1、实验名称及目的

Python 场景控制高级接口验证实验：在进行仿真时，Python 函数通过调用 RflySim3D 的命令接口函数或蓝图接口函数，实现包括发送命令、更新无人机状态、附加无人机等操作。

2、实验原理

RflySim3D 关于 Python 的外部接口都在“UE4CtrlAPI.py”文件中定义，主要是“UE4CtrlAPI”类的场景控制接口，该类的作用是收发 UDP 消息且定义了控制 RflySim3D 场景中物体的方法。该类可将各种消息封装为 UDP 然后发送出去，同时还可以接收 RflySim3D 发送场景中的各类 UDP 消息。其构造函数如下：

```
    # constructor function
        def __init__(self, ip='127.0.0.1'):
            self.ip = ip

            self.udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  # Create sock
et
            self.udp_socket.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
            self.udp_socketUE4 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  # Create s
ocket
            self.udp_socketUE4.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
            self.inSilVect = []
            self.inReqVect = []

            self.stopFlagUE4=True
            self.CoptDataVect=[]
            self.ObjDataVect=[]
            self.CamDataVect=[]
```

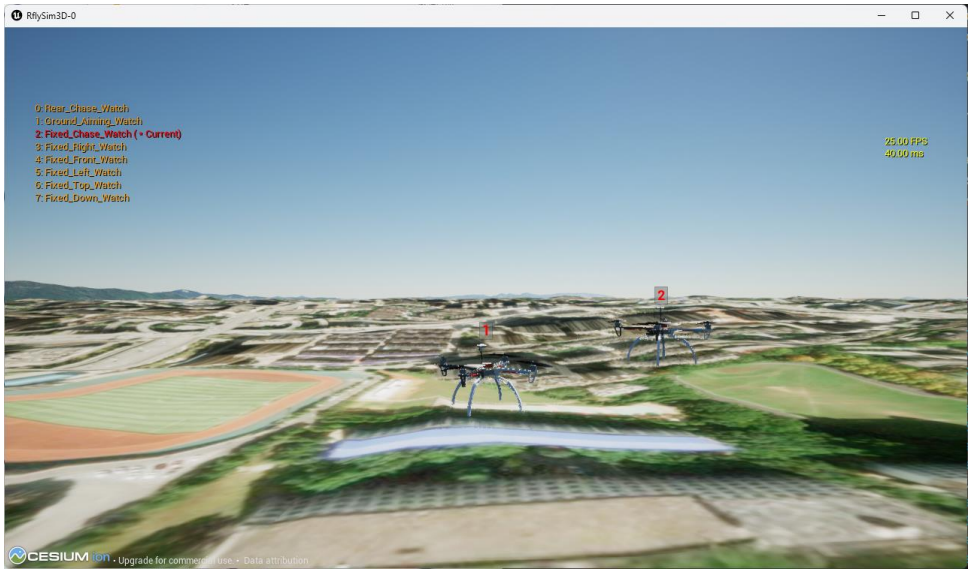
self.udp_socket 用于发送广播消息，它被设置为支持广播，self.udp_socketUE4 用于接收来自 RflySim3D 的数据，它被设置为支持地址重用。本例程只关注发送消息给 RflySim3D 从而控制三维场景内物体的接口，其余接口的使用方法可在 API 文档中的外部接口文件[3]中找到。

利用“UE4CtrlAPI.py”库文件中的“UE4CtrlAPI”类可以传入发送端的端口与 IP 地址进行实例化：

```
    import UE4CtrlAPI as UE4CtrlAPI
    ue = UE4CtrlAPI.UE4CtrlAPI()
```

3、实验效果

本实验利用 python 接口创建了两个飞机，1 号飞机在前进，并且 2 号飞机在围绕着 1 号飞机旋转并随之移动。



效果图

图 1

4、文件目录

文件夹/文件名称	说明
WestTransportC130J	烘焙好的、能被 RflySim3D 识别自定义的飞机模型
PythonCMDDemo.py	此文件调用了“UE4CtrlAPI.py”中的接口
PythonSendUE4Attatch.py	此文件调用了“UE4CtrlAPI.py”中的接口
PythonSendUE4ExtDemo.py	此文件调用了“UE4CtrlAPI.py”中的接口
PythonSendUE4Pos.py	此文件调用了“UE4CtrlAPI.py”中的接口

