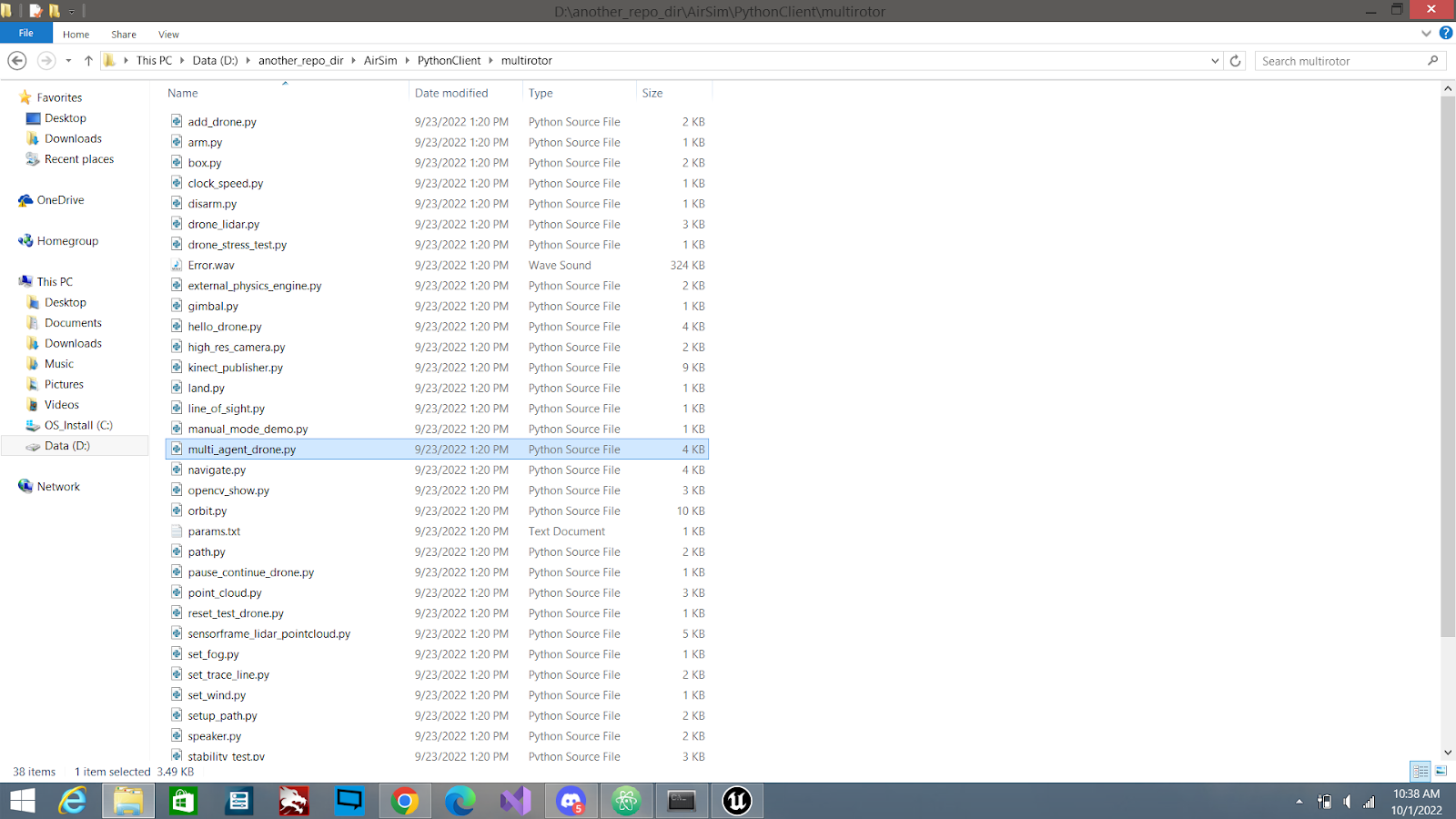
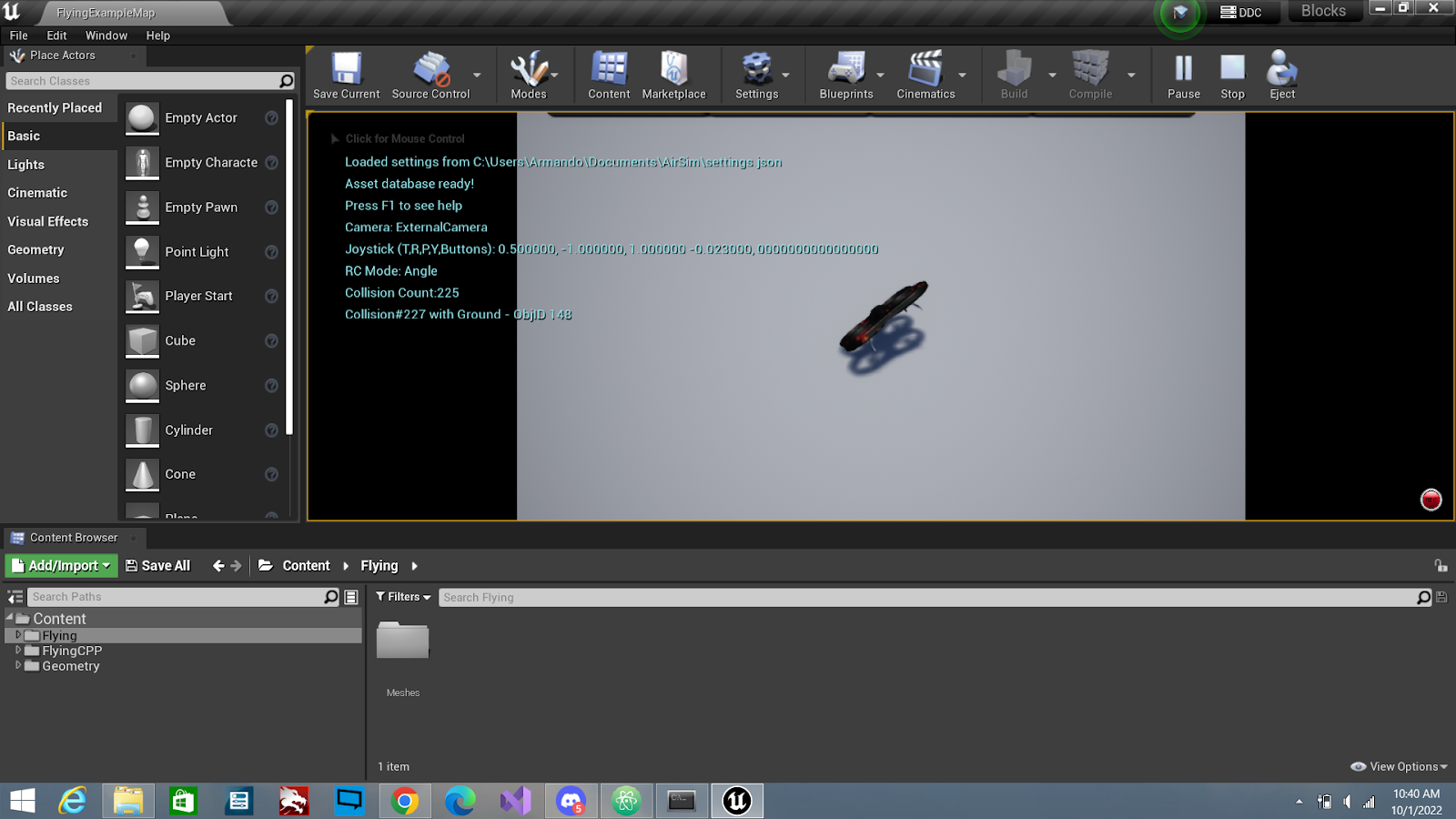
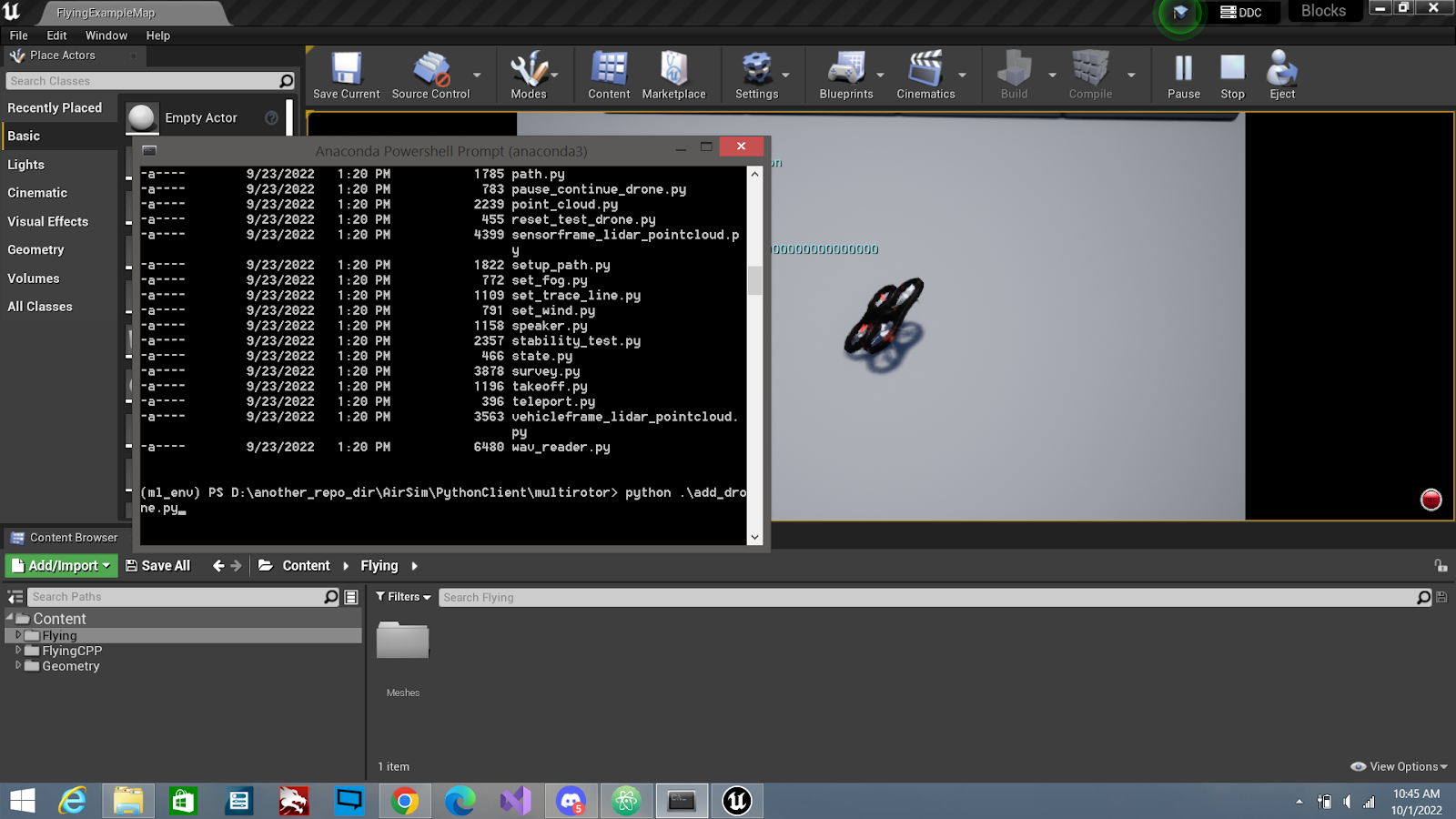
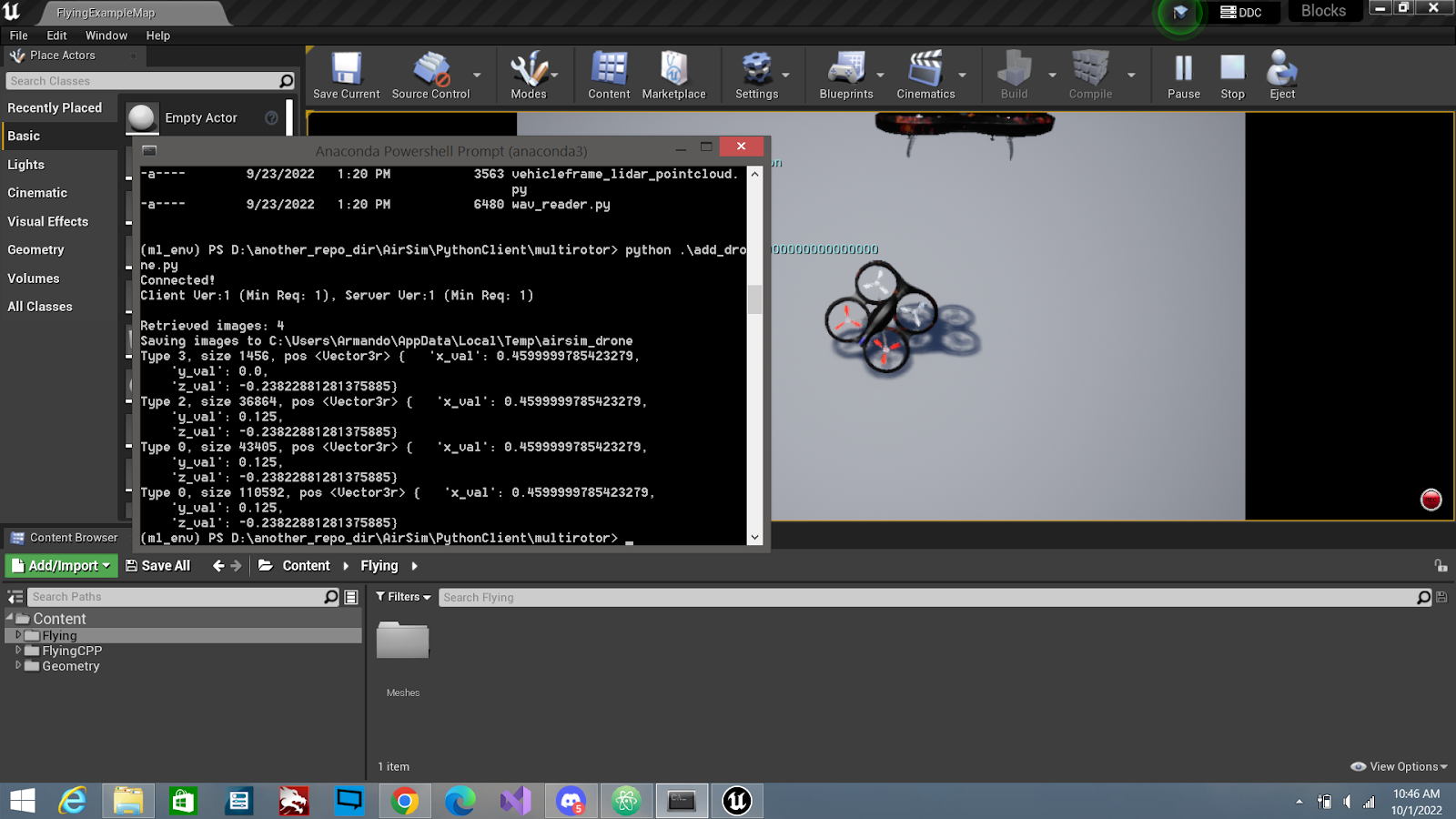
* Getting Started
  + Anaconda Set Up
  + Windows Setup
    - Step 1: Go <https://www.anaconda.com/products/distribution> and install the anaconda navigator
    - Step 2: Create a Conda Virtual Environment
      * Part 1: Go into your search bar and type anaconda. Several suggestions will pop up but what you want is “Anaconda Powershell Prompt”. Go ahead and type: **conda create --name ml\_env python=3.9**
      * Part 2: Next we need to enter the environment:
        + To list all the environments on your machine type: **conda env list** (you should see ml\_env show up)
        + To enter your environment so you can open up the version of python we have on there and other libraries we’ll install, type: **conda activate ml\_env**
        + To leave the environment and return to the base environment just type: **conda deactivate**
