7INVENSUN 七鑫易维

——让世界读懂你的眼神——

aGlass™

虚拟现实眼球追踪配件

aGlass DK II

开发者使用手册

北京七鑫易维科技有限公司

2017 年 9 月

目录

1 aGlass SDK 介绍

七鑫易维虚拟现实眼球追踪配件 aGlass 是为 HTC Vive 虚拟现实头显配套使用的配件。

aGlass SDK 是为广大开发者提供的基于 aGlass 设备的软件开发工具包，在这个工具包的基础上，可以进行二次开发，以实现在 HTC Vive 头显中利用用户的注视点信息进行渲染、交互或分析等功能。

本文档的目的是给开发者一个产品的简单介绍和开发指导。

2 开发环境部署

2.1 SteamVR 安装部署

SteamVR 是 Valve 公司的虚拟现实服务程序。因为 HTC Vive 需要 SteamVR 的支持，使用 aGlass 之前，需要安装部署 SteamVR。

您可以在 https://support.steampowered.com/steamvr/HTC_Vive/上面下载 SteamVR，在 http://store.steampowered.com/上面找到安装和使用的在线说明。

2.2 aGlass Runtime 安装部署

aGlass 眼球追踪配件需要其配套的 Runtime 程序支持，该程序可以在 http://www.aglass.com/development 上面下载。

下载后，双击 Setup.exe 按照说明进行安装即可。

aGlass Runtime 支持的操作系统为 Windows 10 64bit。

2.3 aGlass Runtime 和 SDK 的升级

aGlass Runtime 具有在线自动更新检测和升级功能，在 Runtime 启动时运行。您也可以单击右键菜单中的“帮助->检查更新”主动进行检测。

可以在用户目录 C:\Users\YourUserName\AppData\Local\aGlassRuntime\（注意将 YourUserName 换成您自己的 Windows 用户名）中查看 What’s New.txt 文件，确认 SDK 是否有升级。

如果 SDK 有升级，可以在安装目录 aGlass\Client and Plugins\（绝对路径取决于您安装的路径）中获取最新版的 aGlassClient.dll 和最新版的开发引擎插件（Plugin）。

SDK 文档升级请在 http://www.aglass.com/development 上查看最新版。

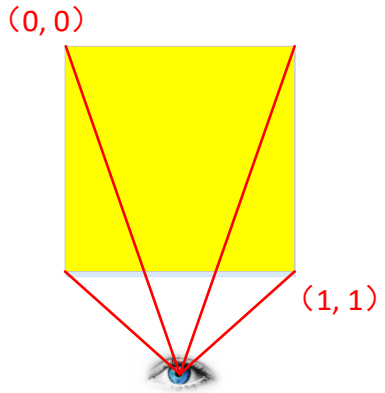
3 基本概念和使用

3.1 注视点和坐标系统

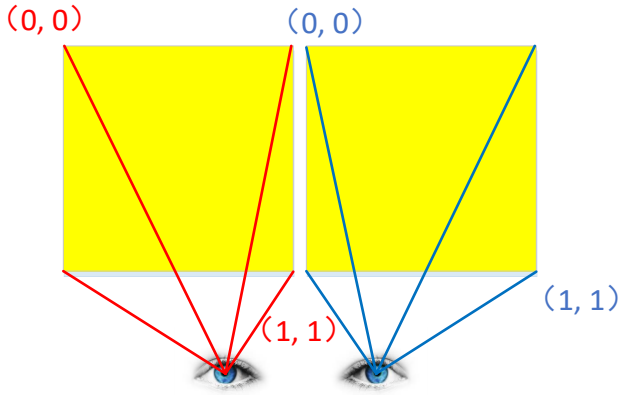
aGlass 的基本功能是追踪使用者的注视点。

注视点是使用者在佩戴 HTC Vive 头显过程中眼睛的视线屏幕的交点，使用的坐标系是屏幕坐标系。

HTC Vive 屏幕分为两部分，每只眼睛对应半个屏幕。对于 aGlass 一只眼睛的配件，映射到分辨率为 1080 像素*1200 像素的半个屏幕上。aGlass SDK 对注视点坐标进行了归一化：其左上角坐标为（0,0），右下角坐标为（1,1），例如（0.5, 0.5）映射到屏幕上位置的像素坐标为（540, 600）。

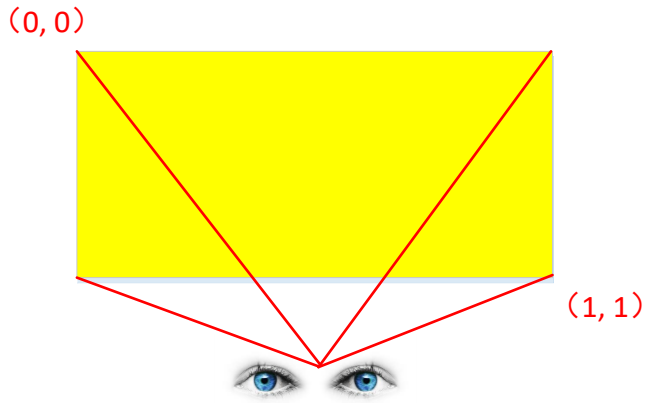


单屏幕的坐标系



双屏幕的坐标系

因为两只眼睛的注视点分别在各自半个屏幕上的映射相同，所以双眼注视点映射到 VR 场景中也采用相同的坐标系。



双眼注视点在 HTC Vive 头显中的坐标系

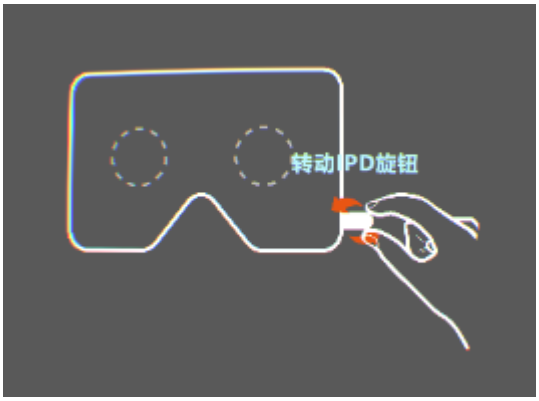
3.2 注视点校准

因为每个人的眼睛特征不同，所以用户使用前需要进行注视点校准。虽然也可以使用默认校准数据，但是仍然建议用户进行校准，因为校准之后才能获得更加准确的注视点信息。

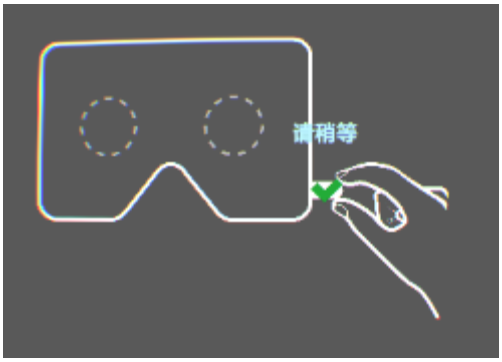
aGlass Runtime 提供一个默认的校准程序，您可以通过单击菜单项中的校准启动。

