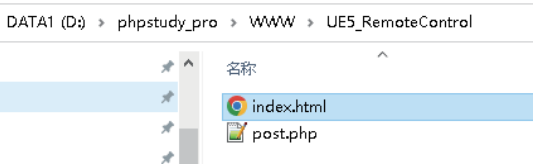

图 1-82 文件夹里有 index.html 和 post.php 两个文件

post.php 文件的全部代码内容如图 1-83 所示。Apache 服务器默认支持 PHP 语言，在 post.php 中使用 PHP 的 curl 拓展来构建一个 HTTP 请求，设置请求的数据内容和发送方式，并实际执行请求发送，最后输出请求收到的反馈。

server.js | index.html | post.php

```
1  <?php
2  //如果接口返回的数据为json，这里需要先定义数据类型为json
3  header("Content-type:application/json;charset=utf-8");
4  $url = "http://127.0.0.1:30010/remote/object/call";
5  $data=array('objectPath'=>
   "/Game/level1.level1:PersistentLevel.Cube_Blueprint_C_0",'functionName'=>
   'SetActorRotation','parameters'=>array('NewRotation'=>array('Roll'=>25,'Pitch'
   =>2,'Yaw'=>10)));
6  $json_data =json_encode($data);
7   $ch = curl_init();
8   curl_setopt($ch,CURLOPT_URL,$url);
9   curl_setopt($ch,CURLOPT_CUSTOMREQUEST,"PUT"); //以PUT的方式发送数据
10  curl_setopt($ch,CURLOPT_POSTFIELDS,$json_data);
11  curl_setopt($ch,CURLOPT_HTTPHEADER,array('Content-Type: application/json',
    'Content-Length: ' . strlen($json_data)));
12  curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
13  $respond = curl_exec($ch);
14  curl_close($ch);
15  print_r($respond);
```

图 1-83 post.php 文件的全部代码内容

如果在计算机上通过浏览器输入网址 http://localhost/UE5_RemoteControl/post.php 来访问网页，网页上会立即显示 UE5 反馈的信息，该信息为一个 JSON 格式的数据 {"ReturnValue":true}，如图 1-84 所示。

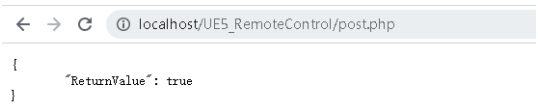

图 1-84　post.php 被调用执行后打印输出了 UE5 服务器的反馈信息

这时可以看到 UE5 中 Cube_Blueprint 蓝图对象的旋转角度确实改变了。通过浏览器直接调用的这个 PHP 程序只是最终用户界面背后的后端程序。需要定制的操作界面网页实际上是 index.html 这个前端页面。index.html 和 post.php 都可以直接用记事本打开进行编辑。

3. 通过 AJAX+PHP 自定义页面向 UE5 发送指令

接下来，可以开始定制 index.html 前端页面。在页面上放置一个 ID 为 btnA 的按钮，通过编写 JavaScript 脚本代码为这个按钮添加一个单击功能，让它被单击后能将 JSON 数据发送给 post.php。在 index.html 页中引入 jquery.js 文件，jquery 是一个 JavaScript 框架，可以让用户编写 JS 脚本的效率大大提高。发送请求的过程就是使用 jquery 框架中的 ajax 组件以 POST 的方式向 post.php 提交变量 data1 中的数据，在收到服务器反馈后，以 alert 弹窗的方式显示反馈信息中的 ReturnValue 键值，如图 1-85 所示。

```
<input type=button value="Action A" Class="btn" id="btnA" />
<input type=button value="Action B" Class="btn" id="btnB" />
<script src="https://cdn.bootcdn.net/ajax/libs/jquery/3.6.0/jquery.js"></script>
    <script>
    var data1={//用变量data1存放要提交的JSON数据
    "objectPath":"/Game/level1.level1:PersistentLevel.Cube_Blueprint_C_0" ,
                "functionName":"SetActorRotation",
                "parameters": {
                    "NewRotation": {
                        "Roll":"10",
                        "Pitch":"20",
                        "Yaw":"50"
                    }
                }
            }
        $('#btnA').on('click', function(msg){//按钮A点击后执行
                jQuery.ajax({ //采用jQuery的ajax组件
                    type: "POST", //以POST的方式发送数据
                    url: "post.php", //发送数据给post.php
                    data: JSON.stringify(data1) , //发送的数据是data1
                    success: function(r) {
                      console.log(r)
                      alert(r.ReturnValue)//提交成功后显示反馈数据中的ReturnValue
                    },
                    error(err) {
                        console.log('请求失败', err);
                     }
                });
            });
    </script>
```

图 1-85　index.html 里使用 js 脚本向 post.php 提交数据的核心代码

此时 post.php 为了能灵活接收 index.html 发来的 JSON 数据，也需要作相应的修改，具体代码如图 1-86 所示。

post.php

```php
<?php
//如果需要PHP返回的数据格式为json格式，则需要在header里如下定义
header("Content-type:application/json;charset=utf-8");
$url = "http://127.0.0.1:30010/remote/object/call";
$input_data = (file_get_contents('php://input'));
$json_data = ($input_data);
 $ch = curl_init(); //使用curl拓展来构建请求
 curl_setopt($ch,CURLOPT_URL,$url); //设置请求的网址为上面的$url变量
 curl_setopt($ch,CURLOPT_CUSTOMREQUEST,"PUT"); //以PUT的方式发送数据
 curl_setopt($ch,CURLOPT_POSTFIELDS,$json_data); //设置请求的数据内容为$json_data
 curl_setopt($ch,CURLOPT_HTTPHEADER,array('Content-Type: application/json',
 'Content-Length: ' . strlen($json_data)));
 //设置了请求的数据格式为json
 curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
 $respond = curl_exec($ch); //发送请求，并将反馈存入$respond变量里
 curl_close($ch);//关闭curl拓展
 print_r($respond);//打印输出反馈
```

图 1-86　post.php 修改后的全部代码内容

这样，通过手机浏览器访问页面 http://192.168.3.44/UE5_RemoteControl/index.html，然后单击页面上的按钮 A，就实现了使用自己设计的网页通知 UE5 中的物体改变属性了！本节讲解的这个流程是利用 HTML 网页请求 PHP 后端程序，由 PHP 向 UE5 内置服务器发送数据，实现远程控制，这个简易的流程介绍主要是希望能起到抛砖引玉的作用，让读者可以在自己的项目中根据项目实际情况定制自己所需的控制界面，从而完美高效地解决远程控制问题。

1.2.3　使用Remote Control Web Interface

如果觉得自己制作开发页面比较麻烦，那么可以认识一下 Remote Control Web Interface 这个 UE5 插件。它能为用户提供一套非常方便的 UI 界面，让用户轻松地借助网页上的控件来操控 UE5 中的对象细节，满足大部分需要远程控制的场景。首先需要在 UE5 项目的 Plugins 菜单里勾选 Remote Control Web Interface 插件并重启 UE5 项目，如图 1-87 所示。

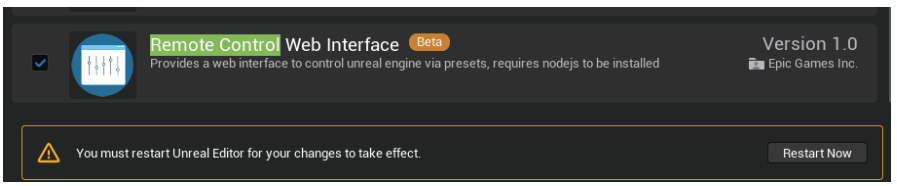

图 1-87　启用 Remote Control Web Interface 插件

需要注意的是，用户需要确保自己的计算机上已经安装了 Node.js，没有的话可以直接从 Node.js 官网下载 Windows 版安装程序，然后双击运行安装即可。在 UE5 的 Content Browser 里的空白处右击，从弹出菜单的 Remote Control 里选择创建一个 Remote Control Preset（远程控制预设），取名为 NewRemoteControlPreset，如图 1-88 所示。

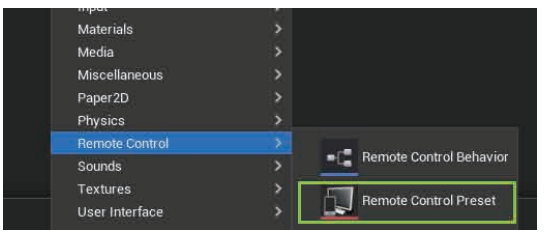

图 1-88　创建一个 Remote Control Preset

双击新建的 NewRemoteControlPreset 对象，会弹出 Remote Control（远程控制）窗

口，在窗口左侧有 Expose 面板和 Details 面板，如图 1-89 所示。

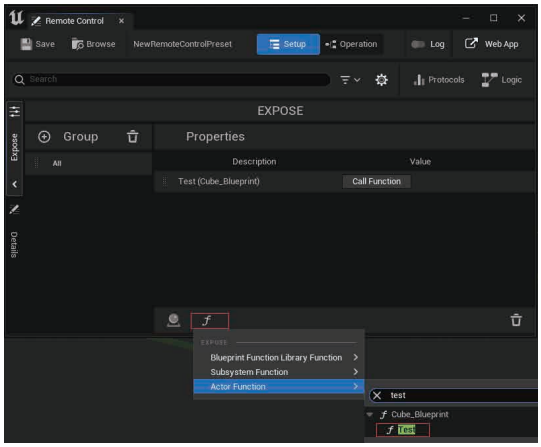

图 1-89　Remote Control 窗口

在 Remote Control 窗口左侧的 Expose 面板中，单击底部的 f_{Test} 按钮可以加入某个 Actor 蓝图里的函数。例如，在上一节中的 Cube_Blueprint 蓝图里所建立的 Test 函数，如图 1-89 中的红框标示所示。

接下来单击右上角的 WebApp 图标，如图 1-90 所示。UE5 会打开浏览器，呈现一个用于自定义构建 UI 界面的网页，可以单击其中的 Build Your Own UI 按钮进一步构建用户自己想要的界面，如图 1-91 所示。

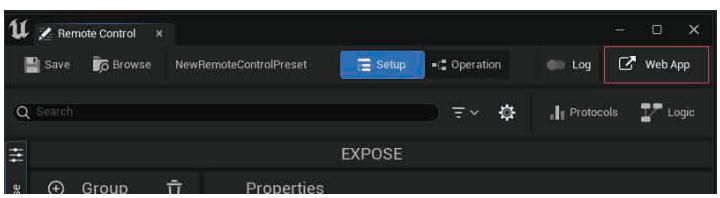

图 1-90　在 Remote Control 窗口里单击 Web App 按钮

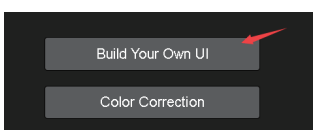

图 1-91　UE5 内置的 Web App 可用于自定义遥控界面

然后可以从左侧的 Properties 标签下找到 Test 条目，把它拖动到右边的区域中，右边区域就是用户最终可以使用的网页界面。具体操作如图 1-92 所示。

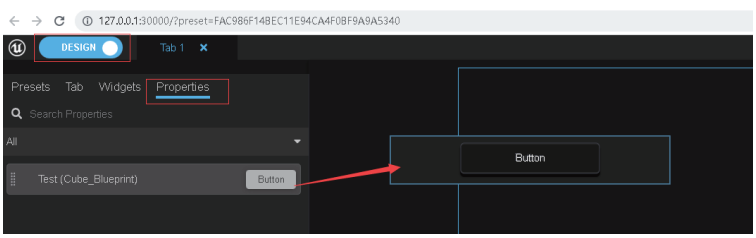

图 1-92　将需要控制的条目拖入编辑区

此时就可以在计算机浏览器地址栏里输入 http://127.0.0.1:30000 来访问这个控制界面了。单击页面中的 Test 按钮就能执行 Cube_Blueprint 对象中 Test 函数内所设定的蓝图内容，实现了把网格体放置到了 x、y、z 坐标值均为 100 的位置上，如图 1-93 所示。

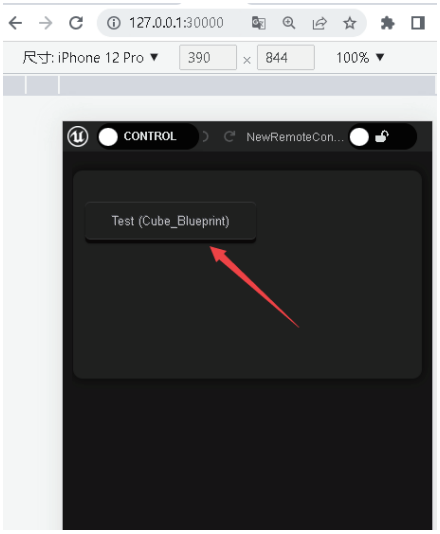

图 1-93　单击页面中的按钮远程执行函数 Test

如果要用手机来操作，当然需要把手机浏览器网址里的 127.0.0.1 换成计算机在局域网内的 IP 地址。且手机和计算机需要在同一个局域网内，也就是连着同一个 Wi-Fi。以上这种方法的一个好处就是不论 UE5 是已经处于运行状态还是尚在编辑状态，都可以用这个网页界面操控 UE5 视口中的物体。

通过这种方式可以远程调用蓝图中的函数，也可以直接远程控制 UE5 中物体的其他属性。接下来在 UE5 场景中放入一个 Point Light（点光源），如图 1-94 所示。

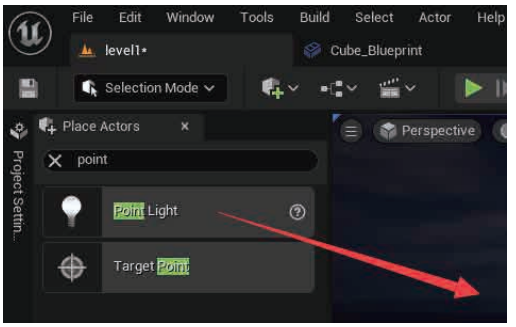

图 1-94　在视口里拖入一个点光源

在确保 Remote Control 窗口处于打开状态的同时，查看 Point Light 的 Details（细节）面板，可以看到它各项属性的右侧都有一个白点图标（三个小白点呈纵向排列），如图 1-95 所示。

图 1-95　单击属性右侧的白点图标把属性开放给远程控制

如果把白点图标的列宽拉大一些，可以看到它其实就是眼睛图标，这个图标的作用是 Expose property（开放属性）。单击 Expose property 可以把左侧对应的属性开放给 Remote Control，从而实现对该属性的远程控制。如果单击这个点光源的 Intensity（强度）和 Light Color（灯光颜色）两个属性，那么这两个属性就可以在 Remote Control 里被远程控制了，如图 1-96 所示。

此时如果单击 Remote Control 窗口右上角的 Web App 按钮（如图 1-90 所示），便可以看到浏览器里的 Properties 标签下多了这两个被开放的属性，如图 1-97 所示。

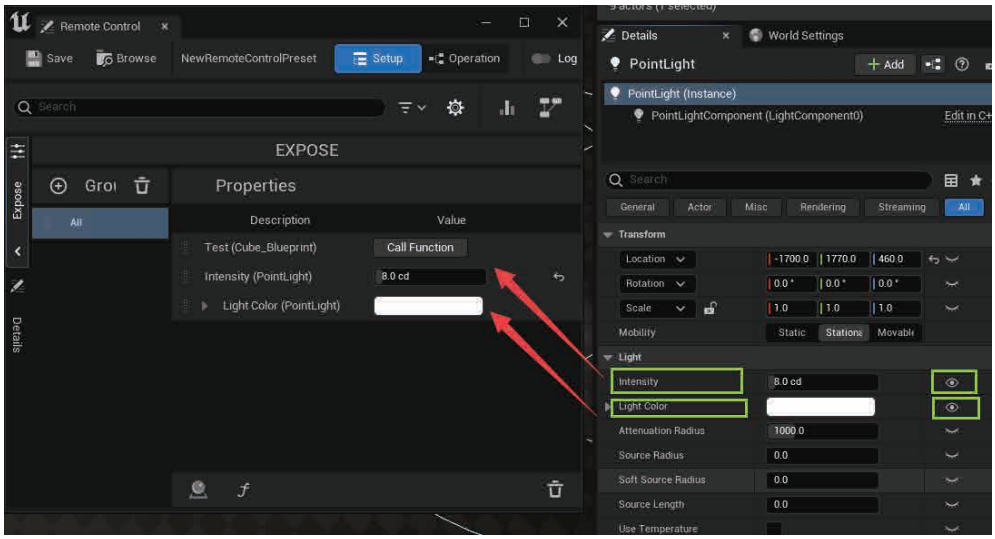

图 1-96　把两个属性开放给远程控制后看到 Remote Control 窗口里的变化

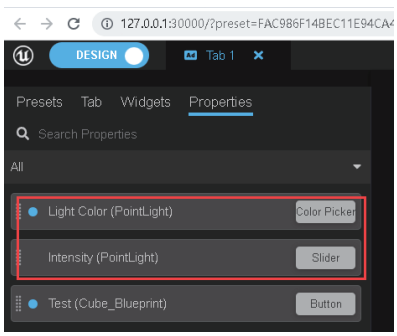

图 1-97　Web App 里也会增加这两个属性

我们可以把这两个条目拖入右侧编辑区域并调整好布局。Web App 里控制属性的组件有多种，如 Color Picker（颜色择取器）、Slider（滑动条）、Dial（表盘）和 Button（按钮）等，如图 1-98 所示。

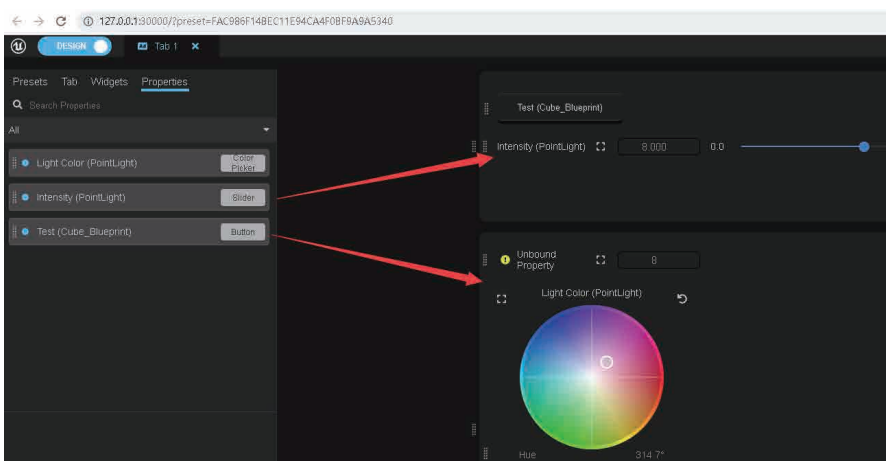

图 1-98　把这两个属性拖入右侧编辑区

此时在手机上通过浏览器打开这个网页，在网页上触摸移动滑块就可以调整 UE5 中灯光的亮度了。而触摸网页上的色盘控件，就能够很方便地改变 UE5 中灯光的颜色，如图 1-99 所示。

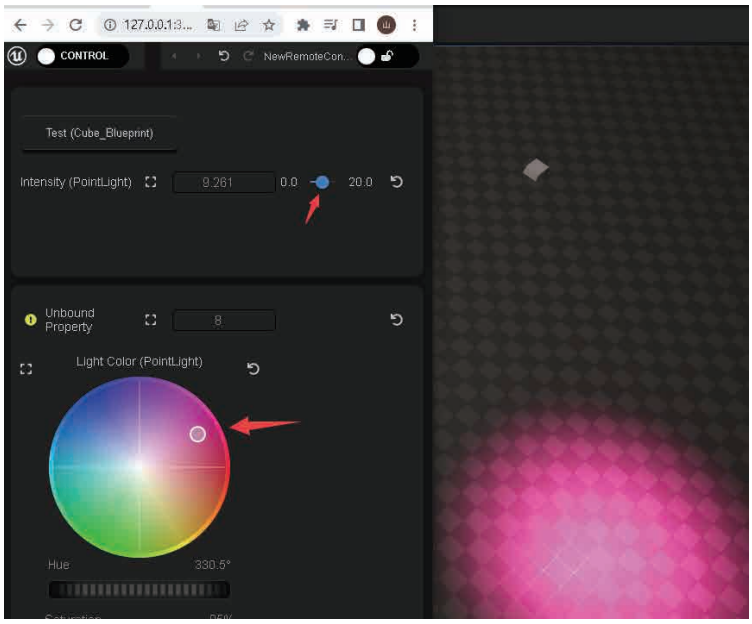

图 1-99　拖动滑块和单击色盘分别可以调整灯光的强度和颜色

注意请确保手机和计算机处于同一个 Wi-Fi 下，而且手机打开的网址里需要把 192.168.3.44 替换为用户自己计算机所在局域网内的 IP 地址。笔者打开的网址是 http://192.168.3.44:30000/。打开网页后，可以修改属性对应的控件类型。例如，可以将 Intensity 属性所对应的控件的类型从 Slider 改为 Dial，如图 1-100 所示。

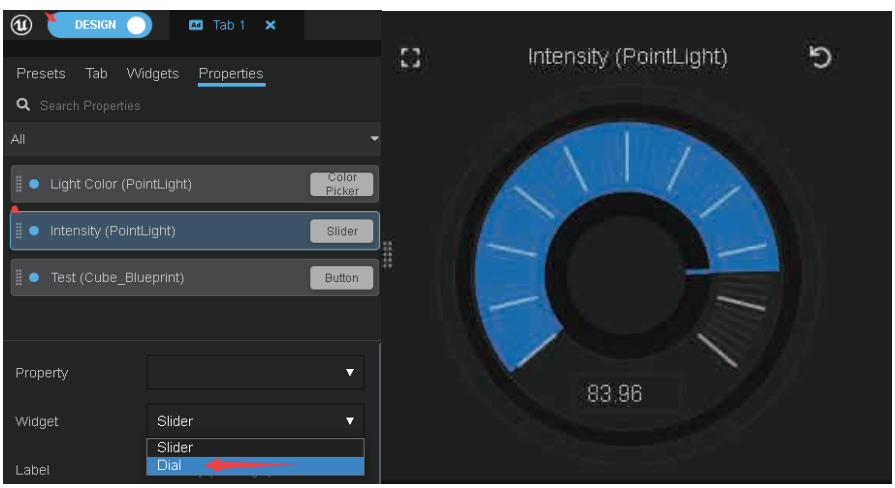

图 1-100　将 Intensity 属性对应的控件类型从 Slider 改为 Dial

这样就可以用拨动表盘的方式来触摸式控制光源的亮度了。结合手机触摸屏，利用各类控件调节灯光的颜色和亮度，感觉是不是相当顺畅而方便呢？

1.2.4　实例：用iPad切换机位并调焦

本实例将采用 iPad 来遥控 UE5，具体而言是在平板设备上通过对网页的触摸操作来切换 UE5 中的摄像机，并控制调整摄像机的焦距和光圈。

继续使用上一节中用到的 UE5 项目文件 RemoteCtrlAPIDemo，在 Content Browser 里新建一个文件夹 Level_Live，然后在这个文件夹里新建一个基础的关卡 Level2。在关卡 Level2 的场景中添加一个 Cube（立方体）并加入两台摄像机，两台摄像机分别取名 Camera1 和 Camera2，一台平视着立方体，另一台则俯视着立方体，如图 1-101 所示。

