

一、AirSim 是什么

AirSim 是微软公司开发的基于游戏引擎的一个开源的跨平台仿真器，它可以用于无人机、无人机车等机器人的物理和视觉仿真。它同时支持基于 PX4 和 ArduPilot 等飞行控制器的软件在环仿真，目前还支持基于 PX4 的硬件在环仿真。

二、python 环境的搭建

win10 系统下 python 环境的搭建包括安装和配置编辑器和解释器，编辑器是用来写 python 代码的，解释器是用来编译运行 python 程序的。
本文的 python 环境配置，不需要手动添加环境变量。

三、解释器 Anaconda 环境的安装与配置

非常推荐使用 Anaconda 对 python 环境进行管理，Anaconda 中可以创建很多不同的虚拟环境，每个环境之间不会相互干涉，而且可以对每个环境进行个性化的配置，比如针对 AirSim 可以创建一个环境，在这个环境中只安装 AirSim 仿真需要使用的第三方包和通用库，而不会对其他的环境造成影响。

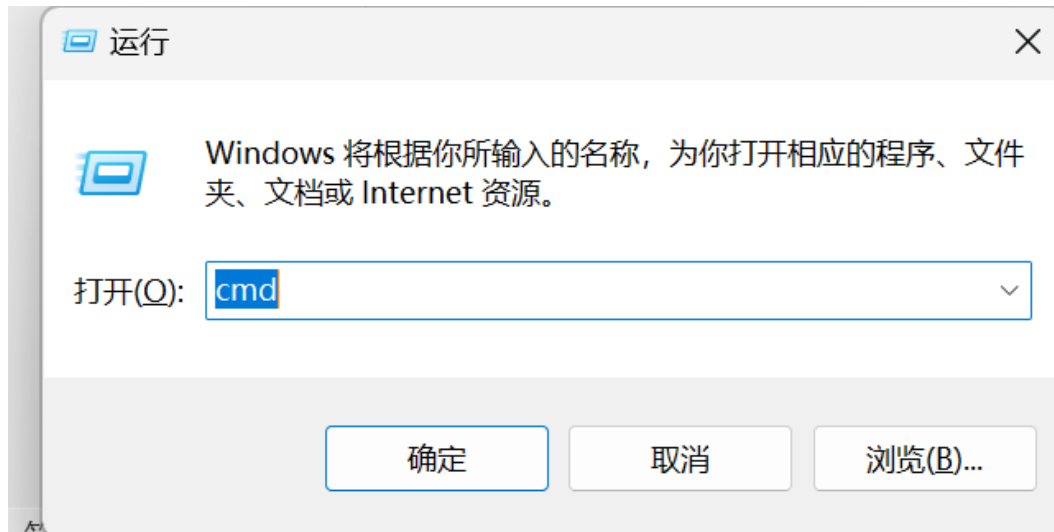
首先下载安装 Anaconda，打开官网 <https://www.anaconda.com/> 点击左上角 Product 中，选择 Individual Edition，然后点击 Download，选择与自己计算机对应的版本安装



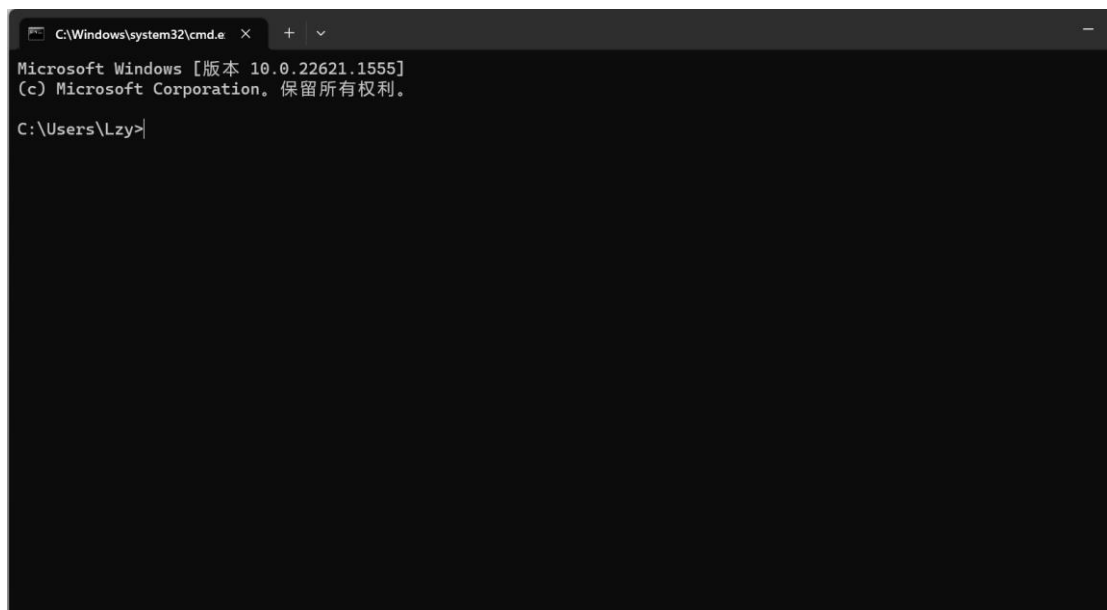
下载好之后，双击打开安装包，按照提示一步一步安装即可。安装成功后，就可以针对 AirSim 专门配置一个虚拟环境。
我们仅使用终端创建 python 虚拟环境就好