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 平台完整版		
3	Python 3.11		

推荐配置请见：https://doc.rflysim.com

6、实验步骤

Step 1：

为了保证 RflySim 平台安装包的大小，本实验中所用到的三维场景、飞机模型等较大文件均已上传至百度网盘中，请在实验前进行下载，下载链接为：https://pan.baidu.com/s/1O98i2oQmsE81pBRJQr_46A

提取码：z85c。下载完成后，进行解压放入本例程文件夹中。注：请勿修改文件夹名称。

Step 1: 在库文件 UE4CtrlAPI.py 中找到对应接口函数定义和用法[2][3][5]

找到“UE4CtrlAPI.py”文件，其中定义了大量 RflySim 平台内置的 python 接口函数，在其中的 UE4CtrlAPI 类[3]下可以找到与 RflySim3D 三维场景内物体控制相关的接口，与 RflySim3D 场景内物体控制相关的函数都带着前缀“sendUE4”。

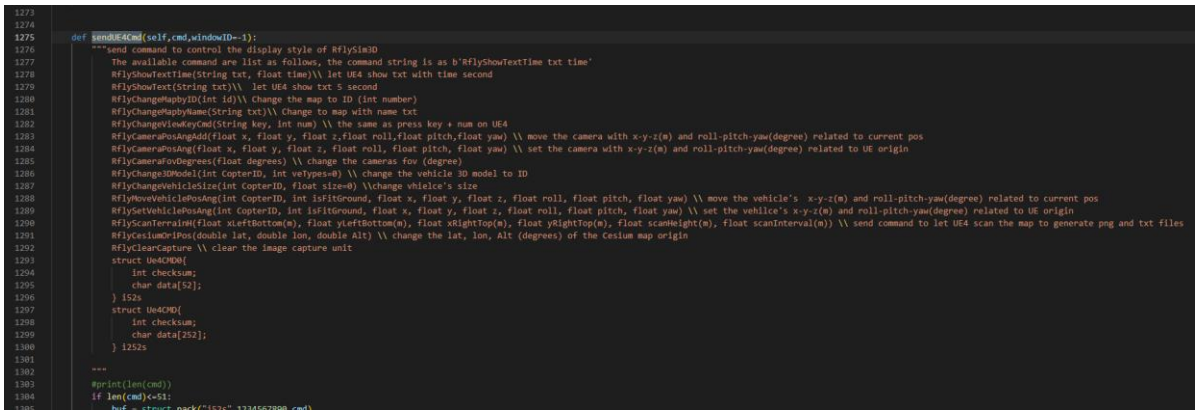


图 2 sendUE4Cmd 函数

该函数是与 RflySim3D 相关的最重要的函数，从它的大量注释可以看到我们之前介绍的 RflySim3D 命令接口[2]基本都在上面。

```
    def sendUE4ExtAct(self,copterID=1,ActExt=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],windowID=-1):
        # struct Ue4ExtMsg {
        #     int checksum;//1234567894
        #     int CopterID;
        #     double runnedTime; //Current  stamp (s)
        #     double ExtToUE4[16];
        # }
        #struct.pack 2i1d16f
        runnedTime = time.time()-self.startTime
        checkSum=1234567894
        buf = struct.pack("2i1d16d",checkSum,copterID,runnedTime,*ActExt)
        if windowID<0:
            if self.ip=='127.0.0.1':
                for i in range(6):
                    self.udp_socket.sendto(buf, (self.ip, 20010+i))
            else:
                self.udp_socket.sendto(buf, ('224.0.0.10', 20009)) #multicast address, send to all RflySim3Ds on all PC in LAN
        else:
            if self.ip!='127.0.0.1' and self.ip!='255.255.255.255':
                self.udp_socket.sendto(buf, ('127.0.0.1', 20010+windowID)) #ensure this PC can reciver message under specify IP mode
            self.udp_socket.sendto(buf, (self.ip, 20010+windowID)) #specify PC's IP to send
        #print('Message Send')
```