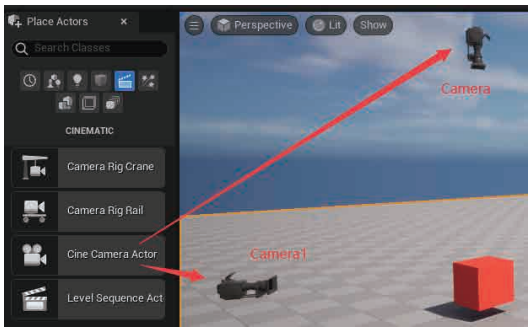

图 1-101　添加两个 Cine Camera Actor

在关卡蓝图中建立两个函数 ViewCamera1 和 ViewCamera2，函数的内容分别是将 Camera1 和 Camera2 设置为当前目标视角。切换镜头视角使用的蓝图节点是 Set View Target with Blend，如图 1-102 所示。

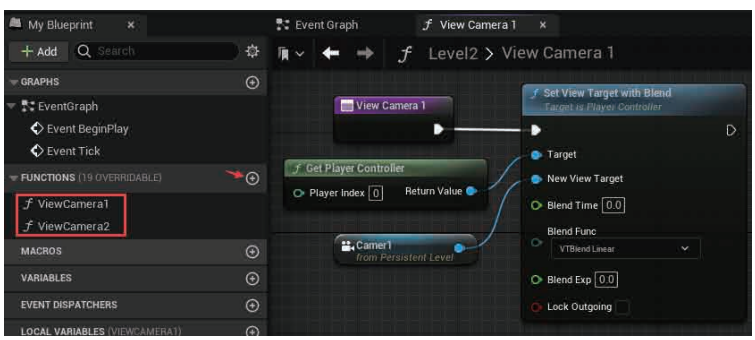

图 1-102　在 Level2 关卡里添加两个函数用于切换机位

在文件夹 Level_Live 里再创建一个 Remote Control Preset（遥控预设），取名为 RC_Preset2，双击打开后可以将 Level2 里两个自定义的函数添加到 Remote Control 里。具体操作如图 1-103 所示。

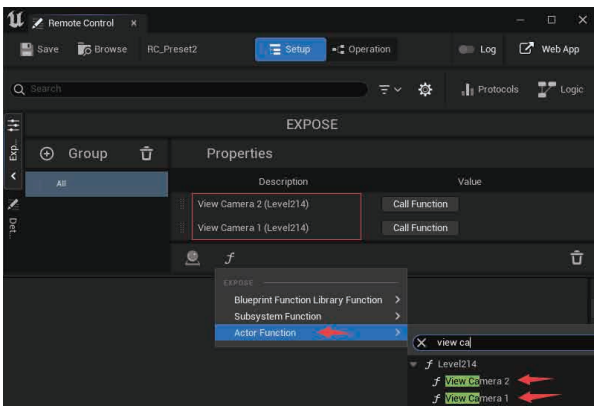

图 1-103　将两个函数开放给远程控制

同时将 Level2 里的 Camera1 选中，从其细节面板里将焦距和光圈这两个属性开放给远程控制，这样在 Remote Control 窗口中就又增加了这两个属性条目，如图 1-104 所示。

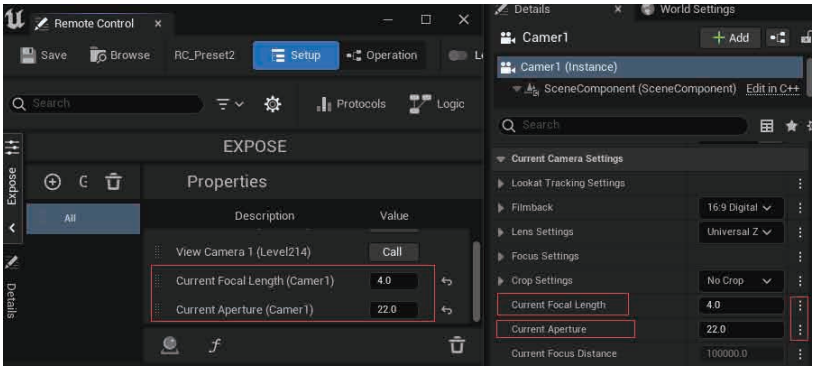

图 1-104　将 Camera1 的焦距和光圈属性开放给远程控制

保存 Remote Control 后打开 Web App，将几个开放给远程控制的属性条目拖到页面区域中，如图 1-105 所示。

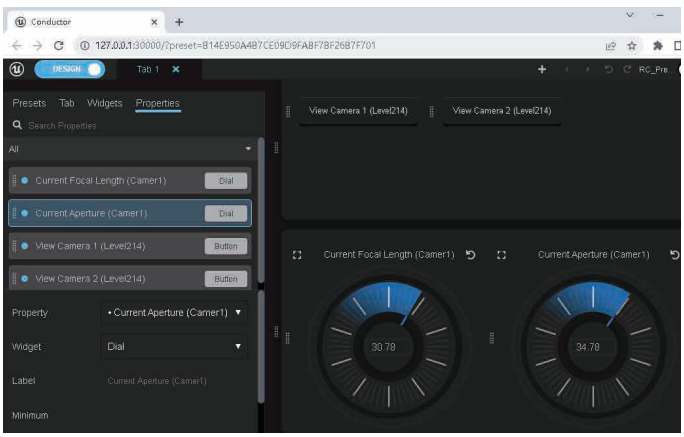

图 1-105　将 4 个条目都拖入 Web App 的右侧页面区域

接下来运行 UE5 的关卡 Level2，然后用 iPad 通过浏览器访问 http://192.168.3.44:30000，这样就可以开始在 iPad 上通过网页来操作切换 UE5 中的摄像机位了，当切换到了 Camera1 时还可以通过网页下方的两个拨盘控件来控制摄像机的焦距和光圈变化，如图 1-106 所示。

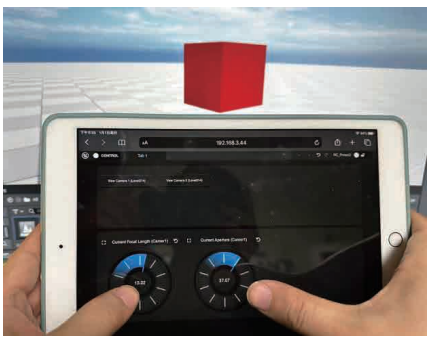

图 1-106　用 iPad 远程控制 UE5

本实例的详细操作步骤可以通过扫描下方二维码来观看。

1.3　DMX 管控 UE5 数字灯光系统

DMX 是 Digital MultipleX 的缩写，意为多路数字传输。DMX512 控制协议是美国舞台灯光协会（USITT ）于 1990 年发布的灯光控制器与灯具设备进行数据传输的工业标准，包括电气特性、数据协议、数据格式等方面的内容。每一个 DMX 控制字节叫作一个指令帧，称作一个控制通道，可以控制灯光设备的一项或几项功能。基于 DMX512 控制协议进行调光控制的灯光系统叫作数字灯光系统。目前，包括电脑灯在内的各种舞台效果灯、调光控制器、控制台、换色器、电动吊杆等各种舞台灯光设备，在全面支持 DMX512 协议的基础上，已全面实现调光控制的数字化，并在此基础上，逐渐趋于计算机化、网络化。因此，对于影视灯光设计与操作人员，理解 DMX512 控制协议的程序结构、控制原理及其应用要点是十分必要的。

最近，在世界各地的举办的各类现场活动中，使用 UE 驱动数字装置为现场增色的案例和需求不断增加。市面上已经出现了越来越多完全依靠 UE 或局部借助 UE 来实现的案例。当 UE 刚刚进入这个市场时，它缺少了某些基础性的可用特性。尽管如此，还是有很多富有创意与激情的人士选择采用 UE 并为之构建了所缺失的插件。利用这些插件，人们可以进一步拓展 UE 带来的可能，实现他们想要的创意目标。

因此，Epic 引入了对 DMX 数据通信的支持，包括 Artnet 和 sACN。Artnet 和 sACN 是允许通过以太网 IP 地址聚合和发送 DMX 数据的网络协议。Artnet 允许通过一根网线发送 32768 个 Universes（数据域）。虽然这是一个较为早期的协议，但它得到了更多设备以及设备的支持。sACN（用于控制网络的流式架构）目前似乎更受欢迎，它允许用户在一根网线上运行 63999 个数据域的 DMX 数据。DMX 在整个行业中用于控制现场活动行业中的各种设备，如照明设备、激光器、烟雾机、机械设备等。

1.3.1　搭建DMX Library配置设备信息

为了能使用 UE5 自带的各类 DMX 灯具素材打开 UE5，从 FILM/VIDEO&LIVE EVENTS 类别里选择 DMX 这个模板来建立新项目。这样，项目中就会内置很多 DMX 灯具可供后续使用，如图 1-107 所示。

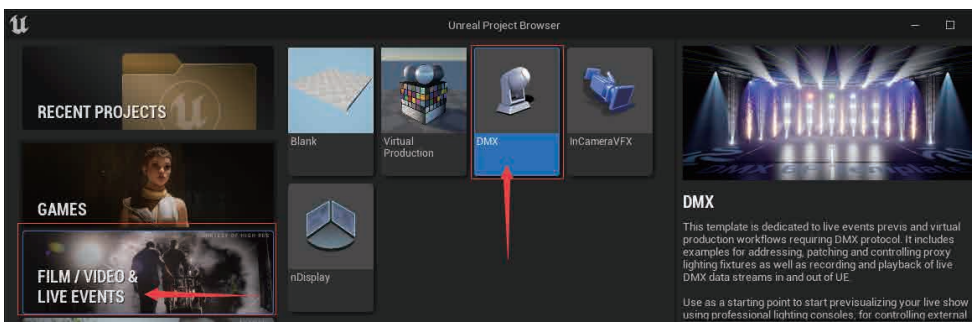

图 1-107　使用 DMX 模板构建 UE5 新项目

在 UE5 的插件设置里，需要把 DMX Engine DMX Fixtures DMX Protocol 等五个插件都进行勾选来启用它们，然后重启 UE5，如图 1-108 所示。

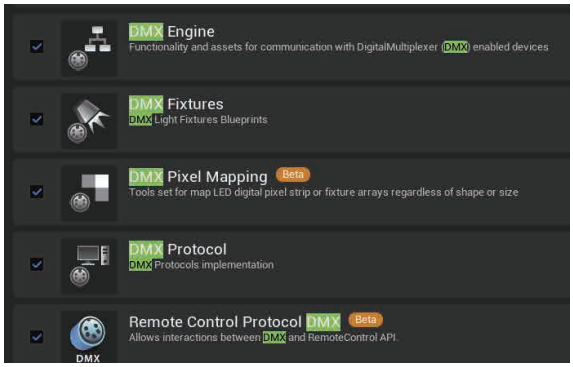

图 1-108　启用 DMX 相关插件

在 Content Browser 里的空白处右击，从弹出菜单里选择 DMX → DMX Library 命令创建一个 DMX Library（DMX 库）对象，将它取名为 NewDMXLibrary。这个库可以用于管理 UE5 项目中用到的各类灯具的具体配置，如图 1-109 所示。

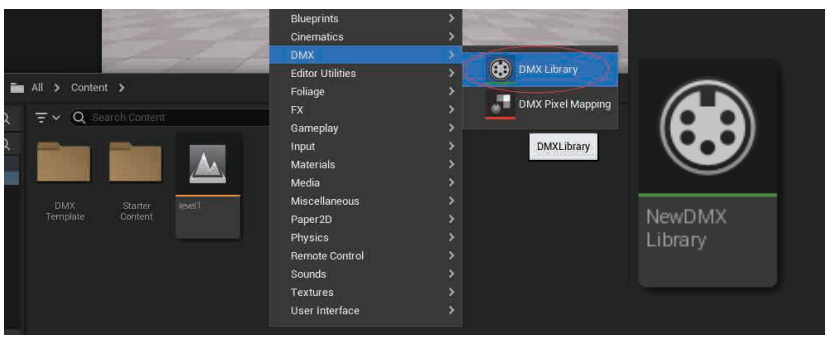

图 1-109　创建一个 DMX Library

双击打开它以后，在 Fixture Types（灯具类型）标签下先添加一个名为 BEAM 的属于 Moving Head（摇头灯）的灯具类型，如图 1-110 所示。

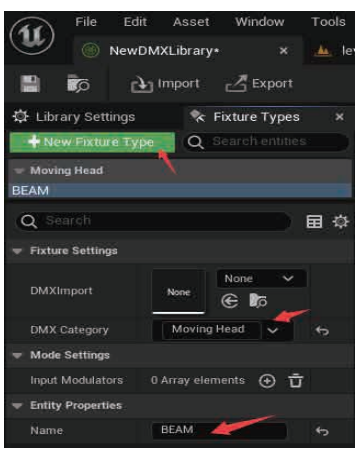

图 1-110　添加灯具类型

然后在右侧为它添加模式，设置信道的数量为 4 个，然后再向右边是设置每个 Channel（信道）的功能，如信道 1 是 Dimmer 功能，对应灯具的开启和关闭，如图 1-111 所示。

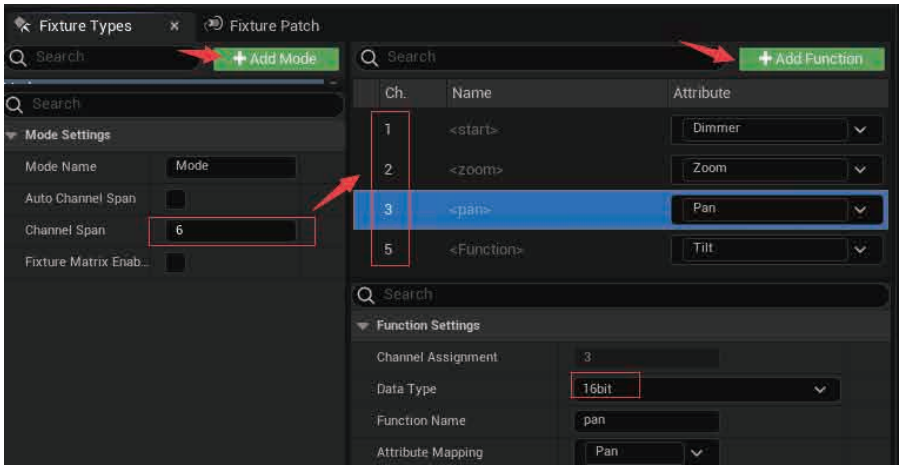

图 1-111　添加灯具类型所对应的模式和各信道的功能

当某个功能的 Data Type 是 16bit 时，通常就会占用两个信道。所以可以看到，在图 1-111 中红框里标识的 1、2、3、5 是不连贯的，其实 3 表示的是 3、4 两个信道，而 5 表示的是 5、6 两个信道。只占一个信道的功能是 8bit 的数据类型。接下来在 Fixture Patch 标签下添加两个具体的灯具，分别命名为 BEAM_1 和 BEAM_2，如图 1-112 所示。

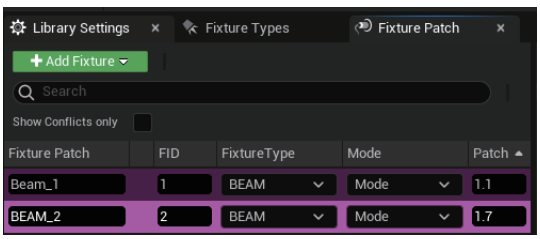

图 1-112　添加两个具体的灯具，都是新建的 BEAM 类型

图 1-112 的最后一列是自动生成的。例如，BEAM_1 的 Patch 内容 1.1 表示它的信道是从第 1 个 Universe 的第 1 个 Channel 开始，由于 Mode 里设置的 BEAM 类型的灯具拥有 6 个信道（#1~#6），所以 BEAM_2 的 Patch 是从第 7 个信道开始，Patch 为 1.7 表示从第 1 个 Universe 的第 7 个信道开始，实际它拥有 #7、#8、#9、#10、#11、#12 这六个信道。每个信道都可以收取 0~255 的数据。Universe 是信号域的意思，每个 Universe 信号域可以包含 512 个 Channel（信道），如图 1-113 所示。

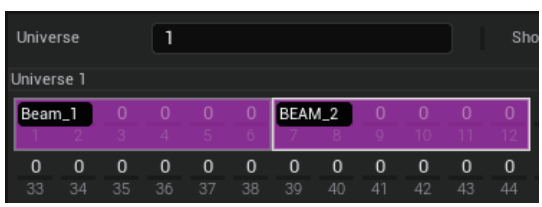

图 1-113　每个灯具具体的信道占用情况包括数据域和信道的编号

以上这些操作在读者初次接触 DMX 时可能会感到不解，实际上这些操作是在为不同的灯具分配不同的信道编号，这样会方便后续往不同的信道发送数据信息时，UE5 可以清楚地知道如何将信息依据信道来源分配给哪个灯具，通知灯具去执行哪个动作，是关灯还是摇摆灯头还是缩小光圈等。不同的灯具具有不同的功能列表。例如，有的灯具类型可以让灯光变色，有的灯具可以 Shutter（频闪），还有的灯具可以实现 Gobo（剪影）等。

1.3.2 用DMX来调控物体的转速

配置好 DMXLibrary 以后，可以使用 DMX 来驱动 UE5 中的一些数据参数，从而营造出动态效果。笔者在本节通过构建一个带参数的材质，使用 DMX 来改变材质参数，从而体现视觉上的动态变化。

1. 构建带参数的材质实现动态效果

在 Content Broswer 里右击空白处，从弹出菜单中选择 Materials → Material 命令可以建立一个材质，取名为 Cylinder_Mat，如图 1-114 所示。

图 1-114 新建材质

双击打开这个材质，进入材质编辑器。设置具体内容如图 1-115 所示。

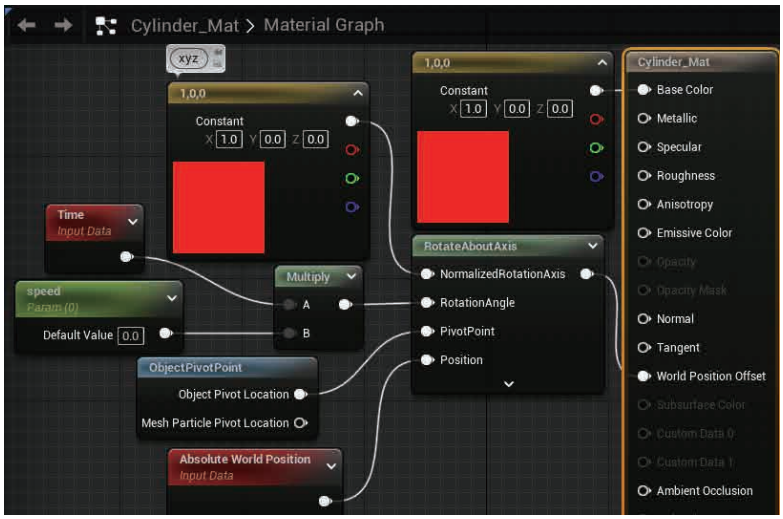

图 1-115 设置材质节点，与蓝图很类似

可以大致理解一下这个材质图，它的左侧有一个叫作 speed 的 param（参数），它使用了 World Postion Offset（世界位置偏移）这个引脚，可以让材质具有动感，让物体可以产生移动起来的感觉（通常草丛或树叶的微动都采用这个做法）。随着 Time（时间）节点值的不断增加，RotateAboutAxis 让 World Postion Offset 旋转起来，转速取决于 speed 参数的大小。RotateAboutAxis 转动的轴心由左上角的 (1,0,0)，指定为 x 轴，如果是 (0,1,0) 就表示为 y 轴，依此类推。PivotPoint 是指旋转点，相当于物体的注册点。

材质就绪后，可以建立一个 Actor 蓝图对象，取名为 Cylinder_Actor，双击打开这个 Actor，在里面添加一个 Cylinder（圆柱体），如图 1-116 所示。

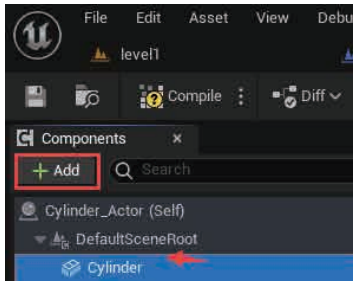

图 1-116 在组件面板里添加 Cylinder 静态网格体组件

选中 Cylinder，在它的 Details 面板里为 Cylinder 设定材质，将材质指定为上面

新建的 Cylinder_Mat，如图 1-117 所示。

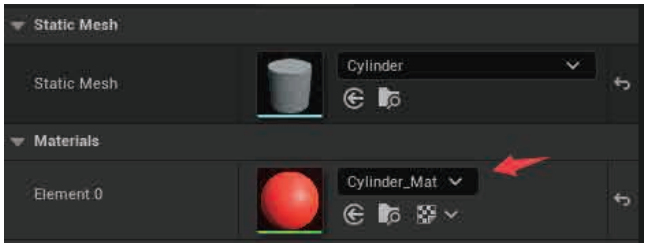

图 1-117　设定材质为 Cylinder_Mat

接着在 Cylinder_Actor 的蓝图里写入如图 1-118 所示的内容，这部分蓝图的含义是设置 Cylinder 变量材质里的 speed 参数值为 1。这样，借助这个材质，Cylinder 就能以相应的转速转动起来了。

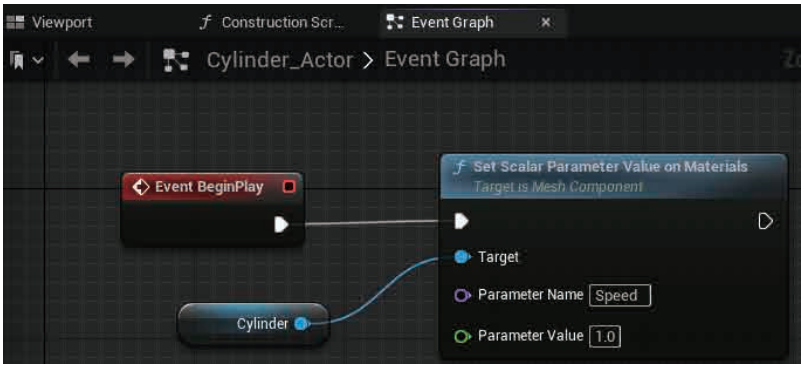

图 1-118　通过蓝图设定材质里的参数值

编译蓝图后，从 Content Browser（内容管理器）里拖曳一个 Cylinder_Actor 放入视口中，如图 1-119 所示。

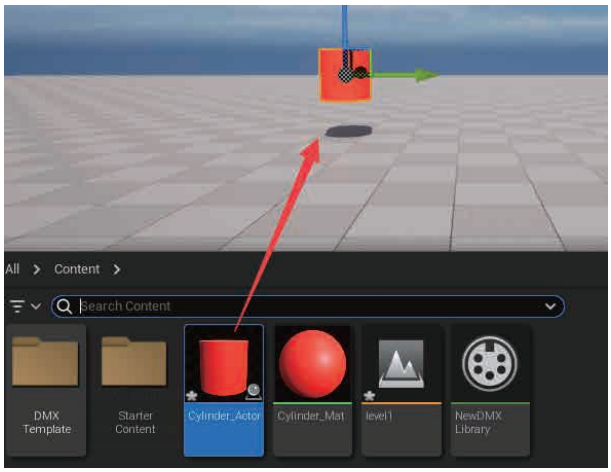

图 1-119　从 Content 里拖曳一个 Cylinder_Actor 放入视口中

运行 UE5，可以看到红色的圆柱 Cylinder 确实转动起来了。接下来要做的就是让它的转动速度由 DMX 数据来驱动！

2. 添加 DMX 组件接收 DMX 数据

保存关卡后，需要在这个 Cylinder 里再加入一个 DMX 组件，在这个 DMX 组件的 Details 面板里设置它的 DMXLibrary 为前面创建的 NewDMXLibrary，并设置 Fixture Patch（灯具装配）为 Beam_1，如图 1-120 所示。

