

In this file you will find instruction about

1. How to install Unreal Engine and create a game
2. How to build UnrealCV plugin
3. How to control the camera in Unreal Engine

Before going to the detailed instruction of installing UnrealCV and it's usage assuming that all of you know how to install Unreal Engine and create a game. If you don't know just follow the following steps---

1. To know about Unreal Engine you can start with it's [Documentation](#)
2. [Here](#) you will find a clear picture to install Unreal Engine
3. You can follow [this link](#) if you want to build Unreal Engine from Source code and set up Qt creator in Linux
4. You also need Microsoft Visual Studio to compile Unreal Engine code, [here](#) you will find this direction

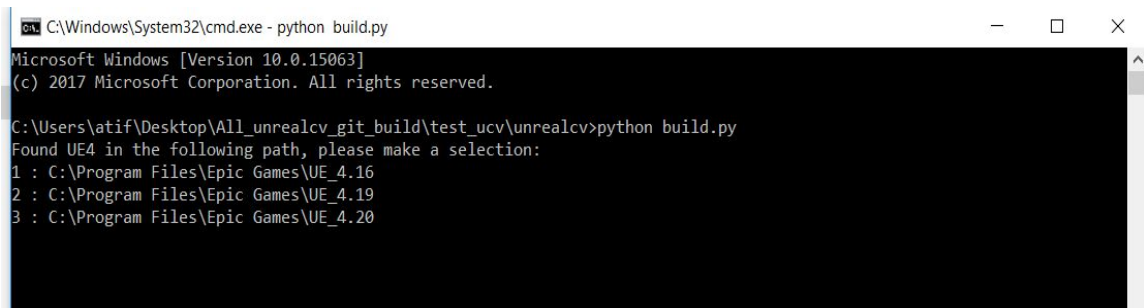
After setting up UE4 and creating the game we can start to build Unreal CV.

To get the info about it follow their [tutorial](#) and [github](#)

Building UnrealCV:

1. Clone the git repo (git clone https://github.com/unrealcv/unrealcv.git)
2. Enter in the folder (cd unrealcv)
3. Change the branch from **master** to **fast** (git checkout fast) as it is the updated branch.
4. Now in command window write **python build.py** to build unrealcv
5. It will ask to press the numeric number corresponding to your installed Unreal Engine version. In my case I have chosen version 4.19 and pressed '2' (omit the inverted comma, just pressed 2 and press 'Enter' button of your keyboard).

Till now no release came for UE4 4.20



```
C:\Windows\System32\cmd.exe - python build.py
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

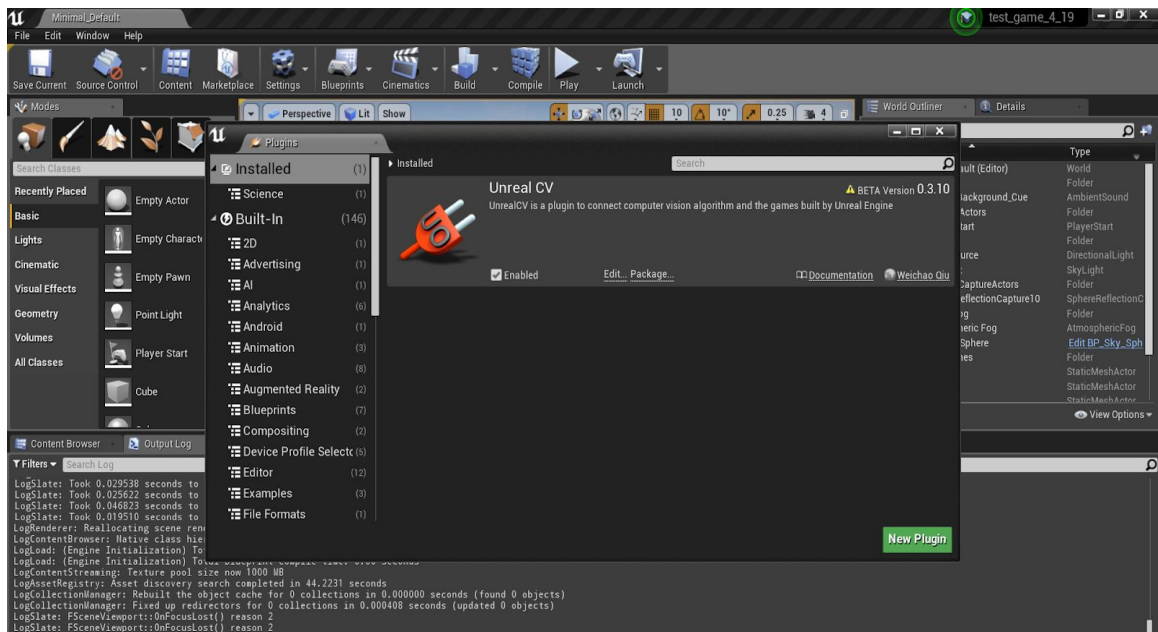
C:\Users\atif\Desktop\All_unrealcv_git_build\test_ucv\unrealcv>python build.py
Found UE4 in the following path, please make a selection:
1 : C:\Program Files\Epic Games\UE_4.16
2 : C:\Program Files\Epic Games\UE_4.19
3 : C:\Program Files\Epic Games\UE_4.20
```

6. After building put the whole folder following [this link](#) .

- After doing it open your solution file of visual studio corresponding to your game. In the following picture this is the red squared marked one.