3.2.1IPD 调节

带好 Vive 头戴显示器后，视野中会出现瞳距（IPD）调节提示，如下图所示：



请按照提示的旋转方向，旋转 Vive 的 IPD 旋钮。出现如下界面后停止旋转：

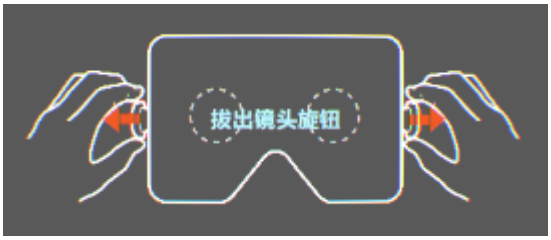


稍等后进入下一个环节。

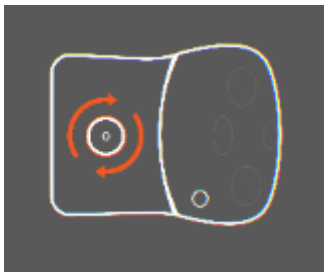
3.2.2镜头距离调节

完成 IPD 调节后，进入镜头距离调节环节。如果您的镜头距离是合适的，这个环节会跳过。

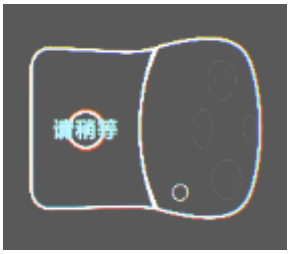
视野中首先会提示您拔出 Vive 的镜头距离旋钮，如下图所示：



然后请您按照视野中的 Vive 镜头距离旋钮的旋转方向，调节镜头距离：

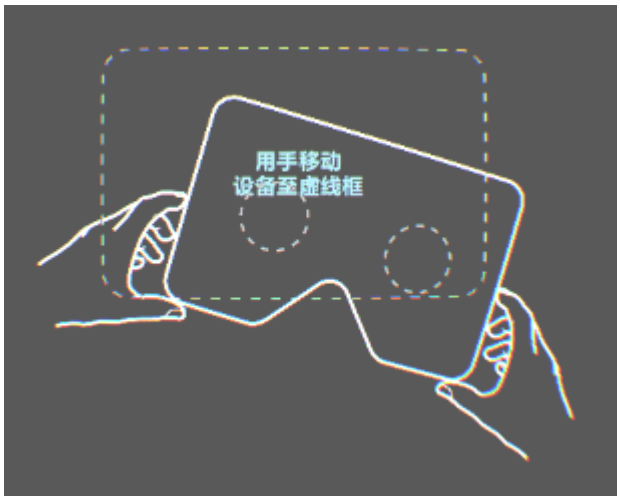


当出现如下界面时，请您停止旋转，稍等后进入下一个环节：

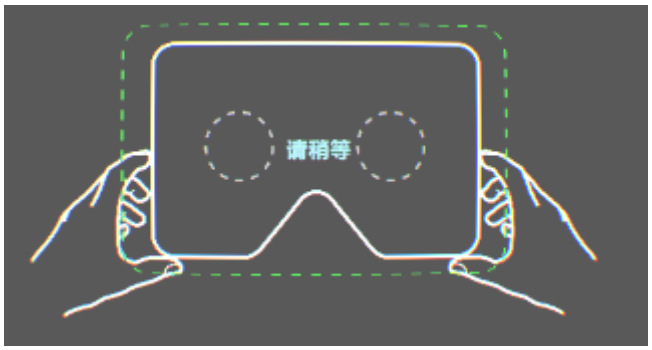


3.2.3佩戴位置调节

完成镜头距离调节后，进入佩戴位置调节，即调节 Vive 头戴显示器和脸部之间的相对位置。其提示界面如下图所示：



当您移动至适当位置后，界面变为下图：



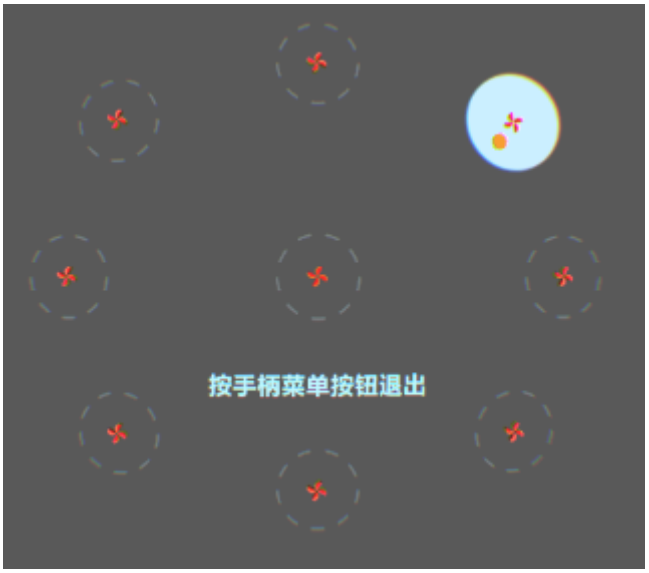
稍候您会自动进入注视点校准环节。

3.2.4注视点校准

校准时，您需要一直注视出现的校准点，如下图所示：

3.2.5校准验证

校准完成后会进入“校准验证”界面，显示多个校验点和用户的注视点标识，您可以通过该界面判断校准是否准确。界面如下图所示：



在该界面，您可以单击 Vive 手柄的菜单按钮或键盘 ESC 键退出。

3.3 用户管理

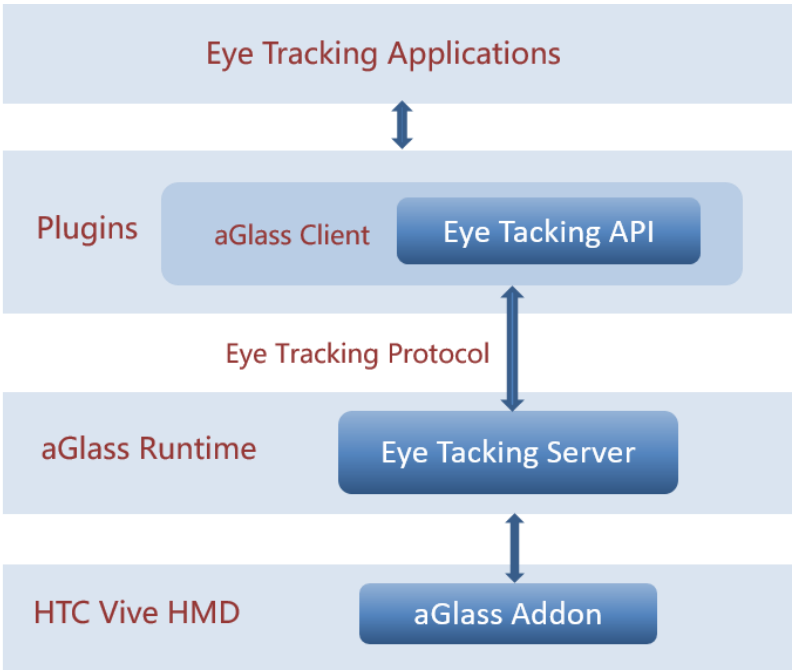
系统具有默认用户账号 Guest，但是建议建立自己的用户账号。

在右键菜单中单击“添加用户”可新建用户账号。新建账号会自动进入注视点校准程序。

使用过程中，如果更换使用者，请切换用户账号。

4 系统架构

aGlass 产品的系统架构如下图所示：



整个系统分为 4 部分：

- 硬件部分
- 眼球追踪服务器
- 眼球追踪客户端
- 眼球追踪应用

4.1 硬件部分

最底层为硬件部分，即装配在 Vive 中的 aGlass 配件。

4.2 眼球追踪服务器

aGlass Runtime 为安装于 PC 机端的软件部分，其主要功能是实现眼球追踪的算法。

软件整体设计为 C/S 架构，所以 aGlass Runtime 的主体为一个 Server，通过基于 TCP/IP 协议的 Eye Tracking Protocol 向 Client 端提供眼球追踪数据。

4.3 眼球追踪客户端

七鑫易维提供一个基于 C++的客户端程序 aGlass Client。aGlass Client 提供开放的 Eye Tracking API，供开发者使用。

为了简化开发者的工作，七鑫易维提供多种主流引擎的 Plugin，如 Unity Plugin、Unreal Plugin 等，并提供范例程序。基于这些 Plugin 进行应用开发更加容易上手。

4.4 眼球追踪应用

眼球追踪应用即具有 aGlass 眼球追踪功能的 VR 应用。

开发者可以基于 Plugin、Eye Tracking API 两种方式开发上层应用，建议使用 Plugin 进行开发。

5 Client API

aGlass Client 从安装目录\aGlass\Client and Plugins\ 内获取，文件夹名称为 Client。

5.1 API 说明

aGlass Client 为开发者提供 C++接口，如下所示：

分类	名称	功能	备注
回调函数	aGlass_eye_callback	用于获取眼球数据的回调函数	
	aGlass_event_callback	用于获取 aGlass 设备事件的回调函数	
设备相关	aGlass_init	初始化	
	aGlass_release	清理释放	
	aGlass_start_device	启动眼球追踪设备	
	aGlass_stop_device	停止眼球追踪设备	
追踪相关	aGlass_start_eye_callback	启动眼球数据回调	
	aGlass_stop_eye_callback	停止眼球数据回调	
	aGlass_start_tracking	启动跟踪	
	aGlass_stop_tracking	停止跟踪	
数据相关	aGlass_get_eye_data_int	获取整数类型眼球数据项	
	aGlass_get_eye_data_int64	获取 64 位整数类型眼球数据项	
	aGlass_get_eye_data_float	获取浮点数类型眼球数据项	