四、安装 AirSim 需要用到的第三方包和一些常用的库。

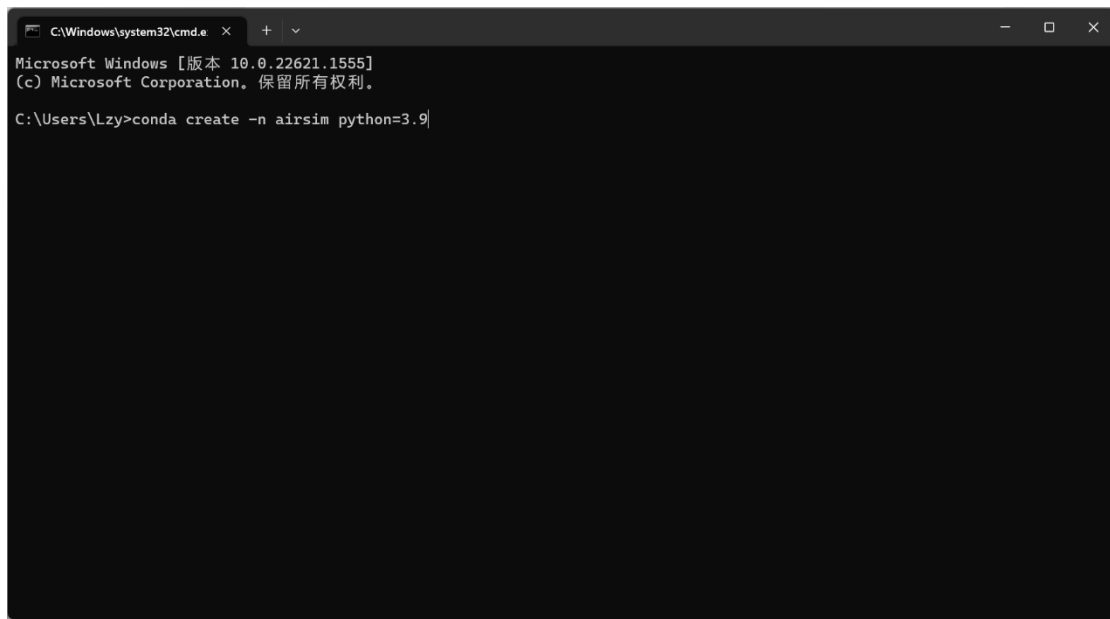
首先创建 python 虚拟环境
按住 Windows 键+R 打开运行框



输入 cmd 打开终端如图所示
输入 “conda create -n airsims python=3.9”



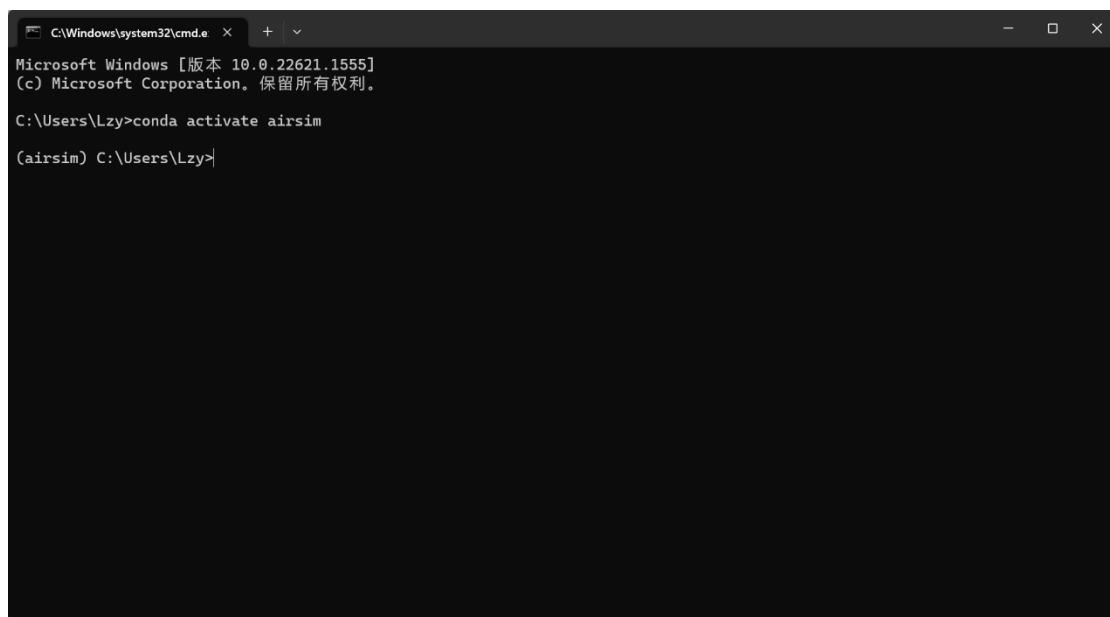
注：“airsims”为环境名，取什么名字都是可以的
“python 后面的是版本号，我这里用的是 3.9”



```
C:\Windows\system32\cmd.e
Microsoft Windows [版本 10.0.22621.1555]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\Lzy>conda create -n airsims python=3.9
```

然后输入“conda activate airsims”进入该虚拟环境



```
C:\Windows\system32\cmd.e
Microsoft Windows [版本 10.0.22621.1555]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\Lzy>conda activate airsims
(airsims) C:\Users\Lzy>
```

接下来就是下载一些需要的包

numpy 是一个矩阵运算库，对于矩阵向量等运算非常方便使用，基本上做估计控制等算法都需要用到。

在进入的虚拟环境中输入：

```
pip install numpy -i https://pypi.tuna.tsinghua.edu.cn/simple
```