图 3 sendUE4ExtAct 函数

该函数可以调用目标无人机的蓝图接口[5]“ActuatorInputsExt”，要求完整版 RflySim3D，有 16 维参数。

```python
def sendUE4PosNew(self,copterID=1,vehicleType=3,PosE=[0,0,0],AngEuler=[0,0,0],VelE=[0,0,0],PWMs=[0]*8,runnedTime=-1,windowID=-1):
    """ send the position & angle information to RflySim3D to create a new 3D model or update the old model's states
        # //输出到模拟器的数据
        # struct SOut2SimulatorSimpleTime {
        #     int checkSum; //1234567890
        #     int copterID;  //Vehicle ID
        #     int vehicleType;  //Vehicle type
        #     float PWMs[8];
        #     float VelE[3];
        #     float AngEuler[3];  //Vehicle Euler angle roll pitch yaw (rad) in x y z
        #     double PosE[3];   //NED vehicle position in earth frame (m)
        #     double runnedTime; //Current Time stamp (s)
        # };
        #struct.pack 3i14f4d
    """
    # if runnedTime<0:
    #     runnedTime = time.time()-self.startTime
    checkSum=1234567890
    # pack for SOut2SimulatorSimpleTime
    buf = struct.pack("3i14f4d",checkSum,copterID,vehicleType,*PWMs,*VelE,*AngEuler,*PosE,runnedTime)
    #print(len(buf))
    if windowID<0:
        if self.ip=='127.0.0.1':
            for i in range(6):
                self.udp_socket.sendto(buf, (self.ip, 20010+i))
        else:
            self.udp_socket.sendto(buf, ('224.0.0.10', 20009)) #multicast address, send to all RflySim3Ds on all PC in LAN
    else:
        if self.ip!='127.0.0.1' and self.ip!='255.255.255.255':
            self.udp_socket.sendto(buf, ('127.0.0.1', 20010+windowID)) #ensure this PC can reciver message under specify IP mode
        self.udp_socket.sendto(buf, (self.ip, 20010+windowID)) #specify PC's IP to send
    #print('Message Send')
```

图 4 sendUE4PosNew 函数

它会发送一次无人机的数据，包括无人机的 ID、ClassID、位置、姿态、速度、各电机数据等。当 RflySim3D 收到这样的数据后，会检测无人机的 ID，如果发现该 ID 不存在于场景中，则会创建一个该飞机，如果已经存在该 ID 的飞机，则它会更新该无人机的各项数据。

```python
def sendUE4Attatch(self,CopterIDs,AttatchIDs,AttatchTypes,windowID=-1):
    """ Send msg to UE4 to attach a vehicle to another (25 vehicles);
    CopterIDs,AttatchIDs,AttatchTypes can be a list with max len 25
    """
    # change the 1D variable to 1D list
    if isinstance(CopterIDs,int):
        CopterIDs=[CopterIDs]

    if isinstance(AttatchIDs,int):
        AttatchIDs=[AttatchIDs]

    if isinstance(AttatchTypes,int):
        AttatchTypes=[AttatchTypes]

    if not isinstance(CopterIDs,list) or not isinstance(AttatchIDs,list) or not isinstance(AttatchTypes,list):
        print('Error: Wrong sendUE4Attatch input Type');
        return

    if len(CopterIDs)!=len(AttatchIDs) or len(CopterIDs)!=len(AttatchTypes) or len(CopterIDs)>25:
        print('Error: Wrong sendUE4Attatch input dimension');
        return

    vLen=len(CopterIDs)
    if vLen<25: # Extend the IDs to 25D
        CopterIDs = CopterIDs + [0]*(25-vLen)
        AttatchIDs = AttatchIDs + [0]*(25-vLen)
        AttatchTypes = AttatchTypes + [0]*(25-vLen)

    if vLen>25:
        CopterIDs=CopterIDs[0:25]
        AttatchIDs=AttatchIDs[0:25]
        AttatchTypes=AttatchTypes[0:25]

    # struct VehicleAttatch25 {
    #   int checksum;//1234567892
    #   int CopterIDs[25];
    #   int AttatchIDs[25];
    #   int AttatchTypes[25];//0: 正常模式, 1: 相对位置不相对姿态, 2: 相对位置+偏航（不相对俯仰和滚转）, 3: 相对位置+全姿态（俯仰滚转偏航）
    # }i25i25i25i

    buf = struct.pack("i25i25i25i",1234567892,*CopterIDs,*AttatchIDs,*AttatchTypes)
    if windowID<0:
```

图 5 sendUE4Attatch

它的作用是将一组无人机附加在另一组无人机上。

附加有几种模式：0：正常模式，1：相对位置不相对姿态，2：相对位置+偏航（不相对俯仰和滚转），3：相对位置+全姿态（俯仰滚转偏航）

Step 2: sendUE4Cmd 函数（发送控制台命令[2][4]）

该函数的逻辑其实就是把一个字符串“cmd”发送给 RflySim3D，如果该 cmd 是正确的“命令接口函数”则会直接调用 RflySim3D 的对应函数，产生的效果与 RflySim3D 控制台命令完全一样。