	aGlass_get_eye_data_byte	获取二进制字节流类型眼球数据项	

5.1.1回调函数

5.1.1.1 眼球数据回调

```
typedef void (AGLASS_CALL * aGlass_eye_callback) (const aGlassEyeData* eye_data, void* user_data)
```

功能	
用于获取眼球数据的回调函数	
参数	
const aGlassEyeData* eye_data	眼球数据
void* user_data	用户数据

5.1.1.2 设备事件回调

```
typedef void (AGLASS_CALL* aGlass_event_callback)(int event_code, void* user_data)
```

功能	
用于获取 aGlass 设备事件的回调函数	
参数	
int event_code	事件代码
void* user_data	用户数据

5.1.2设备相关

5.1.2.1 初始化

aGlassReturnCode AGLASS_CALL aGlass_init(const aGlassInitParam* param)	
功能	
初始化	
参数	
const aGlassInitParam* param	参数
返回值	
参考第 5.2 节“函数返回值”	

5.1.2.2 清理释放

aGlassReturnCode AGLASS_CALL aGlass_release(void)	
功能	
清理释放	
返回值	
参考第 5.2 节“函数返回值”	

5.1.2.3 启动眼球追踪设备

aGlassReturnCode AGLASS_CALL aGlass_start_device (aGlass_event_callback callback, void* user_data)	
功能	
启动眼球追踪设备	
参数	
aGlass_event_callback callback	事件回调函数
void* user_data	用户地址空间的变量的指针
返回值	
参考第 5.2 节“函数返回值”	

5.1.2.4 停止眼球追踪设备

aGlassReturnCode AGLASS_CALL aGlass_stop_device (void)
功能
停止眼球追踪设备
返回值

参考第 5.2 节“函数返回值”	

5.1.3追踪相关

5.1.3.1 启动眼球数据回调

aGlassReturnCode AGLASS_CALL aGlass_start_eye_callback(aGlass_eye_callback callback, void* user_data)	
功能	
启动眼球数据回调	
参数	
aGlass_eye_callback callback	回调函数指针
void* user_data	用户数据
返回值	
参考第 5.2 节“函数返回值”	

5.1.3.2 停止眼球数据回调

aGlassReturnCode AGLASS_CALL aGlass_stop_eye_callback(void)	
功能	
停止眼球数据回调	
返回值	
参考第 5.2 节“函数返回值”	

5.1.3.3 启动追踪

aGlassReturnCode AGLASS_CALL aGlass_start_tracking(void)	
功能	
启动跟踪	
返回值	
参考第 5.2 节“函数返回值”	

5.1.3.4 停止追踪

aGlassReturnCode AGLASS_CALL aGlass_stop_tracking(void)
功能
停止跟踪

返回值	
参考第 5.2 节“函数返回值”	

5.1.4数据相关

5.1.4.1 获取整数类型眼球数据项

aGlassReturnCode AGLASS_CALL aGlass_get_eye_data_int(const aGlassEyeData* data, aGlassEyeDataItem id, int* value)	
功能	
获取整数类型眼球数据项	
参数	
const aGlassEyeData* data	眼球数据
aGlassEyeDataItem id	数据项
int* value	整数值
返回值	
参考第 5.2 节“函数返回值”	

5.1.4.2 获取 64 位整数类型眼球数据项

AGLASS_API aGlassReturnCode AGLASS_CALL aGlass_get_eye_data_int64(const aGlassEyeData* data, aGlassEyeDataItem id, long long* value);	
功能	
获取 64 位整数类型眼球数据项	
参数	
data	眼球数据
id	数据项
value	64 位整数值
返回值	
参考第 5.2 节“函数返回值”	