<https://pypi.tuna.tsinghua.edu.cn/simple> 是清华镜像源的地址，是一个很稳定的地址，可以用来下载各种库

```
C:\Windows\system32\cmd.e X + v
Microsoft Windows [版本 10.0.22621.1555]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\Lzy>conda activate airmim

(airmim) C:\Users\Lzy>pip install numpy -i https://pypi.tuna.tsinghua.edu.cn/simple
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting numpy
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/9d/3b/13404993b5dec7403abc9518569316b5d72d9a3081cd90aca130e6d8b00/numpy-1.24.2-cp39-cp39-win_amd64.whl (14.9 MB)
    14.9/14.9 MB 16.4 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-1.24.2
```

同理再下载 msgpack-rpc-python 和 airmim 包

```
C:\Windows\system32\cmd.e X + v
Successfully installed numpy-1.24.2

(airmim) C:\Users\Lzy> pip install msgpack-rpc-python -i https://pypi.tuna.tsinghua.edu.cn/simple
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting msgpack-rpc-python
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/bf/67/c3cfa7158c46fa3fb1898783ff722f94c52e8b65f601922c853f46405df3/msgpack-rpc-python-0.4.1.tar.gz (7.7 kB)
    Preparing metadata (setup.py) ... done
Collecting msgpack-python
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/8a/20/6eca772d1a5830336f84aca1d8198e5a3f4715cd1c7fc36d3cc7f7185091/msgpack-python-0.5.6.tar.gz (138 kB)
    139.0/139.0 kB 8.6 MB/s eta 0:00:00
    Preparing metadata (setup.py) ... done
Collecting tornado<5,>=3
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/e3/7b/e29ab3d51c8df66922fea216e2bddfc6b430fb29620e5165b16a216e0d3c/tornado-4.5.3.tar.gz (484 kB)
    484.2/484.2 kB 10.1 MB/s eta 0:00:00
    Preparing metadata (setup.py) ... done
Building wheels for collected packages: msgpack-rpc-python, tornado, msgpack-python
  Building wheel for msgpack-rpc-python (setup.py) ... done
    Created wheel for msgpack-rpc-python: filename=msgpack_rpc_python-0.4.1-py3-none-any.whl size=9320 sha256=a02adce317151331111fb3119496d098608edb30c0b617412ceec02defb775fd
    Stored in directory: c:\users\lzy\appdata\local\pip\cache\wheels\6f\89\96\846de2bad02e1b4590da067e3fb694c231d58614883702f48a
  Building wheel for tornado (setup.py) ... done
    Created wheel for tornado: filename=tornado-4.5.3-cp39-cp39-win_amd64.whl size=420791 sha256=6423d18fe80465a39064a1a7302bdb16e3fe9fc148d4b83363e73b0c0f61b389
    Stored in directory: c:\users\lzy\appdata\local\pip\cache\wheels\4c\00\d1\98a4056f94ee9b5565727c73d9a716034f3e863f7f9488c6c2
  Building wheel for msgpack-python (setup.py) ... done
```

```
C:\Windows\system32\cmd.e x + v
Installing collected packages: tornado, msgpack-python, msgpack-rpc-python
Successfully installed msgpack-python-0.5.6 msgpack-rpc-python-0.4.1 tornado-4.5.3

(airsim) C:\Users\Lzy>pip install airsims -i https://pypi.tuna.tsinghua.edu.cn/simple
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting airsims
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/2f/d7/a692b1a82263b6d510bb04b5278154d096e4e4b43fb68cd2482f33ee9a2a/airsim-1.8.1.tar.gz (20 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: msgpack-rpc-python in d:\anaconda3\envs\airsim\lib\site-packages (from airsims) (0.4.1)
Requirement already satisfied: numpy in d:\anaconda3\envs\airsim\lib\site-packages (from airsims) (1.24.2)
Collecting opencv-contrib-python
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/f3/8b/9d010c60a2254e810ea2bd4a8645f9e12dff3ee8f2de12e7b8b38cdec561/opencv_contrib_python-4.7.0.72-cp37-abi3-win_amd64.whl (44.9 MB)
    44.9/44.9 MB 16.4 MB/s eta 0:00:00
Requirement already satisfied: msgpack-python in d:\anaconda3\envs\airsim\lib\site-packages (from msgpack-rpc-python->airsims) (0.5.6)
Requirement already satisfied: tornado<5,>=3 in d:\anaconda3\envs\airsim\lib\site-packages (from msgpack-rpc-python->airsims) (4.5.3)
Building wheels for collected packages: airsims
  Building wheel for airsims (setup.py) ... done
  Created wheel for airsims: filename=airsim-1.8.1-py3-none-any.whl size=21250 sha256=0d59e8fde9f1030f12704c5faef2a970fcc21c9e27585e35f115249c8193e229
  Stored in directory: c:\users\lzy\appdata\local\pip\cache\wheels\0f\81\8d\797ed336a4a2bece0dd91c5d153ec650ccf03a748e3e3edc6f
Successfully built airsims
Installing collected packages: opencv-contrib-python, airsims
Successfully installed airsims-1.8.1 opencv-contrib-python-4.7.0.72

(airsim) C:\Users\Lzy>
```

其他的常用库，我们后面遇到了再安装。

如下图所示，做 AirSim 仿真需要的两个包就安装成功了。如果显示没有 pip 工具，则需要先安装 pip。

四、编辑器 Pycharm 的安装与配置

Pycharm 是一款非常好用的 python 编辑器，具有代码高亮、自动补全、纠错等非常实用方便的功能，界面也很美观。我们只需要下载社区版 Pycharm（免费）即可。