打开该文档目录下的“PythonCMDDemo.py”文件，它非常简单，只有 3 句代码。

```
import UE4CtrlAPI as UE4CtrlAPI
ue = UE4CtrlAPI.UE4CtrlAPI()
ue.sendUE4Cmd(b'RflyChangeMapbyName Grasslands')
```

第一句就是将 UE4CtrlAPI.py 文件的 UE4CtrlAPI 模块引入当前 py 程序；第二句代码表示创建了一个对象，在库文件中已包含了 RflySim 平台的通信的端口，这两句基本是 RflySim3D 的 python 脚本所必备的，不必太关心。第三句就是给 RflySim3D 发送了一个字符串“RflyChangeMapbyName Grasslands”，在 RflySim3D 控制台命令[2]中我们也介绍过了，该字符串表示调用了接口 RflyChangeMapbyName(String txt)，它的作用是将地图切换为“Grasslands”。

打开 RflySim3D，然后运行文档目录下“PythonCMDDemo.py”，可以看到地图被切换了：

图 6

现在可以自由尝试 RflySim3D 控制台命令[2]中介绍的所有命令，或者把这些命令组合起来一同发送。

该函数用 UDP 发送的结构体[4]形式如下：

```
struct Ue4CMD0{
    int checksum;
    char data[52];
} i52s
struct Ue4CMD{
    int checksum;
    char data[252];
```

```
} i252s
```

其中 char 数组保存的就是发送的命令。

Step 3: sendUE4ExtAct 函数（触发扩展蓝图接口[4][5]）

之前使用 RflySim3D 控制台命令接口输入命令，发送信号告诉飞机产生爆炸效果的实验，我们在这里可以使用 python 的方式再做一次：

将此目录下的“WestTransportC130J”文件夹复制到“\PX4PSP\RflySim3D\RflySim3D\Content”目录下，它是一个烘焙好的、能被 RflySim3D 识别自定义的飞机模型。然后打开 RflySim3D，鼠标双击地面，快速按下字母 O+数字 202，即可在地面创建出该飞机。

然后我们再打开此目录下的“PythonSendUE4ExtDemo.py”文件运行，我们可以发现它与直接输入 RflySim3D 控制台命令的效果是一样的。

图 7

该文件也只有三行代码：

```
import PX4MavCtrlV4 as PX4MavCtrl
mav = PX4MavCtrl.PX4MavCtrler(20100)
mav.sendUE4ExtAct(1000,[0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0])
```

第三句的意思就是调用 ID 为 1000 的无人机的蓝图接口“ActuatorInputsExt”，并传入后续 16 维数组作为它的参数。而该飞机传入参数的第 8 位如果是 1，会触发它的爆炸逻辑。

其实根据上一小节的内容，我们可以知道如果使用如下代码：

```
ue.sendUE4Cmd(b'RflySetActuatorPWMsExt 1000 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0')
```

完全可以达成同样的效果，在 RflySim3D 控制台命令接口中[2]，我们介绍过该命令接口，它的作用也是调用无人机的蓝图接口“ActuatorInputsExt”，和 sendUE4ExtAct 函数作用是一样的。那么如果通过上一节中的命令，将整个控制台命令发送给 RflySim3D，那么它也是可以触发飞机的爆炸效果的。

该函数用 UDP 发送的结构体形式[4]如下：

```
struct Ue4ExtMsg {
    int checksum;//1234567894
    int CopterID;
```

```
        double runnedTime; //Current  stamp (s)
        double ExtToUE4[16];
    }
```

它是一个长度为 4×1+4×1+8×1+8×16=144 字节的结构体，其中 pwm[16]就是发送的 16 维电机数据。

Step 4: sendUE4PosNew 函数（创建模型并传入 8 位电机数据[4][5]）

打开 RflySim3D，再启动并运行此目录下的“PythonSendUE4Pos.py”，我们可以看见一个飞机被创建出来了：

图 8

该文件也只有三行：

```
import UE4CtrlAPI as UE4CtrlAPI
ue = UE4CtrlAPI.UE4CtrlAPI()
ue.sendUE4PosNew(10,3,[0,0,-10],[0,0,0],[0,0,0],[0,0,0,0,0,0,0,0])
```

第三句的意思是发送了一个 ID 为 10 的飞机的数据，它的 ClassID 为 3，表示我们创建的是一个四旋翼无人机，[0,0,-10]表示它距离水平面高 10 米。