5.1.4.3 获取浮点数类型眼球数据项

aGlassReturnCode AGLASS_CALL aGlass_get_eye_data_float(const aGlassEyeData* data, aGlassEyeDataItem id, float* value)
功能
获取浮点数类型眼球数据项
参数

const aGlassEyeData* data	眼球数据
aGlassEyeDataItem id	数据项
float* value	浮点数值
返回值	
参考第 5.2 节“函数返回值”	

5.1.4.4 获取二进制字节流类型眼球数据项

`aGlassReturnCode AGLASS_CALL aGlass_get_eye_data_byte(const aGlassEyeData* data, aGlassEyeDataItem id, unsigned char* value, int size)`	
功能	
获取二进制字节流类型眼球数据项	
参数	
const aGlassEyeData* data	眼球数据
aGlassEyeDataItem id	数据项
unsigned char* value	数据缓冲区首地址（由调用者开辟的内存空间）
int size	缓冲区大小
返回值	
参考第 5.2 节“函数返回值”	

5.2 函数返回值

函数返回值 aGlassReturnCode 定义如下：

值	错误码	说明
0	AGLASS_OK	成功
-1	AGLASS_FAILED	失败
-2	AGLASS_NO_RESPONSE	无响应
-3	AGLASS_PARAM_ERROR	参数错误
-101	ENCRYPTION_DEVCE_NO_FOUND	加密设备未找到
-102	ENCRYPTION_DEVCE_NO_MATCH	加密设备不匹配
-103	ENCRYPTION_DEVCE_PWD_ERROR	加密设备密码错误
-104	ENCRYPTION_DEVCE_PWD_EXPIRED	加密设备过期
-105	ENCRYPTION_DEVCE_NO_AUTHORIZATION	加密设备未授权
-106	DEVICE_MULTIPLE_OR_NO_FOUND	多设备或设备未找到
-107	READ_CONFIG_FILE_FAILED	读取配置文件失败

-108	READ_BASE_FILE_FAILED	读取基础数据文件失败
-109	ENCRYPTION_DEVICE_NO_SUPPORTED	不支持该加密设备

5.3 枚举和结构体定义

5.3.1设备事件

枚举名称： aGlassDeviceEventCode

功能：设备事件

枚举值	值	含义
LeftDeviceDisconnect	0	左眼配件未连接
LeftDeviceConnect	1	左眼配件已连接
RightDeviceDisconnect	2	右眼配件未连接
RightDeviceConnect	3	右眼配件已连接

5.3.2眼球数据项

枚举名称： aGlassEyeDataItem

功能：眼球数据

枚举值	数据类型	含义	备注
undefine	——	未定义	
timestamp	long long	时间戳	
gazeX	float	双眼注视点 X 坐标	
gazeY	float	双眼注视点 Y 坐标	
gazeVaild	int	注视点数据有效性	
eyeValid	int	眼球数据有效性	

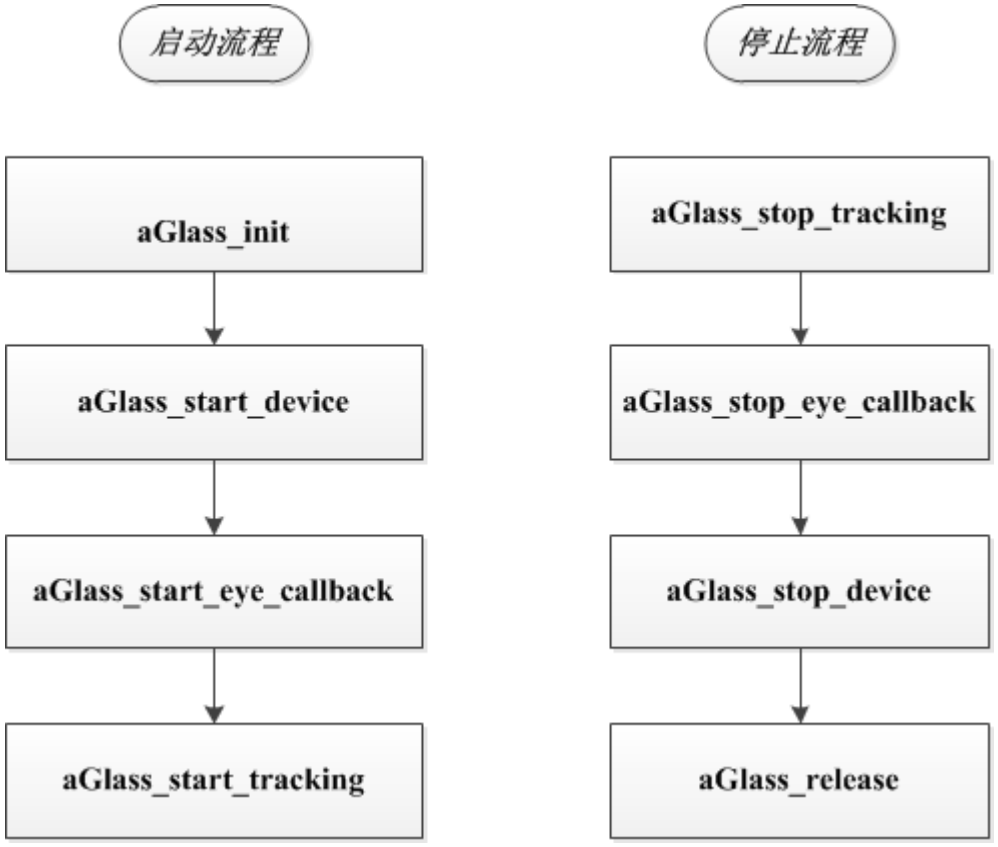
5.3.3初始化参数

结构体名称：aGlassInitParam

功能：初始化参数。

类型	参数	含义
int	devEventPort	用于接收设备事件，使用本地端口。选择一个非系统端口且不被其他程序占用。 该端口号不能与 eyeDataPort 端口号相同。
int	eyeDataPort	用于接收眼球数据，使用本地端口。选择一个非系统端口且不被其他程序占用。 该端口号不能与 devEventPort 端口号相同。
int	calibProcessPort	校准点进度接收端口
int	calibFinishPort	校准点结束状态接收端口

5.4 API 使用流程



5.5 眼球数据的获取

5.5.1时间戳 timestamp

时间戳 timestamp 的数据类型在 C++中表示为 long long 64 位的长整型，通过 aGlass_get_eye_data_byte 接口来获取，缓冲区大小大于等于 8 bytes， C# 需要转换，C\C++可直接内存拷贝至 long long 类型的变量。

