1、实验名称及目的

获取 RflySim3D 内所有动态创建物体位置、碰撞数据实验：通过平台提供的 python 接口获取 RflySim3D 内所有动态创建物体位置、碰撞数据。

2、实验原理

```
import time
import math
import sys
import UE4CtrlAPI as UE4CtrlAPI
```

首先导入必要的依赖库文件

```
ue = UE4CtrlAPI.UE4CtrlAPI()
```

调用 UE4CtrlAPI.py 库文件下的 UE4CtrlAPI 类创建一个通信实例 ue。

```
ue.sendUE4Cmd(b'RflyReqVehicleData 1')
```

发送消息给 RflySim3D，让其将当前收到的飞机数据转发出来，回传到组播地址 224.0.0.10 的 20006 端口 。注：只有飞机位置发生改变时，才会将位置数据传出，因此本语句要放在最前面，确保后续创建的物体（Python 一次性创建）都能被传出

```
ue.initUE4MsgRec()
```

Python 开始飞机数据的监听，数据存储在 inReqUpdateVect 列表（是否更新标志），和 inReqVect 列表（碰撞数据）中。注意：监听语句应该放到 sendUE4Pos 系列语句之前，不然无法捕获创建的障碍物。该方法在 UE4CtrlAPI.py 中的完整定义如下

```
def initUE4MsgRec(self):
        """ Initialize the UDP data linsening from UE4,
        currently, the crash data is listened
        """
        self.stopFlagUE4=False
        #print("InitUE4MsgRec", self.stopFlagUE4)
        self.inSilVect = []
        self.inReqVect = []
        self.inReqUpdateVect = []
        MYPORT = 20006
        MYGROUP = '224.0.0.10'
        ANY = '0.0.0.0'
        self.udp_socketUE4.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
        self.udp_socketUE4.bind((ANY,MYPORT))
        status = self.udp_socketUE4.setsockopt(socket.IPPROTO_IP,
            socket.IP_ADD_MEMBERSHIP,
            socket.inet_aton(MYGROUP) + socket.inet_aton(ANY))
        self.t4 = threading.Thread(target=self.UE4MsgRecLoop, args=())
        self.t4.start()
```

- 首先，将`self.stopFlagUE4`设置为 False，表示不停止 UE4 消息的接收。
- 然后，创建三个空列表`self.inSilVect`、`self.inReqVect`和`self.inReqUpdateVect`，用于存储接收到的数据。
- 接下来，定义了常量`MYPORT`和`MYGROUP`，分别表示端口号和组播地址。
- 然后，将`ANY`设置为`'0.0.0.0'`，表示接收来自任何 IP 地址的数据。
- 接着，通过`setsockopt`方法设置了一些 Socket 选项，其中`socket.SO_REUSEADD

R`表示允许地址重用。

- 然后，通过`bind`方法将 UDP 套接字绑定到指定的 IP 地址和端口号。
- 接下来，通过`setsockopt`方法将套接字加入到指定的 IP 组中，以便接收组内的数据。
- 最后，创建了一个新的线程`t4`，并将其绑定到`UE4MsgRecLoop`方法上，并启动该线程。这个方法将负责接收 UE4 发来的消息。

```
ue.sendUE4Pos(100,30,0,[2.5,0,-8.086],[0,0,math.pi])
ue.sendUE4PosScale(101,2030,0,[10.5,0,-8.086],[0,0,math.pi],[10,10,10])
```

创建 CopterID 分别为 100、101 的障碍物

```
TargetCopterID = 1
PosEF = ue.getUE4Pos(TargetCopterID)
```

通过 getUE4Pos 接口来获取 1 号飞机位置数据，这架飞机是启动软件在环脚本时自动创建的。

PosEF[4] = PosE[3]+Flag(是否有数据)

```
TargetCopterID = 100
PosEF = ue.getUE4Pos(TargetCopterID)
TargetCopterID = 101
PosEF = ue.getUE4Pos(TargetCopterID)
```

通过 getUE4Pos 接口来获取 100 号和 101 号障碍物位置数据

```
TargetCopterID = 102
PosEF = ue.getUE4Pos(TargetCopterID)
```

