

1. 实验名称及目的

RflySim3D 控制台命令接口实验：熟悉通过 RflySim3D 控制台命令接口对场景中的不同对象进行实时控制的方法。

2、实验原理

RflySim3D 内置了一些全局的命令，可以完成 RflySim3D 相关的大部分功能，在 RflySim3D 的程序窗口中按下键盘的波浪号 (~)，可以打开控制台终端，这里可以触发 UE 引擎本身的一些内置命令，还可以触发 RflySim3D 平台内置的命令。这些命令相当于一个个的全局函数。

其中，RflySim 内置命令提供了对不同场景对象、相机、地图等进行操作和调整的功能。主要包括屏幕提示信息显示、地图/视角切换、模型/动画预览及控制、场景中各类数据回传等。

而 UE 引擎自带命令用于监测并平衡各种渲染参数以优化仿真性能。这里性能优化的核心原则包含两点：渲染效率和计算性能。计算性能：减少 CPU 计算次数，把能放在 GPU 中计算的部分放在 GPU 中并行计算；渲染效率：减少每一帧的绘制次数。限制帧率、调整后处理质量、阴影质量等参数，可以在渲染效率方面平衡画面质量和性能需求。同时，通过修改仿真的运行速度和监测各进程的反馈时间，可以评估和优化计算性能。

3、实验效果

本实验利用 RflySim3D 控制台命令实现了仿真过程中的各种交互效果。

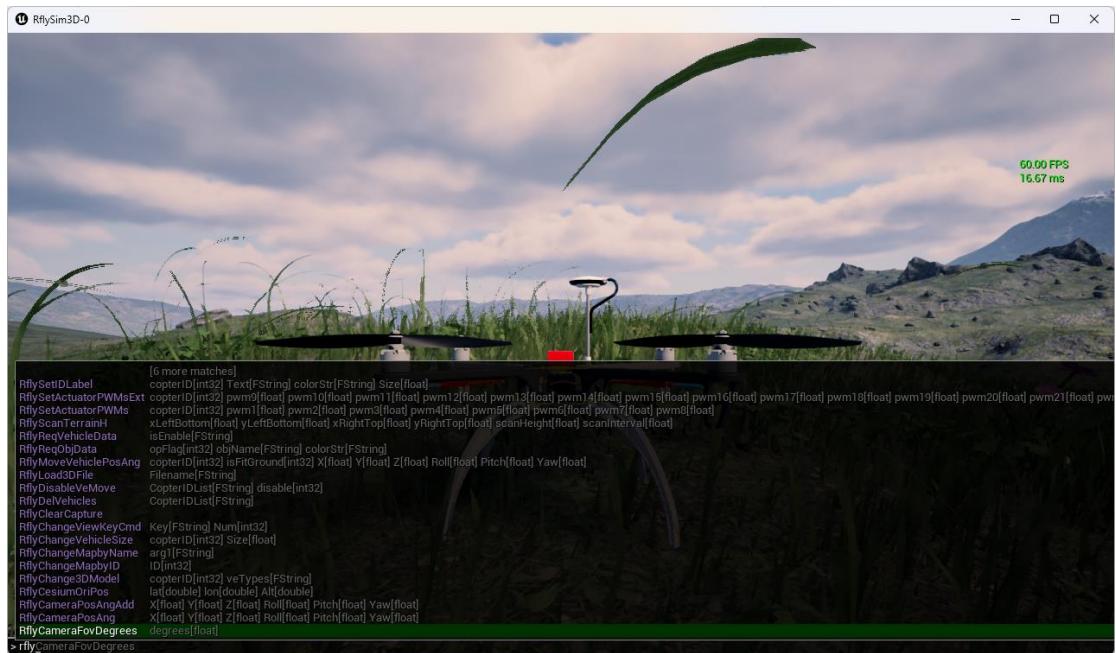


图 1

4、文件目录

文件夹/文件名称	说明
Pictures	动态效果图

5、运行环境

序号	软件要求	硬件要求	
		名称	数量
1	Windows 10 及以上版本	笔记本/台式电脑 ^①	1
2	RflySim 平台 免费版		
3	Wireshark 4.0.7 64-bit		

推荐配置请见：<https://doc.rflysim.com>

6、实验步骤（各命令介绍及效果展示）

Step 1:显示文本

`RflyShowTextTime(String txt, float time)\\让 UE 显示 txt, 持续 time 秒`

打开 RflySim3D，按下键盘左上角的波浪键“~”呼出控制台终端，在其中输入‘Rfly ShowTextTime MyMessage 10’，它的意思是显示字符串“MyMessage”10 秒。可以看见左上角出现了该字符串。



图 2

`RflyShowText(String txt)\\ 让 UE 显示 txt, 持续 5 秒`

与 RflyShowTextTime 函数作用一样，只是它默认显示时间为 5 秒

Step 2:加载 txt 脚本

```
RflyLoad3DFile(FString FileName);\\加载并执行路径下的 TXT 脚本文件
```

RflySim3D 支持 txt 文件作为脚本，这里介绍的命令都可以写成 txt 脚本来使用。我们在 D 盘创建一个“Test.txt”文件，在其中分行输入两条命令：

```
RflyChangeMapbyName Grasslands  
RflyShowText "Hi, Here's TestString"
```



图 3

然后打开 RflySim3D，输入控制台命令：RflyLoad3DFile “D:/Test.txt”，可以看到地图被切换为 Grasslands 了，并且在左上角输出了我们的测试字符串，这正是通过我们编写的脚本的命令完成的。



图 4

Step 3:生成标签（仅支持完整版）

```
RflySetIDLabel(int CopterID, FString Text, FString colorStr, float size);\\设置一个 Copter 的头顶 ID 显示内容（默认显示 CopterID）
```