        + Note: Do not make a bunch of installations in your base environment as that can allow you to have old or interfering versions of packages all over your computer. It is best to isolate your mistakes by making a virtual environment like we just did.
      * Part 3: Installing machine learning packages in the environment:
        + Type: **pip install numpy**
        + Type: **pip install scipy**
        + Type: **conda install pytorch torchvision cudatoolkit=10.0 -c pytorch** (this will install the gpu version of pytorch, for cpu version just take away the “-c pytorch” at the end.
        + Type: Finally, to install opencv for image processing just type: **pip install opencv-python**
        + To check that all packages and dependencies have been installed in your environment type: **pip list** . This will show you a list of all the packages we installed including the additional ones installed by the build wheel for pytorch.
        + That should be all we need for machine learning at the moment.
* Microsoft AirSim Set Up
  + Link for setting up github on Windows: <https://www.slashadmin.co.uk/basic-setup-guide-for-using-github-with-visual-code-and-powershell/>
  + Link for setting up AirSim on Windows (possibly same steps for mac): <https://microsoft.github.io/AirSim/build_windows/>
  + C++ Install on Windows: <https://code.visualstudio.com/docs/languages/cpp>
* Interfacing with Python API
  + Open up Anaconda powershell and navigate to the repo, if your repo is on the D drive then type “D:” in the anaconda prompt to change to your D drive.
  + Also open up your favorite code editor (could be anything) and use that to modify code and write code for interacting with the API.
  + Go ahead and activate the conda environment from the previous instructions.
  + Now go ahead and install this major dependency:
    - **pip install msgpack-rpc-python**
  + Next we need the airsim python package.The airsim package for python is necessary for interacting with the APIs, from your anaconda prompt do
    - **pip install airsim (if you get an error it’s because you didn’t do the previous step!)**
  + Now as a sanity check type **python**, hit enter, type **import airsim**, hit enter, and if you get no error then you have successfully installed the package. Now you’re ready to start interacting with the API in python!
* **Python API Example:**
  + To use the API while the simulator is running live first navigate to: **D:\...\AirSim\PythonClient** on your computer.
  + Notice, in the PythonClient folder there are several folders titled computer\_vision and multirotor (that’s the drone we are working with). Each of these folders contain scripts scripts that can be ran from the anaconda powershell by navigating to them and typing **python scriptName.py**
  + **Make sure before you run any of these scripts you are first running the simulator with the appropriate vehicle!**
  + For example, suppose you wanted to run the multiple drone script:
  + 
  + **First:** Run the Simulator  
    