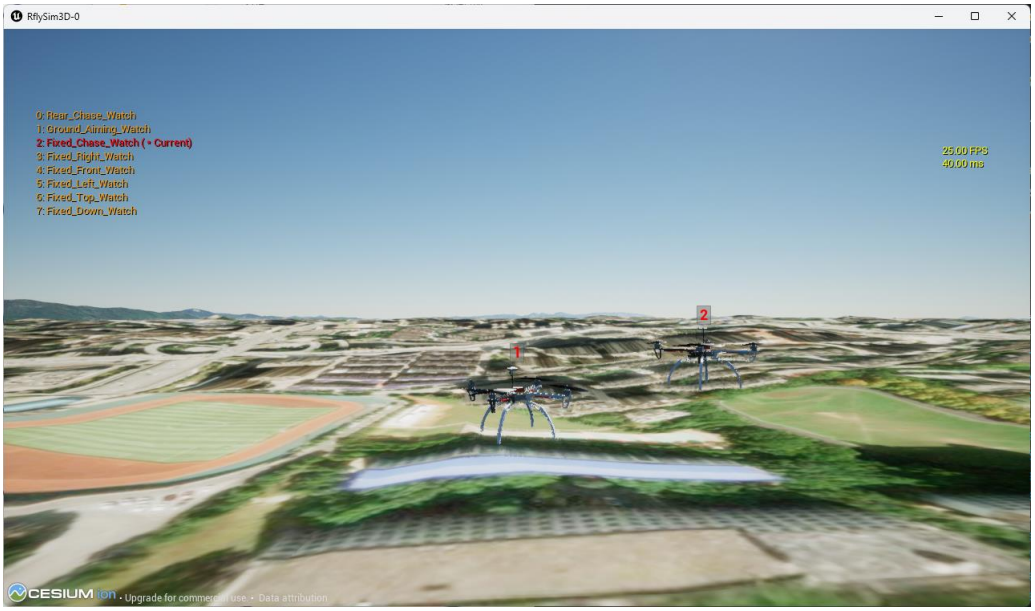
该函数用 UDP 发送的结构体形式[4]如下：

```
struct SOut2SimulatorSimpleTime {
    int checkSum; //1234567890
    int copterID;  //Vehicle ID
    int vehicleType;  //Vehicle type
    float PWMs[8];
    float VelE[3];
    float AngEuler[3];  //Vehicle Euler angle roll pitch yaw (rad) in x y z
    double PosE[3];   //NED vehicle position in earth frame (m)
    double runnedTime; //Current Time stamp (s)
};
```

Step 5: sendUE4Attatch（将一个 Copter 附加到另一个 Copter 上[1] [3] [5]）

打开 RflySim3D，再启动并运行目录下的“PythonSendUE4Attatch.py”，我们可以看见

创建了两个飞机，1 号飞机在前进，并且 2 号飞机在围绕着 1 号飞机旋转并随之移动。



效果图

图 9

该文件代码如下：

```
import time
import UE4CtrlAPI as UE4CtrlAPI
ue = UE4CtrlAPI.UE4CtrlAPI()
ue.sendUE4PosNew(1,3,[0,0,-10],[0,0,0],[0,0,0],[0,0,0,0,0,0,0,0])
ue.sendUE4PosNew(2,3,[0,0,0],[0,0,0],[0,0,0],[0,0,0,0,0,0,0,0])
ue.sendUE4Attatch(2,1,3)
ue.sendUE4PosNew(2,3,[1,0,0],[0,0,0],[0,0,0],[0,0,0,0,0,0,0,0])
ue.sendUE4Cmd(b"RflyChangeViewKeyCmd S -1")
ue.sendUE4Cmd(b"RflyChangeViewKeyCmd N -1")
theta=0
posX=0
while True:
    ue.sendUE4PosNew(1,3,[posX,0,-10],[0,0,theta],[0,0,0],[0,0,0,0,0,0,0,0])
    theta=(theta+0.1)%360
    posX=posX+0.1
    time.sleep(0.1)
```

这串代码先用两个 sendUE4PosNew 函数创建了两个四旋翼飞机，ID 分别为 1 与 2。然后是使用 sendUE4Attatch 函数以 3 号附加模式将 2 号飞机附加到 1 号飞机上。然后再使用了一个 sendUE4PosNew 函数重新设置 2 号飞机的位置（此时设置的是 2 号飞机相对于 1 号飞机的位置，设置在了 1 号飞机前方 1 米处）。