打开 RflySim3D，鼠标双击+O+3 创建出四旋翼飞机，按 S 可以在其头顶显示一个标签，标签内容默认为飞机的 ID。

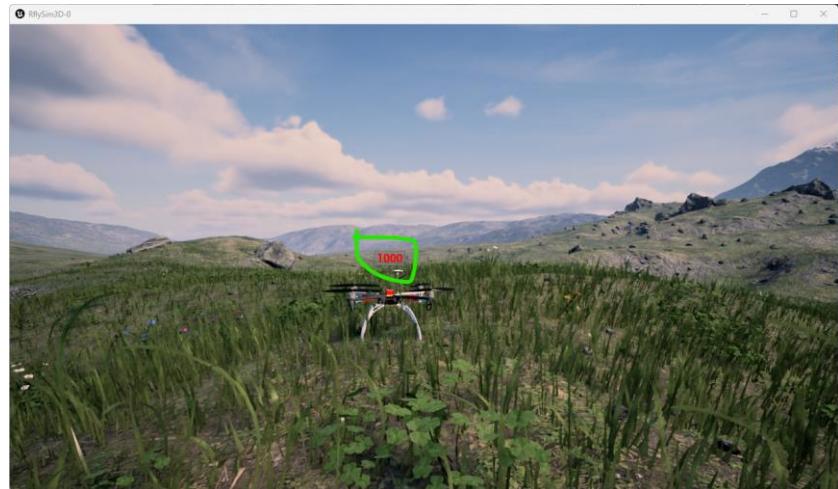


图 5

我们输入命令：RflySetIDLabel 1000 "Hi, here's the test string" FFFF00 20。

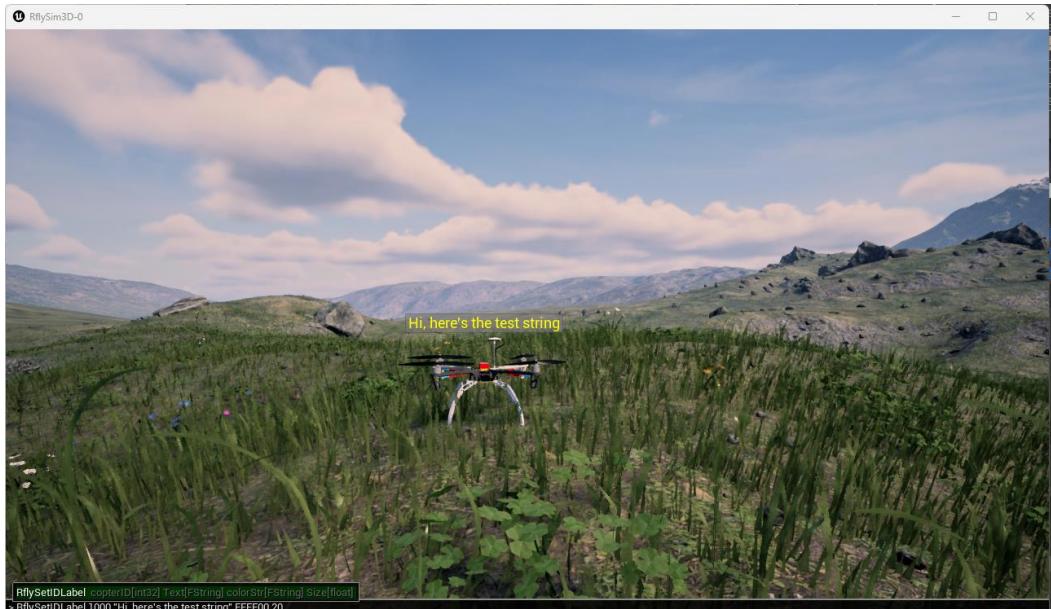


图 6

可以发现，它头上的标签的内容与颜色都发生了改变，这个颜色是 16 进制的 RGB 代码，FFFF00 表示红色与绿色的值为 255，蓝色的值为 0，它们混合后就变成了黄色。

```
RflySetMsgLabel(int CopterID, FString Text, FString colorStr, float size, float time, int flag); \\设置一个 Copter 头顶的 Message 显示的内容
```

此函数与“RflySetIDLabel”函数相似，但它是显示在 ID 下方的，而且可以显示多行。我们输入命令：RflySetMsgLabel 1000 "Hi, here's the test string2" FF00FF 20 5 -1，可以看见紫色的字符串显示在 ID 标签的下方。

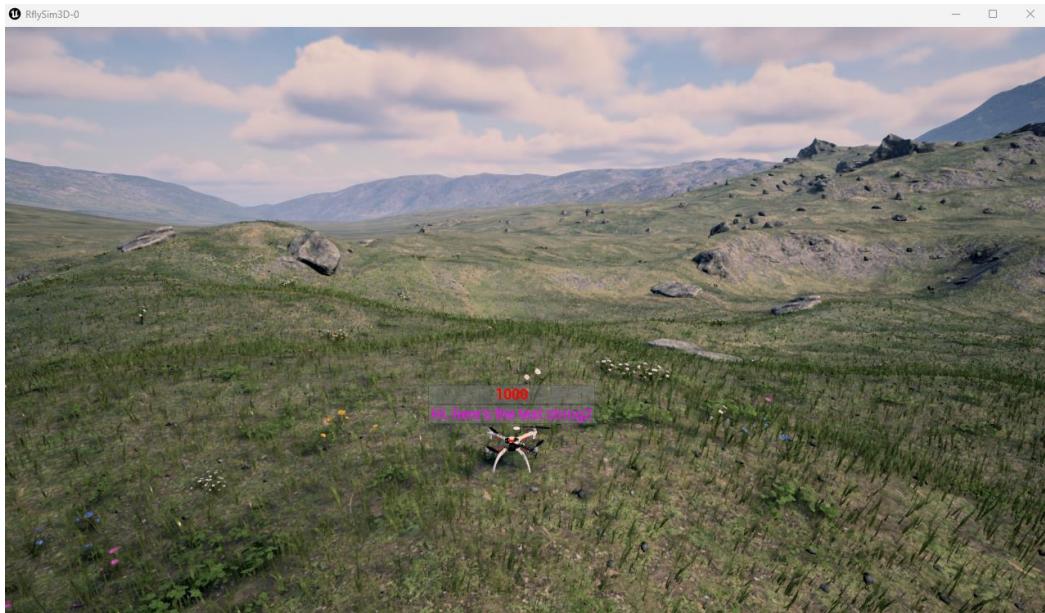


图 7

其中有个参数为“int flag”，它表示操作的是第几行的数据，一共有 5 个消息标签，当 flag<0 时就表示新增一个消息。

Step 4: 切换地图

```
RflyChangeMapbyID(int id)\\ 根据地图的 ID 切换 RflySim3D 场景地图
```

该函数可以根据地图的 ID 快速切换地图，例如在控制台终端中输入“RflyChangeMapbyID 5”，就快速切换到 5 号地图 Grasslands 了。

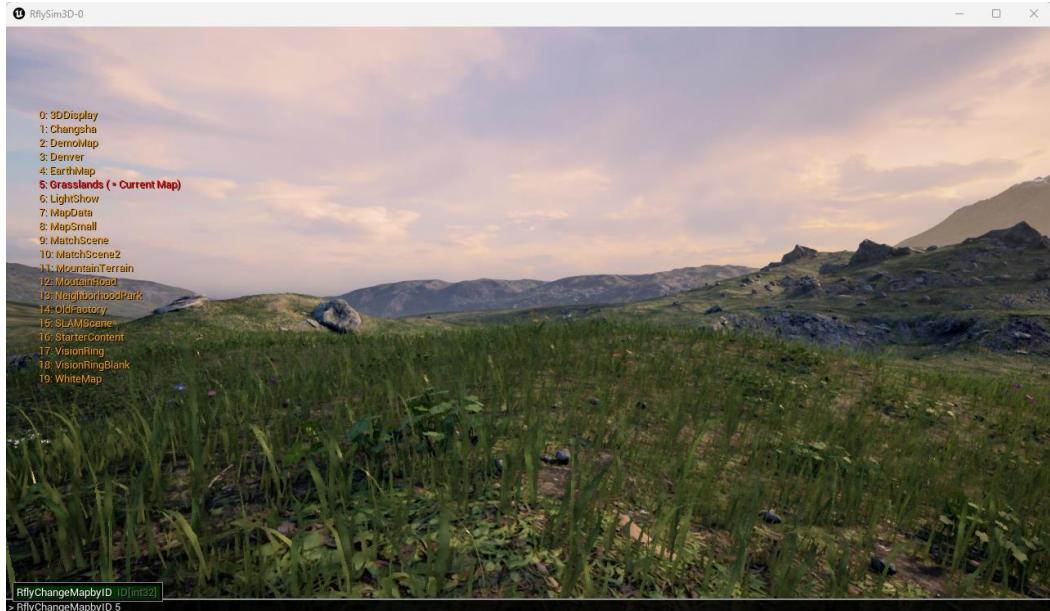


图 8

```
RflyChangeMapbyName(String txt)\\ 根据地图名切换 RflySim3D 场景地图
```

与 RflyChangeMapbyID 函数作用一样，但它是根据地图名进行切换的，例如输入“RflyChangeMapbyName 3DDisplay”，就切换到 0 号地图 3DDisplay 了。



图 9

`RflyChangeViewKeyCmd(String key, int num) \\` 与在 `RflySim3D` 中按一个 `key + num` 效果一致

`RflySim3D` 中有很多快捷键+数字的操作，它们不仅能通过键盘触发，也可以通过此命令来触发，例如我们知道，按快捷键 `M+数字` 就可以切换到指定 ID 的地图，那么我们可以输入命令 “`RflyChangeViewKeyCmd M 5`” 来切换到 5 号地图。