  + **Since we are running a multi-rotor script make sure you are using a script that was designed for the multirotor client!**
  + **Second:** Navigate to the “add\_drone.py” script and have and run it after you’ve started the simulator in unreal.   
      
    
  + Now you should see two drones!
  + You can also modify the scripts with your favorite editors or even make your own.
  + Here’s a semi descent collection script I made:

**import airsim**

**import os**

**import numpy as np**

**import cv2**

**import pprint**

**import time**

**# directory to store pictures**

**imgDir = r'D:\another\_repo\_dir\AirSim\PythonClient\custom\_scripts\collection'**

**# check that directory exists**

**isExist = os.path.exists(imgDir)**

**if not isExist:**

**# make directory if not already there**

**os.makedirs(imgDir)**

**print('Created: ' + imgDir)**

**# set up client object to access multirotor drone**

**client = airsim.MultirotorClient()**

**# connect to AirSim simulator**

**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()**

**# image collection loop**

**while True:**

**# take images**

**responses = client.simGetImages([**

**airsim.ImageRequest("0", airsim.ImageType.DepthVis),**

**airsim.ImageRequest("1", airsim.ImageType.DepthPlanar, True)])**

**print('Retrieved images: %d', len(responses))**

**# grab the current state of collision from the client (aka the drone)**

**collision\_info = client.simGetCollisionInfo()**

**# stop if we encountered a collision**

**if collision\_info.has\_collided:**

**print("Collision at pos %s, normal %s, impact pt %s, penetration %f, name %s, obj id %d" % (**

**pprint.pformat(collision\_info.position),**

**pprint.pformat(collision\_info.normal),**