首先到官网

（<https://www.jetbrains.com/pycharm/download/#section=windows>）下载软件安装包。



Version: 2023.1
Build: 231.8109.197
30 March 2023

[System requirements](#)

[Installation instructions](#)

Download PyCharm

[Windows](#) [macOS](#) [Linux](#)

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

Download

.exe ▼

Free 30-day trial available

Community

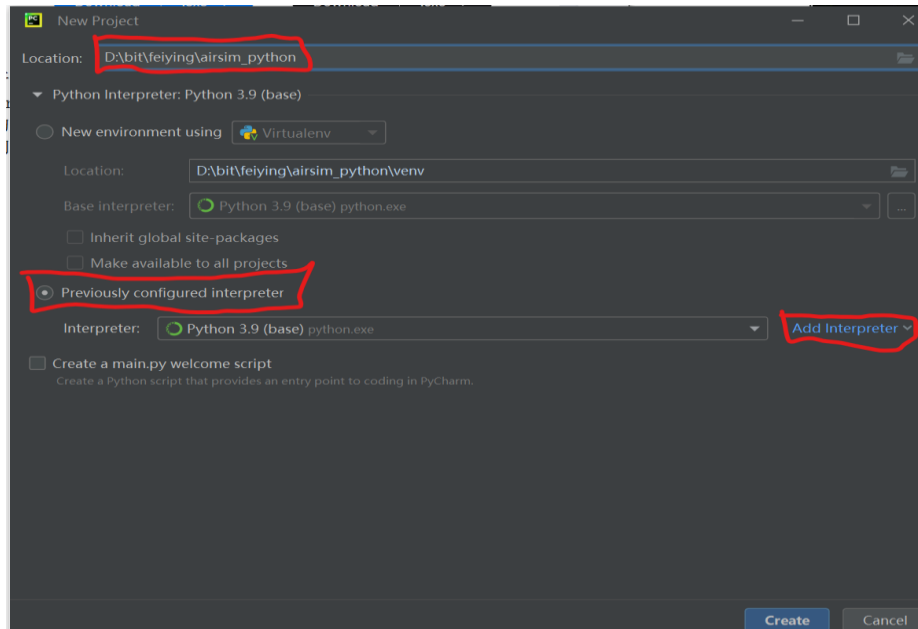
For pure Python development

Download

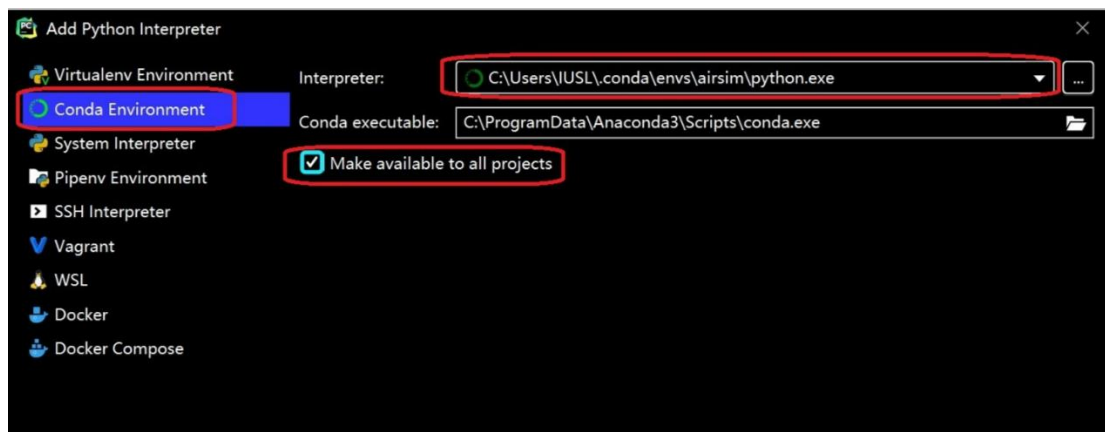
.exe ▼

Free, built on open-source

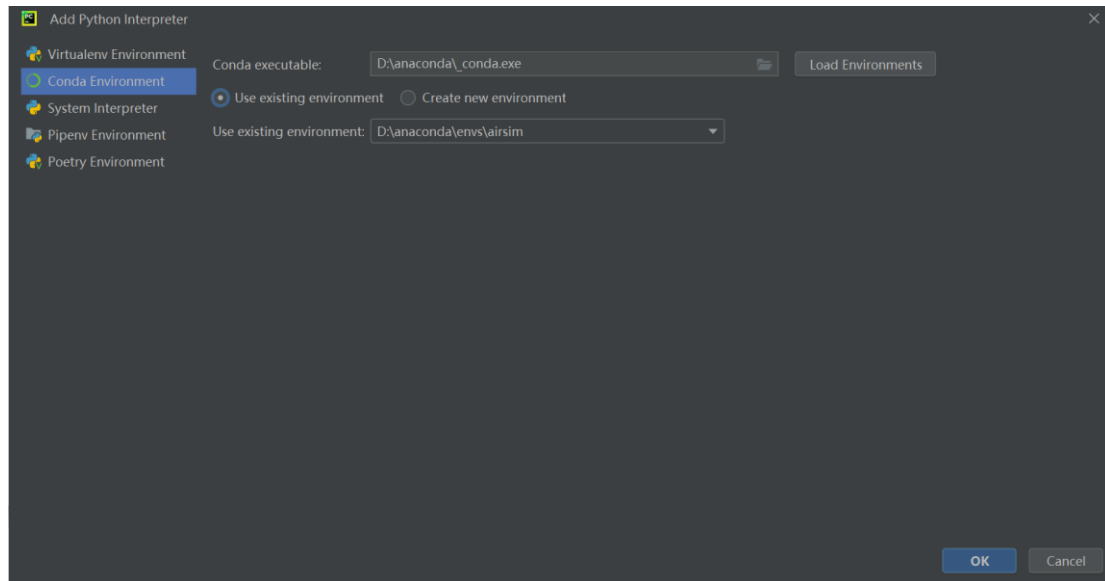
下载好之后，双击打开安装包开始安装，根据提示一步一步安装即可。
打开 pycharm，点击 Create New Project，先选择新建工程的路径，然后选择解释器为上一节创建的 Anaconda 环境，但是注意这里第一次选择的时候是没有这个环境选项的，我们需要手动添加进来。点击 Add interpreter Location 处的位置可以自己选择。