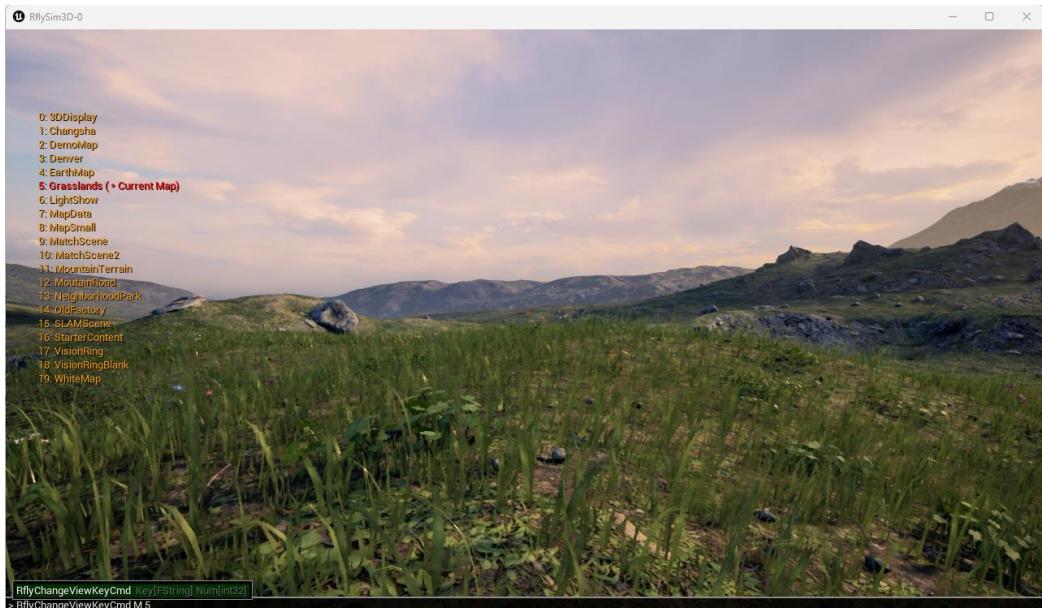


图 10

`RflyCesiumOriPos(double lat, double lon, double Alt) \\` 修改 `Cesium` 的原点位置

`Cesium` 在之前也介绍过了，引入 `Cesium` 是用于构建全球大场景的，该函数可以设置地图原点的经纬度。

使用该接口需要进入使用了 `Cesium` 插件的地图（全球大场景地图），例如 `ChangSha` 地图，并且最好处于联网状态（否则不会加载在线地图影像），开启 `RflySim3D`，按 `M` 切换到 `ChangSha` 地图。

按下 “~”，输入命令 “`RflyCesiumOriPos 39.9042 116.4074 50`”，此处为北京。经纬度

的范围为 [-180,180]与[-90,90]。

Step 5:相机视角

再例如，我们知道按快捷键 B+无人机 ID 可以切换视角跟随的无人机，那么我们就可以使用命令“RflyChangeViewKeyCmd B 1001”，将视角从 ID 为 1000 的无人机上转移到 ID 为 1001 的无人机上。



图 11

RflyCameraPosAngAdd(float x, float y, float z, float roll, float pitch, float yaw) \\ 给摄像机的位置与角度增加一个偏移值

例如想让摄像机向上移动 10 米，并且向下俯视 30°，可以使用命令“RflyCameraPosAngAdd 0 0 -10 0 -30 0”



效果图

图 12 切换相机视角

```
RflyCameraPosAng(float x, float y, float z, float roll, float pitch, float yaw) \\ 设置摄像机的位置与角度(UE 的世界坐标)
```

RflyCameraPosAng 与上面提到的 RflyCameraPosAngAdd 函数类似，但它是直接设置摄像机的位置与角度，而后者是在摄像机本身的位置与角度上再增加一个偏移值。

```
RflyCameraFovDegrees(float degrees) \\ 设置摄像机的视域体 FOV 角度
```

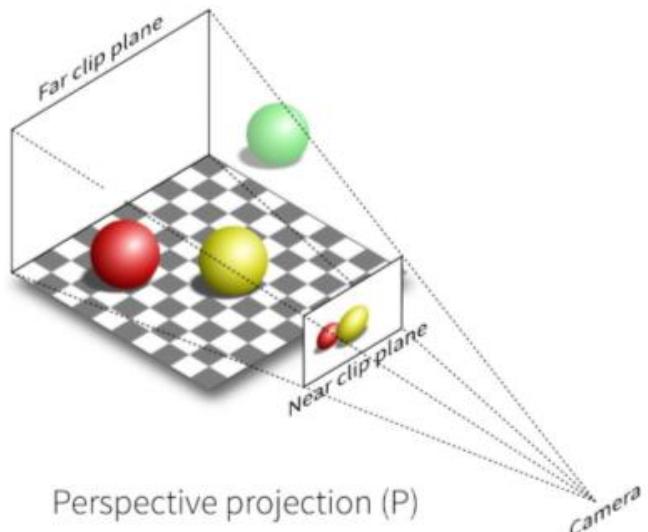


图 13

FOV 的全称是 Field of View(视场角)，此函数会修改摄像机的水平视场角，也就是上图平面上最左侧的中点与最右侧的中点与摄像机形成的角度。一般会默认该角度为 90°。如果想要更改它，可以通过命令“RflyCameraFovDegrees 120”，它会将该角度改为 120°。

Step 6: 调整载具模型

```
RflyChange3DModel(int CopterID, int veTypes=0) \\ 修改一个无人机的模型样式
```

我们知道按 C 键可以修改当前无人机的三维模型的样式，此命令可以修改指定无人机的三维模型样式。可以按 S 显示模型的 ID，例如这里无人机的 ID 为 1000，那么想要将它的样式变为 1 号，可以使用命令 “RflyChange3DModel 1000 1”。



图 14

```
RflyChangeVehicleSize(int CopterID, float size=0) \\修改一个无人机的缩放大小
```

此命令可以让指定无人机变大或变小，使用命令“RflyChangeVehicleSize 1000 10”可以让 ID 为 1000 的无人机变为 10 倍于最初的大小。它与 Ctrl+鼠标滚轮的作用类似。

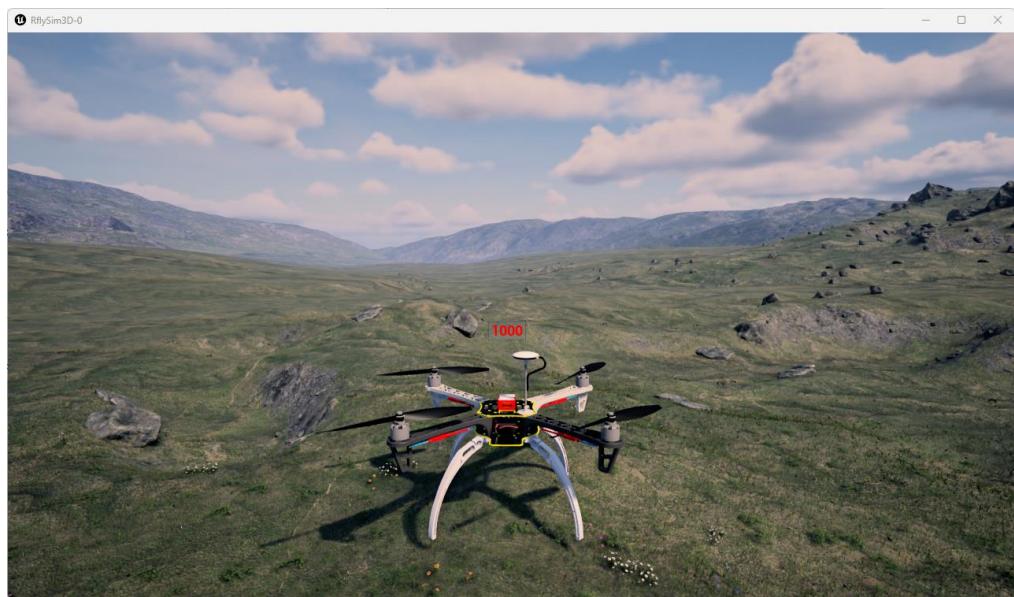


图 15

```
RflyMoveVehiclePosAng(int CopterID, int isFitGround, float x, float y, float z, float roll, float pitch, float yaw) \\ 给无人机的位置与角度设置一个偏移值, isFitGround 设置无人机是否适应地面
```

此命令可以给无人机一个位置和角度的偏移值，例如“RflyMoveVehiclePosAng 1000 1 -10 -10 -10 0 -20 0”可以让飞机移动 (-10, -10, -10) 米，并且 pitch 角度-20°。



图 16

如果 isFitGround 设置为 1，则无人机的 z 坐标会根据地形的高度决定。

```
RflySetVehiclePosAng(int CopterID, int isFitGround, float x, float y, float z, float roll, float pitch, float yaw) \\设置无人机的位置与角度
```