时间戳为 UTC（Coordinated Universal Time，世界统一时间）格式，单位为微秒。

5.5.2注视点坐标

注视点坐标数据包括：

- gazeX：双眼注视点 X 坐标
- gazeY：双眼注视点 Y 坐标

注视点坐标数据在 C++中均表示为 float 类型，通过 aGlass_get_eye_data_float 接口来获取。注视点坐标数据为归一化数据，即从 0 到 1 之间的浮点数。

5.5.3注视点有效性 gazeVaild

注视点有效性 gazeValid 为整型数据，通过 aGlass_get_eye_data_int 接口来获取。

gazeValid 为 0 时表示注视点无效，gazeValid 为 1 时表示注视点有效。

5.5.4眼球数据有效性 eyeValid

眼球数据有效性 eyeValid 为整型数据，通过 aGlass_get_eye_data_int 接口来获取。

eyeValid 为 0 时表示眼球数据无效，eyeValid 为 1 时表示眼球数据有效。

6 Unity Plugin 及其范例工程说明

Unity 是 Unity 公司开发的一个跨平台游戏引擎，可以开发 2D 或 3D 的游戏。使用 Unity 引擎可以开发游戏、仿真、训练软件等应用产品。

VR 眼球追踪配件 aGlass DK II 为 Unity 2017.1（或更高版本）提供开发插件（Plugin）和范例工程。Unity 更详细的信息可参考 https://docs.unity3d.com/Manual/GettingStarted.html。

6.1 Unity Plugin

6.1.1获取 Unity Plugin

Unity Plugin 可以通过以下方法获取：

- 安装路径 aGlass\Client and Plugins\Unity\ 中的 Plugins 文件夹。
- 从 http://www.aglass.com/development 下载 UnityPlugin.rar，下载后解压，文件夹 Plugins 即为 Unity Plugin。

- 从 http://www.aglass.com/development 下载 aGlass Unity Demo 工程 UnityDemo.rar，下载至本地后解压，在路径 UnityDemo\Assets\ 中，文件夹 Plugins 即为 Unity Plugin。

6.1.2API 说明

6.1.2.1 开启设备

定义：aGlass.Instance.aGlassStart()

功能：开启 aGlass 设备

返回值：参考 6.2 节“函数返回值”

6.1.2.2 关闭设备

定义：aGlass.Instance.aGlassStop()

功能：关闭 aGlass 设备

返回值：参考 6.2 节“函数返回值”

6.1.2.3 判断注视点有效性

定义：aGlass.Instance.GetEyeValid()

功能：判断注视点是否有效

返回值：

- True：有效
- False：无效

6.1.2.4 获取注视点坐标

定义：aGlass.Instance.GetGazePoint()

功能：获取注视点坐标

返回值：

- float aGlass.Instance.GetGazePoint().x：注视点数据 X 坐标
- float aGlass.Instance.GetGazePoint().y：注视点数据 Y 坐标

6.2 范例工程

aGlass DK II 提供基于 Unity Plugin 的 Unity 范例工程。

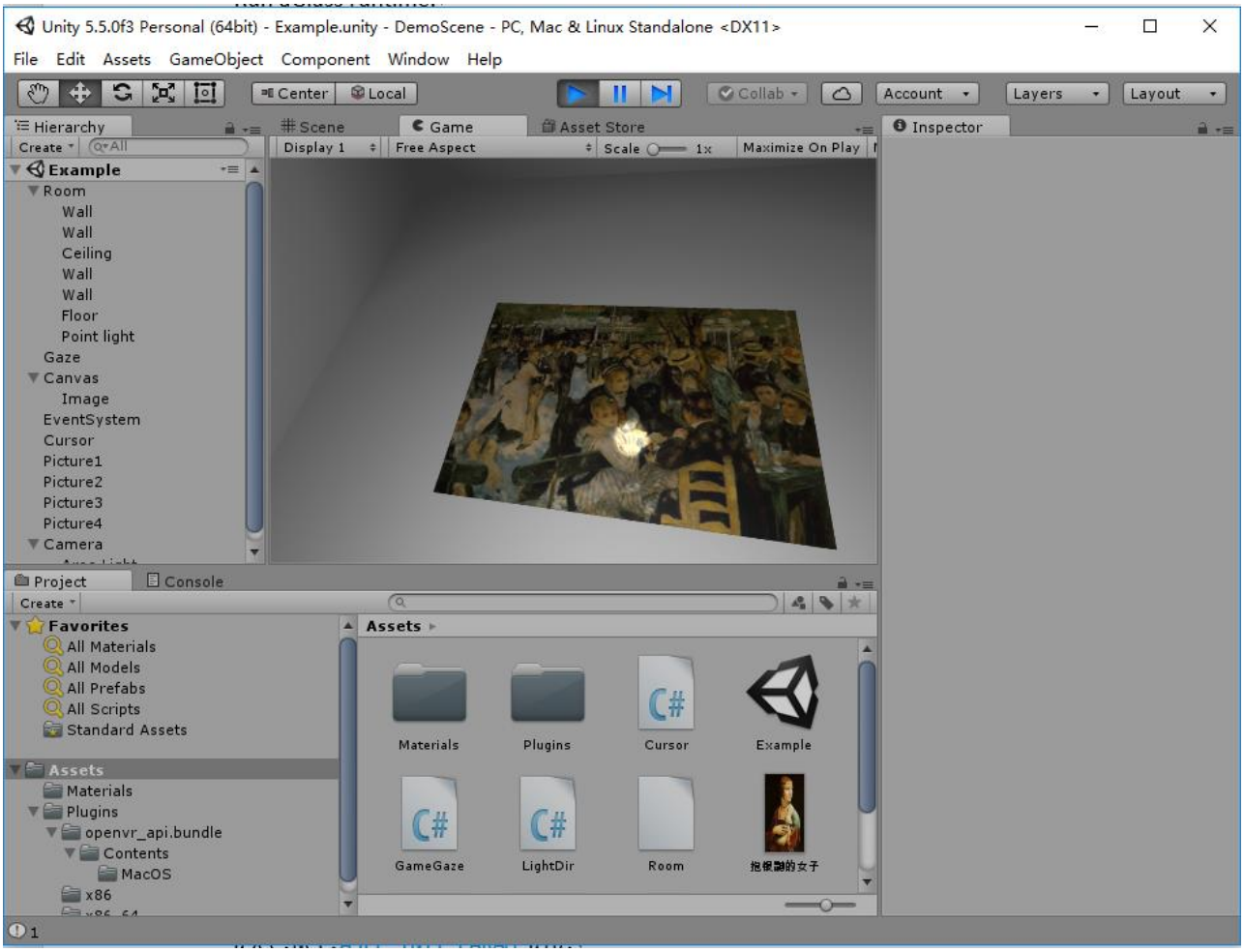
从 http://www.aglass.com/development 下载名称为 UnityDemo.rar 的文件并解压，该工程即为 Unity 范例工程。