按照自己的路径（先前所创建的 airsims 虚拟环境的路径）进行选择，设置好后点击 ok，再点击 create 即可创建。



但是据我尝试的结果，新版 Pycharm 安装好后会出现配置 conda 虚拟环境时，无法找到 conda 的可执行文件，以至于无法创建 new project。
以下是遇到无法找到 conda executable（conda 可执行文件）的解决方法：只需要在 ‘Conda executable 的输入框里面，找到 anaconda 路径（安装 anaconda 时的路径）下的 “_conda.exe”’，再点击 ‘load environments’，即可出现 ‘use existing environment’ 的输入框，如下图所示。



五、控制无人机起飞和降落

在新建的 python 工程中新建一个 python File，将下面的代码复制进去：

```
"""
test python environment
"""

import airsims

# connect to the AirSim simulator
client = airsims.MultirotorClient()
client.confirmConnection()

# get control
client.enableApiControl(True)

# unlock
client.armDisarm(True)

# Async methods returns Future. Call join() to wait for task
to complete.
client.takeoffAsync().join()
client.landAsync().join()

# Lock
client.armDisarm(False)

# release control
client.enableApiControl(False)
```

六、airsim 使用的基本介绍

Unreal 引擎中的坐标系与 AirSim 定义的坐标系是不同的，甚至长度单位都不同。Unreal 的长度单位是厘米，而 AirSim 的长度单位是米。不过不用担心，AirSim 已经非常好的处理了这个问题，你不用管 Unreal 的坐标系是什么，只需要按照 AirSim 的坐标系设置即可，AirSim 会帮你自动转换的。

本文先说明两个坐标系的定义：全局坐标系、机体坐标系。

全局坐标系是固连到大地的，x, y, z 三个坐标轴的指向分别是北，东，地，也就是朝北是 x 轴的正方向，朝南就是 x 轴的负方向。全局坐标系的原点位置是大地的某一点（可以在 settings 文件中设置）。

机体坐标系是固连到四旋翼机身的，x, y, z 三个坐标轴的指向分别是前，右，下，也就是飞机的前方是 x 轴的正方向，飞机后方是 x 轴的负方向。机体坐标系的原点位置是机体的重心位置。因此向上飞 z 为负数。

七、airsim 的 api

`client.moveToZAsync(-3, 1).join()` # 高度控制
`moveToZAsync(z, velocity)` 是高度控制 API，第一个参数是高度，第二个参数是速度。实现的效果是以 1m/s 的速度飞到 3 米高。`.join()` 后缀的意思是程序在这里等待直到任务完成，也就是四旋翼达到 3 米的高度。如果不加 `.join()` 后缀，则不用等待任务是否完成，函数直接返回，程序继续往下执行。

```
def moveToPositionAsync(  
    self,  
    x,          # 位置坐标（北东地坐标系）  
    y,  
    z,  
    velocity,   # 速度  
    timeout_sec=3e38,  
    drivetrain=DrivetrainType.MaxDegreeOfFreedom,  
    yaw_mode=YawMode(),  
    lookahead=-1,  
    adaptive_lookahead=1,  
    vehicle_name="",  
)
```

输入参数包括：

x, y, z: 位置坐标（全局坐标系 - 北东地）

velocity: 飞行速度 (m/s)

timeout_sec: 如果没有响应, 超时时间
drivetrain, yaw_mode: 设置飞行朝向模式和 yaw 角控制模式
lookahead, adaptive_lookahead: 设置路径飞行时候的 yaw 角控制模式
vehicle_name: 控制的四旋翼名字

x, y, z, velocity 这四个参数是必须要设置的量, 指示四旋翼以多大的速度飞往哪个坐标点。后面的几个参数都有其默认值, 不用设置也可以。

lookahead 和 adaptive_lookahead 这两个参数是设置当四旋翼飞轨迹时候的朝向, 目前还用不到。

vehicle_name 是将指令发送给哪个四旋翼, 当做多个四旋翼协同飞行控制的时候, 这个参数就派上用场了。

drivetrain 和 yaw_mode 这两个参数的组合可以设置四旋翼的偏航角控制模式, 下面详细介绍。

drivetrain 参数可以设置为两个量:

airsim.DrivetrainType.ForwardOnly: 始终朝向速度方向

airsim.DrivetrainType.MaxDegreeOfFreedom: 手动设置 yaw 角度

yaw_mode 必须设置为 YawMode() 类型的变量, 这个结构体类型包含两个属性:

YawMode().is_rate: True - 设置角速度; False - 设置角度

YawMode().yaw_or_rate: 可以是任意浮点数

下面分几种情况讨论:

情况 1 (不允许):

```
drivetrain = airsims.DrivetrainType.ForwardOnly
yaw_mode = airsims.YawMode(True, 0)
client.moveToPositionAsync(x, y, z, velocity, drivetrain=drivetrain,
yaw_mode=yaw_mode).join()
```

当 drivetrain = airsims.DrivetrainType.ForwardOnly 时, 四旋翼始终朝向其飞行的方向, 这时 yaw_mode 不允许设置为 YawMode().is_rate = True。因为前面的参数要求四旋翼朝向运动方向, 而 yaw_mode 要求四旋翼以一定的角速度旋转, 这是矛盾的。

情况 2:

```
drivetrain = airsims.DrivetrainType.ForwardOnly
yaw_mode = airsims.YawMode(False, 90)
client.moveToPositionAsync(x, y, z, velocity, drivetrain=drivetrain,
yaw_mode=yaw_mode).join()
```

这种情况下, 四旋翼的朝向始终与前进方向相差 90 度, 也就是四旋翼始终向左侧方向运动。例如: 当四旋翼在绕着一个圆心转圈时, 其朝向始终指向圆心,

这里的 90 度可以任意设置。

情况 3:

```
drivetrain = airsims.DrivetrainType.MaxDegreeOfFreedom
yaw_mode = airsims.YawMode(False, 0)
client.moveToPositionAsync(x, y, z, velocity, drivetrain=drivetrain,
yaw_mode=yaw_mode).join()
```

这种情况下，不管速度方向是什么，四旋翼的 yaw 角始终等于 0，也就是其朝向始终指向正北方向。如果是 90 度，则始终指向正东方向，而-90 度，则始终指向正西方向。

情况 4:

```
drivetrain = airsims.DrivetrainType.MaxDegreeOfFreedom
yaw_mode = airsims.YawMode(True, 10)
client.moveToPositionAsync(x, y, z, velocity, drivetrain=drivetrain,
yaw_mode=yaw_mode).join()
```

这种情况下，四旋翼不管速度方向是什么，yaw 角以 10 度/秒的速度旋转。

下面总结一下这两个参数的设置对效果的影响：

	ForwardOnly	MaxDegreeOfFreedom
is_rate=True	不允许	yaw 角以 yaw_or_rate 度/秒旋转
is_rate=False	yaw 角相对于速度方向偏差 yaw_or_rate 度	yaw 角相对正北方向偏差 yaw_or_rate 度

无人机起飞降落

起飞

```
client.takeoffAsync(timeout_sec = 20, vehicle_name = '')
```

降落

```
client.landAsync( timeout_sec = 60, vehicle_name = '')
```

回到起始位置

```
client.goHomeAsync(timeout_sec = 3e+38, vehicle_name = '')
```

悬停

```
client.hoverAsync(vehicle_name = '')
```

八、获取无人机状态

1、获取估计状态

这个状态是由传感器估计的状态，并不是无人机状态的真值。

AirSim 默认的无人机底层飞控 `simple_flight` 并不支持状态估计，所以如果是 `simple_flight` 飞控，此函数得到的状态与真值相同。

使用 PX4 飞控可以获取估计的状态

```
state = client.getMultirotorState(vehicle_name = '')
```

其中无人机的状态变量 `state` 包含如下：

```
class MultirotorState(MsgpackMixin):
    collision = CollisionInfo()                # 碰撞信息
    kinematics_estimated = KinematicsState()   # 状态信息
    gps_location = GeoPoint()                  # GPS 信息
    timestamp = np.uint64(0)                   # 时间戳
    landed_state = LandedState.Landed          # 是否是降落状态
    rc_data = RCData()                         # 遥控器数据
    ready = False
    ready_message = ""
    can_arm = False
```

碰撞信息的定义先列在这里，后面再具体讲。

```
class CollisionInfo(MsgpackMixin):
    has_collided = False
    normal = Vector3r()
    impact_point = Vector3r()
    position = Vector3r()
    penetration_depth = 0.0
    time_stamp = 0.0
    object_name = ""
    object_id = -1
```

状态信息的定义：

```
class KinematicsState(MsgpackMixin):
    position = Vector3r()                # 位置
    orientation = Quaternionr()          # 姿态角
    linear_velocity = Vector3r()         # 速度
    angular_velocity = Vector3r()        # 机体角速率
    linear_acceleration = Vector3r()     # 加速度
    angular_acceleration = Vector3r()    # 机体角加速度
```

GPS 信息包含：

```
class GeoPoint(MsgpackMixin):
    latitude = 0.0
    longitude = 0.0
    altitude = 0.0
```

2、获取状态真值

```
kinematics_state = client.simGetGroundTruthKinematics(vehicle_name =  
'')
```

九、无人机控制

1、位置控制

x, y, z 是全局坐标系下的三维位置坐标指令

```
client.moveToPositionAsync(x, y, z, velocity, timeout_sec = 3e+38,  
drivetrain = DrivetrainType.MaxDegreeOfFreedom, yaw_mode =  
YawMode(), lookahead = -1, adaptive_lookahead = 1, vehicle_name = '')
```

2、速度控制

vx, vy, vz 是全局坐标系(NED)下的速度指令

```
client.moveByVelocityAsync(vx, vy, vz, duration, drivetrain =  
DrivetrainType.MaxDegreeOfFreedom, yaw_mode = YawMode(), vehicle_name  
= '')
```