此命令与 RflyMoveVehiclePosAng 函数类似，但它是直接设置无人机的位置，而不是给出一个增量。

Step 7: 调用蓝图动画

```
RflySetActuatorPWMs(int CopterID, float pwm1, float pwm2, float pwm3, float pwm4, float pwm5, float pwm6, float pwm7, float pwm8); \\传入 8 个值，并触发目标无人机的蓝图的接口函数
```

它其实就是在直接设置无人机的 8 位电机数据，一般该数据是由 UDP 发送来的，但可以用此命令直接设置。

关闭 CopterSim，重启 RflySim3D，鼠标双击地面按字母 O+数字 3，可以看见创建了一个 ID 为 1000 的无人机在双击的地方。此时由于没有外部输入 UDP 数据，无人机处于静止状态。然后我们使用命令“RflySetActuatorPWMs 1000 10 10 10 10 0 0 0 0”。



效果图

图 17

可以看见该旋翼无人机螺旋桨开始转动，因为对此无人机而言，这 8 位数的前 4 位控制着它的 4 个旋翼的旋转速度。现在它旋转速度为每秒 10 度。

一般而言，这 8 个电机数据是由 UDP 发送的，例如：之后会介绍的 python 的 sendUEPosNew 函数，发送的 UDP 中有一项 PWMs 就是这 8 位数据。

这 8 维数据实际上控制的是 XML 文件中定义的各致动器，在 XML 接口章节中会提到。

Step 8:清除载具

```
RflyDelVehicles(FString CopterIDList); \\删除一些 Copter (逗号是分隔符)
```

该函数可以根据 ID 删除当前存在于场景的无人机。RflySim3D 并不会主动删除场景中的无人机，哪怕已经没有接收到它们的数据了。（这也是为什么关闭 CopterSim 后无人机并没有从场景中消失），必须明确的要求 RflySim3D 移除这些无人机。

我们打开 RflySim3D，鼠标双击+O+3，创建 2 个无人机，ID 分别会默认为 1000 与 1001，我们再输入命令：RflyDelVehicles "1000,1001"，即可发现无人机消失了。



效果图

图 18

但需要注意的是删除飞机之前应当停止向 RflySim3D 发送无人机的数据，如果 RflySim3D 仍接收着数据，那么很可能它将无人机删除后又因为收到了无人机的信息而立刻再次创建。

Step 9: 获取地形数据

```
RflyScanTerrainH(float xLeftBottom(m), float yLeftBottom(m), float xRightTop(m), float yRightTop(m), float scanHeight(m), float scanInterval(m))
```

该函数可以扫描三维地形，生成一个 png 的高度图与 txt，CopterSim 程序会需要它才知道 UE 有哪些地形、以及它们的高程。但该函数需要知道地形的大小和高度、扫描的精度。

例如命令 “RflyScanTerrainH -1000 -1000 1000 1000 200 1”，表示地形左下角坐标为 (-1000, -1000) 米，右上角为 (1000, 1000) 米，最高不超过 200 米，每隔 1 米采样一次，最终会形成一个 png 的图像与 txt 文件。



图 19

然后可以在“\PX4PSP\RflySim3D”找到此地图的 2 个同名文件。

A screenshot of a Windows File Explorer window. The path is displayed as: “此电脑 > Windows (C:) > PX4PSP > RflySim3D”. The search bar contains the text “在 RflySim3D 中搜索”。The table below lists the files found in the RflySim3D folder:

名称	修改日期	类型	大小
Engine	2023/4/14 11:25	文件夹	
RflySim3D	2023/3/30 17:33	文件夹	
3DDisplay.png	2023/4/14 14:16	PNG 文件	884 KB
3DDisplay.txt	2023/4/14 14:16	Text Document	1 KB
ClickLog.txt	2023/4/14 14:15	Text Document	1 KB
CreateLog.txt	2023/4/14 14:15	Text Document	1 KB
RflySim3D.exe	2022/12/17 1:05	应用程序	142 KB
RflySim3D.txt	2023/3/31 15:35	Text Document	1 KB

图 20

再将它们放到“PX4PSP\CopterSim\external\map”中即可让 CopterSim 识别此地图，并且它也可以用于 Simulink 获取高度图，后面会更详细的使用它。

名称	修改日期	类型	大小
3DDisplay.png	2022/2/20 15:55	PNG 文件	2,426 KB
3DDisplay.txt	2022/2/20 15:55	Text Document	1 KB
Changsha.png	2021/12/18 19:07	PNG 文件	354 KB
Changsha.txt	2022/2/9 0:46	Text Document	1 KB
Denver.png	2021/12/18 19:07	PNG 文件	1,197 KB
Denver.txt	2022/2/9 0:47	Text Document	1 KB
EarthMap.png	2021/12/18 19:08	PNG 文件	274 KB
EarthMap.txt	2022/2/9 0:47	Text Document	1 KB
Grasslands.png	2022/2/20 15:55	PNG 文件	2,426 KB
Grasslands.txt	2022/2/20 15:55	Text Document	1 KB
LightShow.png	2020/4/12 17:29	PNG 文件	1 KB

图 21

Step 10: UDP 收发

RflyReqVehicleData(FString isEnabled);如果'isEnabled'不为 0，则 RflySim3D 会开始发送所有 Copter 的数据。

默认情况下，RflySim3D 只会接收来自外部的 Copter 的信息，收到的 UDP 数据会类似于这样的结构体：

```
struct SOut2SimulatorSimple {
    int checkSum;
    int copterID;
    int vehicleType;
    float MotorRPMSMean;
    float PosE[3];
    float AngEuler[3];
}
```

RflySim3D 默认不会往外发送三维场景中的无人机的这些数据（位置、姿态），但是如果有必要，可以调用此命令“RflyReqVehicleData 1”，当 RflySim3D 接收到此命令时会进入“数据回传模式”，开始使用 UDP 发送请求的 Copter 的数据。此模式下一旦 RflySim3D 接收到了 Copter 的更新数据，RflySim3D 会再将相关数据发送出来：

```
struct reqVeCrashData {
    int checksum; //数据包校验码 1234567897
    int copterID; //当前飞机的 ID 号
    int vehicleType; //当前飞机的样式
    int CrashType;//碰撞物体类型, -2 表示地面, -1 表示场景静态物体, 0 表示无碰撞, 1 以上表示被碰飞机
    的 ID 号

    double runnedTime; //当前飞机的时间戳
    float VelE[3]; // 当前飞机的速度
    float PosE[3]; //当前飞机的位置
    float CrashPos[3];//碰撞点的坐标
    float targetPos[3];//被碰物体的中心坐标
    float AngEuler[3]; //当前飞机的欧拉角
    float MotorRPMS[8]; //当前飞机的电机转速
    float ray[6]; //飞机的前后左右上下扫描线
    char CrashedName[20] = {0};//被碰物体的名字
}
```

}

reqVeCrashData 共有 48 个数据项，每个数据项的大小根据类型不同而不同。总共有 4 个 int（每个占 4 个字节）、1 个 double（占 8 个字节）、29 个 float（每个占 4 个字节）、以及一个 20 字节的字符数组 CrashedName。可以看见它是一个 $4 \times 4 + 1 \times 8 + 29 \times 4 + 20 \times 1 = 160$ 字节的包，再加上 udp 包头固定的 8 字节，它就是 168 字节的包。我们可以打开 Wireshark 软件进行抓包来观察它：在 wireshark 的应用显示过滤器中输入“`udp.length == 168`”回车，它会为我们匹配长度为 168 的 udp 协议包，可以发现此时并没有这个长度的包。