通过 getUE4Pos 接口来获取不存在的物体位置数据

下面的程序直接通过 inReqUpdateVect 列表和 inReqVect，定时检查接收到的数据是否有更新，并打印出更新后的数据。

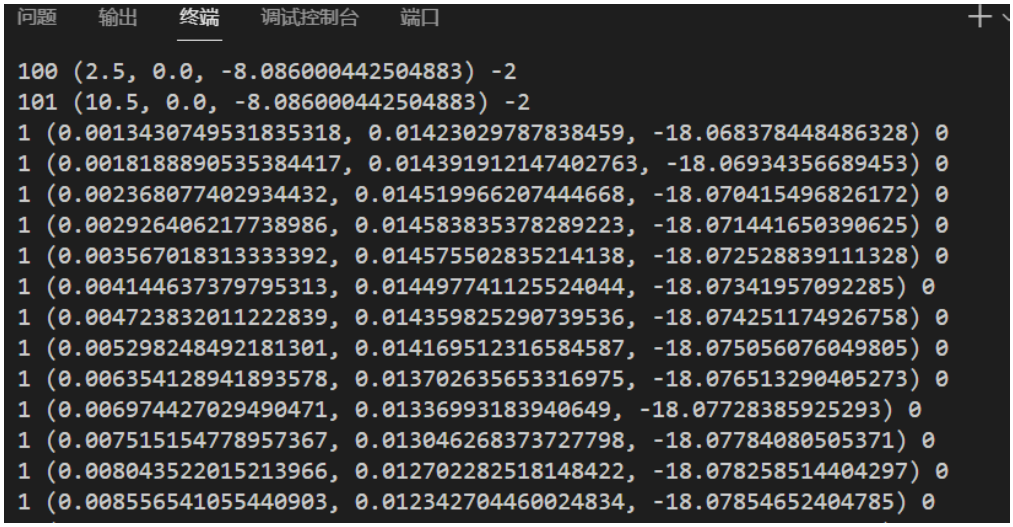
```
lastTime = time.time()
num=0
lastClock=time.time()
lastCount=0
while True:
    lastTime = lastTime + 1/30.0
    sleepTime = lastTime - time.time()
    if sleepTime > 0:
        time.sleep(sleepTime)
    else:
        lastTime = time.time()

    for i in range(len(ue.inReqVect)):
        if ue.inReqUpdateVect[i]: # 如果 i 号数据有更新
            print(ue.inReqVect[i].copterID,ue.inReqVect[i].PosE,ue.inReqVect[i].CrashTyp
e)
            ue.inReqUpdateVect[i]=False
```

inReqVect[i]列表中的 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。这是在 reqVeCrashData 结构体[3]中定义的。

3、实验效果

本实验通过 python 接口获取获取 RflySim3D 内所有动态创建物体位置、碰撞数据。



4、文件目录

文件夹/文件名称	说明
GetUE4PosAPI.bat	软件在环仿真实验脚本
GetUE4PosAPI.py	Python 实验脚本

5、运行环境

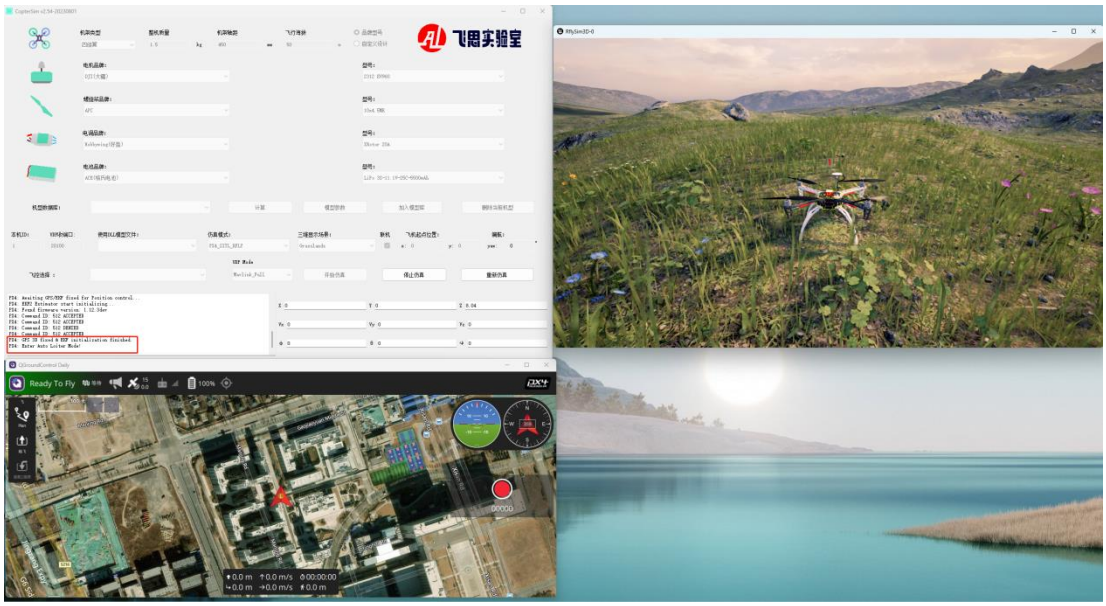
序号	软件要求	硬件要求	
		名称	数量(个)
1	Windows 10 及以上版本	笔记本/台式电脑①	1
2	RflySim 平台个人高级版及以上		
3	Visual Studio Code		