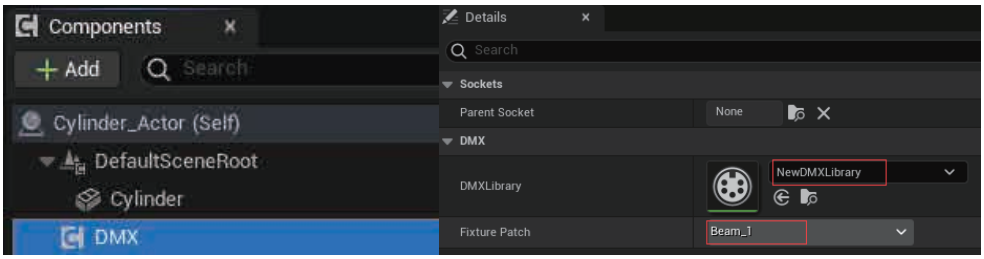

图 1-120　添加 DMX 组件并设置组件对应的具体灯具

保持 DMX 组件处于选中状态，从它的细节面板里单击 On Fixture Patch Received 事件右侧的加号按钮，添加灯具接收 DMX 信号的事件，如图 1-121 所示。

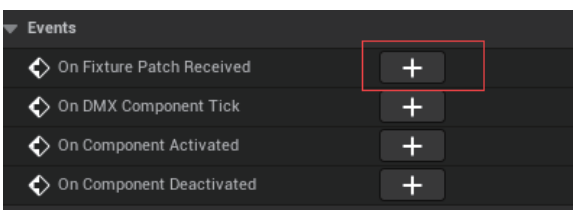

图 1-121　添加 On Fixture Patch Received 事件

这个事件表示灯具在自己所占用的信道上收到 DMX 数据会触发的事件。用户可以在该事件节点下添加蓝图内容，如图 1-122 所示。

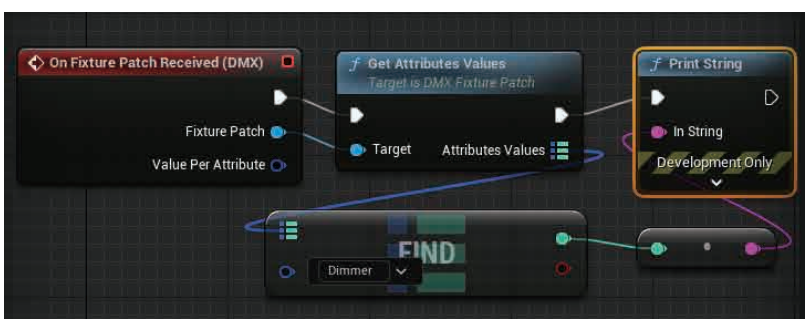

图 1-122　收到 DMX 数据后打印数据里的 Dimmer 属性信息

这部分蓝图内容表示：当灯具收到 DMX 数据时，会从 DMX 数据里找到 Dimmer 属性对应的信息并打印输出。编译蓝图后，运行 UE5，从视口上并不会看到任何打印输出内容，这是因为还没有任何 DMX 信号传送到 UE5 来。为了方便测试，可以先通过 DMX Output Console（DMX 输出控制台）模拟构建 DMX 数据信号输送给 UE5，如图 1-123 所示。

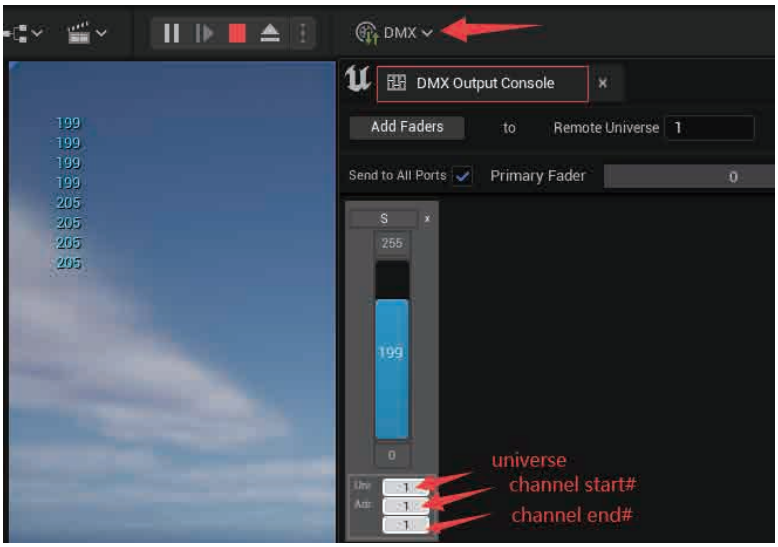

图 1-123　在 DMX Output Console 里单击 Add Faders 添加调节器

单击 Add Faders 按钮就可以添加一个调节器，调节器底部的三个数字输入区分别表示 Universe（数据域）以及起始信道和结束信道，即表示这个调节器是在发送针对哪些信道的信号。用鼠标拖动调节器上蓝色的滑块区域就可以模拟发送 0~255 的数值到指定信道。当看到视口中有数据打印出来时，就表示 DMX 数据被蓝图接收到了，因为从上一节内容可以看到第一个 Universe 里的信道 1~ 信道 6 都被分配给了 Beam_1 这个灯具，而从 Beam_1 所属的灯具类型 BEAM 的设定里可以看到信道 1 对应的是灯具的 Dimmer 属性。

调节器顶部的名称字符是可以自定义改写的，这里改为 Dimmer 比较合适，方便以后查看时能清晰地知道这个调节器是用来控制什么属性的。接下来可以进一步修改蓝图内容，如图 1-124 所示。

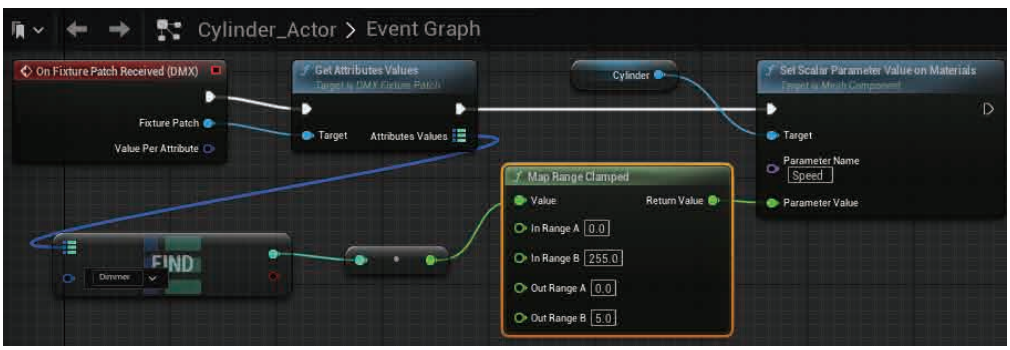

图 1-124　将 DMX 数据与材质参数值关联起来

这样发送来的 Dimmer 属性的数据就被用于控制材质参数 speed 的大小变化了。这里用到了一个 Map Range Clamped 的方法，把收到的 0~255 的数据按比例转化到了 0~5 的数据，这样 speed 值最终就不至于变得太大。

编译后再次运行 UE5 并调节 DMX 输出控制台中的蓝色滑块的值，就可以控制 Cylinder 转动的速度快慢了，如图 1-125 所示。

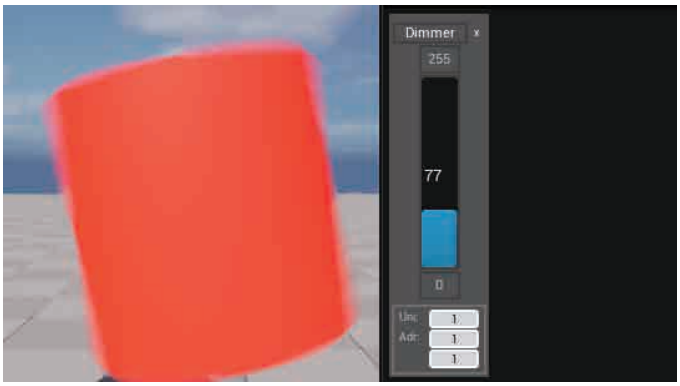

图 1-125　通过 DMX 数据的调节来控制物体转速

3. 使用 DMX 驱动灯具的各项功能

由于这个 UE5 项目是基于 DMX 模板创建并启用了 DMX Fixtures 插件，所以在图 1-126 所示的路径里可以找到很多灯具和器材。如果没有看到这个路径，可以单击 Content Browser 区域右侧的齿轮图标 Settings 然后勾选 Show Engine Content（显示引擎内容）和 Show Plugin Content（显示插件内容）选项。

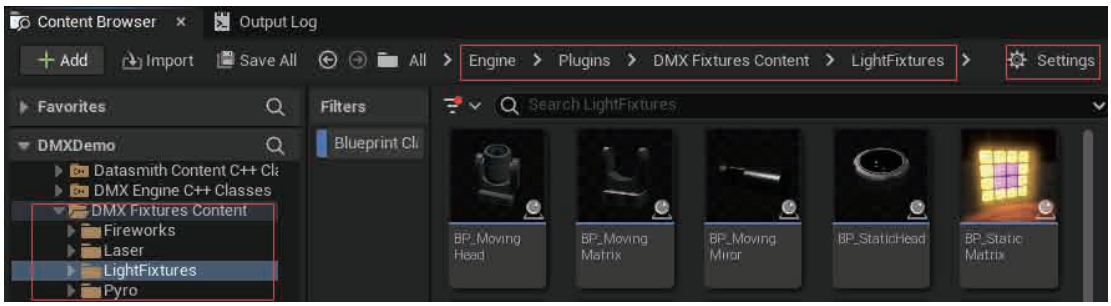

图 1-126　查看插件文件夹里 DMX Fixtures Content 的灯具文件夹 LightFixtures

LightFixtures 文件夹里有多种不同的灯具类型，如图 1-127 所示。

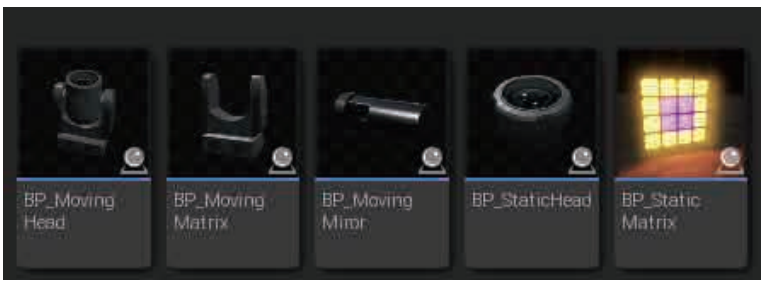

图 1-127　几种比较常见的灯具（摇头灯、矩阵灯、静态头灯）

可以从中拖曳一个 BP_MovingHead 对象放到关卡 Level1 的场景中，设置它的 DMX 库为 NewDMXLibrary，把 Fixture Patch（灯具装配）指定为 BEAM_2 这个灯。基于之前在 NewDMXLibrary 里的设置，BEAM_2 占用了 6 个信道。也就是说 #7、 #8、#9、#10、#11、#12 这六个信道里的数据将会影响这个摇头灯，如图 1-128 所示。

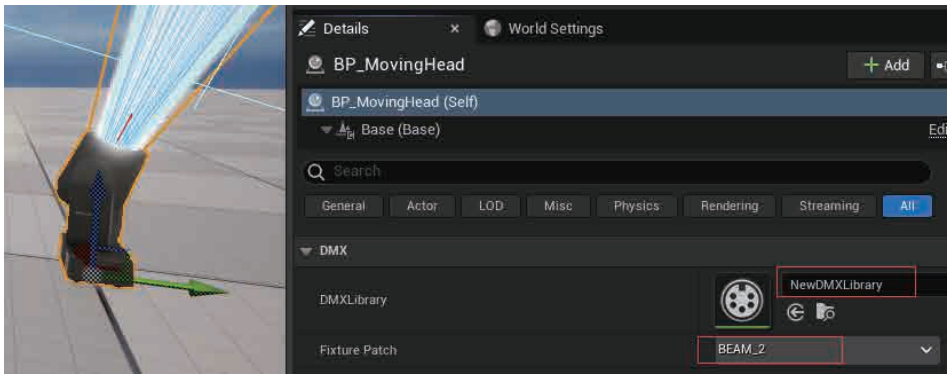

图 1-128　设置摇头灯对象的 DMX 属性，实际是为它分配信道

为了便于观察，可以把场景中的日光 DirectionalLight 调暗一些，并将灯光颜色设定为暗一些的颜色，具体设置如图 1-129 所示。

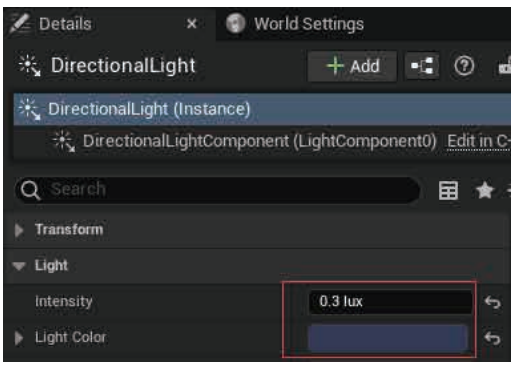

图 1-129　调暗场景中的 DirectionalLight

在 DMX 输出控制台里添加一个调节器，按图 1-130 所示进行设置，这样就可以拖动蓝色滑块来控制灯的开启与熄灭了。因为 7 号信道正是 BEAM_2 这个灯占用的第 1 个信道。也就对应了 BEAM_2 这个灯的 Dimmer 属性。而 Dimmer 属性就是灯的开关功能。

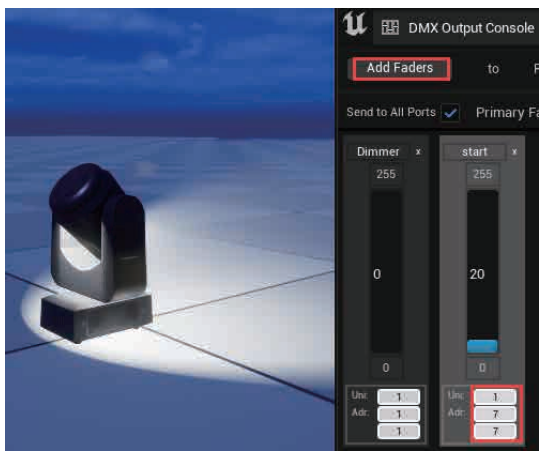

图 1-130　增加一个调节器控制 7 号信道

我们可以回顾之前在 NewDMXLibrary 里设置的各信道对应的属性，理解 7 号信道是 BEAM_2 这个灯的第一个信道，即 Dimmer 属性对应的信道，如图 1-131 所示。

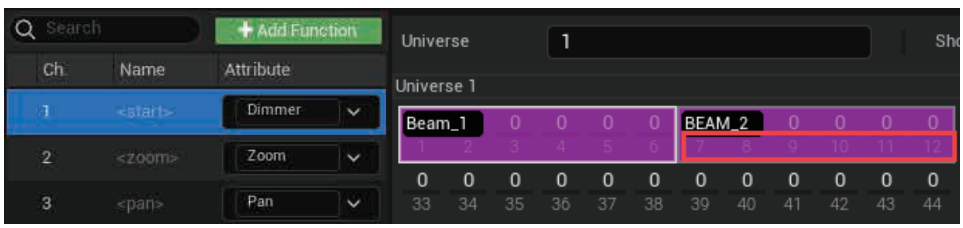

图 1-131　回顾 7 号信道的分配归属是 BEAM_2 这个灯具

依次再增加 3 个调节器，可以分别控制这个摇头灯的其他几个功能。于是可以通过 DMX 信号让这个灯抬头、摆头，以及调节光柱的粗细，如图 1-132 所示。

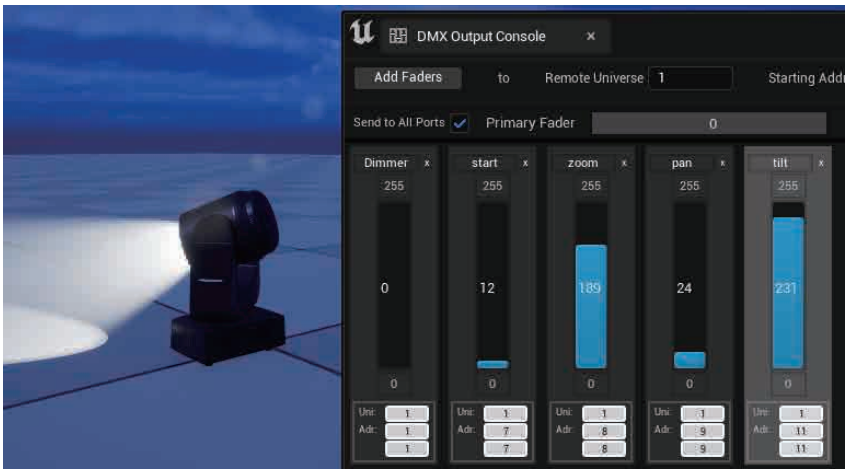

图 1-132　用 DMX 控制 BEAM_2 的四个属性 Dimmer、Zoom、Pan、Tilt

4. 基于 DMX 构建 Sequence 动画

如果想在 UE5 里制作一段动画来表现灯具的变幻，可以单击 Add Level Sequence 来建立 Sequencer 时间轴动画，在 Sequencer 里单击 Track 按钮找到用户建立的 NewDMXLibrary，选中它添加进来，如图 1-133 所示。

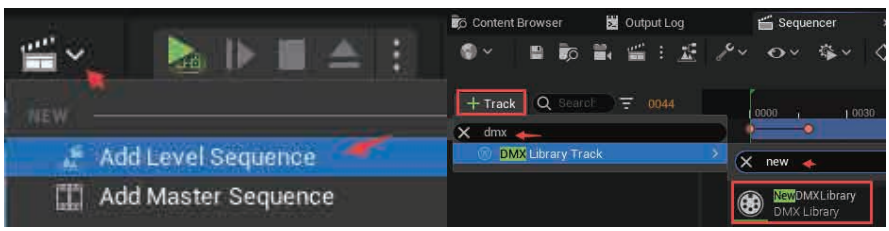

图 1-133　创建 Level Sequence 添加 DMX Library Track

这样在 Sequencer 面板的左侧就可以看到这个 DMX Library 以及它所涉及的各个灯具和每个灯具所包含的各种属性。为这些属性添加对应的关键帧就可以构建一段灯光秀动画，如图 1-134 所示。

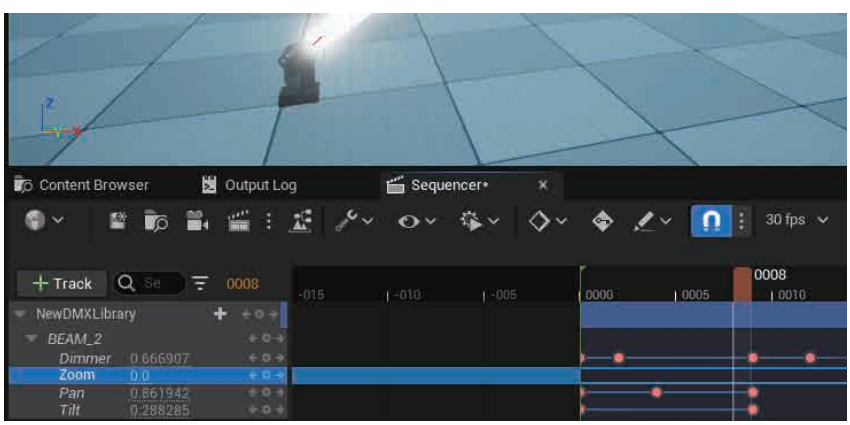

图 1-134　在 Sequencer 里为某个灯添加各属性的关键帧动画

5. 使用蓝图发送 DMX 数据

如果想通过蓝图来发送 DMX 数据，也是可以实现的。在关卡蓝图里写入如图 1-135 所示的内容，表示在 1 号信号域里分别向 1 号信道发送数字 5，向 7 号信道发送数字 60，向 11 号信道发送数字 100。

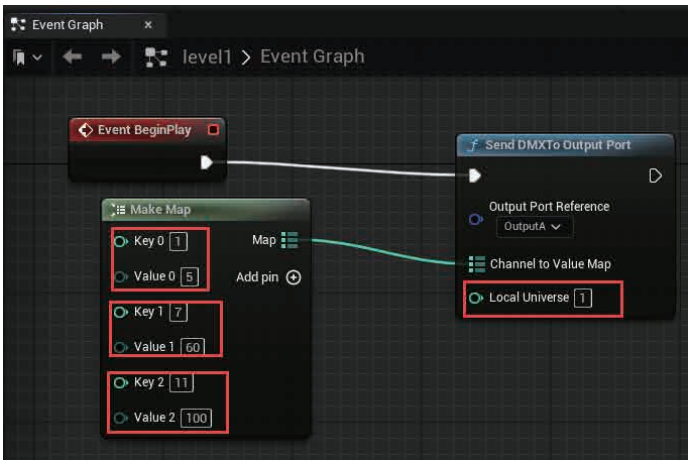

图 1-135　用蓝图发送 DMX 数据，给不同的信道发送不同数值

也可以换个写法，如图 1-136 所示的做法就是直接向指定的灯具发送 DMX 数据，向 BEAM_2 灯具的 Tilt 功能所对应的信道发送数值 200。

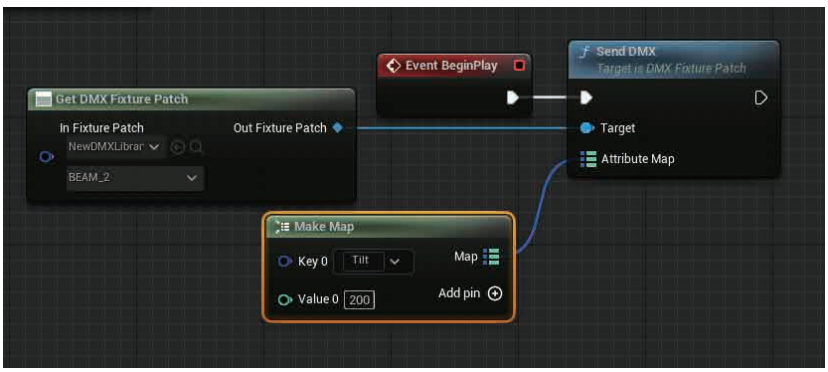

图 1-136　用蓝图发送 DMX 数据的另一种写法

掌握了这些技术细节后，再结合前面的 WebSocket 远程控制技术，也就可以实现远程发送 DMX 数据来遥控虚拟灯具了。

1.3.3　使用外部软件来发送DMX给UE5

数字多路传输（Digital Multiplex，DMX）是广泛用于控制现场灯光、声效、烟花、动画演播等内容的数据通信标准。UE5 拥有了诸多 DMX 插件后，用户可通过更专业的控制台硬件设备来控制现场活动的各种声光效果，并且可以将 UE5 中的虚拟灯光秀与真实的灯光设备联动起来。如图 1-137 所示的是两种控制 DMX 灯具的控制台设备。

