AirSim +UnrealEngine安装和使用教程（基于ubuntu22.04）

说明：在ubunut22.04系统上安装AirSim和UnrealEditor, 使用UnrealEditor创建自定义场景，然后在py文件中调用airsim API， 实现无人机在自定义场景中实现算法。

一 安装UnrealEngine并搭建场景

参考：https://microsoft.github.io/AirSim/build_linux/

1 安装UnrealEngine

- Make sure you are registered with Epic Games. This is required to get source code access for Unreal Engine.
- Clone Unreal in your favorite folder and build it (this may take a while!). **Note**: We only support Unreal >= 4.27 at present. We recommend using 4.27.

```
# go to the folder where you clone GitHub projects
git clone -b 4.27 git@github.com:EpicGames/UnrealEngine.git
cd UnrealEngine
./Setup.sh
./GenerateProjectFiles.sh
make
```

2 搭建场景

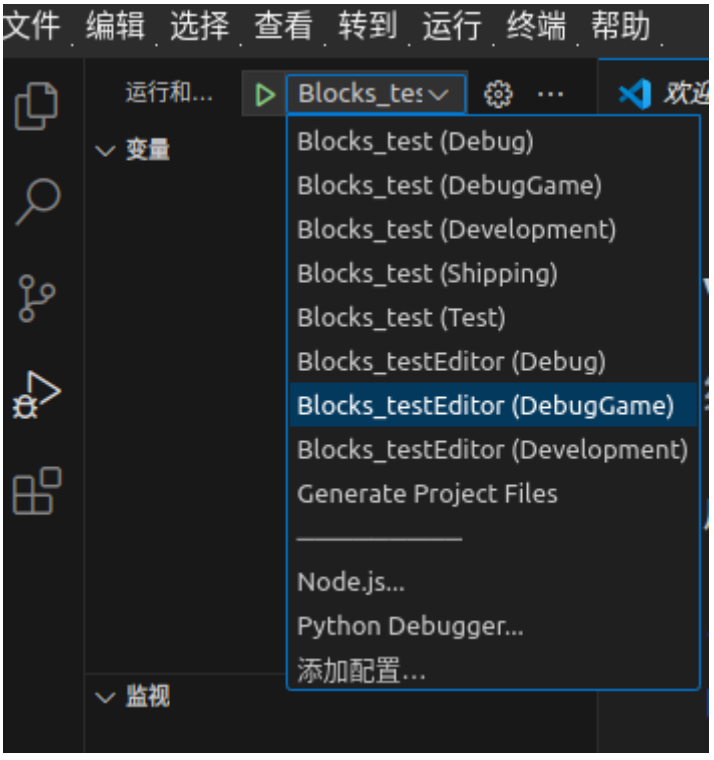
创建空白蓝图，注意项目的保存路径为：/home/usrname/Documents/UnrealProjects, 否则重新编译时可能出错，注意也不能命名为test,后面编译的时候会有冲突。然后新建C++类（不需要设置），将编译后的AirSim中的/Unreal/Plugins文件夹拷贝到项目目录下，注意一定要是编译后，很重要。不需要下载AirSim，因为就算下载了AirSim，它在ubunut22.04中编译会出各种各样的依赖问题，因为我们需要的知识Plugins/AirSim这个文件，所以只需要有这个文件就好，可以在github上找一下编译后的Plugins/AirSim文件，我是没有找到。所有我使用了Docker，拉取ubuntu18.04的镜像，创建容器并在容器内下载AirSim并完成编译，然后把编译后的Plugins/AirSim拷贝到宿主机上。

修改项目中的projectname.uproject文件，注意除了"FileVersion"，"EngineAssociation"保持和源文件一样，修改“Modules”中的“Name”项为项目名称，其他如下：

```
{
    "FileVersion": 3,
    "EngineAssociation": "{0004001E-08DD-3E3D-6203-B4B6FD6934D8}",
    "Category": "",
    "Description": "",
    "Modules": [
        {
            "Name": "Blocks_test",
            "Type": "Runtime",
            "LoadingPhase": "Default",
            "AdditionalDependencies": [
                "AirSim"
            ]
        }
```

```
    ],
    "Plugins": [
        {
            "Name": "AirSim",
            "Enabled": true
        },
        {
            "Name": "SteamVR",
            "Enabled": false
        },
        {
            "Name": "OculusVR",
            "Enabled": false
        }
    ]
}
```