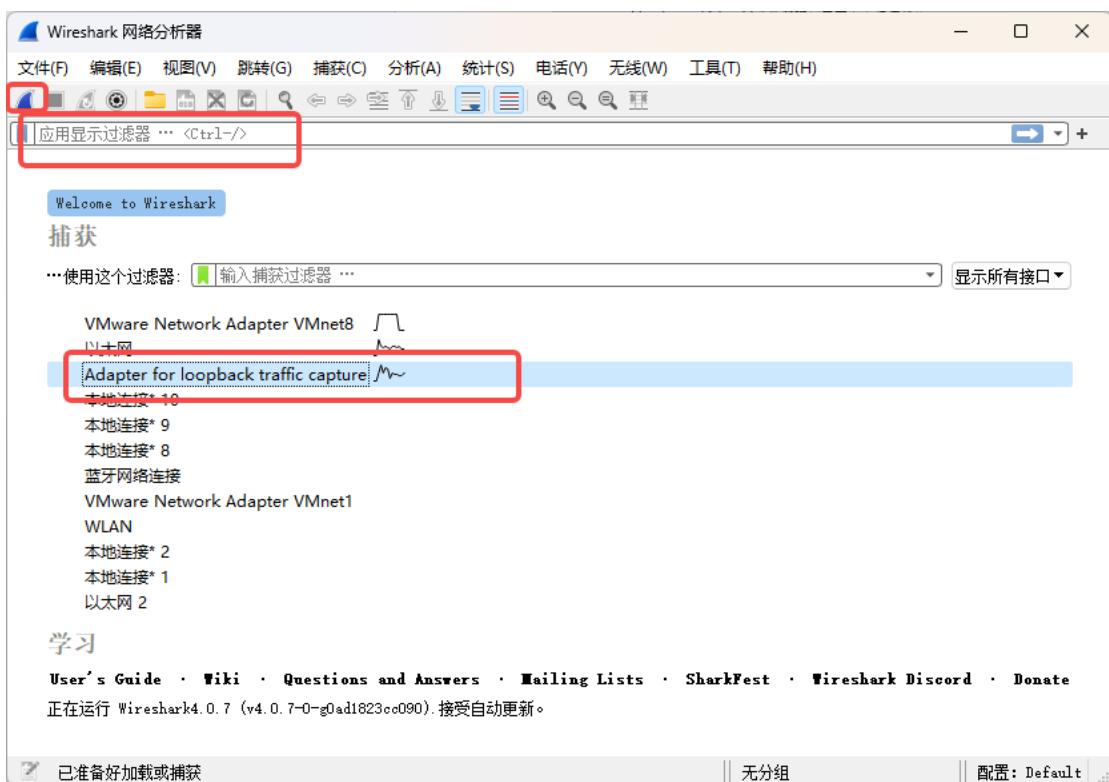


图 22

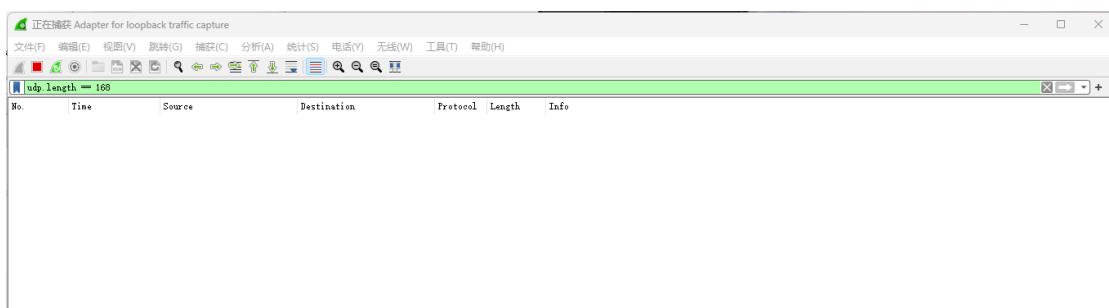


图 23

然后我们打开 RflySim3D 与 CopterSim，输入“RflyReqVehicleData 1”，开启数据回传模式。

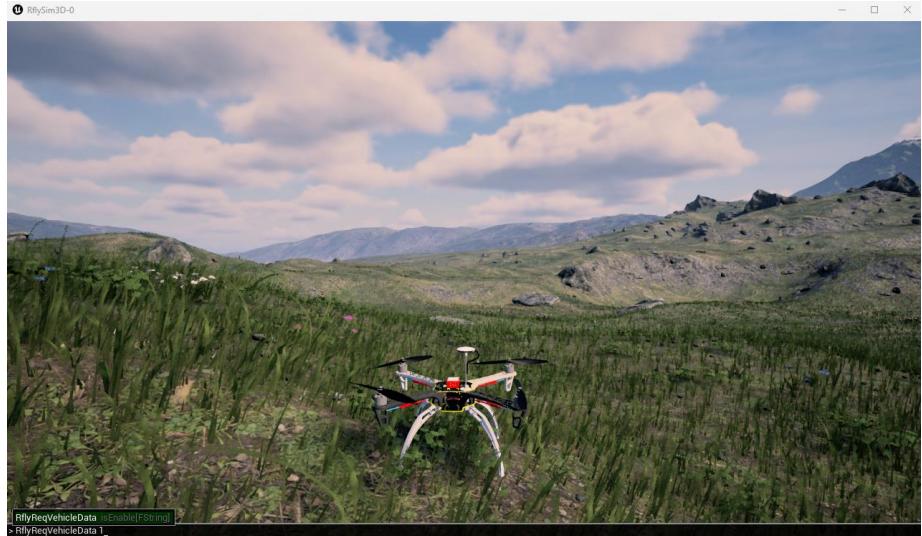


图 24

我们可以发现 Wireshark 中开始抓到这样的包了：

No.	Time	Source	Destination	Protocol	Length	Info
598	13.130313	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160
610	13.888929	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160
621	14.128570	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160
670	15.028464	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160
679	15.361894	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160
700	16.028587	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160
709	16.694984	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160
753	17.029836	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160
780	17.699269	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160
800	18.036789	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160
821	18.704947	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160
830	19.042461	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160
871	19.711042	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160
RRI	20.000000	127.0.0.1	224.0.0.10	UDP	192	52607 → 20006 Len=160

图 25

我们还可以看见，这个包是发往组播地址 224.0.0.10 的 20006 端口的，那么任何程序只要加入该 udp 组播，并监听 20006 端口，再把收到的数据按结构体 “reqVeCrashData” 解析即可拿到这些数据。

具体的接收逻辑并不需要额外编写，平台的接口已经写好了接收它的逻辑，之后介绍 python 接口与 simulink 接口时会介绍，这里只是介绍命令接口的函数，因此使用 Wireshark 进行检验。

```
RflySetPosScale(float scale); \\ 全局位置的缩放
```

该函数会影响通过 UDP 传入的 Copter 的位置信息，默认 scale==1。我们上面提到了，RflySim3D 接收到的数据是类似于这样的：

```
struct SOut2SimulatorSimple {
    int checkSum;
    int copterID;
    int vehicleType;
    float MotorRPMSMean;
    float PosE[3];
    float AngEuler[3];
}
```

但 RflySim3D 在收到位置信息后还会进行一个处理：“`PosE[i]= PosE[i]*scale`”，进行一个全局的缩放变换。它主要是用于单位的统一，RflySim3D 中的空间单位是 cm，如果外部

传入的数据的单位是 m，则可以使用命令“RflySetPosScale 100”，这样就不必额外进行单位转换的逻辑了。

Step 11:回传场景不同对象数据

```
RflyReqObjData(int opFlag, FString objName, FString colorStr);\\请求获取三维场景中物体的数据
```

该函数类似“RflyReqVehicleData 函数”，它也会回传三维场景中物体的一些数据。但它一次只回传一个指定目标的数据。

当 opFlag==0 时，它会返回指定 ID 的、用于捕获图像的相机的数据：

```
struct CameraData { //56
    int checksum = 0;//1234567891
    int SeqID; //相机序号
    int TypeID;//相机类型
    int DataHeight;//像素高
    int DataWidth;//像素宽
    float CameraFOV;//相机视场角
    float PosUE[3]; //相机中心位置
    float angEuler[3];//相机欧拉角
    double timestamp;//时间戳
};
```

它是和回传图像有关的接口，RflySim3D 是支持从三维场景中输出捕获的图像的，它可以根据请求创建一个相机用于捕获图像，而相机的数据可以通过该函数来获得。

RflyReqObjData 0 0 FFFFFF

当 opFlag==1 时，它会返回指定 ID 的 Copter 的数据：

```
struct CoptReqData { //64
    int checksum = 0; //1234567891 作为校验
    int CopterID;//飞机 ID
    float PosUE[3]; //物体中心位置（人为三维建模时指定，姿态坐标轴，不一定在几何中心）
    float angEuler[3];//物体欧拉角
    float boxOrigin[3];//物体几何中心坐标
    float BoxExtent[3];//物体外框长宽高的一半
    double timestamp;//时间戳
};
```