图 1-137　各类 DMX 控制台硬件

控制台硬件设备可以同时控制真实的数字灯光设备和 UE5 里虚拟的灯光设备。请参看图 1-138 中展示的样子。

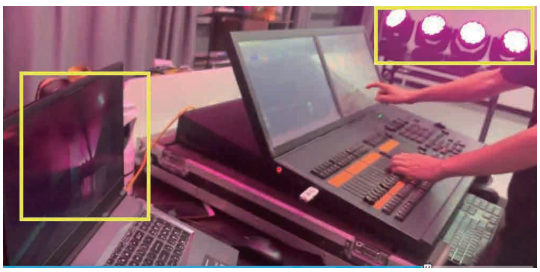

图 1-138　DMX 控制台硬件

它可以同时控制 UE 虚拟灯具和外部真实灯具，也可以在同一个局域网内的其他计算机上通过控制软件发送 DMX 数据来实现更为复杂的 DMX 数据流控制，从而得到更为精彩的虚拟灯光秀。这样的 DMX 控制台软件有很多，如知名的 grandMA、Resolume Arena 和 WYSIWYG 等。这里笔者推荐使用 MA Lighting 公司出品的 dot2 onPC 这款简单易学的控制软件。

1. 安装和配置 dot2 onPC 软件

首先进入 malighting.com 的官网，通过搜索 dot2 onPC 可以找到相应的下载地址，如图 1-139 所示。

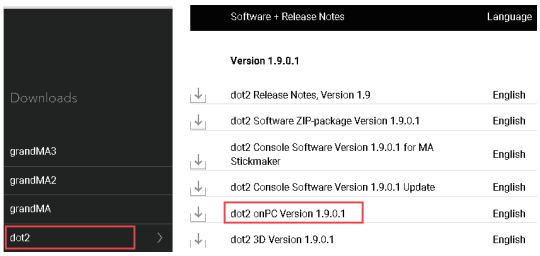

图 1-139　从 malighting.com 官网下载免费的 dot2 onPC 软件

安装好 dot2 onPC 软件后，打开程序，在界面下方单击 Setup 按钮可以依次设置 Network Interface（网络接口）、Sessions（会话）和 Network Protocols（网络协议），从而确保这个软件能与计算机上的 UE5 通信，如图 1-140 所示。

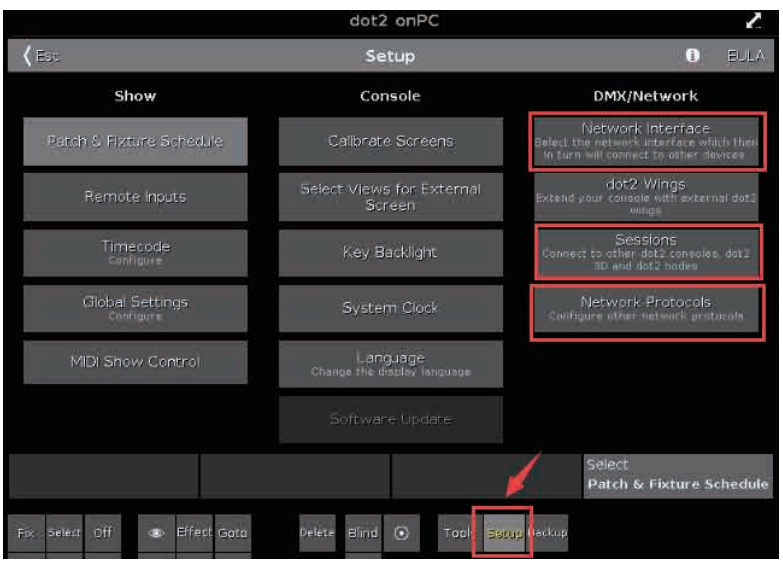

图 1-140　在 dot2 onPC 软件里设置

先单击图 1-140 中右上方红框标示的 Network Interface 后，会要求用户选择一个网络适配器，请选择自己计算机当前所用的网络适配器（俗称网卡）即可。此时，dot2 onPC 会重新启动。

然后单击 Sessions 进行设置，看到如图 1-141 所示的界面。

图 1-141　单击 Sessions 后看到的界面

单击右上角的 Start new or join an existing session（开始新会话或加入已有的会话）按钮，然后单击 Add 按钮添加一个已连接的设备（也就是当前计算机）即可，如图 1-142 所示。

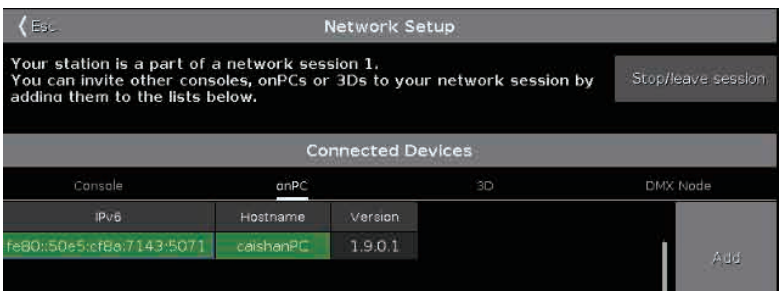

图 1-142　单击 Add 按钮添加已连接的设备

此时单击左上角的 Esc 图标退出当前界面，然后单击 Network Protocols 进入图 1-143 所示的界面。

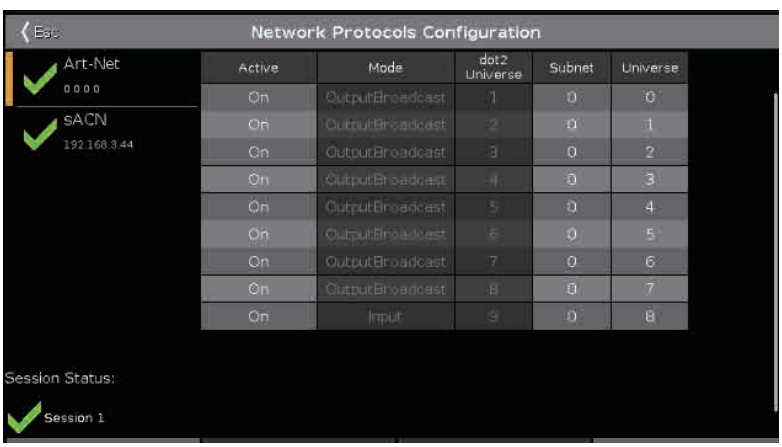

图 1-143　单击 Network Protocols 后看到的界面

将图 1-143 所示的三个绿色勾都打上后，就可以让软件向 UE5 发送 DMX 数据了。

2. 在 UE5 中配置灯具接收外部 DMX 数据

要让 UE5 能收到 dot2 onPC 发来的 DMX 信息，还需要在 UE5 的“Project Settings”（项目设置）里搜索找到 DMX，按照如图 1-144 所示进行设置，尤其是要将输入端的协议名设为 sACN 协议，对应的网卡地址改为 0.0.0.0。

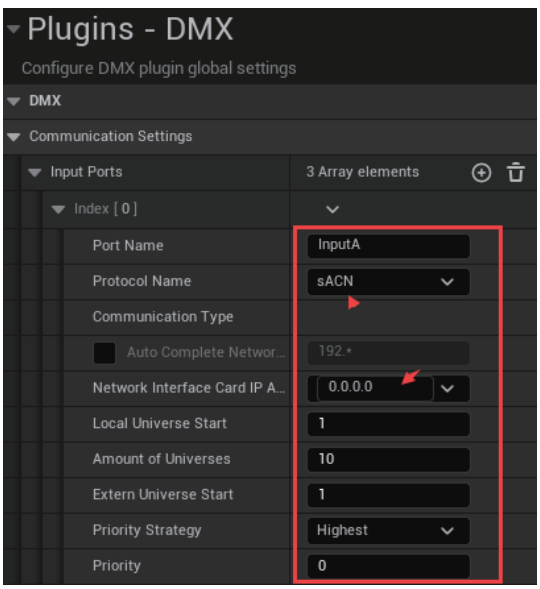

图 1-144　在 UE5 项目设置里将 DMX 输入端的协议名设为 sACN 协议

这样就可以让 dot2 onPC 与 UE5 开始 DMX 通信了。接着在 UE5 里新建一个文件夹 DMX_dot2，在文件夹中新建一个 DMXLibrary 文件，取名为 DMXLib_dot2，打开它后新建了一个灯具类型，取名为 LEDpar，如图 1-145 所示。

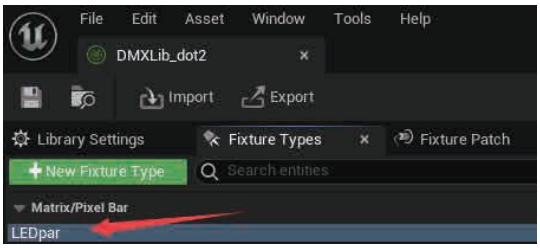

图 1-145　新建灯具类型 LEDpar

接着给它添加 4 个功能属性并配置对应的信道，将 2、3、4 信道分别对应灯光颜色的 RGB，也就是把 Channel 2 设置为控制灯的 Red 颜色，把 Channel 3 设置为控制灯的 Green 颜色，把 Channel 4 设置为控制灯的 Blue 颜色，如图 1-146 所示。

图 1-146　添加 4 个功能属性并配置了对应的信道

然后在“Fixture Patch”（灯具装配）标签里创建 10 个 LEDpar 类型的灯具，如图 1-147 所示。

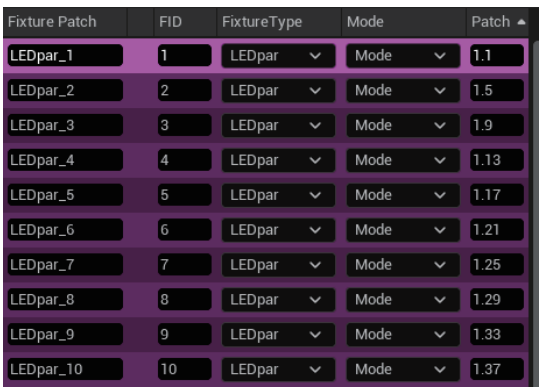

图 1-147　添加了具体的 10 个灯

由于每个灯均占用 4 个通道，所以 10 个灯在 Universe 1 里的信道分布情况如图 1-148 所示。

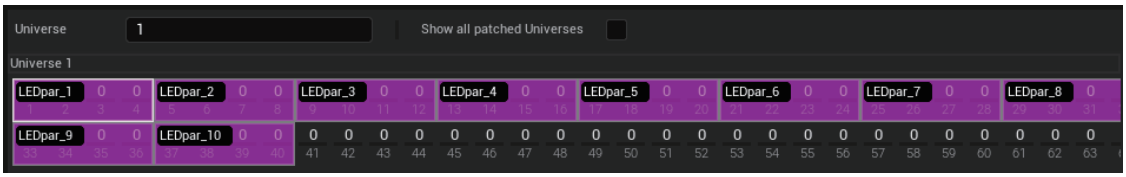

图 1-148　新增的 10 个灯具的信道分配

3. 在 dot2 onPC 中设置对应灯具

在 UE5 里保存 DMXLib_dot2 后，回到 dot2 onPC 软件里，再次单击 Setup 按钮，接着需要单击 Patch&Fixture Schedule 按钮来设置 dot2 onPC 软件里的灯具信息。当 dot2 onPC 软件中的灯具信道信息与 UE5 中的完全匹配后，就可以利用 dot2 onPC 软件的诸多功能来进行较为复杂的灯具 DMX 管控了。操作步骤如图 1-149 所示。

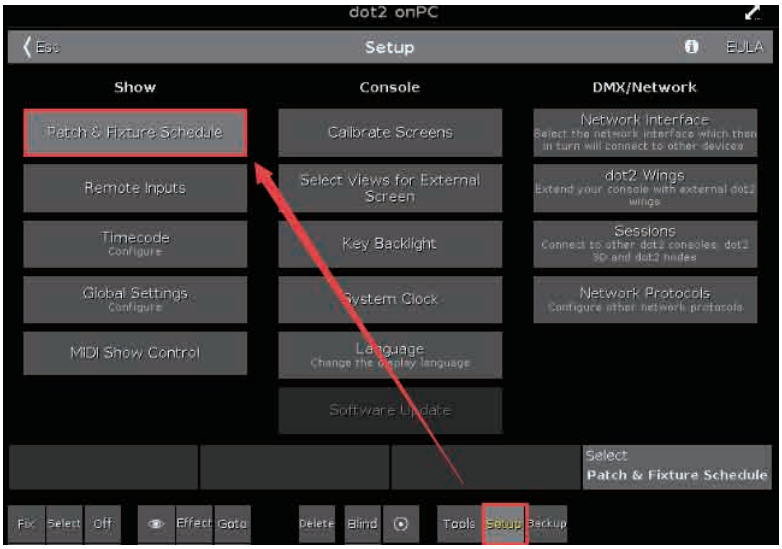

图 1-149　在 dot2 onPC 软件里单击 Patch&Fixture Schedule 按钮

接着单击右上角的 Add New Fixtures 按钮来添加新灯具，然后会弹出如图 1-150 所示的界面，单击 Select other 按钮。

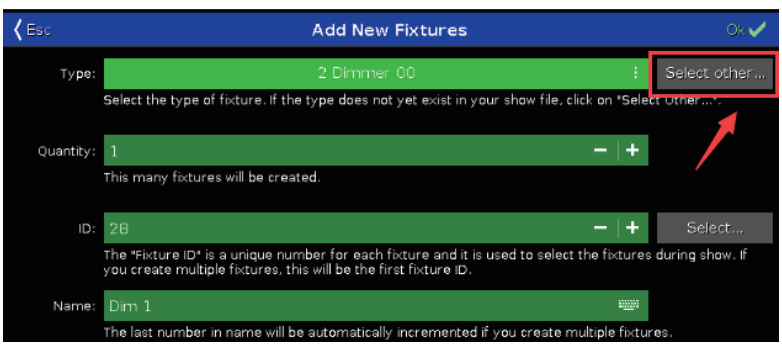

图 1-150　dot2 onPC 软件里添加新灯具的界面

dot2 onPC 软件内置了各大灯具厂商提供的数字灯具配置信息，可以输入 par 搜索到如图 1-151 所示的一款灯具。

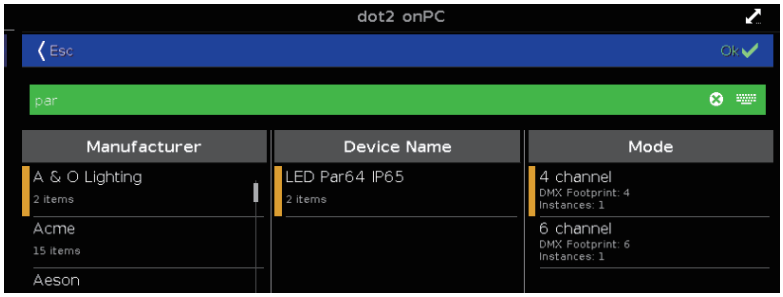

图 1-151　输入 par 搜索到 A&O lighting 制造商的 LED Par64 IP65 灯具

单击右上角的 OK 按钮，在弹出的窗口里 Quantity（数量）后面的输入框中输入 10，表示增加 10 个这样的灯，下方的 Patch 默认为 1.001 表示从 Universe（数据域） #1 的 1 号信道开始，如图 1-152 所示。

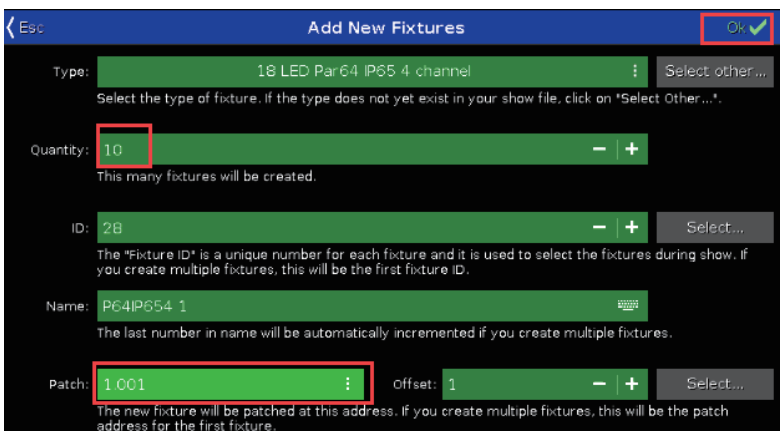

图 1-152　设置添加灯具的数量为 10 个

再次单击右上角的 OK 按钮，就可以看到 10 个灯的数据已经配置好了，其中数据列 Patch 显示了每个灯的起始信道，如图 1-153 所示。

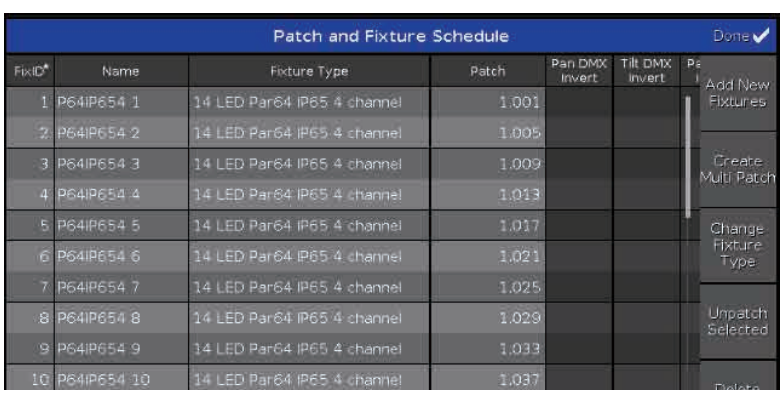

图 1-153　显示了 10 个新增的灯具的信息明细表

单击右上角的 Done 按钮，然后再单击出现的 Apply All Changes 按钮应用所作的修改，如图 1-154 所示。

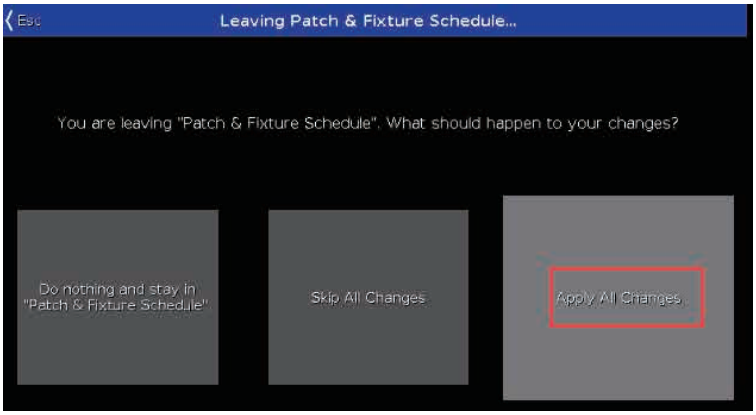

图 1-154　单击 Apply all Changes 按钮应用所有的修改

于是就可以看到有 10 个图标出现在了软件的左侧画面中，它们就表示这 10 个具体的灯，如图 1-155 所示。

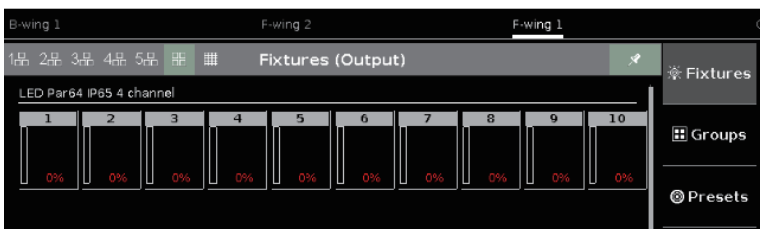

图 1-155　出现 10 个图标表示 10 个 LED 灯

这时候在 UE5 的菜单栏上打开 DMX 信道监视器（DMX Channel Monitor），可以看到 10 个灯的 3 个颜色信道（分别代表 R、G、B）都已经有了默认值 255。而且每个灯的第一个信道的初始值都是 0，即表示灯是关闭的，如图 1-156 所示。

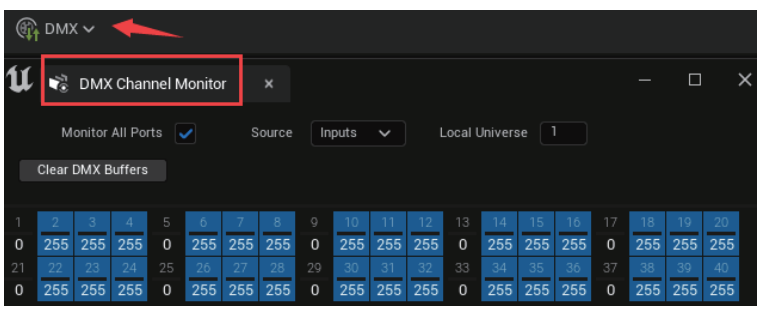

图 1-156　DMX 信道监视器能显示 dot2 onPC 软件发来的各信道的默认值

4. 在 UE5 中布置灯光秀场景

为了能在 UE5 场景中看到具体的 10 个灯，可以新建关卡，取名为 level_dot2。从 UE5 的灯具素材文件夹中拖曳 10 个 BP_StaticHead（静态头灯）放入关卡场景中，如图 1-157 所示。

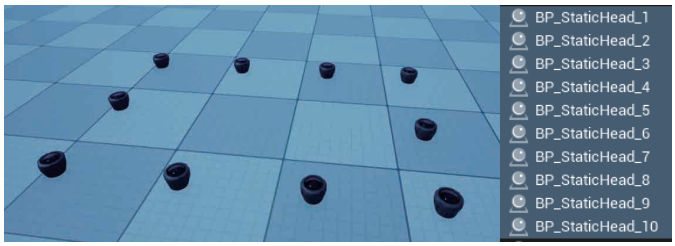

图 1-157　在关卡里摆放 10 个静态头灯

可以同时选中场景中的这 10 个灯，为它们批量配置 DMX 库并设定对应的灯具。从 DMX 菜单里选择 Open Patch Tool，这是一个可以批量修改灯具设置的工具。在弹出的窗口里设定 DMX Library 为 DMXLib_dot2，将 Fixture Patch（灯具装配）属性指定为 LEDpar_1，如图 1-158 所示。

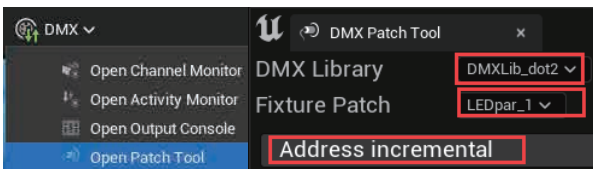

图 1-158　使用 Open Patch Tool 批量配置灯具

然后单击 Address incremental 按钮完成修改，这样这 10 个灯都会采用 DMXLib_dot2 这个文件作为它们的 DMX Library，而它们的 Fixture Patch 属性会分别自动对应上 LEDpar_1、LEDpar_2、LEDpar_3……依次类推。

批量处理完毕后，可以抽检查看一下，场景中的每个灯的细节面板里确实都已经配置好了相应的灯具信息，如图 1-159 所示，抽查第 10 个灯的细节面板里的 DMX 设置。