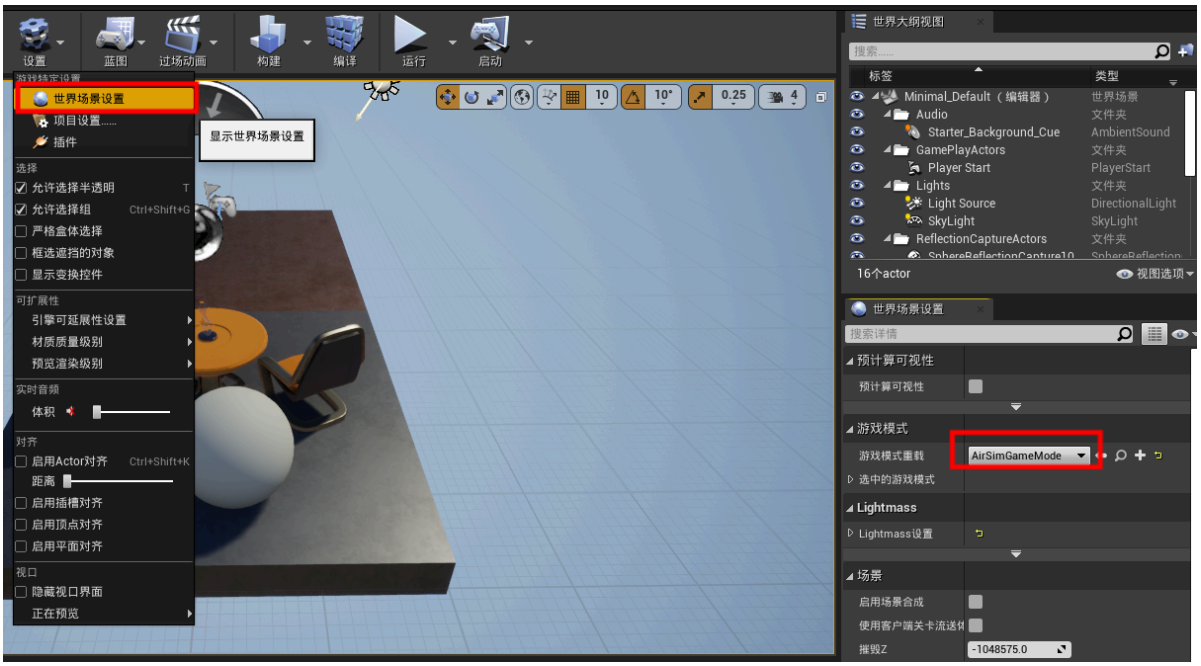
使用Vscode编译，在VScode中打开项目文件夹，选择projectnameEditor(Debug Game)，编译后会自动开启UE4Editor。



解决编译后打开项目点击projectname.uproject后，被要求再次编译的问题。

找到Engine/Source/Developer/DesktopPlatform/Private/DesktopPlatformBase.cpp文件，然后修改Arguments += " -Progress -NoEngineChanges -NoHotReloadFromIDE"为Arguments += " -Progress"，再次编译项目文件，然后再次点击.projectname.uproject, 点击rebuild,就可以了。以后再次打开时就不会被要求编译了。

然后将游戏模式重载设置为AirSimGameMode，到此所创建的环境中就包含了AirSim插件了。

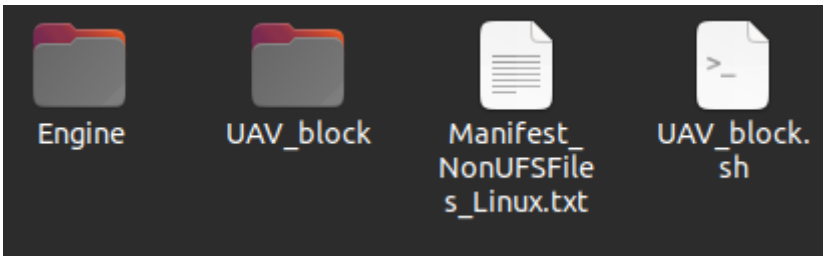


3 打包项目(可选)

将项目打包成可执行文件，在可执行文件中画面更流畅。

打包成Linux项目

打包成功后的文件包括，UAV_block.sh为启动脚本，可执行程序是/UAV_block/UAV_block/Binaries/Linux/UAV_block



二 测试

1 在宿conda环境中安装airsim包

```
pip install msgpack-rpc-python
```

```
pip install airsim
```