然后给 RflySim3D 发送了两个按键命令[1]，相当于手动在 RflySim3D 中按下 S 键、N 键，-1 表示不加数字键，（在 3.2.3 中介绍过 S 键是显示飞机的 ID，N 键是启动上帝视角，这样视角就不会跟着飞机转了，否则容易头晕）。

接下来是一个 while true 的循环，可以看见该程序每隔 0.1s 就向 RflySim3D 发送一次 1 号飞机的坐标，每次 1 号飞机的 x 坐标会增大 0.1 米，绕 z 轴的旋转角 yaw 增大 0.1 度。

可以看见，我们并没有更新 2 号飞机的位置，但是 2 号飞机仍在运动，这是因为它已

经被“附加在 1 号飞机上”了，1 号飞机运动时 2 号飞机会自动跟随运动。

Step 6:其他函数

还有许多函数，但功能大同小异，或者曾介绍过类似的功能，这里就只简单介绍它们的功能了，可以将例程稍加修改实验它们的作用。

```
sendUE4LabelID(self,CopterID=0,Txt='',fontSize=30,RGB=[255,0,0],windowID=-1)
```

它与之前介绍的“RflySetIDLabel”命令作用一致，在目标无人机头上显示一个字符串。

```
sendUE4LabelMsg(self,CopterID=0,Txt='',fontSize=30,RGB=[255,0,0],dispTime=0,dispFlag=-1,windowID=-1)
```

它与之前中介绍的“RflySetMsgLabel”命令作用一致，在目标无人机头顶的 ID 标签下方再显示消息标签。

```
sendUE4Pos(self,copterID=1,vehicleType=3,MotorRPMSMean=0,PosE=[0,0,0],AngEuler=[0,0,0],windowID=-1)
```

该函数是 sendUE4PosNew 函数的简化版本，但作用是一样的。

```
sendUE4Pos2Ground(self,copterID=1,vehicleType=3,MotorRPMSMean=0,PosE=[0,0,0],AngEuler=[0,0,0],windowID=-1)
```

该函数与 sendUE4PosNew 相似，但由它生成的无人机的 z 坐标不起作用，其 z 坐标会贴合当前的地面。

```
sendUE4PosScale(self,copterID=1,vehicleType=3,MotorRPMSMean=0,PosE=[0,0,0],AngEuler=[0,0,0],Scale=[1,1,1],windowID=-1)
```

该函数作用与 sendUE4PosNew 也相似，也是发送无人机的数据，只是更新的数据有所不同，它额外发送了一个缩放数据 Scale，可以控制无人机的缩放大小。

```
sendUE4PosScale2Ground(self,copterID=1,vehicleType=3,MotorRPMSMean=0,PosE=[0,0,0],AngEuler=[0,0,0],Scale=[1,1,1],windowID=-1)
```

该函数与 sendUE4Pos2Ground 函数相似，也是发送的数据稍有不同，它也发送了一个无人机的缩放数据 Scale

```
sendUE4PosFull(self,copterID,vehicleType,MotorRPMS,VelE,PosE,RateB,AngEuler,windowID=-1)
```

该函数也是发送无人机的数据，更新的数据稍有不同，附加了无人机的加速度、角速度、GPS 位置等信息。

```
sendUE4PosSimple(self,copterID,vehicleType,PWMs,VelE,PosE,AngEuler,runnedTime=-1,windowID=-1)
```

该函数也是发送无人机的数据，与 sendUE4PosNew 很相似。

```
sendUE4PosScale100(self,copterID,vehicleType,PosE,AngEuler,MotorRPMSMean,Scale,isFitGround=False,windowID=-1)
```

该函数可以一次发送 100 架无人机的数据，将它们放在同一个结构体里，增加大规模集群仿真时的信息密度。

```
sendUE4PosScalePwm20(self,copterID,vehicleType,PosE,AngEuler,Scale,PWMs,isFitGround=False,windowID=-1):
```

该函数可以一次发送 20 架无人机的数据。

7、参考资料

[1]. RflySim3D 快捷键接口总览..\..\..\API.pdf

[2]. RflySim3D 控制台命令接口总览..\..\..\API.pdf

[3]. UE 场景控制 python 接口总览..\..\..\API.pdf

[4]. RflySim3D 外部交互接口数据协议..\..\..\API.pdf

[5]. RflySim3D 模型导入及蓝图控制规则..\..\..\API.pdf

8、常见问题

1. 无