打开 RflySim3D，鼠标双击地面+按下字母 O+数字 3，创建一个 ID 飞机（ID 默认为 1000），输入命令“RflyReqObjData 1 1000 FFFFFF”。



图 26

在 Wireshark 中可以看到捕获到了长度为 $64+8=72$ 字节的包。

No.	Time	Source	Destination	Protocol	Length	Info
1514	53.941095	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1540	54.940968	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1552	55.940980	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1593	56.941125	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1620	57.940985	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1647	58.941139	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1662	59.941700	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1678	60.941237	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1719	61.941371	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1750	62.941833	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1773	63.941897	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1810	64.941111	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1822	65.941116	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1854	66.941887	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1869	67.941173	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1896	68.941108	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1922	69.941872	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64
1942	70.941082	127.0.0.1	224.0.0.10	UDP	96	56084 - 20006 Len=64

图 27

当 $opFlag==2$ 时，它会返回指定 ID 的物体的数据：

```
struct ObjReqData { //96
    int checksum = 0; //1234567891 作为校验
    int seqID = 0;
    float PosUE[3]; //物体中心位置（人为三维建模时指定，姿态坐标轴，不一定在几何中心）
    float angEuler[3];//物体欧拉角
    float boxOrigin[3];//物体几何中心坐标
    float BoxExtent[3];//物体外框长宽高的一半
    double timestamp;//时间戳
    char ObjName[32] = { 0 }; //碰物体的名字
};
```

在 RflySim3D 中，并不是所有物体都是 Copter 类的对象（例如地形、障碍、建筑等），而此接口可以用于获取这些非 Copter 的物体的信息。

打开 RflySim3D，双击地面，我们可以看到鼠标点击地面后，屏幕上输出了一些信息，其中第一个单词表示击中物体的名字，也就是说“地形的名字为 Landscape_1”

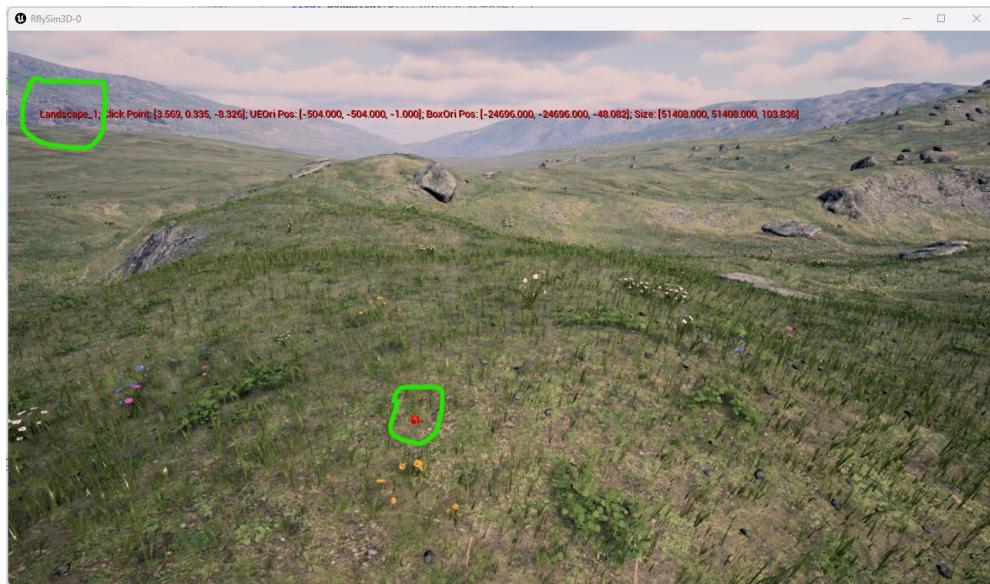


图 28

然后我们使用命令“RflyReqObjData 2 Landscape_1 FFFFFF”，表示我们想知道名为“Landscape_1”的对象的信息。

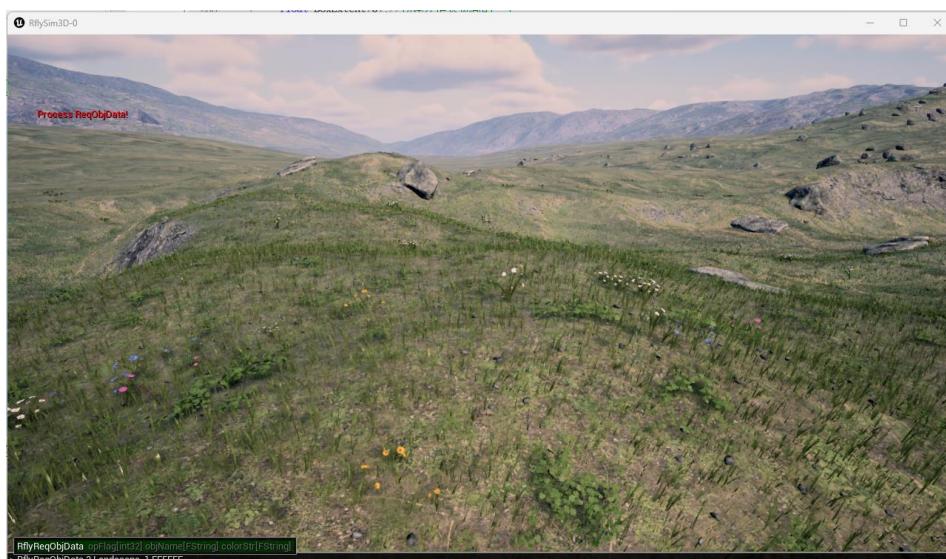


图 29

然后在 Wireshark 中检验，我们可以看见它抓到了长度为 $96+8=104$ 的包。

图 30

Step 12:清空取图

```
RflyClearCapture(int seqID);\\ 清空抓取的图像
```

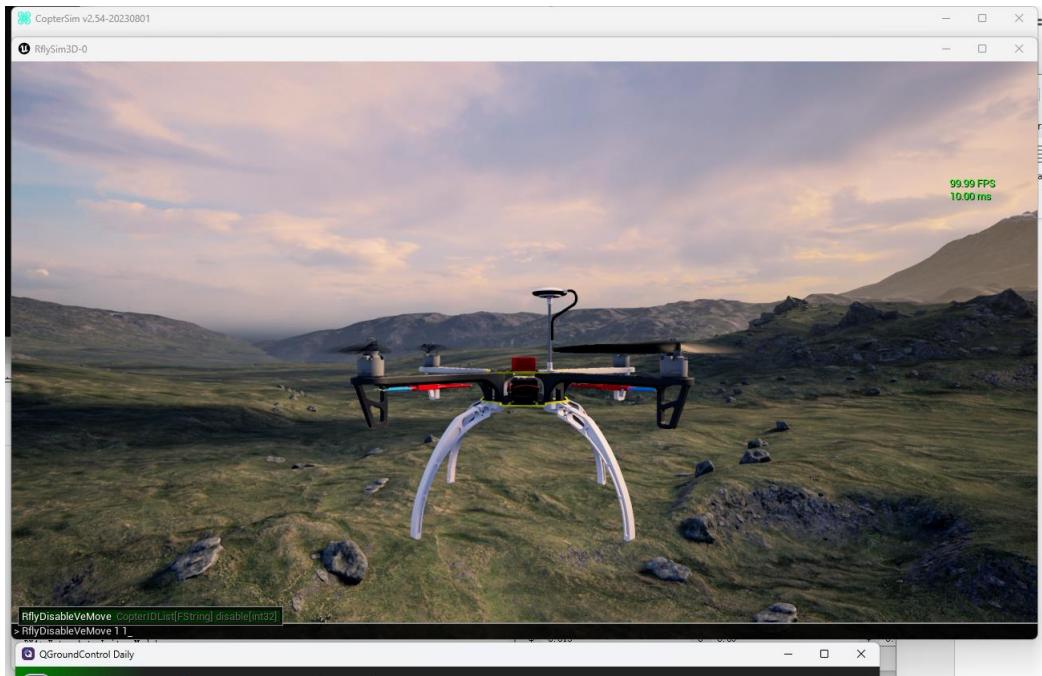
该函数会清空指定 seqID 的图像传感器捕捉的图像的内存，如果 seqID<0 则清空所有相机的图像的内存数据。