2 可以修改 /Documents/AirSim/文件夹下的setting.json文件来配置UE的启动，具体可以配置那些参数请移步到G 老师

3 测试，使用Hello Dorne的py文件测试AirSim启动及控制是否正常。

启动UE项目，可以以Editor模式（UnrealEngineEditor）或以Game模型（打包项目文件中的可执行文件）启动，然后运行hello_drone.py

```
# ready to run example: PythonClient/multirotor/hello_drone.py
import airsim
import os

# connect to the AirSim simulator
client = airsim.MultirotorClient()
client.confirmConnection()
client.enableApiControl(True)
client.armDisarm(True)

# Async methods returns Future. Call join() to wait for task to complete.
client.takeoffAsync().join()
client.moveToPositionAsync(-10, 10, -10, 5).join()

# take images
responses = client.simGetImages([
    airsim.ImageRequest("0", airsim.ImageType.DepthVis),
    airsim.ImageRequest("1", airsim.ImageType.DepthPlanar, True)])
print('Retrieved images: %d', len(responses))

# do something with the images
for response in responses:
    if response.pixels_as_float:
        print("Type %d, size %d" % (response.image_type,
len(response.image_data_float)))
        airsim.write_pfm(os.path.normpath('/temp/py1.pfm'),
airsim.get_pfm_array(response))
    else:
```

```
        print("Type %d, size %d" % (response.image_type,
len(response.image_data_uint8)))
        airsim.write_file(os.path.normpath('/temp/py1.png'),
response.image_data_uint8)
```

测试键盘控制无人机运动，修改/Documents/AirSim/setting.json文件如下：

```
{
  "SeeDocsAt":
"https://github.com/Microsoft/AirSim/blob/main/docs/settings_json.md",
  "SettingsVersion": 1.2,
  "SimMode": "Multirotor",
  "ClockSpeed": 1.0,
  "Vehicles": {
    "Multirotor1": {
      "VehicleType": "SimpleFlight",
      "DefaultVehicleState": "Armed",
      "X": 0,
      "Y": 0,
      "Z": 0,
      "EnableCollisionPassthrogh": false,
      "EnableCollisions": true,
      "AllowAPIAlways": true,
      "RC": {
        "RemoteControlID": 0,
        "AllowAPIWhenDisconnected": false
      }
    },
   "Multirotor2": {
      "VehicleType": "SimpleFlight",
      "DefaultVehicleState": "Armed",
      "X": 2,
      "Y": 0,
      "Z": 0,
      "EnableCollisionPassthrogh": false,
      "EnableCollisions": true,
      "AllowAPIAlways": true,
      "RC": {
        "RemoteControlID": 1,
        "AllowAPIWhenDisconnected": false
      }
    },
    "Multirotor3": {
      "VehicleType": "SimpleFlight",
      "DefaultVehicleState": "Armed",
      "X": 4,
      "Y": 0,
      "Z": 0,
      "EnableCollisionPassthrogh": false,
      "EnableCollisions": true,
      "AllowAPIAlways": true,
      "RC": {
        "RemoteControlID": 2,
        "AllowAPIWhenDisconnected": false
      }
```

```
    },
    "Multirotor4": {
      "VehicleType": "SimpleFlight",
      "DefaultVehicleState": "Armed",
      "X": 6,
      "Y": 0,
      "Z": 0,
      "EnableCollisionPassthrogh": false,
      "EnableCollisions": true,
      "AllowAPIAlways": true,
      "RC": {
        "RemoteControlID": 3,
        "AllowAPIWhenDisconnected": false
      }
    }
  }
}
```