① ：推荐配置请见：https://doc.rflysim.com

6、实验步骤

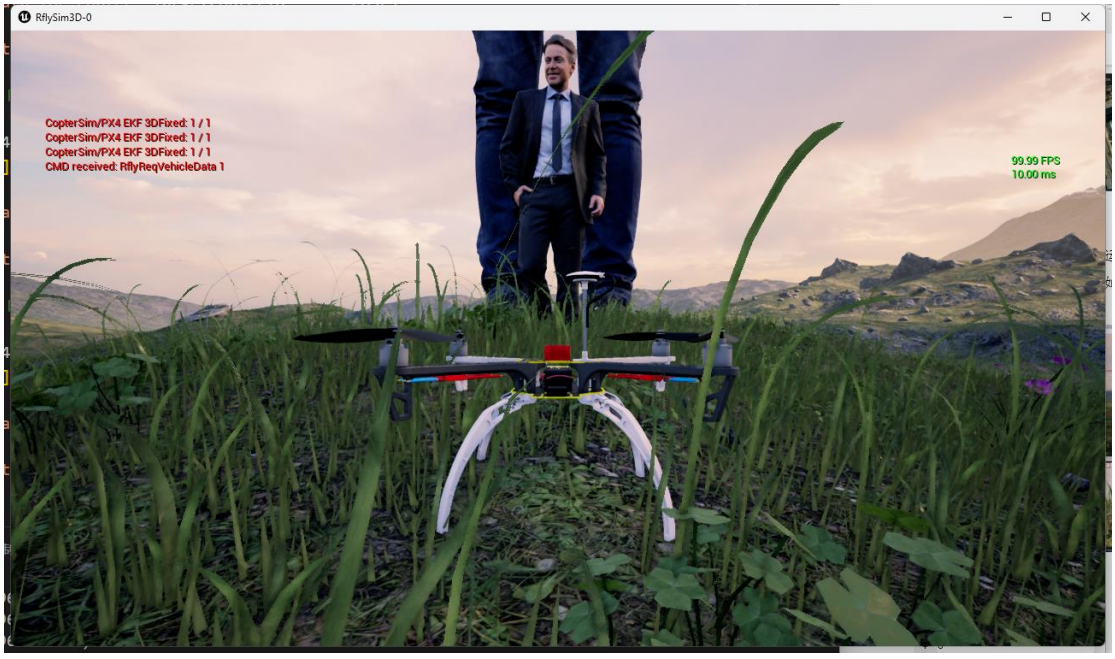
Step 1：

以管理员方式运行 GetUE4PosAPI.bat，开启一个飞机的软件在环仿真。将会启动 1 个 QGC 地面站，1 个 CopterSim 软件，且其软件下侧日志栏必须打印出“GPS 3D fixed & EKF initialization finished”字样代表初始化完成，并且 RflySim3D 软件内有 1 架无人机，其 CopterID 为 1。



Step 2：

用 VScode 运行 GetUE4PosAPI.py 文件，可以获取到场景内飞机和障碍物的信息，创建的几个物体，如下图所示。



此时飞机在地面上未起飞，因此碰撞物体类型为-2 表示地面

```
76
问题    输出    终端    调试控制台    端口

1 (0.0, 0.0, -8.039999961853027) -2
1 (0.0, 0.0, -8.039999961853027) -2
1 (0.0, 0.0, -8.039999961853027) -2
1 (0.0, 0.0, -8.039999961853027) -2
1 (0.0, 0.0, -8.039999961853027) -2
1 (0.0, 0.0, -8.039999961853027) -2
1 (0.0, 0.0, -8.039999961853027) -2
1 (0.0, 0.0, -8.039999961853027) -2
1 (0.0, 0.0, -8.039999961853027) -2
1 (0.0, 0.0, -8.039999961853027) -2
1 (0.0, 0.0, -8.039999961853027) -2
1 (0.0, 0.0, -8.039999961853027) -2
1 (0.0, 0.0, -8.039999961853027) -2
```

Step 3：

在 QGC 中控制飞机起飞，碰撞物体类型为 0 表示无碰撞

```
1 (0.018122879788279533, 0.014640222303569317, -18.168258666992188) 0
1 (0.01829248107969761, 0.01514009665697813, -18.16765022277832) 0
1 (0.0184441190212965, 0.015618368051946163, -18.167024612426758) 0
1 (0.018617399036884308, 0.01622229814529419, -18.166173934936523) 0
1 (0.018723340705037117, 0.01664811000227928, -18.165538787841797) 0
1 (0.01880689337849617, 0.017049528658390045, -18.1649112701416) 0
1 (0.01888456754386425, 0.01754816435277462, -18.164091110229492) 0
1 (0.018926674500107765, 0.01800825446844101, -18.163293838500977) 0
1 (0.01893548108637333, 0.018327251076698303, -18.162717819213867) 0
1 (0.018924251198768616, 0.01862180419266224, -18.16216468811035) 0
1 (0.01889261044561863, 0.018890563398599625, -18.16164207458496) 0
1 (0.018841609358787537, 0.01913336291909218, -18.16115951538086) 0
1 (0.018773049116134644, 0.019350789487361908, -18.160722732543945) 0
1 (0.018688026815652847, 0.019543945789337158, -18.16033363342285) 0
```

Step 4：

在下图“GetUE4PosAPI.bat”脚本开启的命令提示符 CMD 窗口中，按下回车键（任意键）就能快速关闭 CopterSim、QGC、RflySim3D 等所有程序。

Step 5：

在下图 VS Code 中，点击“终止终端”，可以彻底退出脚本运行。

7、参考文献

[1]. RflySim3D 快捷键接口总览

[2]. RflySim3D 控制台命令接口总览

[3]. RflySim3D 外部接口文件总览

8、常见问题

Q1：无

A1：无