**pprint.pformat(collision\_info.impact\_point),**

**collision\_info.penetration\_depth, collision\_info.object\_name, collision\_info.object\_id))**

**break**

**time.sleep(0.1)**

**# save images**

**for idx, response in enumerate(responses):**

**filename = os.path.join(imgDir, str(idx))**

**if response.pixels\_as\_float:**

**print("Type %d, size %d" % (response.image\_type, len(response.image\_data\_float)))**

**airsim.write\_pfm(os.path.normpath(filename + '.pfm'), airsim.get\_pfm\_array(response))**

**elif response.compress: #png format**

**print("Type %d, size %d" % (response.image\_type, len(response.image\_data\_uint8)))**

**airsim.write\_file(os.path.normpath(filename + '.png'), response.image\_data\_uint8)**

**else: #uncompressed array**

**print("Type %d, size %d" % (response.image\_type, len(response.image\_data\_uint8)))**

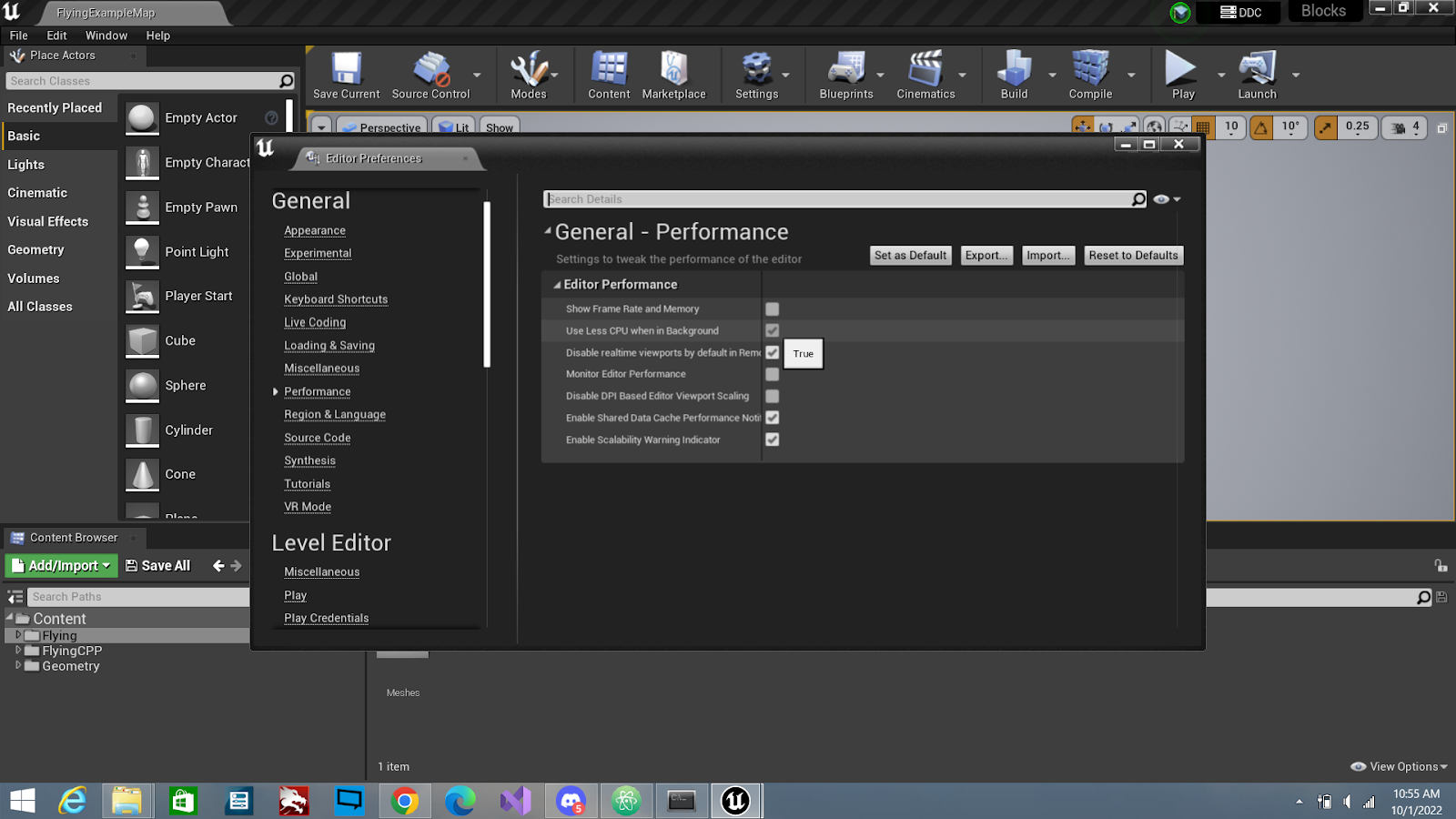
**img1d = np.fromstring(response.image\_data\_uint8, dtype=np.uint8) # get numpy array**

**img\_rgb = img1d.reshape(response.height, response.width, 3) # reshape array to 4 channel image array H X W X 3**

**cv2.imwrite(os.path.normpath(filename + '.png'), img\_rgb) # write to png**

**# end connection**

**client.enableApiControl(False)**

* To write your own scripts just go and test out the example and even start reading the doc from here: <https://microsoft.github.io/AirSim/image_apis/>
* To improve performance by running scripts please uncheck the box highlighted in this image **(Use Less CPU when in Background)**:  
  **Found by going into Edit -> Editor Preferences -> Performance**  
* This will increase the performance by a lot so you can run scripts and play the simulator in real time!