This PC > Local Disk (F:) > test_game_4_19				
Name	Date modified	Type	Size	
.vs	10/12/2018 23:59	File folder		
Binaries	11/12/2018 00:00	File folder		
Config	10/12/2018 23:59	File folder		
Content	11/12/2018 00:18	File folder		
Intermediate	11/12/2018 18:14	File folder		
Plugins	11/12/2018 00:20	File folder		
Saved	11/12/2018 00:21	File folder		
Source	10/12/2018 23:59	File folder		
test_depth.npy	11/12/2018 01:23	NPY File	1.201 KB	
test_game_4_19.sln	11/12/2018 01:11	Visual Studio Solut...	8 KB	
test_game_4_19.uproject	11/12/2018 01:19	Unreal Engine Proj...	1 KB	
test_lit.png	11/12/2018 01:22	PNG File	375 KB	
test_normal.png	11/12/2018 01:22	PNG File	66 KB	
test_obj.png	11/12/2018 01:20	PNG File	27 KB	

- After opening build the game (on the toolbar you will see **Build** ) and if it compiled successfully **Debug** the game from the toolbar. It will start the game.
- You can now check whether the Plugin has integrated with UE4 or not. Go from the menu bar **Edit>Plugins** then the following view will come.

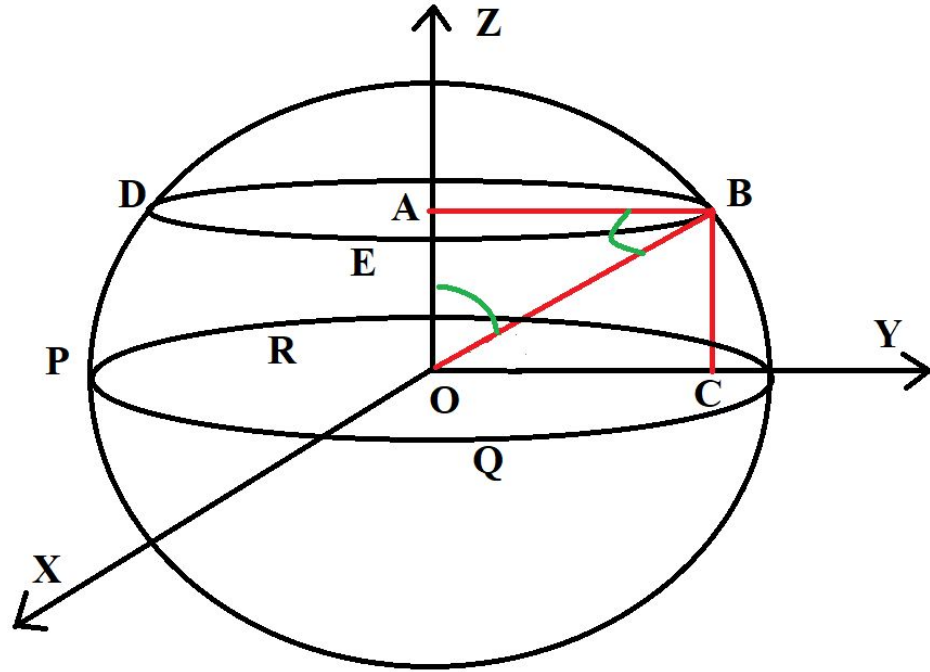


- To know about the basic command of UnrealCV follow [this link](#)

11. It's better if you can perform [this tutorial](#) from their site. It will give you some basic idea of UnrealCV.

### **Spherical movement of camera around the object:**

1. In Unreal CV we can see some commands to set and get the location and rotation of the camera. like **vget /camera/0/location** to get the camera location and **vset /camera/0/location x y z** to set the location of the camera.
2. We can also get the location and rotation information of object using **vget/object/object\_name/location** . To get the name list of the object use **vget /objects** .
3. Here the idea is to circulate the camera on a spherical surface where object is located in the centre of that sphere. At Equator the diameter of the sphere is situated and there you can assume the biggest circle is situated. Actually, the sphere is nothing but a stack of circle from North to South. If you want to travel from North to South then you have to change your **Polar/ Elevation angle** while for travelling from West to East you have to change your **Azimuthal angle**. The lines covers the trajectory from North pole to South is called **Latitude** and from West to East the covering line known as **Longitude**([Follow this](#))
4. If you change the Elevation and Azimuthal Angle then you will find the surface coordinate of that sphere if you know the centre and radius of that surface. To get the full picture please visit [this link](#) . After getting the x,y,z coordinates just pass this to the command by which you can refresh the location of the camera(**vset /camera/0/location X Y Z**).
5. Another point of this task is to always look at the object. To resolve this problem the idea of **Roll, Pitch, Yaw** came. In Unreal Engine viewport, camera is pointing at the object in a manner while you will rotate the camera around the Y axis of the world coordinate system then it will cover the Latitude line with respect to the change of the elevation angle by using different Pitch value.
6. For a specific Elevation angle only a specific Pitch exists. While the Elevation angle will change then the Pitch will also have to change to look at the camera. Regarding the change of Azimuthal angle Yaw will have to be changed to look at the object for every rotation.
7. Here I have tried to demonstrate the idea how Pitch value is calculated in this task.



Suppose, in **X-Y-Z** plane a sphere is situated whose centre is **O** and here is the biggest circle **PQR** is situated on the **XY** plane. Assuming the camera will rotate along the **Z** axis that means the trajectory of the camera will be the circumference of **PQR** circle. Here the elevation angle(angle **ZOC**) is 90 degree and azimuthal angle will vary from 0 degree to 360 degree. If now Elevation angle decrease(angle **ZOB**) and camera goes to the **B** point on the surface of sphere we have to find out the angle **ABO**.

In triangle **AOB**,  $\text{angle}(\mathbf{ABO}) = 180 - (\text{angle } \mathbf{AOB} + \text{angle } \mathbf{OAB})$ . So, in every iteration angle **AOB** or Elevation angle will be changed and return a new **Pitch** value.