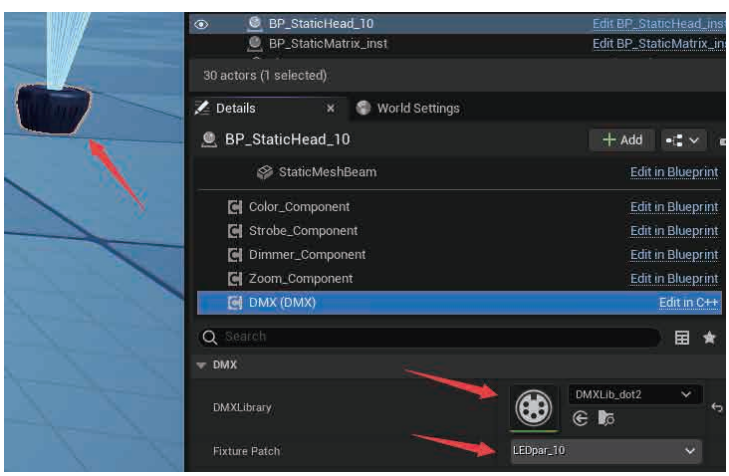

图 1-159　批量修改完毕后检查其中的某一个灯的 DMX 设置

5. 使用 dot2 onPC 操控 UE5 中的灯具

回到 dot2 onPC 软件里，用鼠标左键框选 10 个灯具图标，在右侧单击 Dimmer，然后再单击出现的 Open 按钮即可让这十个灯对应的第一个信道的值都变为 255，如图 1-160 所示。

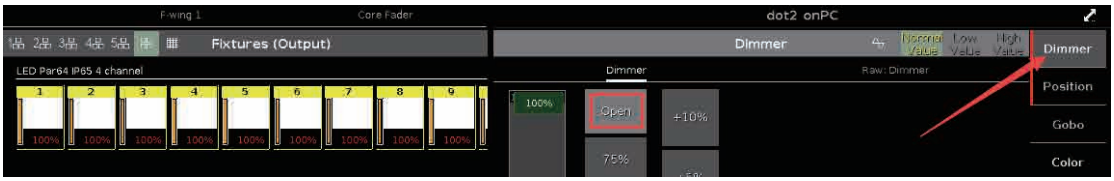

图 1-160　框选 10 个灯然后单击 Dimmer 项里的 Open 按钮

如果切换到 UE5 关卡里，也会看到场景中这 10 个灯确实都亮了，而且亮度还挺高，如图 1-161 所示。

图 1-161　UE5 场景中的 10 个灯都亮了且第一个信道的值均变为 255

我们可以在 dot2 onPC 软件里拖动如图 1-162 所示的最左侧的柱状滑块，从而很方便地调节灯的亮度，也可以单击 Close 按钮来熄灭灯具。

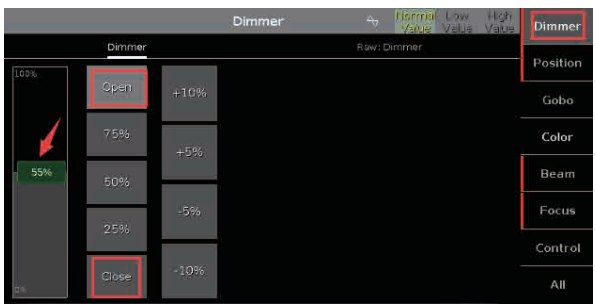

图 1-162　在 Dimmer 标签下单击 Open 和 Close 按钮来开关灯

还可以只选中某一个灯的图标，单独对其进行类似的调控。可以留意到右侧白色高亮的标签除了 Dimmer 外还有 Gobo、Color 等，在 dot2 onPC 软件里不同厂家、不同类型的灯具在右侧能够使用的功能标签不尽相同。通过单击右侧的 Color 按钮则可以很方便地调整对应灯具的颜色，如图 1-163 所示。

图 1-163　单击 Color 标签可以通过颜色选择器改变灯的颜色

通过上述一系列的步骤，我们演示了通过 dot2 onPC 这款外部软件来控制 UE5 灯具的过程，dot2 onPC 有很多内置的 Effect（特效）功能，能为灯具的各类属性设置动画效果，包括频闪动画、摇摆位置动画、颜色变换动画、亮度强弱变化动画等，这样将可以大大提升读者打造灯光秀控制的精彩程度。本节所使用 dot2 onPC 文件已经保存在了 UE5 项目 DMXDemo 的 dot2 文件夹里，文件名为 ue5show.show.gz。

1.3.4 实例：可遥控的数字灯光秀

实例一方面巩固复习前面的知识点，一方面也会加入新的知识点。同时，实例都配有视频讲解，可以避免单靠图文表述可能存在的含糊不清的地方。本节实例是通过在 UE5 所在台式计算机之外的另外一台笔记本上安装 dot2 onPC 软件后来控制台式计算机上的 UE5。

我们需要确保笔记本与台式计算机都连接在同一个 Wi-Fi 下。在台式计算机上，建立一个新的基于 DMX 模板的 UE5 项目 DMXDemo_live，打开后建立一个新的关卡，取名 Level1，再建立一个 DMX Library，取名 DMXLib_Live。在关卡 Level2 里放入了两个 BP_FireWorksLauncher（礼花发射器）、7 个 BP_MovingHead（摇头灯）和 6 个 BP_PyroModule（焰火模块），如图 1-164 所示。

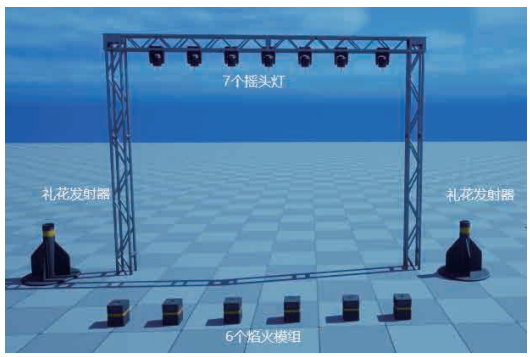

图 1-164　关卡里摆放了三种类型的共计 15 个灯具设备

打开 DMXLib_Live 对象，在里面新建了三种灯具类型，分别是 Sharp、Fireworks 和 pyro，如图 1-165 所示。

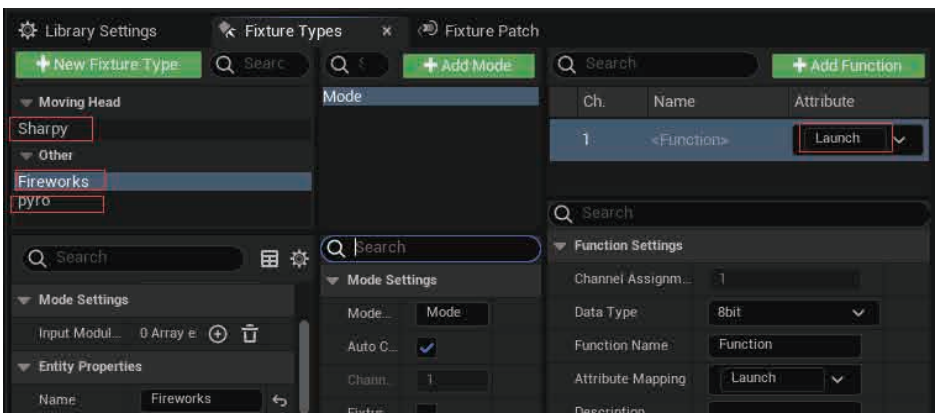

图 1-165　在 DMX Library 里添加 3 个灯具类别

其中 Fireworks 这种类型比较简单，只使用了一个信道，信道对应的属性就是 Launch，也就是发射礼花的功能。而 pyro 类型使用了两个信道，分别是 ModeStartStop 和 Burst，如图 1-166 所示。

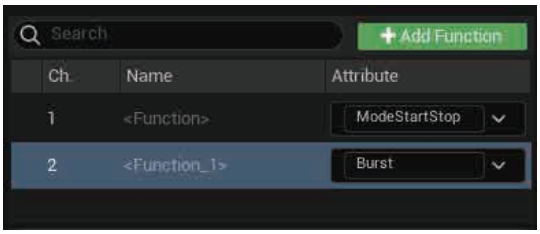

图 1-166　pyro 灯具类型里添加了两个属性 ModeStartStop 和 Burst

而对 Sharpy 这个类型，一共设定了 16 个信道，之所以这么做是因为 Sharpy 这个类型后续会提供给摇头灯使用，通过查看 BP_MovingHead 蓝图的组件面板可以看到，UE5 里的摇头灯有 8 个核心的 DMX 组件，分别对应着它的 8 个核心功能：Pan、Tilt、Color、Gobo、Dimmer、Strobe、Zoom 和 Frost。而 dot2 onPC 软件里有一款供应商的灯具正好涵盖了这八种功能但却总共有 16 个信道。为了让 UE5 里灯具的信道设置与 dot2 onPC 软件里的保持一致，所以在 UE5 里为 Sharpy 灯具类型设置了 16 个信道，只是有些信道直接对应了空值，没有具体属性。具体设置如图 1-167 所示。

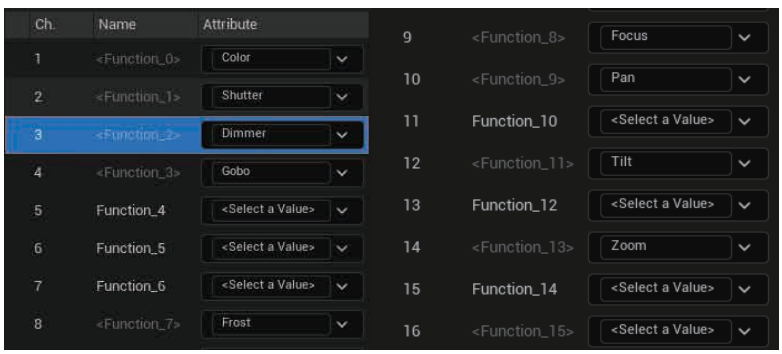

图 1-167　Sharpy 类型里添加了 16 个信道

接着在 Fixture Patch（灯具装配）标签下具体地添加这 15 个灯，并将每个灯的信道分配好。最后结果如图 1-168 所示。

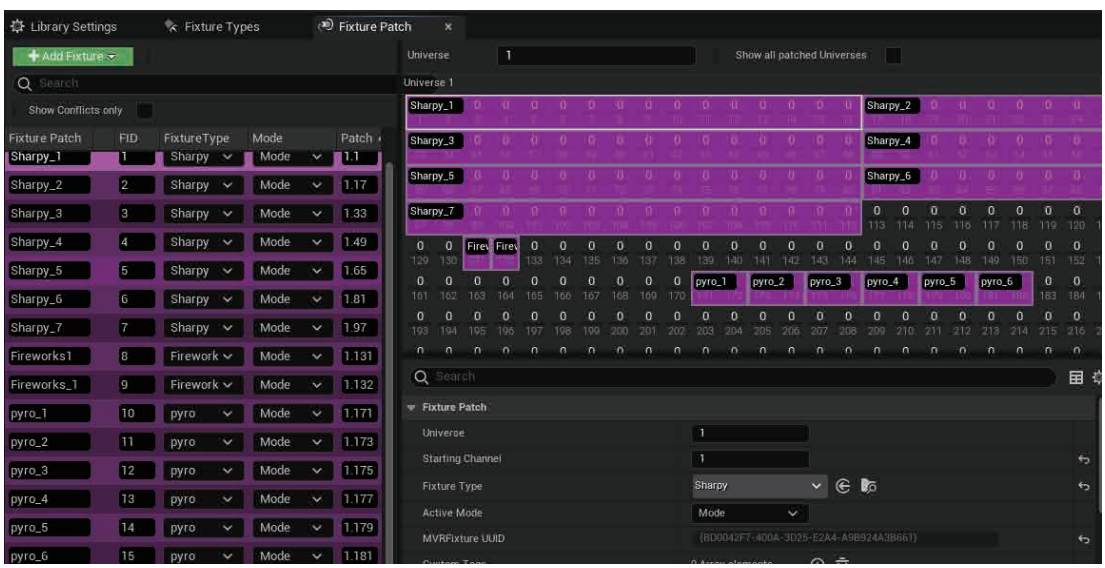

图 1-168　15 个灯具各自的信道分布

在 dot2 onPC 软件里不能像 UE5 那样自定义创建灯具类型，它只能从现有供应商列表里选择实际的灯类。在笔记本电脑上打开 dot2 onPC 软件后，在软件里搜索 sharp，建立 7 个 Sharpy 灯具，因为这款灯的信道配置正好包含了 UE5 摇头灯所拥有的 8 个核心功能，但这种灯在 dot2 onPC 软件里共有 16 个信道，每个信道对应的功能可以从图 1-169 中看到。

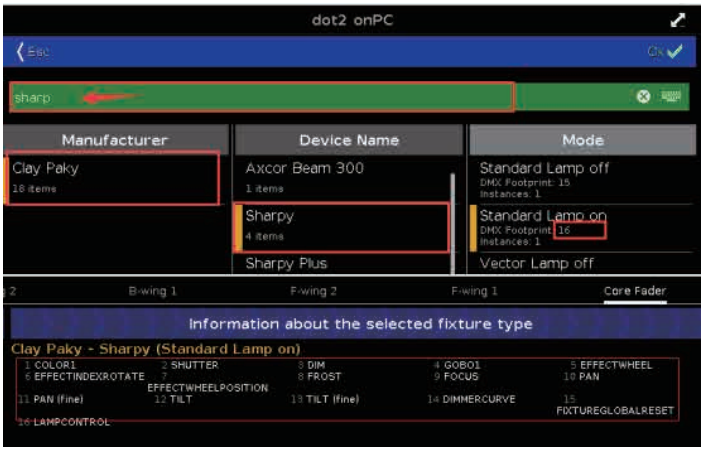

图 1-169　在 dot2 onPC 软件里选择灯具类型

接下来可以建立两个 Punch DayLight 类型的灯，因为 dot2 onPC 软件里的这种灯具只使用一个信道，对应 UE5 中的 Fireworks 类型正好合适，如图 1-170 所示。

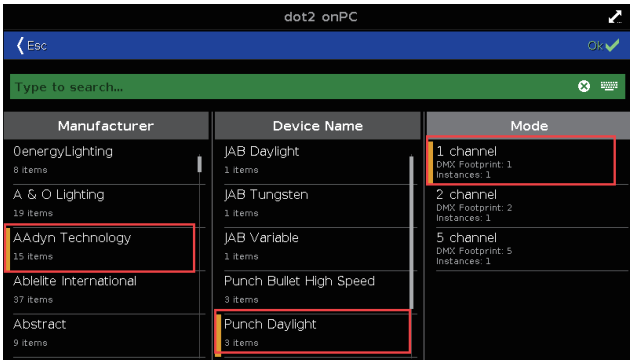

图 1-170　选择只有一个信道的 Punch DayLight 1 channel 类型的灯

最后再建立 6 个 ALS 150 D 18-38 类型的灯，因为 dot2 onPC 软件里的这种类型的灯使用两个信道，正好可对应 UE5 中 pyro 类型的灯具，如图 1-171 所示。

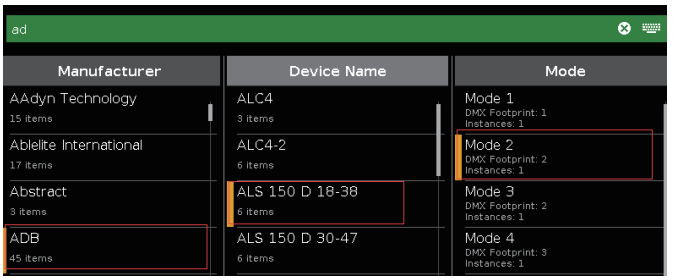

图 1-171　选择 ALS 150 D 18-38 Mode2 类型的灯

在 dot2 onPC 软件里配置好灯具数量和相应的信道后，就可以利用里面的 Effect 施加各类效果给到灯具了，如图 1-172 所示。

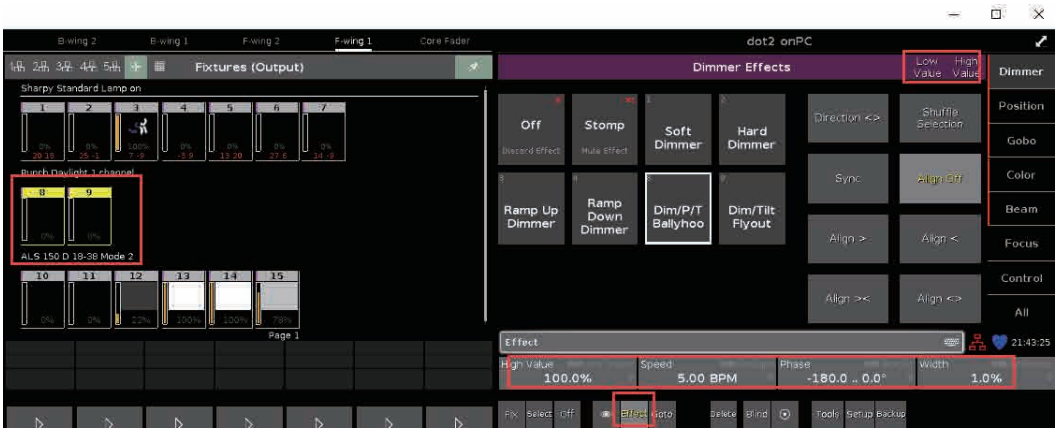

图 1-172　使用 dot2 onPC 软件里的 Effect 特效

每个属性（如 Dimmer、Color 等）对应的 Effect 中都有多个特效，每个特效都有相应的参数可以修改。利用 dot2 onPC 软件里丰富的 Effect 可以让灯具绚丽地舞动起来。灯具表演的过程可以通过 UE5 进行录制，在 UE5 中选择 Window → Cinematics → Take Recorder 命令就可以开始录制，如图 1-173 所示。

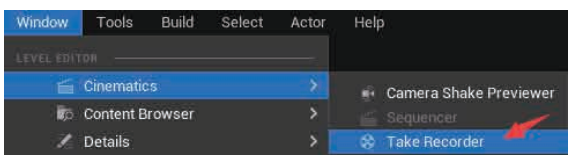

图 1-173　选择 Take Recorder 录制

在弹出的 Take Recorder 窗口中，单击 +Source 按钮添加相应的 DMX 库文件，然后单击 Add all Fixture Patches 按钮把所有的灯具对应添加进来，单击右上角的红色圆形按钮后就可以开始录制了，如图 1-174 所示。

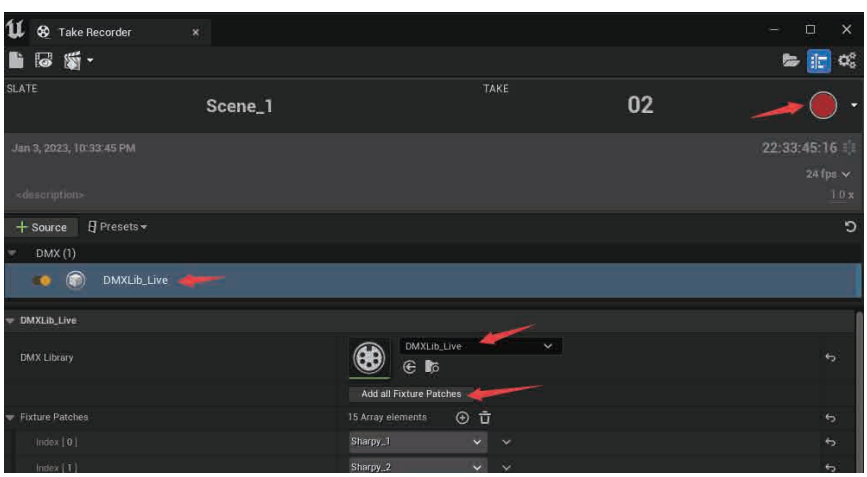

图 1-174　添加 DMX 库和灯具后开始录制

如果要停止或结束录制，可以单击屏幕右下角的 Stop 按钮，如图 1-175 所示。

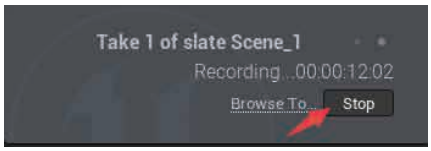

图 1-175　停止录制

录制得到的是一个 Level Sequence 文件，而且里面的关键帧是不可编辑的，如果希望录制得到可以编辑的 Level Sequence 文件，则需要在 Take Recorder 窗口中的右上角单击“显示用户设置”图标，然后从最底部取消对 Auto Lock 项的勾选，如图 1-176 所示。

图 1-176　取消对 Auto Lock 项的勾选才能录制得到可编辑的 Sequence 文件

录制完成后可以从 Content 下找到出现的 Cinematics 文件夹，里面有录制好了的 Sequencer 文件，双击打开，可以按空格键或是单击 Sequencer 面板底部的播放按钮来播放整个录制的内容。

但是由于录制的是 DMX 信号相关的动画，在播放时需要将工具栏上的 DMX 工具按钮下的 Receive DMX 项取消勾选，这一点很容易忽视，从而导致无法看到动画。其他情况下，尤其是通过外部软件发送 DMX 信号给 UE5 或是使用 UE5 的 DMX 工具里的 Open Output Console 调试时记得要把 Receive DMX 项先勾选上，如图 1-177 所示。

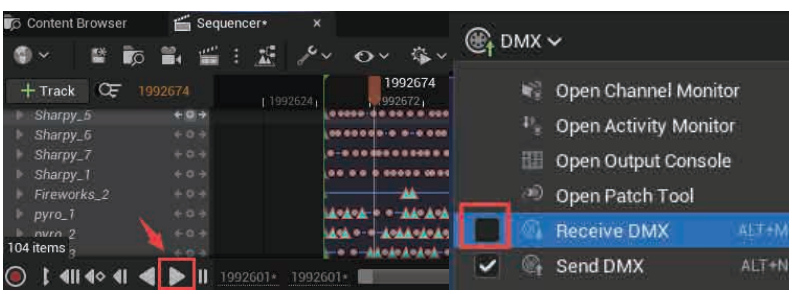

图 1-177　播放 Sequencer 时要先取消勾选 Receive DMX

运行 UE5，再单击播放这个 Sequencer，就可以看到漂亮的灯光焰火秀了。把 dot2 项目文件保存到 U 盘给到笔记本，从笔记本打开 dot2 软件，载入 U 盘里的项目文件，即可实现通过笔记本来遥控灯光秀，如图 1-178 所示。

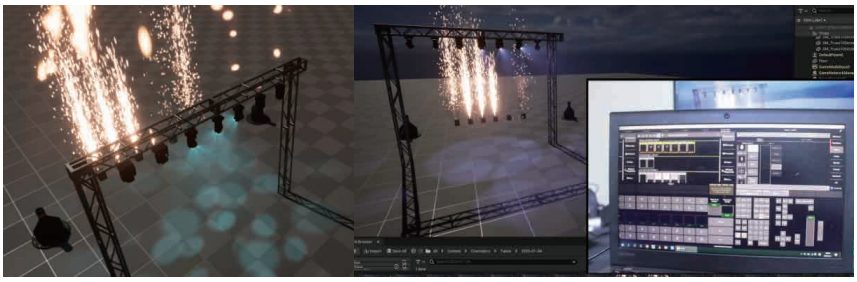

图 1-178　笔记本上载入台式计算机上调试好了的 dot2 项目文件来遥控 UE5

本实例详细的操作步骤可以通过扫描下方二维码来观看。

1.4　用 OSC 来控制 UE5 中的物体

1.4.1　用TouchOSC定制UE5的控制界面

OSC（Open Source Control）是经过现代网络技术优化过的一种用于计算机、声音合成器与其他多媒体设备之间通信的协议，它将现代网络技术的优势带进了电子乐器的世界。OSC 的优势包括互操作性、准确性、灵活性，以及增强的组织和文档化。这个简单而强大的协议提供了实时控制声音和其他媒体处理所需的一切，而且使用起来非常灵活、简单。在 UE5 中已经内置了 OSC 插件，它让数字互动又多了一种数据通信的方式。OSC 所发送的数据由两部分组成，即地址和数据，其中地址是一个类似 aaa/bbb 这样的字符串，而数据是与之对应的具体值。