修改完setting.json文件后需要重新启动游戏运行，配置后的setting.json同样对打包后的项目生效。键盘控制无人机的程序如下：

```
import sys
import time
import airsim
import pygame
import cv2
import numpy as np

# pygame settings #
pygame.init()
screen = pygame.display.set_mode((320, 240))
pygame.display.set_caption('keyboard ctrl')
screen.fill((0, 0, 0))

#   AirSim settings  #
# 这里改为你要控制的无人机名称(settings文件里面设置的)

AirSim_client = airsim.MultirotorClient()
AirSim_client.confirmConnection()
AirSim_client.enableApiControl(True, 'Multirotor1')
# AirSim_client.enableApiControl(True, 'Multirotor2')
# AirSim_client.enableApiControl(True, 'Multirotor3')
# AirSim_client.enableApiControl(True, 'Multirotor4')
AirSim_client.armDisarm(True, 'Multirotor1')
# AirSim_client.armDisarm(True, 'Multirotor2')
# AirSim_client.armDisarm(True, 'Multirotor3')
# AirSim_client.armDisarm(True, 'Multirotor4')
AirSim_client.takeoffAsync(vehicle_name='Multirotor1').join()
# AirSim_client.takeoffAsync(vehicle_name='Multirotor2').join()
# AirSim_client.takeoffAsync(vehicle_name='Multirotor3').join()
# AirSim_client.takeoffAsync(vehicle_name='Multirotor4').join()

# 基础的控制速度(m/s)
vehicle_velocity = 2.0
# 设置临时加速比例
```

```
speedup_ratio = 10.0
# 用来设置临时加速
speedup_flag = False

# 基础的偏航速率
vehicle_yaw_rate = 5.0

while True:
    yaw_rate = 0.0
    velocity_x = 0.0
    velocity_y = 0.0
    velocity_z = 0.0

    time.sleep(0.02)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    scan_wrapper = pygame.key.get_pressed()

    # 按下空格键加速10倍
    if scan_wrapper[pygame.K_SPACE]:
        scale_ratio = speedup_ratio
    else:
        scale_ratio = speedup_ratio / speedup_ratio

    # 根据 'A'; 和 'D' 按键来设置偏航速率变量
    if scan_wrapper[pygame.K_a] or scan_wrapper[pygame.K_d]:
        yaw_rate = (scan_wrapper[pygame.K_d] - scan_wrapper[pygame.K_a]) *
scale_ratio * vehicle_yaw_rate

    # 根据 'UP'; 和 'DOWN'; 按键来设置pitch轴速度变量(NED坐标系, x为机头向前)
    if scan_wrapper[pygame.K_UP] or scan_wrapper[pygame.K_DOWN]:
        velocity_x = (scan_wrapper[pygame.K_UP] - scan_wrapper[pygame.K_DOWN]) *
scale_ratio

    # 根据 'LEFT' 和 'RIGHT' 按键来设置roll轴速度变量(NED坐标系, y为正右方)
    if scan_wrapper[pygame.K_LEFT] or scan_wrapper[pygame.K_RIGHT]:
        velocity_y = -(scan_wrapper[pygame.K_LEFT] -
scan_wrapper[pygame.K_RIGHT]) * scale_ratio

    # 根据 'W' 和 'S' 按键来设置z轴速度变量(NED坐标系, z轴向上为负)
    if scan_wrapper[pygame.K_w] or scan_wrapper[pygame.K_s]:
        velocity_z = -(scan_wrapper[pygame.K_w] - scan_wrapper[pygame.K_s]) *
scale_ratio

    print(f"Expectation gesture: {velocity_x}, {velocity_y}, {velocity_z},
{yaw_rate}")

    # 设置速度控制以及设置偏航控制
    AirSim_client.moveByVelocityBodyFrameAsync(vx=velocity_x, vy=velocity_y,
vz=velocity_z, duration=0.02,
                                               yaw_mode=airsim.YawMode(True,
yaw_or_rate=yaw_rate),vehicle_name='Multirotor1')
```

```
    # AirSim_client.moveByVelocityBodyFrameAsync(vx=velocity_x, vy=velocity_y,
vz=velocity_z, duration=0.02,
    #                                         yaw_mode=airsim.YawMode(True,
yaw_or_rate=yaw_rate),vehicle_name='Multirotor2')
    # AirSim_client.moveByVelocityBodyFrameAsync(vx=velocity_x, vy=velocity_y,
vz=velocity_z, duration=0.02,
    #                                           yaw_mode=airsim.YawMode(True,
yaw_or_rate=yaw_rate),vehicle_name='Multirotor3')
    # AirSim_client.moveByVelocityBodyFrameAsync(vx=velocity_x, vy=velocity_y,
vz=velocity_z, duration=0.02,
    #                                           yaw_mode=airsim.YawMode(True,
yaw_or_rate=yaw_rate),vehicle_name='Multirotor4')

    Multirotor1_state =
AirSim_client.getMultirotorState(vehicle_name='Multirotor1')
    print(f"Drone1 state: {Multirotor1_state}")

    if scan_wrapper[pygame.K_ESCAPE]:
        AirSim_client.reset()
        pygame.quit()
        sys.exit()
```