3、水平速度控制（指定高度）

vx, vy 是全局坐标系(NED)下的速度指令，z 是全局高度指令

```
client.moveByVelocityZAsync(vx, vy, z, duration, drivetrain =  
DrivetrainType.MaxDegreeOfFreedom, yaw_mode = YawMode(), vehicle_name  
= '')
```

4、水平姿态角控制（指定高度）

pitch, roll 是水平姿态角指令，z 是全局高度指令，yaw 是偏航角指令

```
client.moveByAngleZAsync(pitch, roll, z, yaw, duration, vehicle_name  
= '')
```

5、水平姿态角控制（高度油门控制）

pitch, roll 是水平姿态角指令，throttle 是油门量指令，yaw_rate 是偏航角速率指令

```
moveByAngleThrottleAsync(pitch, roll, throttle, yaw_rate, duration,
```

```
vehicle_name = '')
```

6、高度控制

z 是全局高度指令，velocity 是垂直速度

```
client.moveToZAsync(z, velocity, timeout_sec = 3e+38, yaw_mode =  
YawMode(), lookahead = -1, adaptive_lookahead = 1, vehicle_name = '')
```

7、偏航角控制

```
client.rotateToYawAsync(yaw, timeout_sec = 3e+38, margin = 5,  
vehicle_name = '')
```

8、偏航角速率控制

```
client.rotateByYawRateAsync(yaw_rate, duration, vehicle_name = '')
```

9、沿路径飞行

```
client.moveOnPathAsync(path, velocity, timeout_sec = 3e+38,  
drivetrain = DrivetrainType.MaxDegreeOfFreedom, yaw_mode =  
YawMode(), lookahead = -1, adaptive_lookahead = 1, vehicle_name = '')
```

10、遥控器控制

由遥控器信号控制，vx_max, vy_max 是速度限制，z_min 是最低高度限制。

```
client.moveByManualAsync(vx_max, vy_max, z_min, duration, drivetrain  
= DrivetrainType.MaxDegreeOfFreedom, yaw_mode = YawMode(),  
vehicle_name = '')
```

11、遥控器控制（程序定义遥控器信号量）

```
client.moveByRC(rcdata = RCData(), vehicle_name = '')
```

rcdata 包括：

timestamp

pitch

roll

throttle

yaw

switch1

switch2

switch3

switch4

```
switch5  
switch6  
switch7  
switch8  
is_initialized  
is_valid
```

十、无人机底层飞控

1、直接控制四个电机

```
client.moveByMotorPWMsAsync(front_right_pwm, rear_left_pwm,  
front_left_pwm, rear_right_pwm, duration, vehicle_name = '')
```

2、姿态角控制、指定高度控制

roll, pitch, yaw 姿态角指令, z 高度指令

```
client.moveByRollPitchYawZAsync(roll, pitch, yaw, z, duration,  
vehicle_name = '')
```

3、姿态角控制、高度油门控制

```
client.moveByRollPitchYawThrottleAsync(roll, pitch, yaw, throttle,  
duration, vehicle_name = '')
```

4、俯仰翻滚姿态角、偏航角速率、高度油门控制

```
client.moveByRollPitchYawrateThrottleAsync(roll, pitch, yaw_rate,  
throttle, duration, vehicle_name = '')
```

5、俯仰翻滚姿态角、偏航角速率、指定高度控制

```
client.moveByRollPitchYawrateZAsync(roll, pitch, yaw_rate, z,  
duration, vehicle_name = '')
```

6、姿态角速率、指定高度

```
client.moveByAngleRatesZAsync(roll_rate, pitch_rate, yaw_rate, z,  
duration, vehicle_name = '')
```

7、姿态角速率、高度油门控制

```
client.moveByAngleRatesThrottleAsync(roll_rate, pitch_rate,  
yaw_rate, throttle, duration, vehicle_name = '')
```

8、更改角速度环控制器参数

```
client.setAngleRateControllerGains( angle_rate_gains=AngleRateControllerGains(), vehicle_name = '')
```

9、更改角度环控制器参数

```
client.setAngleLevelControllerGains(angle_level_gains=AngleLevelControllerGains(), vehicle_name = '')
```

10、更改速度环控制器参数

```
client.setVelocityControllerGains(velocity_gains=VelocityControllerGains(), vehicle_name = '')
```

11、更改位置环控制器参数

```
client.setPositionControllerGains(position_gains=PositionControllerGains(), vehicle_name = '')
```

十一、关于 Async 的解释和使用

这里面的很多控制函数都有 `dutation` 或者 `timeout_sec`，而且函数有 `Async` 后缀。这些方法的调用会立即返回，即使其所指定的任务还没有完成，这样程序可以继续往下执行，而不用在这里等待仿真中的无人机把任务执行完。如果想要程序在这里等待任务执行完，则需要在方法后面加上 `.join()`。例如，让四旋翼起飞，如果想要程序一直在这里等待四旋翼起飞完成，然后再执行后面的语句，则可以用 `client.takeoffAsync().join()`

十二、关于任务覆盖

在代码执行时，如果使用的都是立即返回，那么很有可能在仿真中，无人机的上一个任务还没有执行完，新的任务就又到来了，此时旧的任务会被取消，新的任务会覆盖掉旧的任务，无人机会立即执行最新的任务。