1. TouchOSC 程序的安装与配置

从 hexler.net 官网可以下载安装 TouchOSC 程序，这套软件在 Windows 和 Mac 计算机以及手机上都可以安装。笔者下载的是 Windows 版本，如图 1-179 所示。

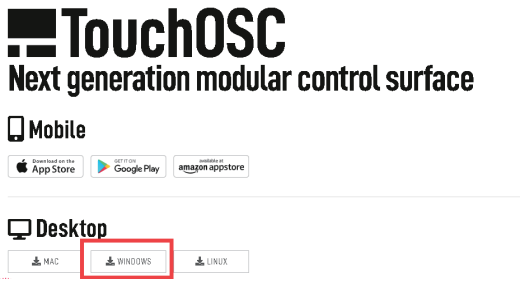

图 1-179　下载 TouchOSC

安装完毕后打开 TouchOSC，为了让它能与 UE5 通信，需要先配置一下它的链接信息。在工具栏上单击链接图标，在弹出的 Connections（连接）窗口里单击 OSC 标签，设置通信协议为 UDP，并设置 Host 地址（也就是 UE5 所在计算机的局域网 IP 地址）和端口号 8000。如果 TouchOSC 与

UE5 安装在同一台计算机上，则在 Host 地址里输入 127.0.0.1 即可，如图 1-180 所示。

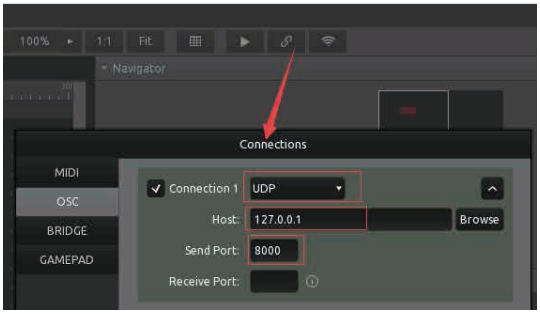

图 1-180　TouchOSC 配置 OSC 连接设置

单击 Done 按钮确认后，在界面的空白处右击，从弹出菜单里选择 BUTTON，建立一个按钮。

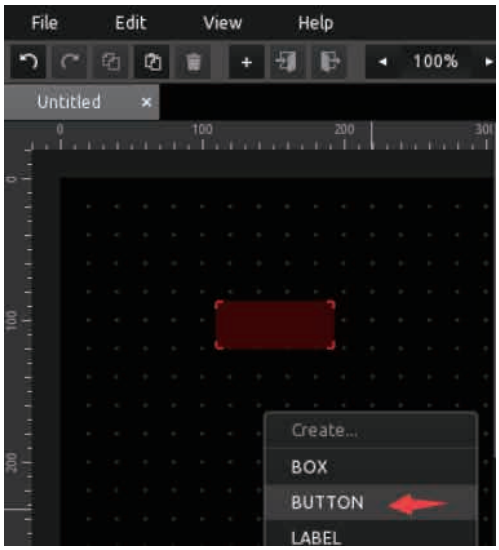

图 1-181　在 TouchOSC 编辑界面创建一个按钮

在右侧的控件属性面板里可以看到这个按钮的 OSC 相关的信息，选中 touch 表示在触摸按钮时会触发，RISE 则强调是在手指脱离触摸的瞬间触发，如图 1-182 所示。TouchOSC 里把触碰事件分为 RISE 和 FALL，分别表示开始接触和脱离接触。ANY 则既包括 RISE 又包括 FALL。如果选择 ANY，那么在单击按钮的过程中会触发两次发送数据的事件。

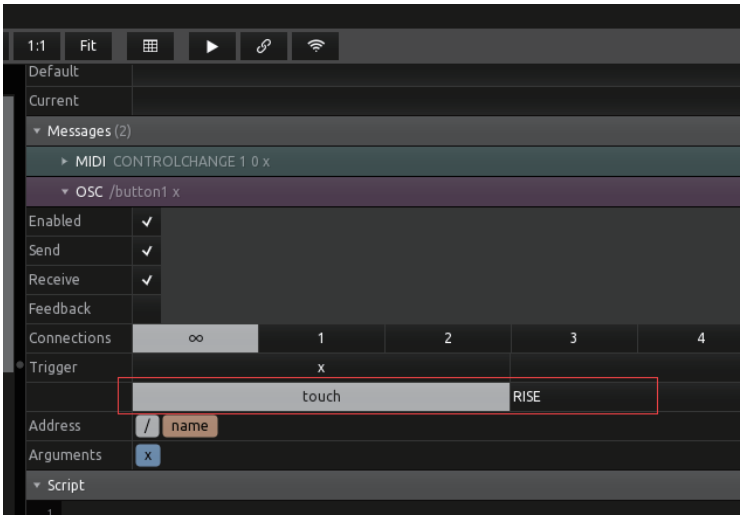

图 1-182　Trigger 选择 touch 表示触摸时触发，x 表示鼠标触发

而具体发送的数据可以通过单击右侧加号来加入，如图 1-183 中的 name，如果想修改，可以选中 name 直接按键盘上的 Delete 键先删掉它，然后单击加号，选择 CONSTANT（常量），表示添加一个常量。

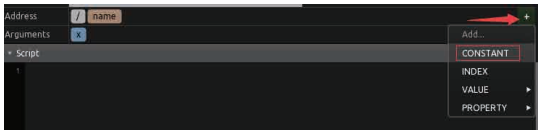

图 1-183　添加要发送的内容

如图 1-184 所示，在 Address 里就加入

了一个 test 常量，这样发送出去的地址信息就是 /test。而地址下面的 Arguments（参数）表示要发送的数据值。

图 1-184　添加具体的常量值

按钮默认的参数 X 的值就是 0。可以选中这个 X 删掉它，还可以通过单击参数右侧的加号按钮加入更多类型的参数，如加入一个字符类型的常量 Hello，如图 1-185 所示。

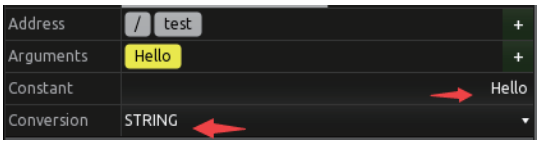

图 1-185　添加不同类型的参数

2. 使用不同类型的控件

在 TouchOSC 软件的控制界面可以加入更多其他类型的控件，如再添加一种名为 XY 的控件，它的作用是让用户在一个方块区域内触摸时，捕获到手指或鼠标在红色方块内的 X 坐标和 Y 坐标。方块的左下角是（0，0）点，右上角是（1，1）点，这个坐标系统和人们熟知的三维模型设计软件中的 UV 定位方式很相似，如图 1-186 所示。

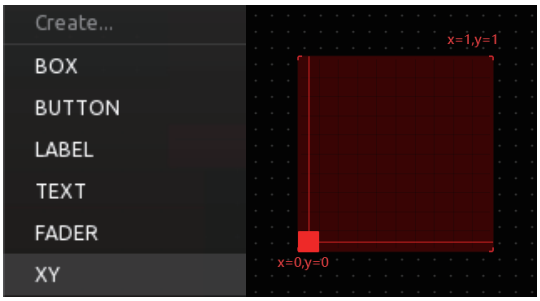

图 1-186　添加 XY 类型的控件

如图 1-187 所示，在右侧的控件属性的 OSC 部分进行设置。红框里的设置表示当用户在方块内拖动鼠标时会连续发送数据，事件的两个 Arguments（参数）x 和 y 分别表示鼠标在方块区域内对应的 x 坐标和 y 坐标。图 1-187 右下角的 0 和 1 表示 x 参数值的范围是 0 到 1。

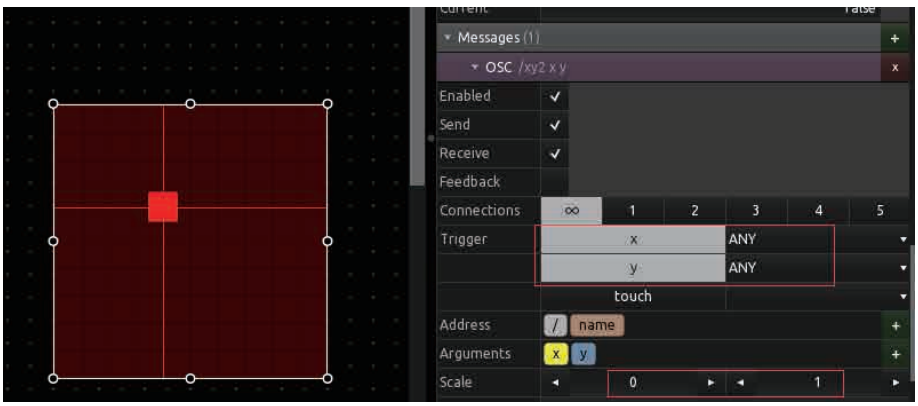

图 1-187　XY 类型的控件的 OSC 属性设置

可以再添加一个 FADER 控件（纵向滑块），这个控件比较常用。对应 OSC 属性设置的细节如图 1-188 所示。

图 1-188　FADER 控件的设置

参数 X 表示滑块滑动到了什么位置，在最底部的值是 0，在最高处的值是 1。

添加完这三个控件后，可以进一步调整好它们的位置布局，然后单击工具栏上的运行按钮（白色三角按钮），就可以让这个 TouchOSC 程序开始与 UE5 连接并发送数据了，如图 1-189 所示。

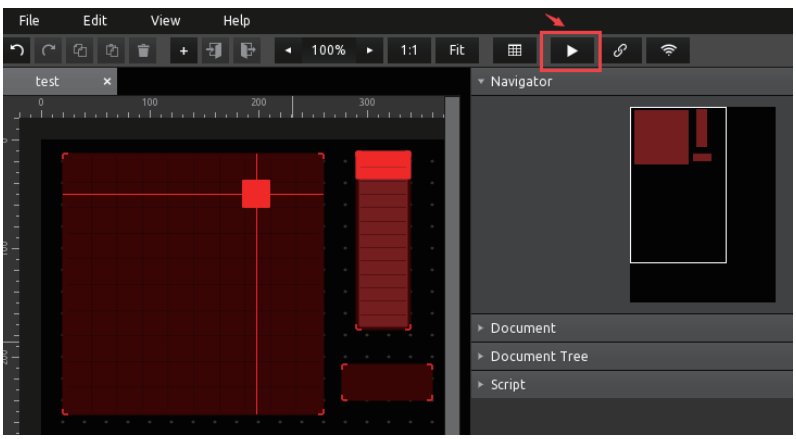

图 1-189　单击 TouchOSC 程序的运行按钮

1.4.2　用蓝图接收OSC信号

在 UE5 中使用蓝图接收并处理 OSC 数据信息，就可以让外部 OSC 软件起到操控 UE5 对象的作用。要想在 UE5 中接收到外部发送过来的 OSC 数据，需要先启用 OSC 插件（本节 UE5 项目源文件在 OSCDemo 文件夹中），如图 1-190 所示。

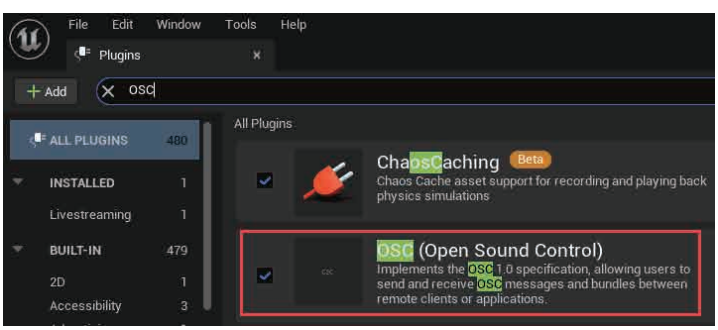

图 1-190　UE5 启用 OSC 插件

重启 UE5 后，在 OSCDemo 项目里新建了一个关卡，取名 Level1，接着在关卡蓝图里写入如图 1-191 所示的内容。

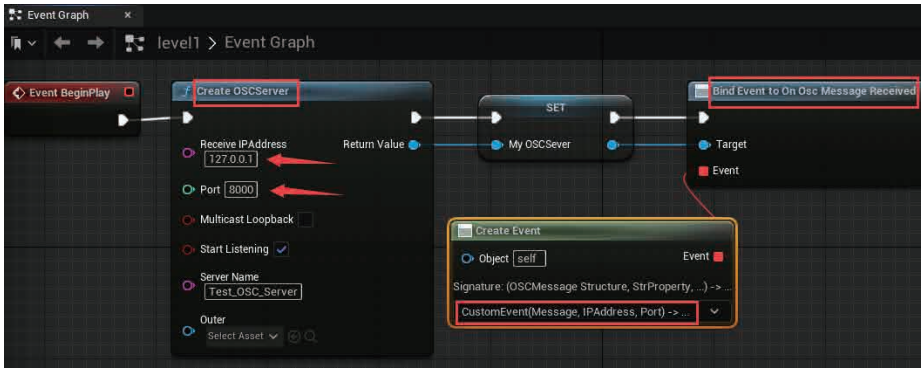

图 1-191　用蓝图创建 OSC 服务器

蓝图内容的意思是在 BeginPlay（关卡运行）时，UE5 会构建一个 OSC 服务器，其 IP 地址为 127.0.0.1，监听端口为 8000，这与刚才在 TouchOSC 中的设置完全吻合。然后把服务器对象存入到了一个名为 MyOSCServer 的变量里，接着为这个服务器对象绑定一个 OSC 信息接收事件。而这个接收事件对应的蓝图内容如图 1-192 所示。

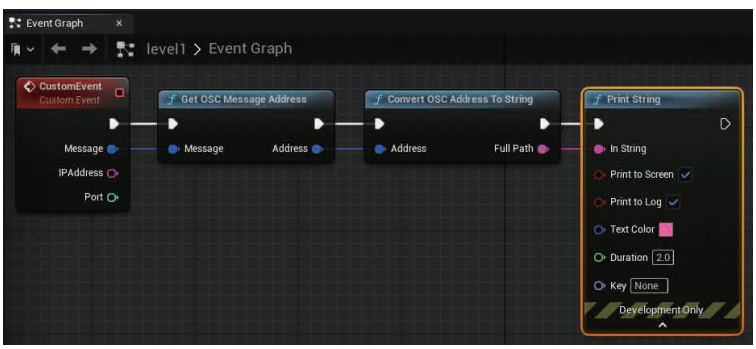

图 1-192　接收 OSC 消息地址并打印输出

这部分蓝图内容的意思是：UE5 在接收到外部的 OSC 数据时，会将接收到的 OSC 事件信息中的消息地址转为字符串格式打印出来。Compile（编译）关卡蓝图并保存，接着运行 UE5。然后在 TouchOSC 运行的界面试试单击各个控件，观察 UE5 视口中的文本输出，如图 1-193 所示（提示：如果要关闭 TouchOSC 运行的 UI 界面，而不是关掉整个 TouchOSC 程序，需要单击图 1-193 中红色箭头所指的右上角的灰色小圆点）。

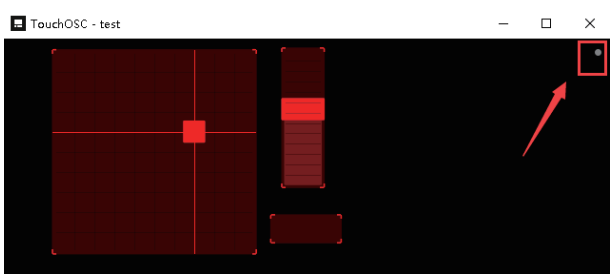

图 1-193　在 TouchOSC 运行的界面上操作

如果单击 TouchOSC 界面上的矩形按钮，会发现 UE5 视口中能打印输出 /test 字样，这就是 TouchOSC 程序发送过来的地址信息，如图 1-194 所示。

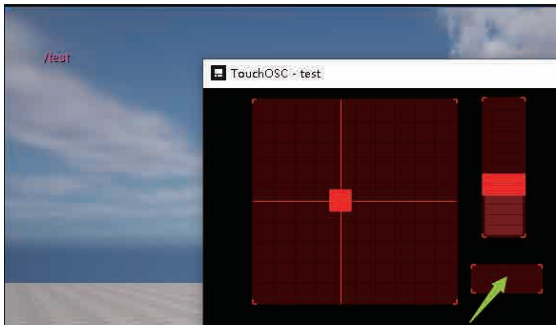

图 1-194　单击按钮会发送 /test 给到 UE5 打印出来

如果在 UE5 中还想获得来自 TouchOSC 界面上的 XY 控件和 FADER 控件的 OSC 数据，则可以进一步完善关卡蓝图，详细内容如图 1-195 所示。

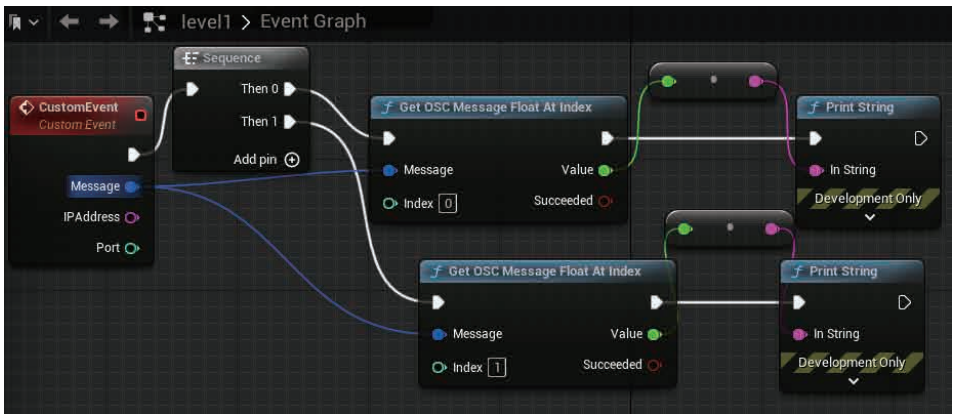

图 1-195　蓝图接收 OSC 消息的多个参数值

这表示将收到的 OSC 信息中的第一个浮点（Float）类型的数据参数和第二个浮点类型的参数分别打印出来。所以这时如果在方块区域内按住鼠标左键并拖曳移动鼠标，那么 UE5 会不断打印出鼠标的坐标位置的 x 和 y 值，两者都是介于 0 到 1 的浮点小数，如图 1-196 所示。

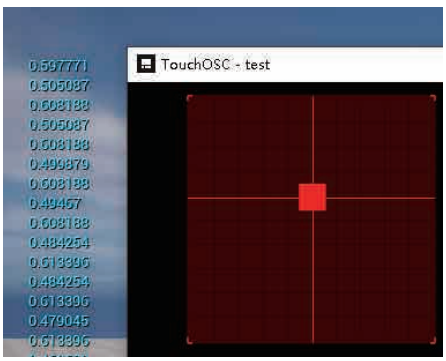

图 1-196　UE5 打印输出 XY 控件的 XY 参数值

1.4.3　实例：用OSC控制UE5的人物动作

本节实例是基于第三人称游戏模板建立的一个 UE5 项目，项目名称为 OSCDemo_live。通过这个项目一起来实践如何利用外部软件发送的 OSC 数据来操控 UE5 中的对象。本实例用到的 TouchOSC 文件 test.tosc 也保存在 OSCDemo_live 文件夹里。选用 Third Person（第三人称）模板的操作如图 1-197 所示。

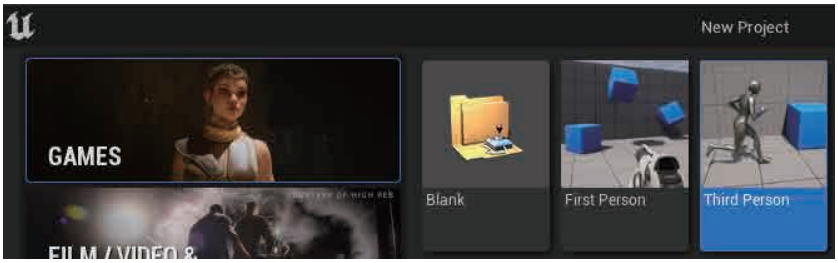

图 1-197　新建第三人称游戏模板的 UE5 项目

UE5 项目中要记得开启 OSC 插件，重启后把项目中的 BP_ThirdPersonCharacter 蓝图对象复制一份，取名 BP_Robot，如图 1-198 所示。

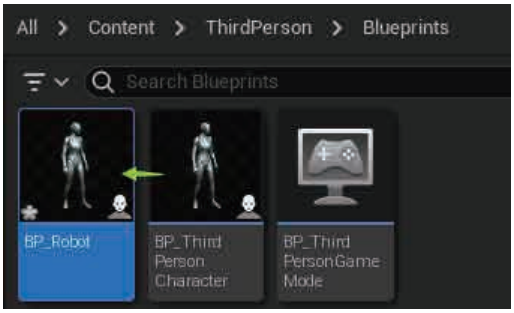

图 1-198　复制蓝图对象

双击打开 BP_Robot 进行编辑，从它的组件面板里选中其中的 Mesh 组件，如图 1-199 所示。

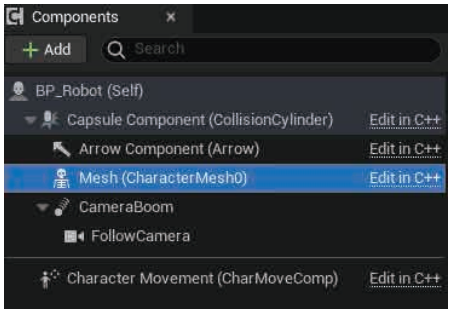

图 1-199　从组件面板里选中 Mesh 组件

从它对应的细节面板里找到 Animation 部分的属性，作如图 1-200 所示的修改。

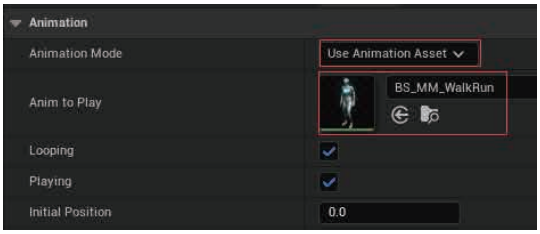

图 1-200　设置 Animation Mode 和 Anim to Play 属性

将这个蓝图编译并保存，然后从 Content Browser 里拖曳一个 BP_Robot 放置到默认关卡中。选中关卡场景中的这个 BP_Robot 对象，然后在关卡蓝图中写入如图 1-201 所示的内容。

在蓝图区域的空白处右击可以获取对场景中 BP_Robot 的引用，使用 Set Play Rate 节点来设置 BP_Robot 对象的 Mesh 组件的 Play Rate（播放速率）属性，也就是动画播放速度。这样，UE5 接收到的 OSC 信息里的值就会影响 Mesh 动画播放速度。蓝图细节参数如图 1-202 所示。