6.2.1准备工作

- 安装 Unity 2017.1 或更高版本
- 连接好 HTC Vive 设备
- 连接好 aGlass 配件
- 启动 SteamVR
- 启动 aGlass Runtime
- 使用 Unity 打开范例工程 UnityDemo
- 在 Asset Store 中查找 SteamVR Plugin，下载并导入 Unity 工程
- 点击 Edit->Project Settings->Player->Virtual Reality Supported->Virtual Reality SDKs->OpenVR，启用 OpenVR

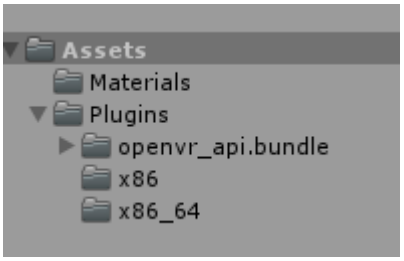
6.2.2运行范例程序

在 Unity 中运行范例程序 UnityDemo。场景为一个房间，房间的每面墙上都有一副画作，有一束探照灯照在注视点的方向，当你注视点移动时，探照灯随着一起移动，如下图所示：



6.2.3工程结构

在 Unity 的 Project 选项卡中，在 Assets 文件夹中，可以看到整个工程用到的所有资源，如下图所示：



文件夹 Plugins 内的资源是 Unity Plugin。

文件夹 Materials 里面包含范例工程中用到的材质。

Assets 根目录下，有一些范例场景和相关资源，如下图所示：

6.2.4范例工程脚本说明

6.2.4.1 GameGaze.cs

GameGaze.cs 是范例工程的核心脚本，其功能是获取用户的注视点数据。

6.2.4.1.1 启动 aGlass

```
void Start ()
{
    print(aGlass.Instance.aGlassStart());
}
```

6.2.4.1.2 ESC 退出和注视点数据打印

```
void Update ()
{
    if (Input.GetKeyDown (KeyCode.Escape) && (SceneManager.GetActiveScene ().buildIndex
    == 0 || SceneManager.GetActiveScene ().buildIndex == 1)) {
        Application.Quit ();
    }
    if (aGlass.Instance.GetEyeValid())
    {
        print(aGlass.Instance.GetGazePoint().x);
        print(aGlass.Instance.GetGazePoint().y);
    }
 }
```

6.2.4.1.3 探照灯跟随注视点

```
public void GetPos (GameObject c, ref float cx, ref float cy)
{
    if (aGlass.Instance.GetEyeValid()) {
        count = 0;
        if (!c.activeSelf) {
            c.SetActive (true);
        }
        cx = aGlass.Instance.GetGazePoint().x;
```

```
            cy = aGlass.Instance.GetGazePoint().y;
        } else {
            count++;
            if (count > 10 && c.activeSelf) {
                c.SetActive (false);
            }
        }
    }
```

6.2.4.1.4 程序结束时退出眼球追踪

```
    void OnDestroy ()
    {
        print(aGlass.Instance.aGlassStop());
     }
```

6.2.4.2 Cursor.cs

脚本 Cursor.cs 的功能是获取注视点数据并将其转化为光标数据。

6.2.4.2.1 获取对象“Gaze”

```
    void Start()
    {
        gaze = GameObject.Find ("Gaze").GetComponent<GameGaze>();
    }
```

6.2.4.2.2 更新光标位置

```
    void Update () {
        gaze.GetPos (light.gameObject, ref x, ref y);
        cursor.transform.localPosition = new Vector2 ((x - 0.5f) * 1512, (0.5f - y) * 1680);
    }
```

6.2.4.3 LightDir.cs 和对象击中

脚本 LightDir.cs 的功能是创建一个探照灯，指向注视点光标位置。

在该脚本中，实现了“对象击中”，当用户的注视点观察油画旁边两个立方体时，视线与立方体发生“碰撞”，程序会返回该立方体的 ID，同时立方体的颜色会发生变化。实现代码如下：

```
    void Update ()
    {
        lighter.transform.LookAt(this.gameObject.transform);
        leftCube.GetComponent<Renderer>().material.color = Color.white;
        rightCube.GetComponent<Renderer> ().material.color = Color.white;
        RayDetect();
     }

    void RayDetect()
    {
        print(1);
        RaycastHit hit;
        if (Physics.Raycast(cam.transform.position, this.gameObject.transform.position –
```

```
            cam.transform.position, out hit))
        {
            if (hit.transform.gameObject.tag == "Box")
            {
                hit.transform.gameObject.GetComponent<Renderer>().material.color = Color.red;
                print(hit.transform.gameObject.name);
            }
        }
    }
```

7 Unreal Plugin

7.1 获取 Unreal Plugin

Unreal Plugin 可以通过以下方法获取：

- 安装路径 aGlass\Client and Plugins\Unreal\ 中的 Plugins 文件夹。
- 从 http://www.aglass.com/development 下载 UnrealPlugin.rar，下载后解压即为 Unreal Plugin。