十三、关于仿真环境使用

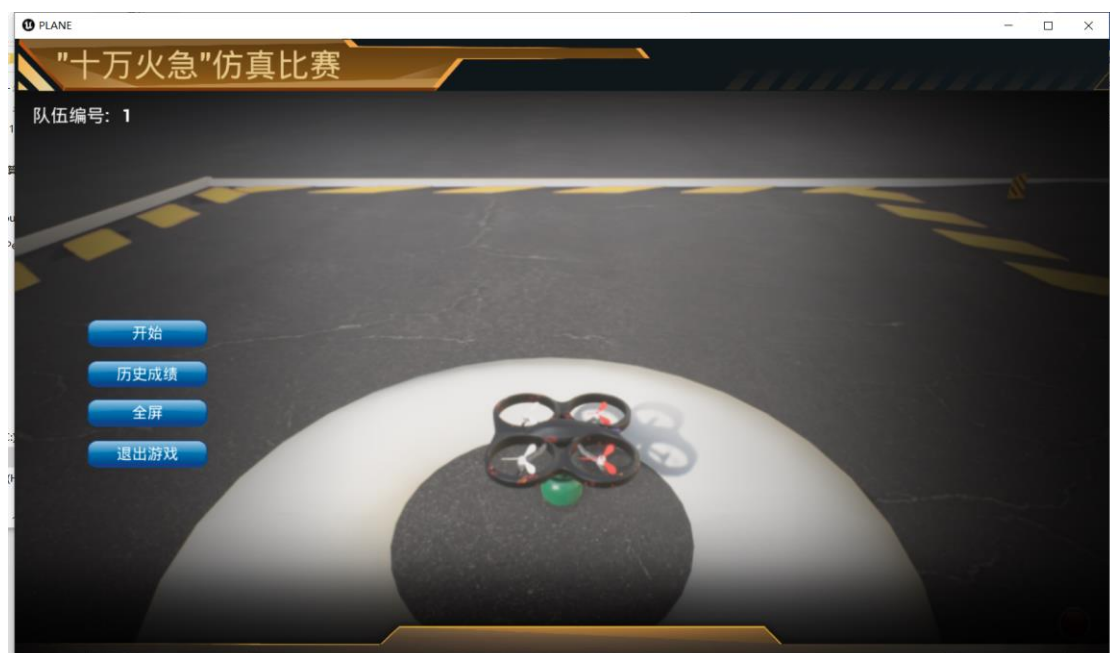
下载附件中的 `PlaneV4.zip` 并解压，运行解压后文件中的

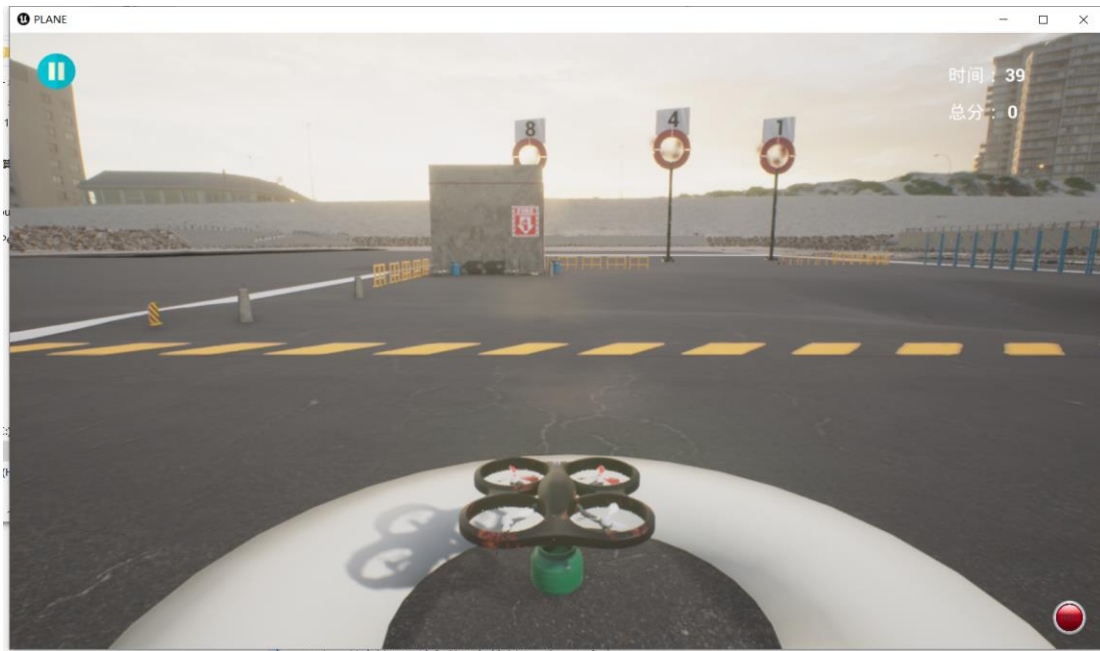
PLANE.exe 文件，然后关闭文件。

名称	修改日期	类型	大小
Engine	2023/3/5 14:18	文件夹	
PLANE	2023/3/5 14:18	文件夹	
Manifest_NonUFSFiles_Win64.txt	2023/3/5 14:18	文本文档	3 KB
PLANE.exe	2023/3/5 14:15	应用程序	142 KB

将“/此电脑/文档/AirSim”中的 settings.json 文件替换为附件中提供的 settings.json 文件。（注：也可自己调整无人机的偏航角，但是无人机初始位置不能更改）

仿真环境启动后界面如下图所示：





附录：AirSim 使用教程参考文档链接：

<https://www.zhihu.com/column/multiUAV>

该文档由飞鹰队视觉部刘梓越、王诗绮、谭佳明所作