图 1-201　关卡蓝图里绑定 OSC 信息接收事件

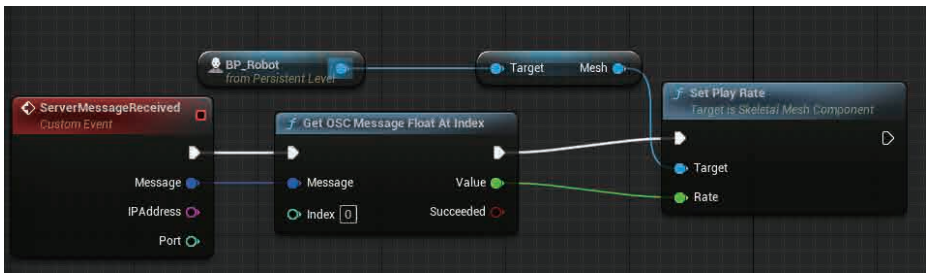

图 1-202　用 OSC 信息参数来设置 Mesh 组件的 Play Rate

编译保存关卡蓝图后，运行关卡。接着双击源代码文件夹里的 test.tosc 文件，启动 TouchOSC 程序界面，再单击 TouchOSC 工具栏上的运行按钮（白色三角按钮），此时 TouchOSC 会连接 UE5 中用蓝图创建的 OSC 服务器，接下来就可以通过拖动 TouchOSC 界面上的 FADER 滑块来控制人物动作的速度快慢，如图 1-203 所示。

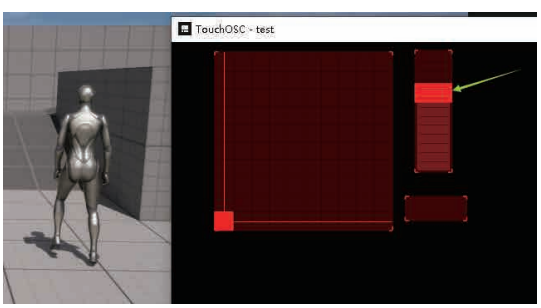

图 1-203　用 FADER 控件控制人物动画速度

由于 TouchOSC 软件界面控件丰富、调整布局和参数设置灵活，而且在 PC 端和移动端上都能安装运行，因此大大提高了远程控制 UE5 时的界面定制效率。

本实例详细的操作步骤可以通过扫描下方二维码来观看，视频教程里含有对更多物体的控制演示。

1.5　在 UE5 中使用 MIDI 乐器数字接口

1.5.1　使用虚拟琴键发送MIDI信号

MIDI（Musical Instrument Digital Interface，乐器数字化接口）是编曲界最广泛的音乐标准格式，可称为“计算机能理解的乐谱”。MIDI 传输的不是声音信号，而是音符、控制参数等指令，它指示 MIDI 设

备要做什么以及怎么做，如演奏哪个音符、多大音量等。它们被统一表示成 MIDI 消息（MIDI Message）。UE5 已经拥有很多为 MIDI 量身定制的插件，借助这些插件，UE5 可以通过 MIDI 信号与外界的诸多电子音乐设备进行通信交互，让视听互动变得简单易行。

目前市面上已经有大量的现代乐器支持接收和输出 MIDI 信号。而很多音乐编辑软件也能发送 MIDI 信号，它们被统称为 DAW （Digital Audio Workstation，数字音频工作站），是一种用于录音、混音、音频剪辑以及数字音频处理的软件（也有硬件单元，但是不常见）。常见的 DAW 软件有 Ableton Live、Pro Tools、Logic、GarageBand 和 REAPER 等。所有的 DAW 都是可以录制、编辑、处理以及缩混数字音频的。大部分 DAW 都带有 MIDI 功能，可以通过 MIDI 控制器对音符进行输入、编辑并通过诸如合成器之类的虚拟乐器音进行播放。

在本节中，将以 REAPER 这款软件为例，为读者们讲解 DAW 如何发送 MIDI 信号到 UE5 以及在 UE5 中如何查看这些信号。

1. REAPER 和 loopMIDI 的组合运用

首先，需要下载并安装 REAPER 程序，如图 1-204 所示。

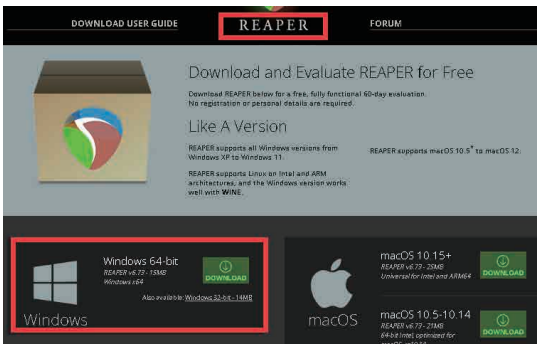

图 1-204　下载安装 Windows 版 REAPER 软件

然后，还需要下载 loopMIDI 这款软件，它的作用是成为一个 MIDI 信号中转站，可以把各类 DAW 软件里发出的 MIDI 信号传送给计算机上的其他软件（如 UE5），如图 1-205 所示。

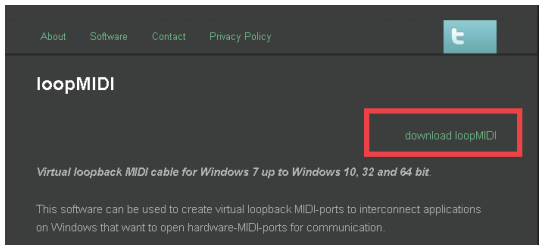

图 1-205　下载 loopMIDI 软件

运行 loopMIDI，通过单击左下角的加号按钮添加一个 MIDI 的信号传输口。

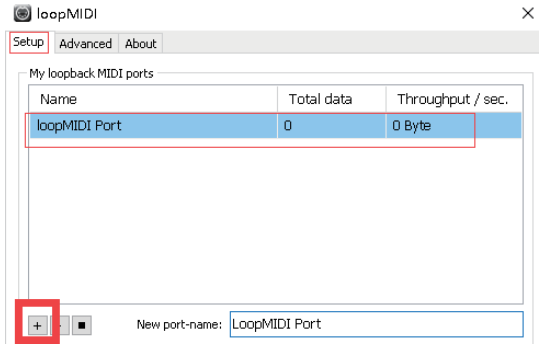

图 1-206　loopMIDI 运行的窗口

注意：这个窗口不要关闭。接着可以启动 REAPER 软件，新建一个项目，取名为 Virtual_MIDI_Keyboard。从菜单栏里选择 View → Virtual MIDI Keyboard 命令，如图 1-207 所示。

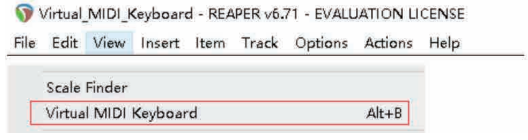

图 1-207　在 REAPER 里开启虚拟琴键功能

于是会弹出一个琴键窗口，右击选择 Dock Virtual MIDI keyboard in Docker，如图 1-208 所示，可以将它固定在软件界面的底部区域。

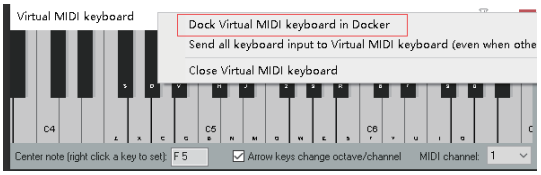

图 1-208　调整虚拟琴键的布局

接着鼠标双击 REAPER 界面左上角空白区域建立一个音轨（Track），如图 1-209 所示。

图 1-209　新建一个音轨（Track）

从主菜单上的 Options → Preferences... 打开偏好设置，将 Device 项相关参数设置为如图 1-210 所示。

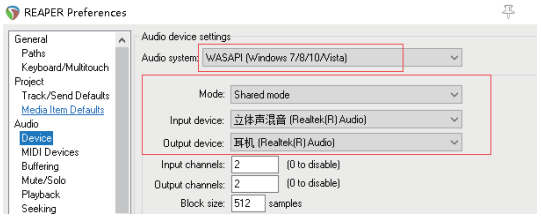

图 1-210　修改 Device 设置让计算机能播放音轨的声音

其中 Output device 需依据计算机实际的音频外放设备来设置，这样就可以让计算机播放出 REAPER 的音轨音效了。之后还需要把 MIDI Devices 中底部的 loopMIDI Port 设置为 Enabled，如图 1-211 所示。

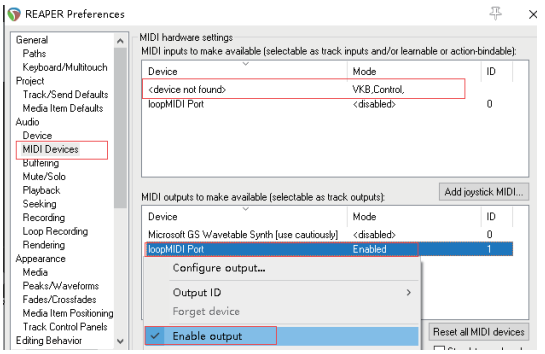

图 1-211　设置 loopMIDI Port 为 Enabled

单击音轨的左侧暗红色圆形按钮，让它变为亮红色，同时单击小喇叭图标让它处于 ON 状态，如图 1-212 所示。

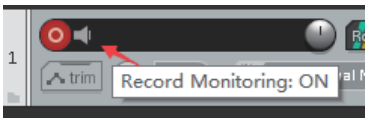

图 1-212　设置音轨为录制模式

在音轨右下方出现的 IN FX 条上右击，指定 Input:MIDI，也即是通过虚拟琴键作为音轨的输入源，如图 1-213 所示。

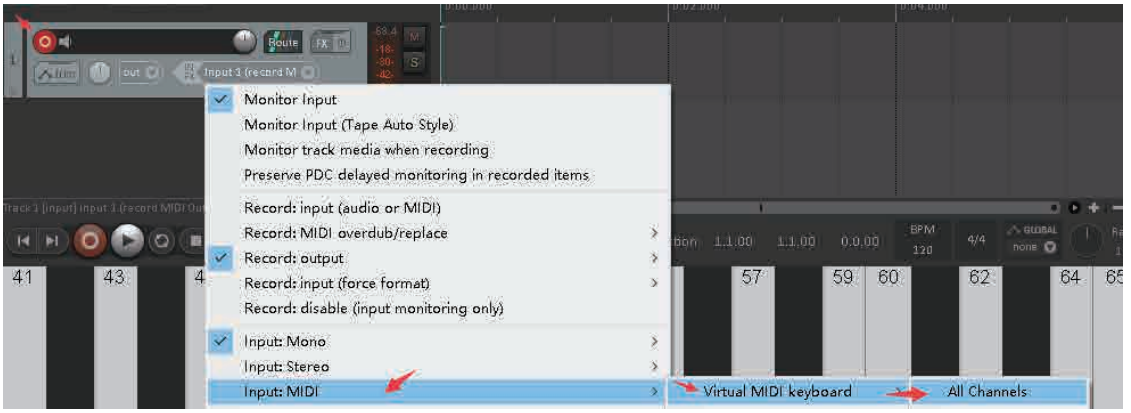

图 1-213　设置音轨的输入来自于虚拟琴键

接着单击音轨上的 Route 按钮，在弹出的窗口里将 MIDI Hardware Output 设置为 loopMIDI Port，如图 1-214 所示。

图 1-214　将 MIDI Hardware Output 设置为 loopMIDI Port

这时单击虚拟琴键，可以看到音轨上的指示器以及 loopMIDI 软件上的数据值都会相应地发生变动。这就说明 REAPER 的 MIDI 数据已经发送到 loopMIDI 这个信号中转站了，如图 1-215 所示。

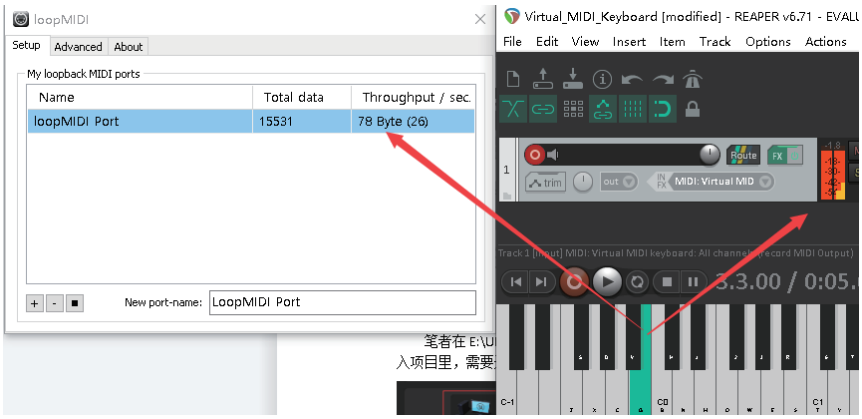

图 1-215　单击虚拟琴键看 loopMIDI 是否有数据变化

2. 在 UE5 中使用 Remote Control Protocol MIDI 插件

接下来，笔者在 E:\UE5_Tutorials\Cai_NewBook 路径下建立一个空白的 UE5 项目，取名为 MIDIDemo。进入项目里，需要开启以下两个插件，如图 1-216 所示。

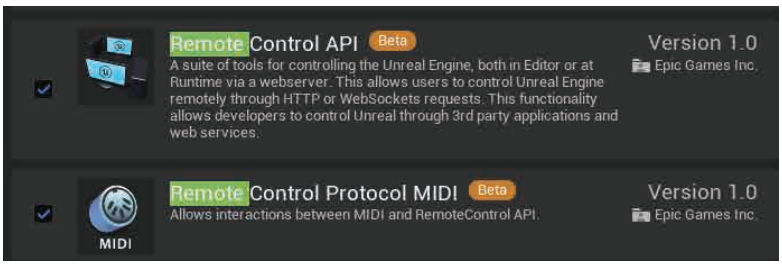

图 1-216　启用 Remote Control API 和 Remote Control Protocol MIDI 插件

重启 UE5 项目后，打开项目设置（Project Settings），搜索 midi，设置 Device Name 为 loopMIDI 软件里新建的那个端口名 loopMIDI Port，如图 1-217 所示。

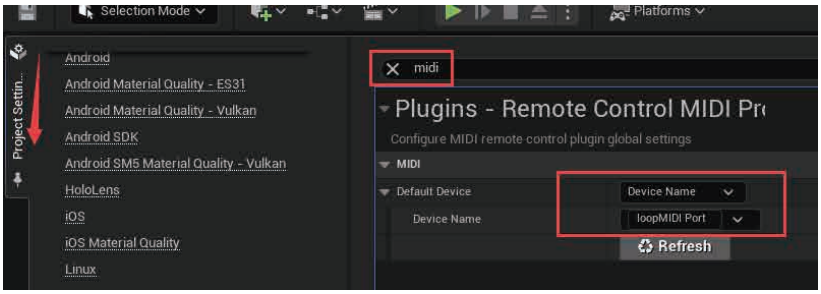

图 1-217　在 UE5 项目设置中设置 MIDI 设备名为 loopMIDI Port

然后，建立一个基础的关卡，取名为Level1，在关卡（也叫场景）中放置一个点光源（Point Light）。选中这个 Point Light，通过其细节面板设置它的 Light Color 为红色。在 Content Browser 中右击创建一个 Remote Control Preset，取名为 MIDI_RemoteControl，如图 1-218 所示。

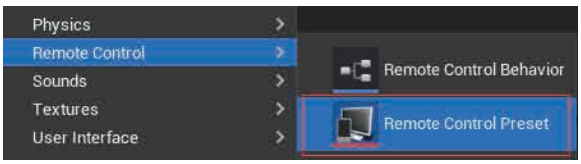

图 1-218　新建一个 Remote Control Preset

双击打开 MIDI_RemoteControl，把弹出的 Remote Control 窗口调小一些，确保场景中的点光源仍处于选中状态。单击它的细节面板中 Intensity 属性最右侧的三个小白点，从弹出的菜单里点选 Unexpose Property 表示将该属性暴露给 Remote Control（远程控制）。单击后 Intensity 属性会出现在 Remote Control 窗口里，如图 1-219 所示。

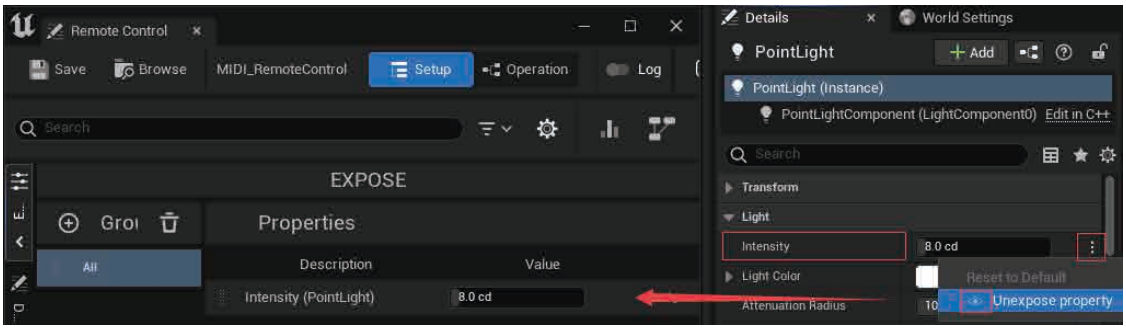

图 1-219　把 Point Light 的 Intensity 属性开放给远程控制

接下来，单击 Remote Control 窗口顶部右侧的 Log 开关按钮，此时如果再去单击 REAPER 里的虚拟琴键，所发出的 MIDI 信号就会在 Log 区域显示出来。从 Log 里可以看到每条信息都包含诸如事件类型 NoteOn、通道为 Channel 、信息数据 1 为 12、信息数据 2 为 127 等信息。注意：单击白色琴键的不同部位，MessageData2 的返回值是不同的，在白色琴键的底部单击时发出的 MessageData2 值为 127，如图 1-220 所示。

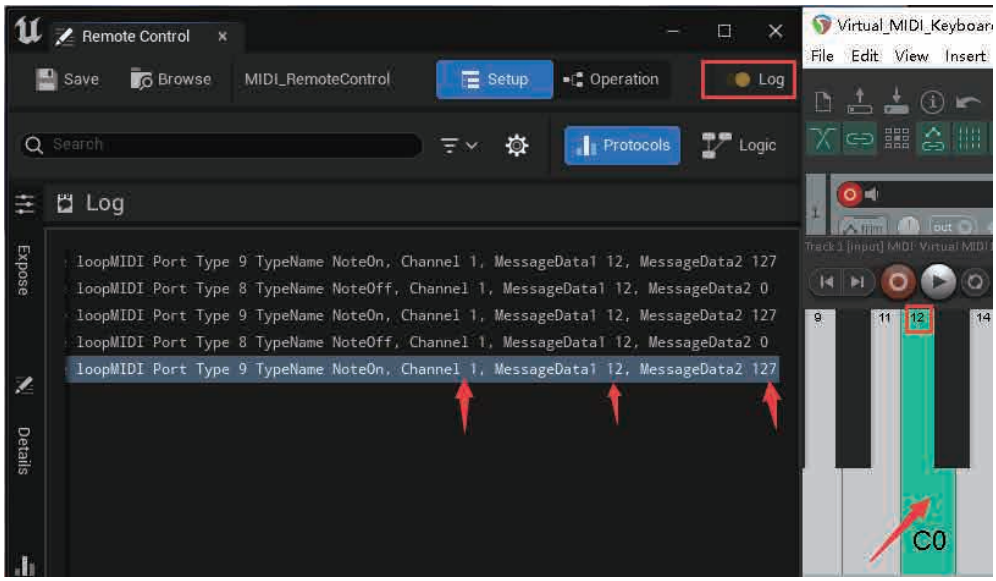

图 1-220　使用 Log 面板查看接收到的 MIDI 信息数据

到这里，就实现了让 UE5 接收音乐编辑软件发送过来的 MIDI 数据！通过对 Log 面板中各行数据的观察，可以很清晰地了解所接收到的数据中的各个参数的名称。

1.5.2　UE5处理MIDI信号的两种方式

既然 UE5 能接收到来自于外部的 MIDI 信号，那么就可以利用 MIDI 信号来控制 UE5。使用接收到的 MIDI 信号来驱动 UE5 中的对象，通常有两种方式：使用 Remote Control Preset 进行属性绑定和使用蓝图调用 MIDI。

1. 使用 Remote Control Preset 进行属性绑定

双击打开在上一节构建的 Remote Control Preset，切换到 Protocols（协议）标签下，点选 Intensity（强度）这个暴露出来的属性，单击 +Add Binding（添加绑定）按钮添加 MIDI 信号绑定，Event Type（事件类型）选择 Note On 表示琴键按下时，Channel 输入框里填写 1 而 Mapped Channel Id 输入框里填写 12，表示会观察 MIDI 信号，当信号的 Channel 为 1 而 MessageData1 为 12 时，就会让 Intensity 这个属性值在底部填写的 Ranges 区间里切换。如图 1-221 所示，当 Input 为 0 时，也就是当信号的 MessageData2 为 0 时，Intensity 这个属性将会变为 0。而当 Input 为 127 时，也就是当信号的 MessageData2 为 127 时，Intensity 的值将会变为 20，如图 1-221 所示。

注意：这里 Input 值为 0 和 127 都是指 MIDI 信号的 MessageData2 属性值。保存后，按下 12 号琴键，会看到 UE5 视口里的点光源开始发光。如果松开 12 号琴键，点光源就会熄灭。这就是一种比较简便的使用 MIDI 控制 UE5 场景物件的方法，如图 1-222 所示。

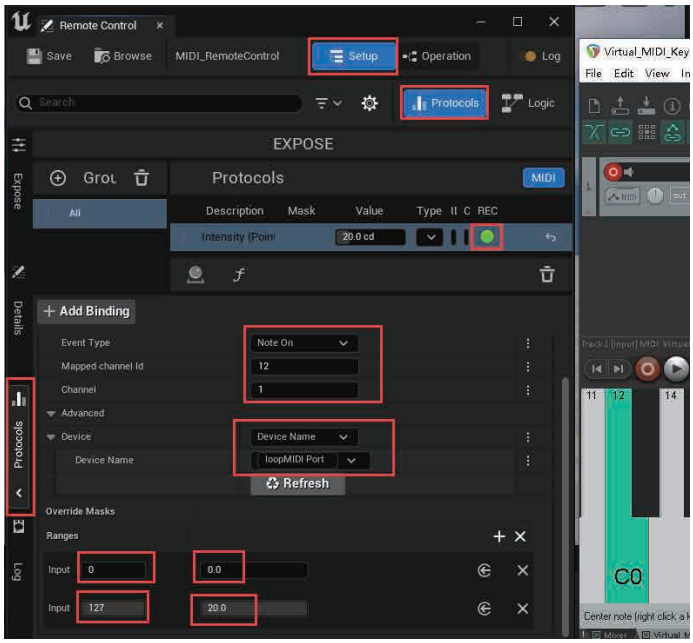

图 1-221　为 Intensity 属性绑定信息事件和不同信息的对应值

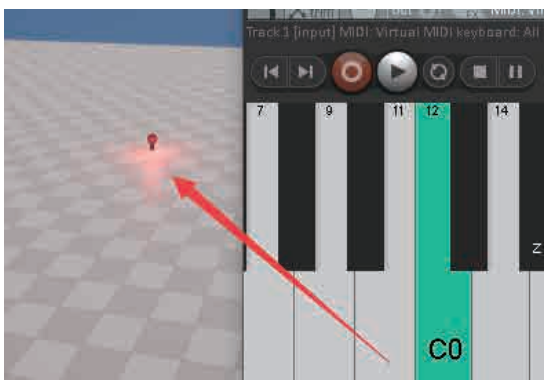

图 1-222　用虚拟琴键控制 Point Light 的亮度

2. 使用蓝图来读取 MIDI 信号

另外一种方法是通过蓝图来读取 MIDI 信号，可以试着在关卡蓝图中写入如图 1-223 所示的内容，通过 Create MIDIDevice Input Controller 节点来连接外部的 MIDI 控制器。