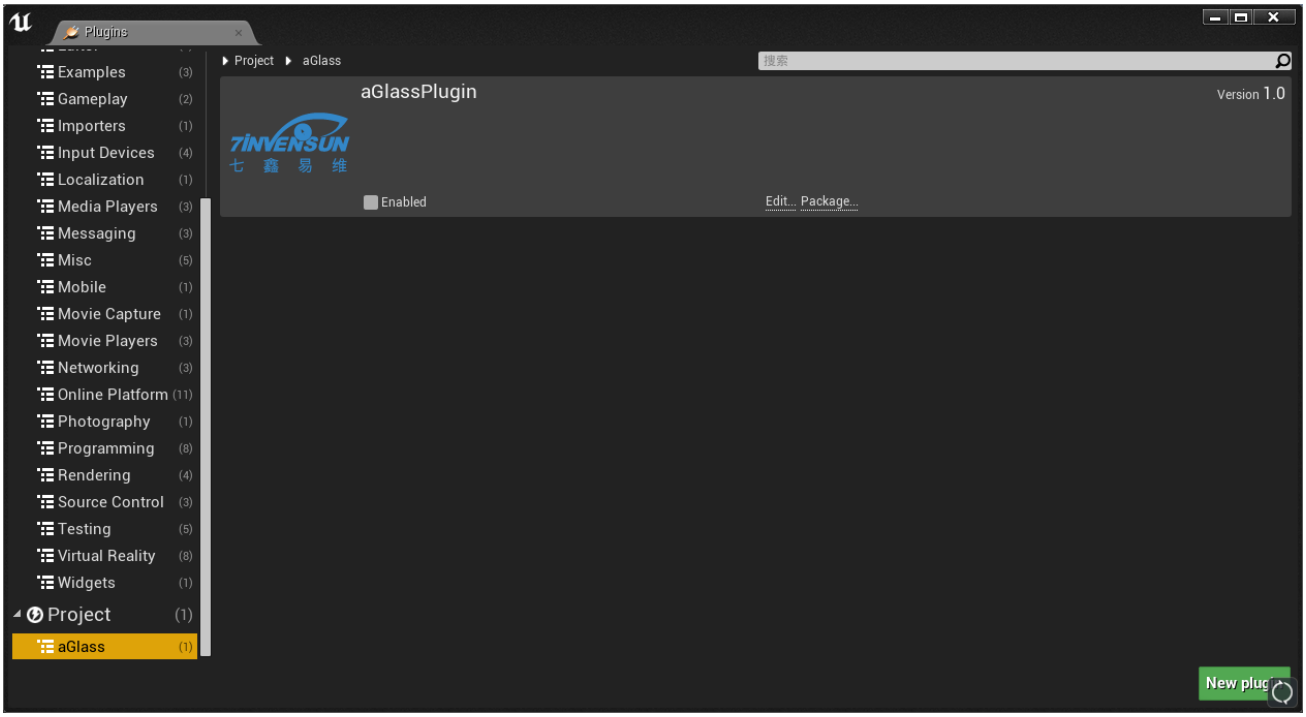
7.2 准备工作

- 安装 Unreal 引擎（推荐 4.17 或以上版本）
- HTC Vive 正确安装
- aGlass 配件安装和连接正常
- 启动 SteamVR
- 启动 aGlass Runtime
- 创建一个新的 Unreal 工程（建议使用 C++工程。如果您采用蓝图工程，需要在 Unreal Editor 中增加一个 C++类）

7.3 导入 Unreal Plugin

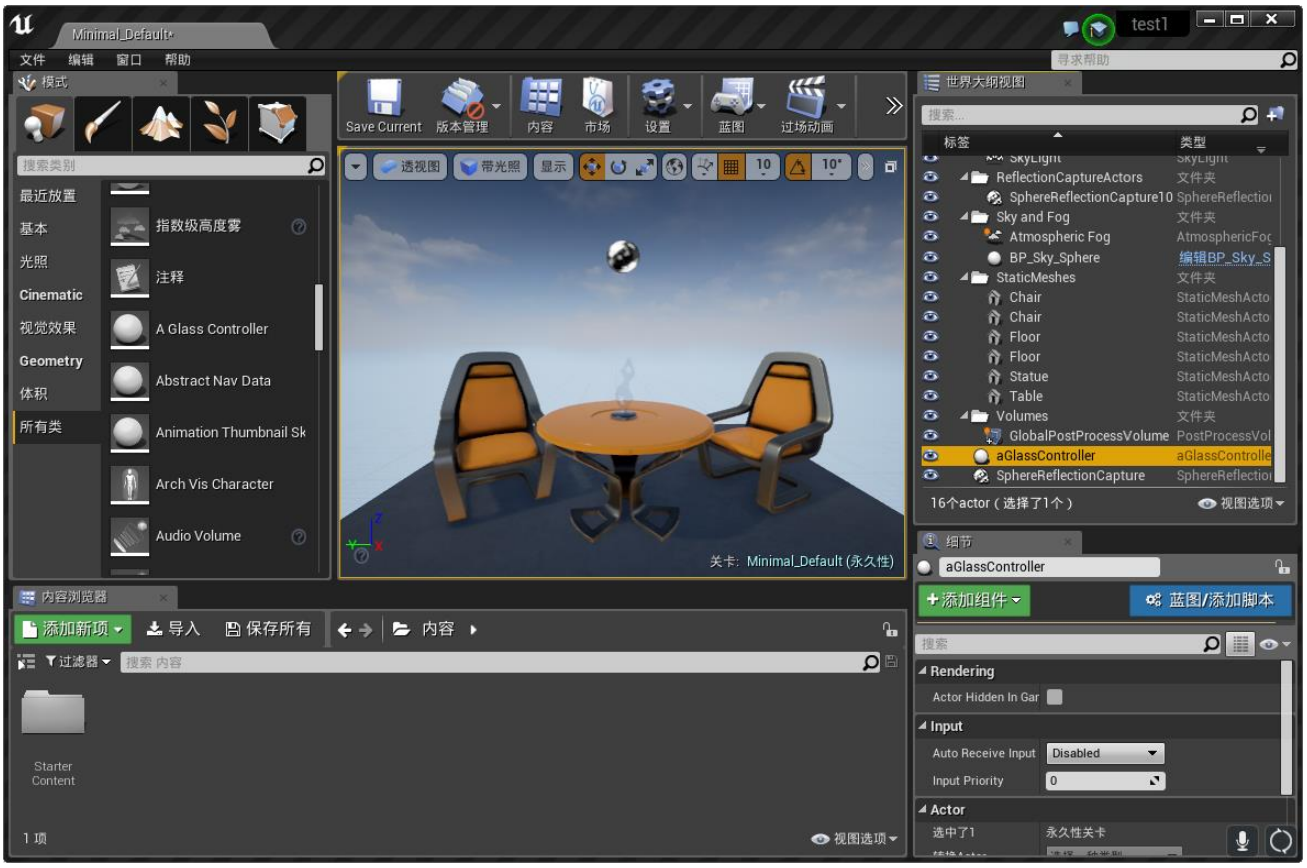
将 Unreal Plugin 的 Plugins 文件夹拷贝至 Unreal 工程的根目录下。

关闭并重新打开您创建的工程，单击菜单：编辑->Plugins，打开 Plugins 选项卡，在选项卡左边的列表中，可以看到 Project 分类中有 aGlass 一项，如下图所示：



勾选 aGlassPlugin 界面的 Enable 选项，按照 Unreal 的提示，重新启动工程，以使 aGlassPlugin 生效，重启过程中按照提示进行重新编译。