Step 13:拒收指定消息

```
RflyDisableVeMove(FString CopterIDList, int disable);\\拒接接收指定 ID 的 Copter 的信息（逗号是分隔符）
```

使用该函数可以让 RflySim3D 根据 ID 拒绝接收一些无人机的数据，相当于禁止了这些 ID 的 Copter 的创建、移动、姿态变化。

我们打开在 “\Desktop\RflyTools” 打开软件在环仿真 SITLRun，使用 QGC 让无人机处于移动状态，然后我们使用命令：RflyDisableVeMove 1 1，会发现 QGC 中无人机仍在移动，CopterSim 中的模拟坐标仍在变化，但 RflySim3D 中的无人机停下来了。



效果图

图 31

但是可以看见它并没有停止执行器的旋转，这是因为发给 RflySim3D 的执行器数据是“旋转速度”，不再收到新的执行器数据只是意味着旋转速度不变了，而不是执行器停止旋转。

Step 14: 显示常用渲染参数

stat fps 可以显示更新率，实时监测帧率表现，以评估渲染效率并进行调整。

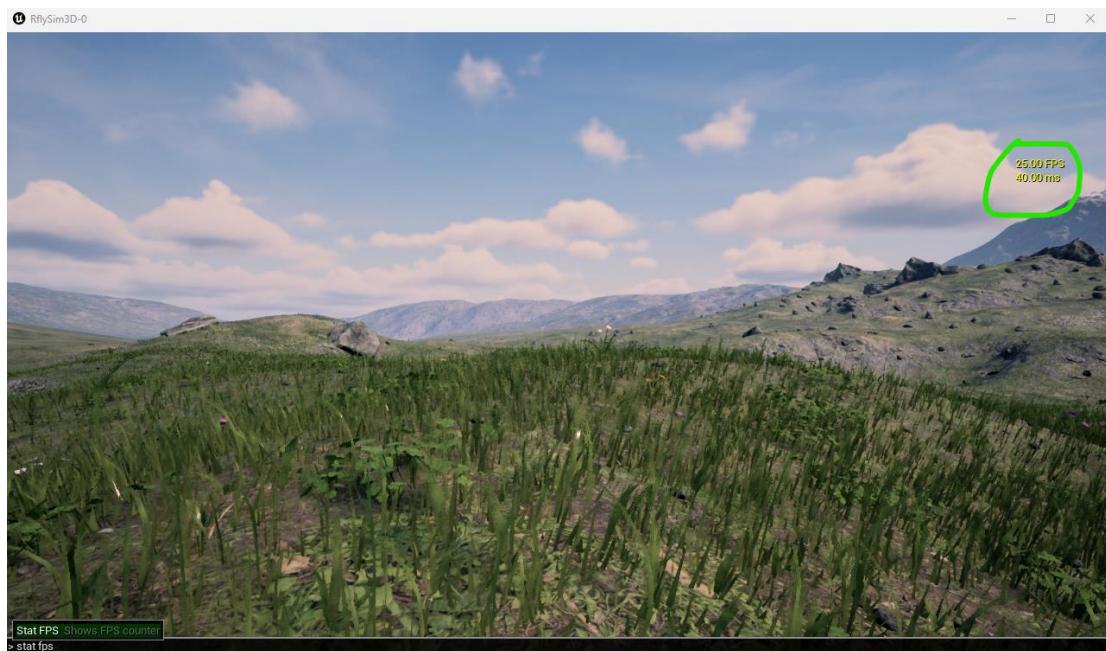
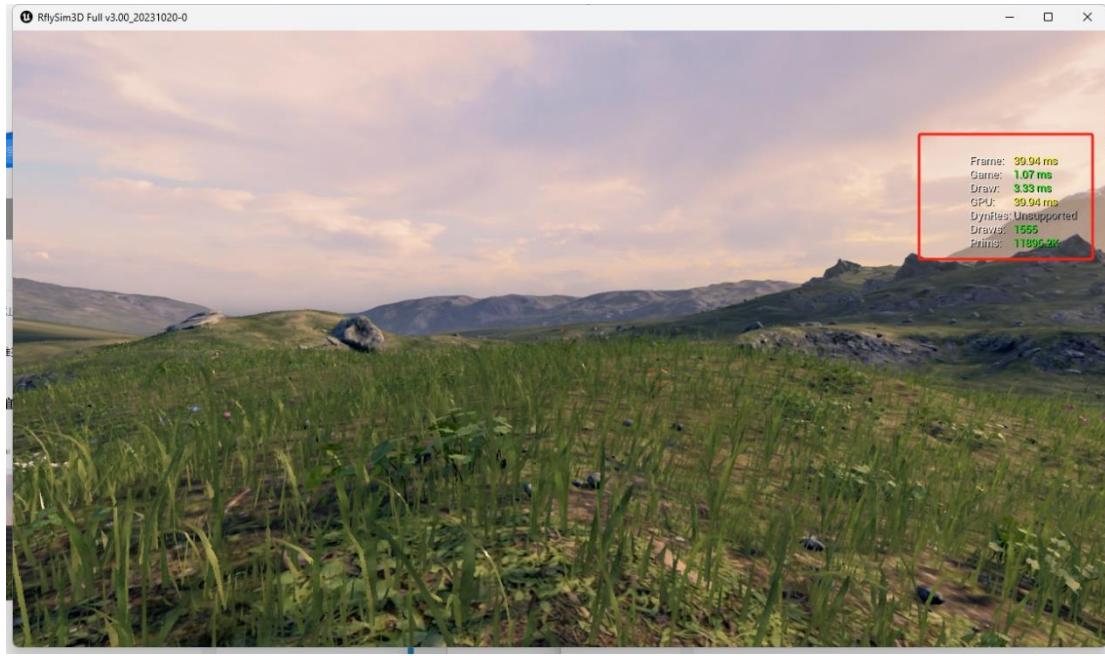
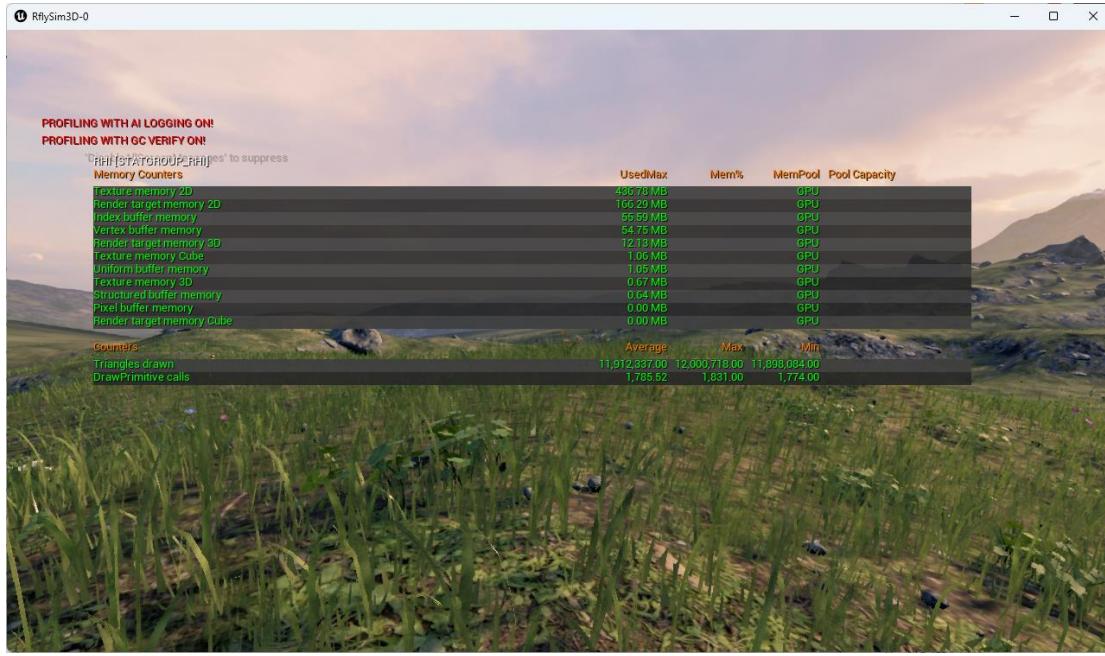