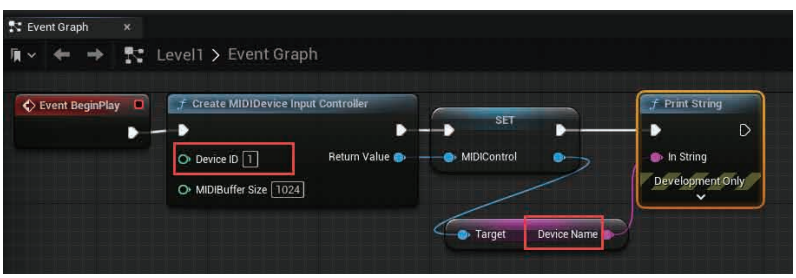

图 1-223　蓝图打印输出 MIDI 设备名称

运行 UE5，可以在视口中看到打印输出，内容是 loopMIDI Port。如果没有看到输出这个 loop MIDI Port，则可以保存 UE5，取消插件里的 Remote Control Protocol MIDI 和 Remote Control API，重新启动 UE5 再尝试。

为了了解更多使用蓝图读取 MIDI 的方法和技巧，可以增加如图 1-224 所示的蓝图节点。

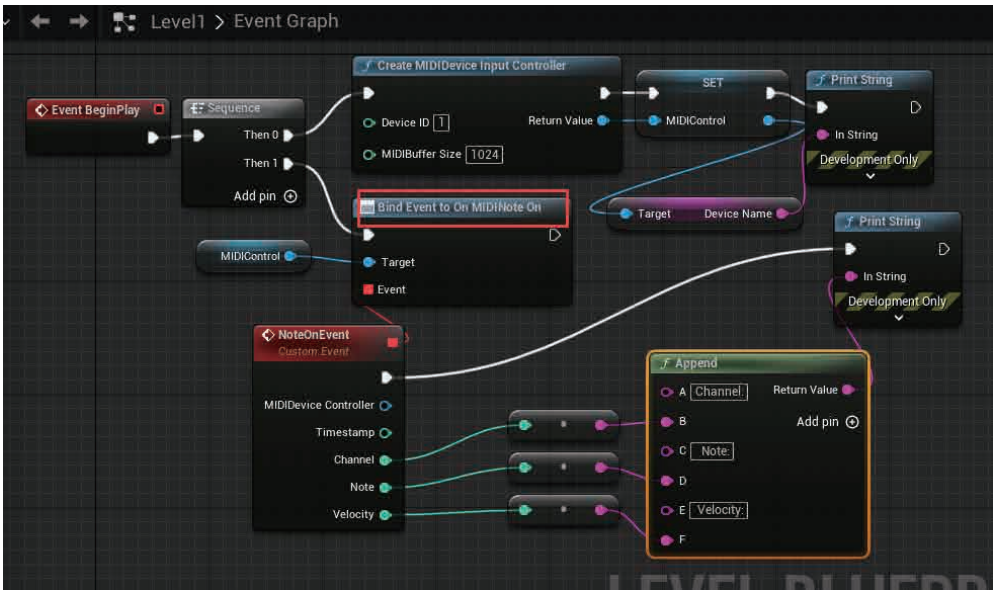

图 1-224　绑定 MIDI 设备的琴键按下事件

其中：MIDIControl 是一个变量，类型为 MIDIDevice Input Controller，用于存储外部的 MIDI 设备，如图 1-225 所示。

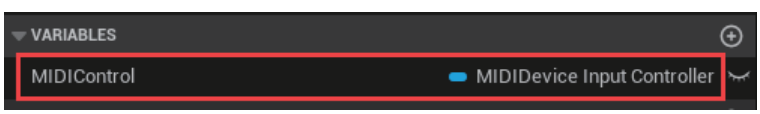

图 1-225　MIDIControl 变量的类型

它是在执行 Create MIDIDevice Input Controller 节点时，通过将其引脚 Return Value 拖出来后单击 Promote to variable（提升为变量）自动构建出来的，如图 1-226 所示。

图 1-226　将 Create MIDIDevice Input Controller 的返回值提升为变量

然后，使用了 Bind Event to MIDINote On 节点给 MIDIControl 绑定了琴键按下的事件，当事件发生时，会打印输出事件里所含带的 Channel Note Velocity 等信息，如图 1-227 所示。

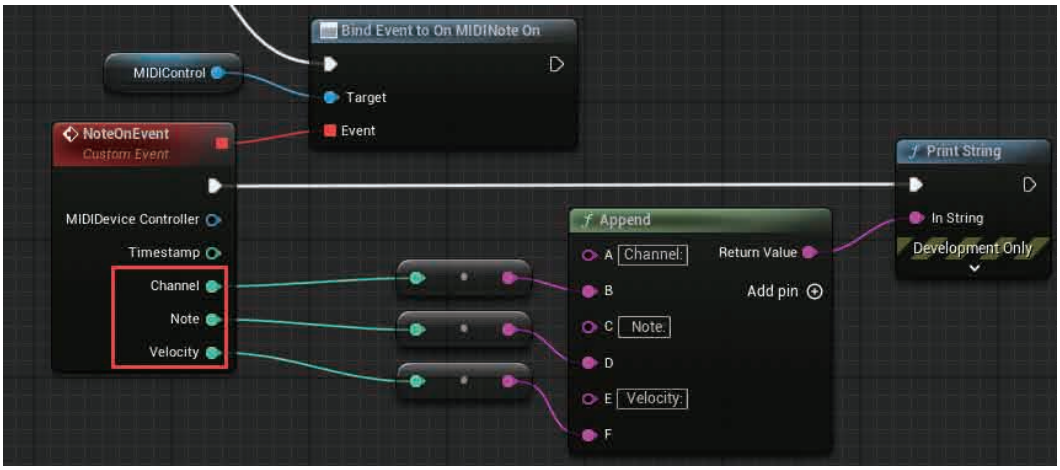

图 1-227　琴键按下时会打印输出 MIDI 数据信息

运行 UE5，按下虚拟琴键上的 12 号键，能看到了 UE5 视口左上角打印输出的三项数据内容，如图 1-228 所示。

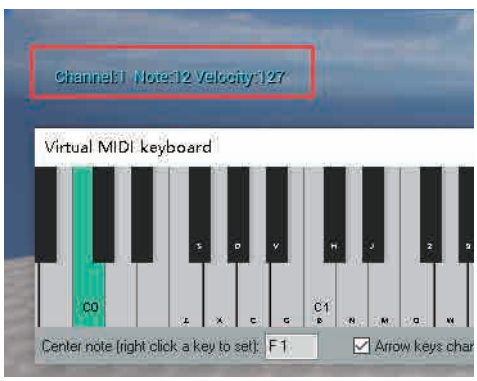

图 1-228　UE5 打印输出 MIDI 信息中的 Channel+Note+Velocity

据此可以明白，事件里收到的 Note 值其实就是之前在 Log 中看到的 MessageDate1 的值，而 Velocity 其实就是之前看到的 MessageDate2 的值。更进一步，可以把松开琴键的事件也绑定上来，如图 1-229 所示。

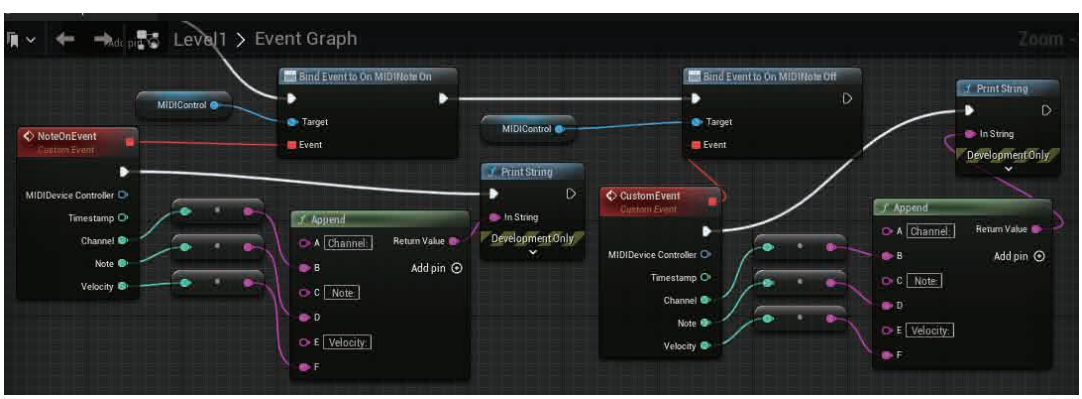

图 1-229　为 MIDI 设备绑定琴键松开事件

采用 Bind Event to MIDINote Off 节点可以为 MIDIControl 绑定松开琴键的事件。编译保存后测试，可以看到在松开 12 号琴键时，MIDI 数据里 Note 值为 12，Velocity 值为 0，如图 1-230 所示。

图 1-230　松开琴键时看 UE5 输出的 MIDI 信号数据

基于目前掌握的这些知识，可以进一步使用蓝图来控制灯的开关，如实现在按下琴键后开灯。具体的蓝图内容如图 1-231 所示。

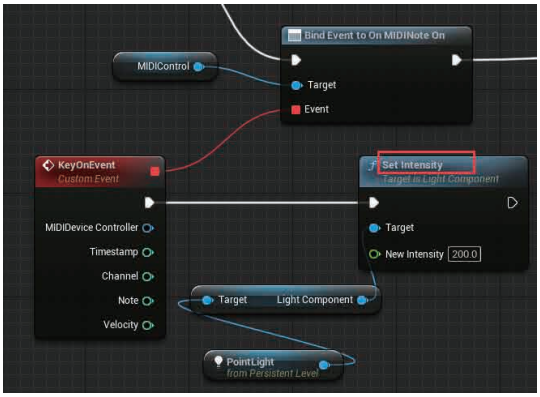

图 1-231　按下琴键后让灯的亮度变为 200

类似地，实现松开琴键后灭灯的蓝图内容如图 1-232 所示。

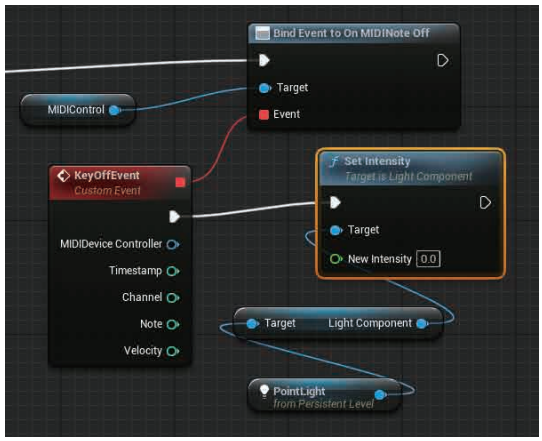

图 1-232　松开琴键后让灯的亮度为 0

编译关卡蓝图后运行 UE5，接着单击虚拟琴键，就可以看到开关灯的互动了，效果如图 1-233 所示。

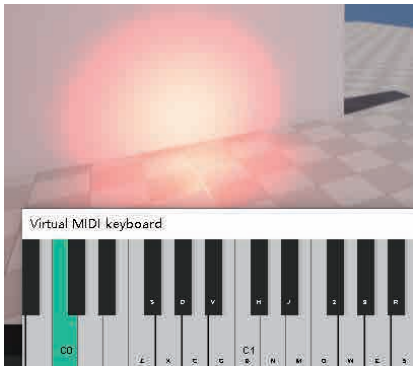

图 1-233　在 REAPER 里按动虚拟琴键测试 UE5

1.5.3　实例：使用nanoKONTROL2操作UE5

MIDI 控制器 nanoKONTROL2 是一款物美价廉的电音控制器，nanoKONTROL2 支持 USB 连接和蓝牙无线连接，可以通过它连接到计算机上直接向 UE5 发送 MIDI 数据，这样就不需要 loopMIDI 软件作为中转桥梁了。nanoKONTROL2 的外观如图 1-234 所示。

图 1-234　nanoKONTROL2 的整体外观

要用 nanoKONTROL2 来控制 UE5，需要在为它通电之前，先同时按住 SET 键和后退键，然后再插入电源线，当后退键开始闪烁时表示已经进入 DAW 模式，然后就可以松开这两个键了。之后再次使用 nanoKONTROL2 都会默认进入这个 DAW 模式，如图 1-235 所示。

图 1-235　第一次使用时先同时按住 SET 键和后退键然后再插入电源线

继续沿用上一节用到的 UE5 项目文件，此时已经不需要继续使用 loopMIDI 软件。关闭 loopMIDI 软件，运行 UE5，通过如图 1-236 所示的关卡蓝图部分，观察视口里打印输出的 MIDI 控制器的设备名称。Sequence 节点用于把复杂的蓝图内容分成多条执行线依次执行，让蓝图看起来更加整洁清晰。

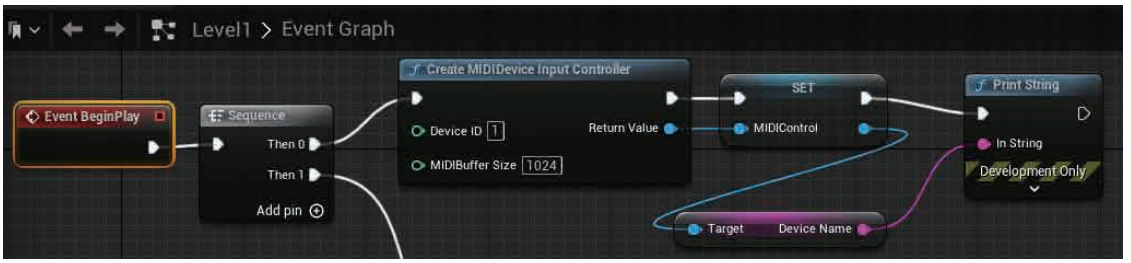

图 1-236　打印输出 MIDI 控制器的设备名称

UE5 会在视口左上角打印输出 MIDI 控制器的设备名称，如果能看到输出 nanoKONTROL2 字样，就说明 nanoKONTROL2 控制器已经成功连接到 UE5 了（如果没有看到的话，建议先连接好控制器硬件，然后再启动 UE5 项目尝试）。而针对 MIDI 控制器按键和松键的事件，则可以用手指尝试按下 nanoKONTROL2 设备上的各个键，通过视口可以相应看到打印输出的 Note 值。通过观察可以知道 nanoKONTROL2 设备上的 CYCLE 键下的五个方块按键所对应的 Note 值从左往右依次是 91、92、93、94 和 95，而这些方块

按键所对应的 Channel 值都是 1，如图 1-237 所示。

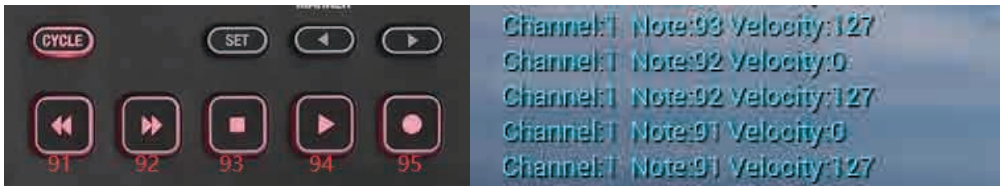

图 1-237　观察 nanoKONTROL2 设备上各个键按下和松开时打印输出的信息

nanoKONTROL2 控制器上除了按键外还有 8 个推拉键。推拉键在被用户滑动的时候会连续发送 MIDI 值到 UE5。要接收这样的信息值，需添加如图 1-238 所示的蓝图内容。

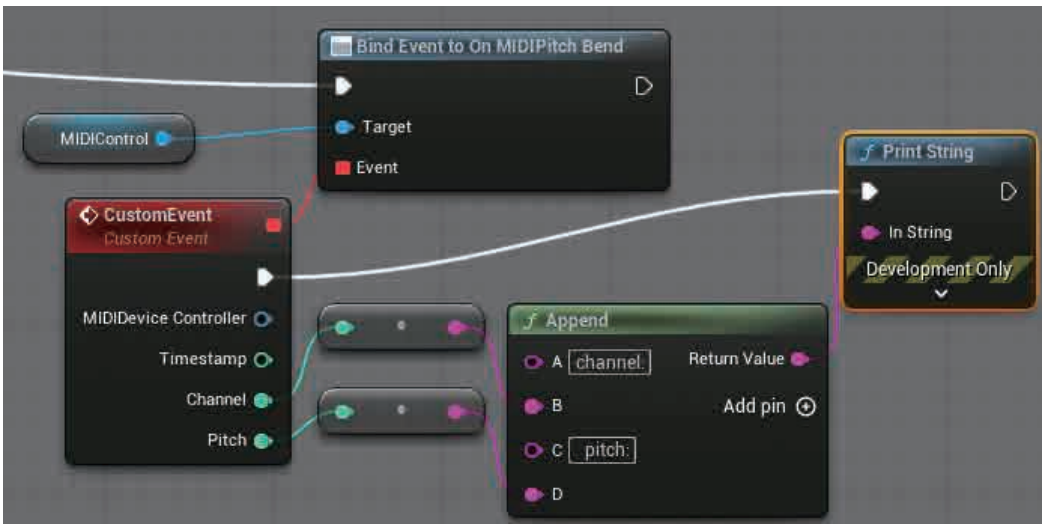

图 1-238　为推拉键绑定 Bend 事件接收相应数据并打印输出

在蓝图中使用 Bind Event to On MIDIPitch Bend 节点为推拉键的推拉事件添加绑定，于是 UE5 就能收到推拉事件发送来的 MIDI 信息了。在 UE5 运行的同时，如果上下滑动 nanoKONTROL2 控制器上的第一个推拉键，UE5 视口上会打印输出这个键对应的 Channel 值和 Pitch 值，如图 1-239 所示。

图 1-239　上下滑动 nanoKONTROL2 设备上的推拉键

对照输出值的变化，可以看到当把推拉键拉到最底部时，Pitch 值为 0，而当把它推到最顶部时 Pitch 值变为 16383。每个推拉键对应的 Channel 值都不同，键盘上从左到右的 8 个推拉键的 Channel 值分别为 1~8，如图 1-240 所示。

图 1-240　测试输出推拉键的 Channel 和 Pitch 值

接下来，可以使用推拉键对 UE5 中的物体对象进行控制。首先在场景里放入一个球体（Sphere），注意要在球体的细节面板里将 Mobility 属性设为 Movable（可移动的）。然后修改关卡蓝图，利用推拉事件里的 Pitch 值来影响球体的 Z 坐标。采用 Set Actor Location 节点来设置球体的 X、Y、Z 坐标值，在 New Location X 和 New Location Y 里分别输入球体在场景中的 X、Y 坐标值，如图 1-241 所示。

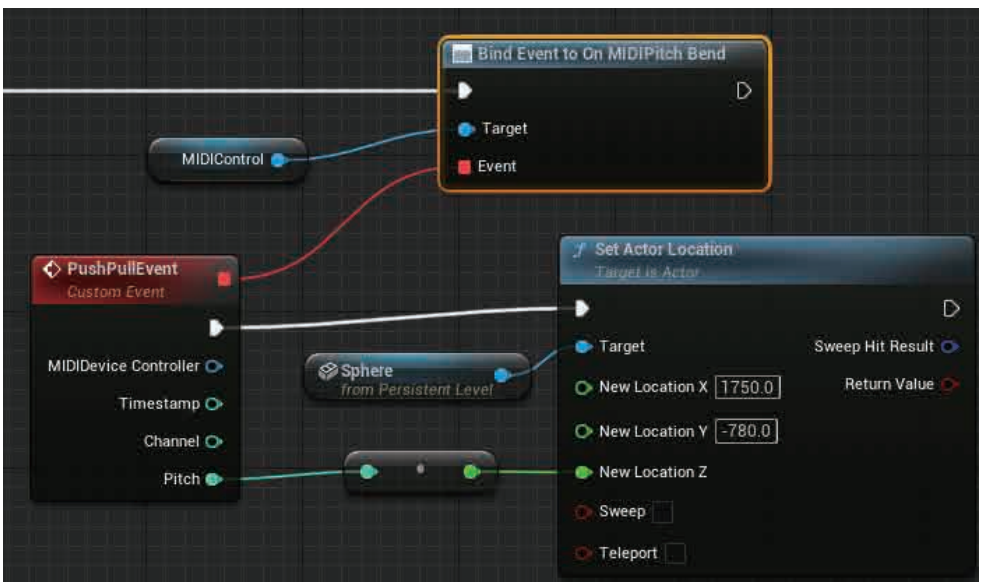

图 1-241　用接收到的 MIDI 数据中的 Pitch 值来控制球的 Z 坐标值

运行 UE5 后，就可以通过滑动 nanoKONTROL2 控制器上的推拉键来控制 UE5 中的球体上下运动了。滑动推拉键可以让球体的 Z 坐标值在 0~16383 变动，如果不希望球体被推得太高，可以利用 Map Range Clamped 节点将 Pitch 值映射在某个合理区间内，从而让球体只能在某个 Z 值范围内上下移动，如图 1-242 所示。

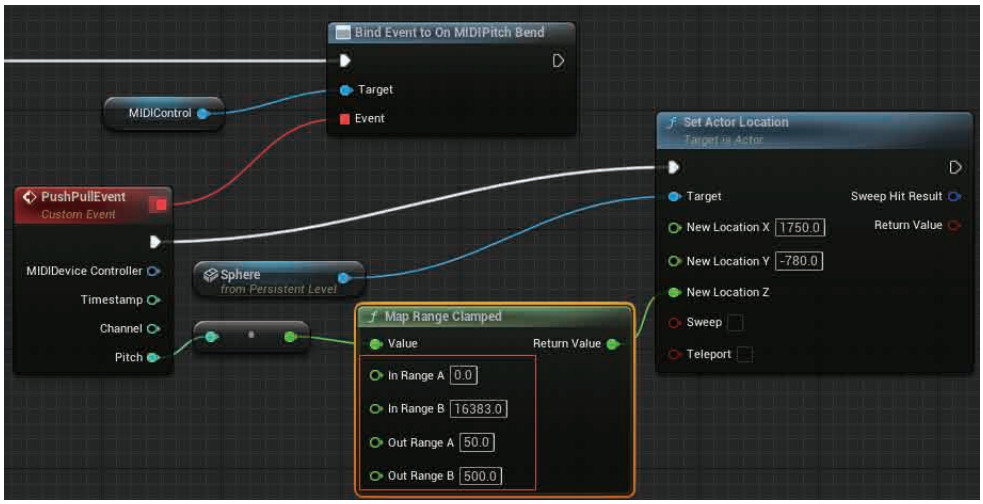

图 1-242　用 Map Range Clamped 节点将 0~16383 映射到 50~500

利用 Map Range Clamped 节点把 Ptich 值映射为 50~500 的对应值，这样球体就可以在一个比较合适的高度区间运动了。也就是说当 Pitch 值为 0 时，球的 Z 值将变为 50，而当 Pitch 值为 16383 时，球的 Z 值将变为 500。当 Pitch 值为 0~16383 的中间值时，Z 值也将变为 50~500 的中间值，这就是 Map Range Clamped 映射的作用。最终互动效果如图 1-243 所示。

图 1-243　滑动 nanoKONTROL2 设备上的推拉键来控制球体的升降

关于本节实例里涉及的详细的操作步骤，可以通过扫描下方二维码来观看，在视频教程里还进一步拓展演示了对更多物体进行控制的方法。