7.4 建立 aGlass 控制器



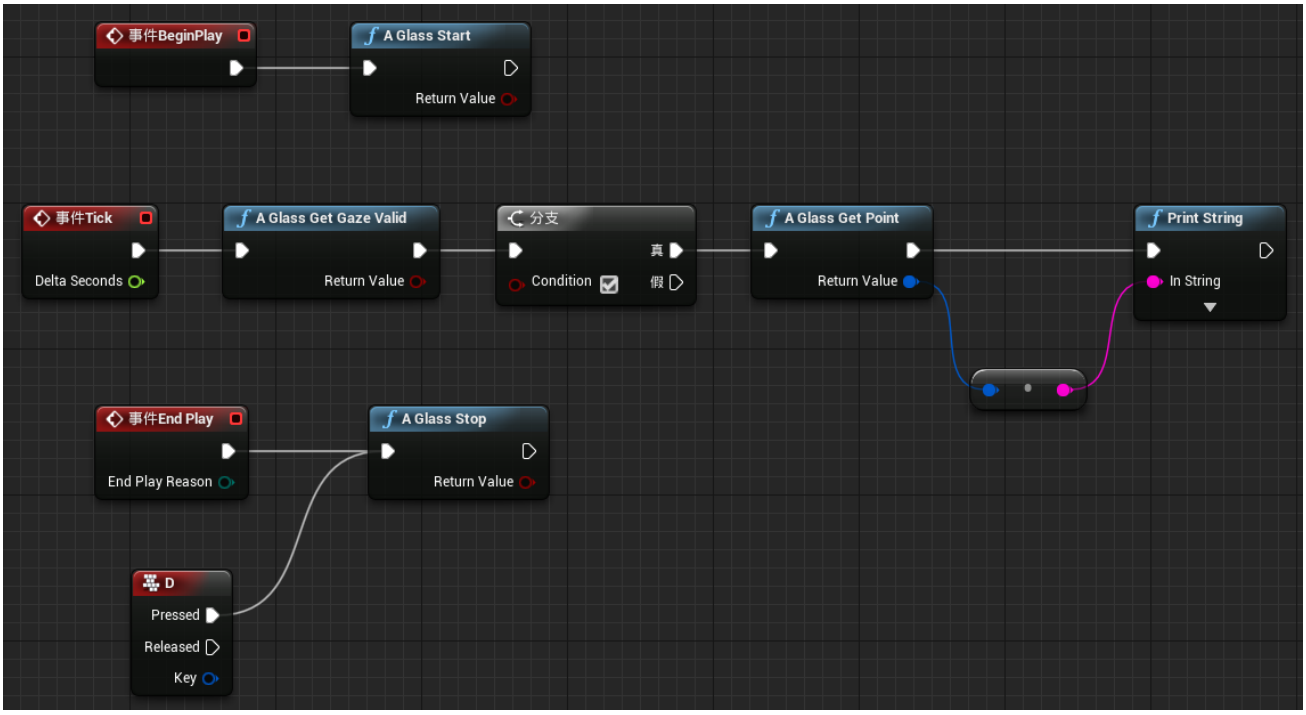
在“模式->所有类”中，选择 A Glass Controller，将其拖拽至主窗口，将在世界大纲视图中出现，将其命名为 aGlassController。

7.5 创建蓝图

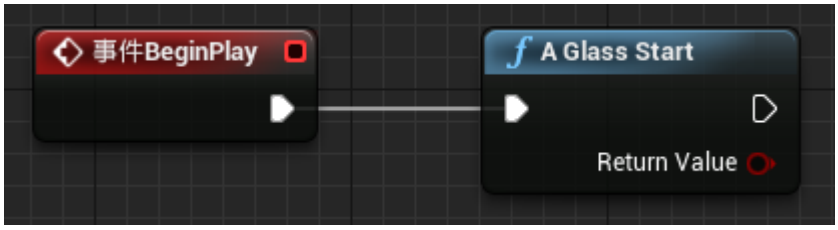
点击“蓝图 -> 打开关卡蓝图”，如下图所示：


Minimal_Default - 关卡蓝图
文件 编辑 视图 调试 窗口 帮助
我的蓝图
添加新项
图表
事件图表
函数(20 可重载)
宏
变量
事件调度器
编译
Search
类设置
类默认值
播放
事件图表
Minimal_Default > 事件图表
缩放-7
右击来创建新节点
关卡蓝图
细节
Actor更新
Start with Tick E
Tick Interval (se
Allow Tick Befor
Replication
Only Relevant to
Always Relevant
Replicate Moven
Net Load on Clie
Net Use Owner F
Replicates
Net Cull Distanc
Net Update Freq
Min Net Update I
Net Priority
Input
Block Input
Auto Receive Inp
Input Priority
Actor
可以被损坏
Generate Overla

最终的蓝图如下图所示：


事件BeginPlay
A Glass Start
Return Value
事件Tick
Delta Seconds
A Glass Get Gaze Valid
Return Value
分支
Condition
真
假
A Glass Get Point
Return Value
Print String
In String
事件End Play
End Play Reason
A Glass Stop
Return Value
D
Pressed
Released
Key

7.5.1启动 aGlass 设备



上面蓝图的功能是启动 aGlass 设备。在本例中，运行范例程序后立刻启动 aGlass 设备。

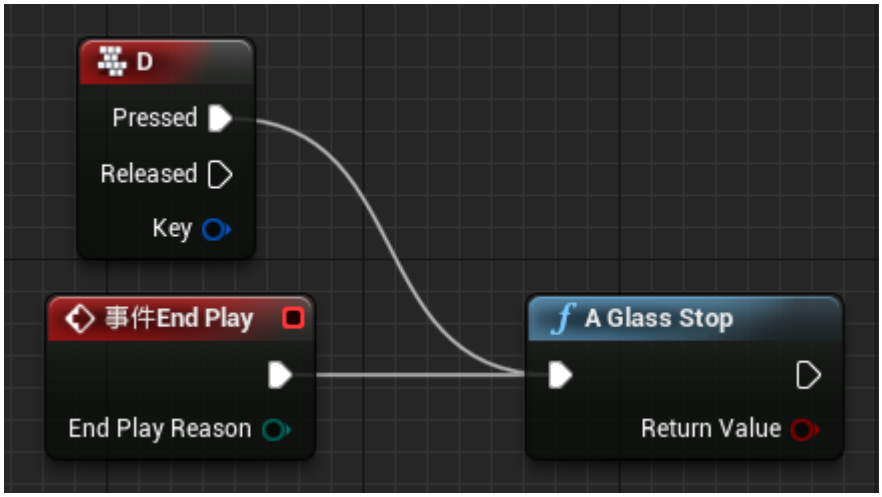
7.5.2获取注视点数据

上面蓝图的功能是在每一帧获取注视点数据。

“A Glass Get Gaze Valid”检查注视点数据是否合法，如果合法，通过“A Glass Get Point”获取到注视点数据并打印。

7.5.3停止 aGlass 设备

有两种方法可以停止 aGlass 设备的工作。一个方法是单击键盘上的 D 键，另一个方法是当你停止范例程序时，aGlass 设备自动停止。蓝图如下所示：



7.6 运行范例工程

单击“预览->虚拟现实预览”，然后带上 HTC Vive 头显，然后您可以在头显的屏幕上看到代表注视点坐标的字符串。

8 第三方权利声明

HTC Vive 为 HTC 公司的品牌，一切权利归 HTC 公司所有。

Steam 和 SteamVR 为 Valve 公司品牌，一切权利归 Valve 公司所有。

Unity 为 Unity 公司品牌，一切权利归 Unity 公司所有。

Unreal 为 Epicgames 公司品牌，一切权利归 Epicgames 公司所有。