图 32

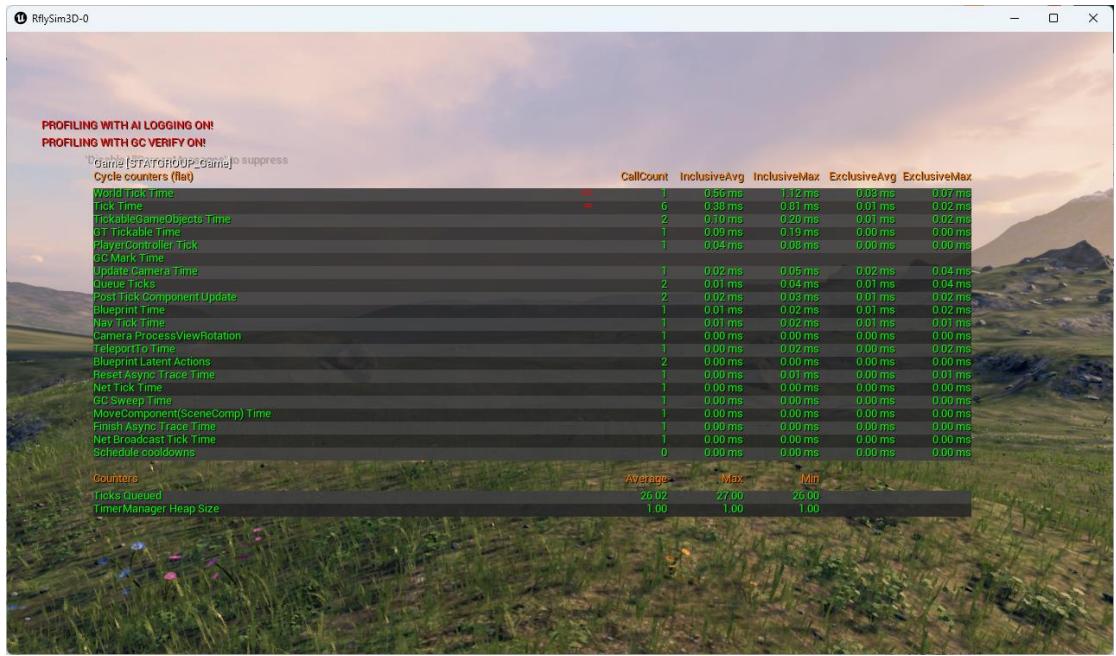
stat unit: 显示各类消耗，包括 CPU、GPU 和渲染线程等的消耗情况，有助于分析渲染性能瓶颈所在，并进行相应优化。



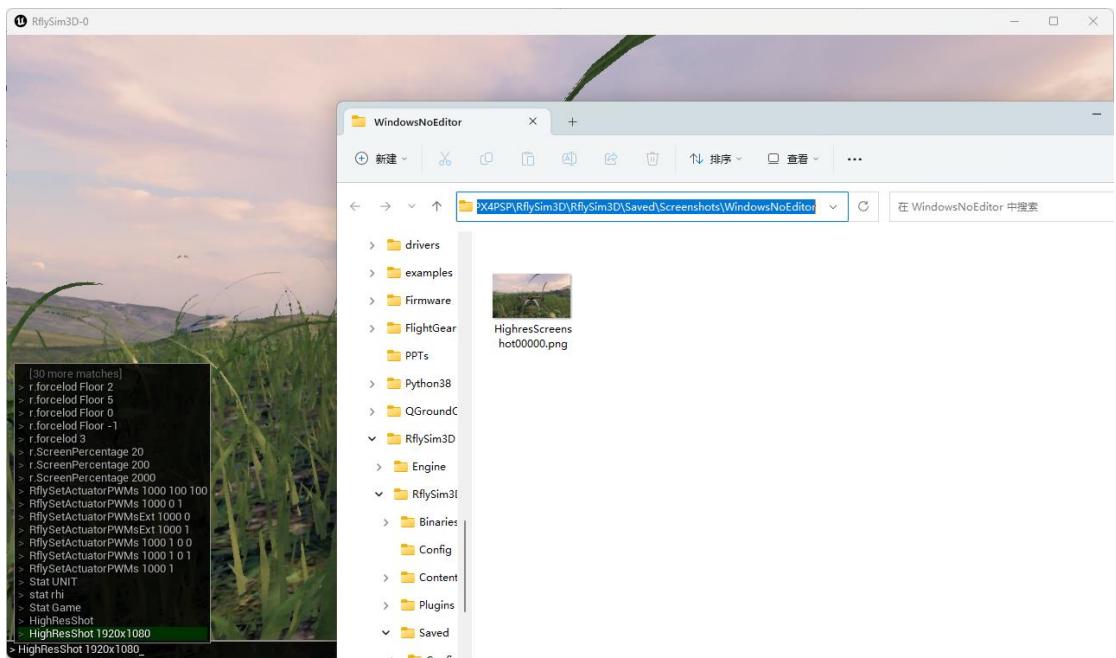
stat rhi: 显示 GPU 上的消耗细节，包括顶点、像素和 GPU 时间等信息，用于定位和优化渲染的瓶颈。



stat game: 显示各进程 Tick 反馈时间，可评估计算性能的消耗情况，找到性能瓶颈并进行相应优化。



HighResShot: 进行自定义尺寸的截图，用于观察和分析不同渲染参数下的性能表现，以便调整优化。



Step 15: 修改常用渲染参数

't.Maxfps 60' 可以设置最大帧率为 60。



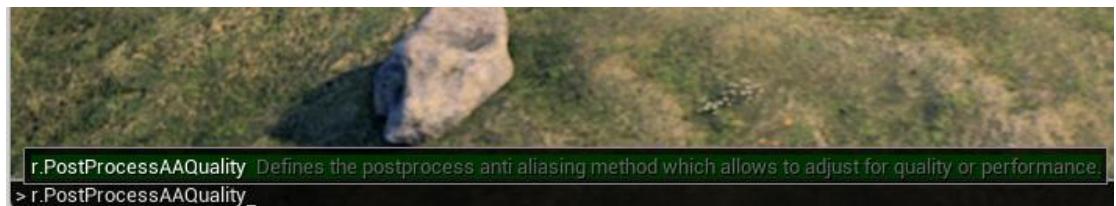
图 33

r.forcelod: 控制 LOD (细节层次) 的点数和面数，以平衡模型细节和性能消耗。



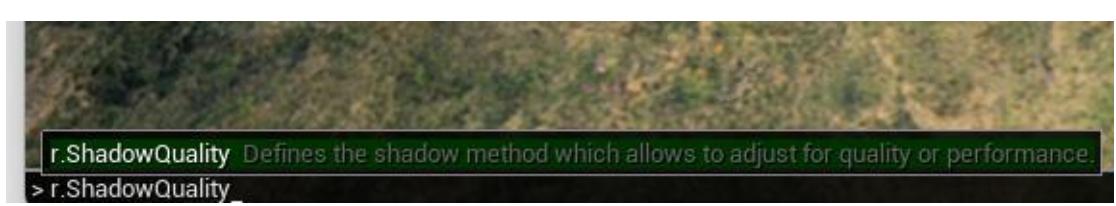
效果图

r.PostProcessAAQuality: 调整后处理抗锯齿质量，平衡画面质量与性能需求。



效果图

r.ShadowQuality: 设置阴影质量，平衡渲染效果和性能需求。



效果图

slomo: 修改仿真的运行速度，可用于测试不同计算负载下的性能表现和优化效果。



效果图

7、参考资料

- [1]. RflySim3D 控制台命令接口总览 [\(见 API 文档\)](#)
- [2]. [命令行参数 | 虚幻引擎文档 \(unrealengine.com\)](#): <https://docs.unrealengine.com/4.27/zh-CN/ProductionPipelines/CommandLineArguments/>
- [3]. [\[UE4\] 常用控制台命令_ue4 控制台命令行大全_somnusand 的博客-CSDN 博客](#): <https://blog.csdn.net/somnusand/article/details/115511383>

8、常见问题

Q1: ****

A1: ****