**Python scripts and Blender 2.80.**

Tutorial series: <https://www.blender.org/support/tutorials/>

Video (external to tutorial series): <https://www.youtube.com/watch?v=rHzf3Dku_cE>

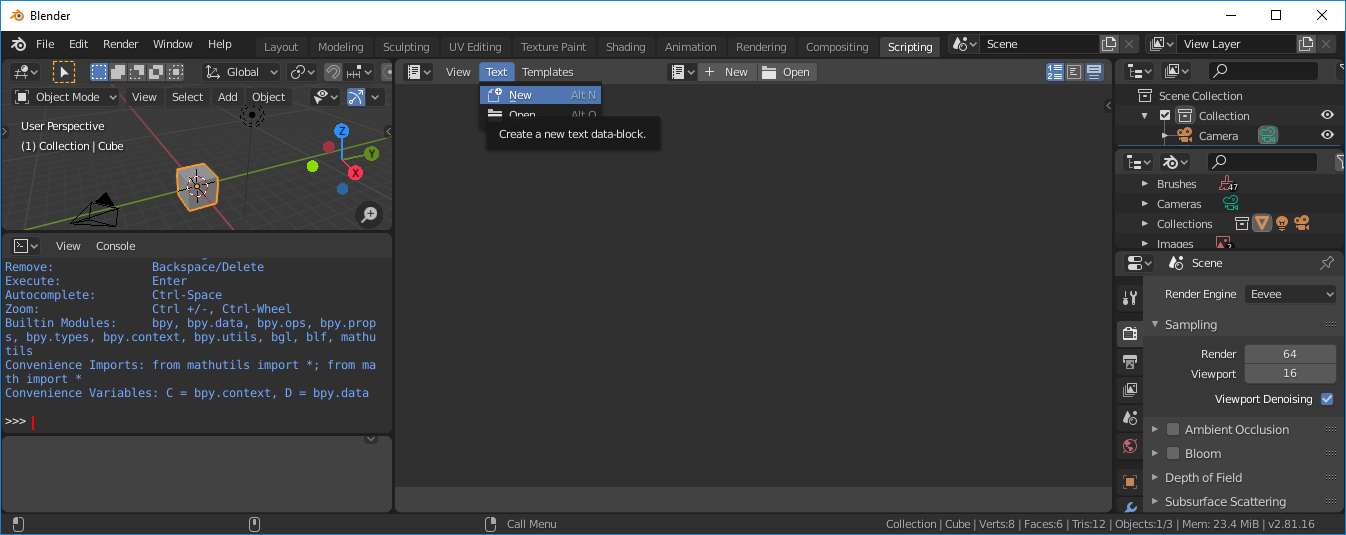
**1. Introduction**

You normally build images and animations in blender using the standard GUI. Blender does support a programming interface built on Python scripts. If you get good at writing scripts, you can create your own Blender plugins and add buttons and options to add new functionality to the standard user interface. You can use scripts to do most of the things you can do using the GUI. Scripts are less intuitive and require knowledge of programming but do allow you to do some things with far less effort including creating a large numbers of key frames, using data about object behaviour from external files, automating the generation of scene based on a set of parameters and exporting motion data.

Let us see what we can do…

**2. Becoming familiar with the Scripting interface**

Launch blender and click on the Scripting tab at the top of the window and then create a new text file.



Copy the following script and paste it into the Text editor Window on the RHS of the screen.

import bpy

# useful shortcut to reference the scene later

scene = bpy.context.scene

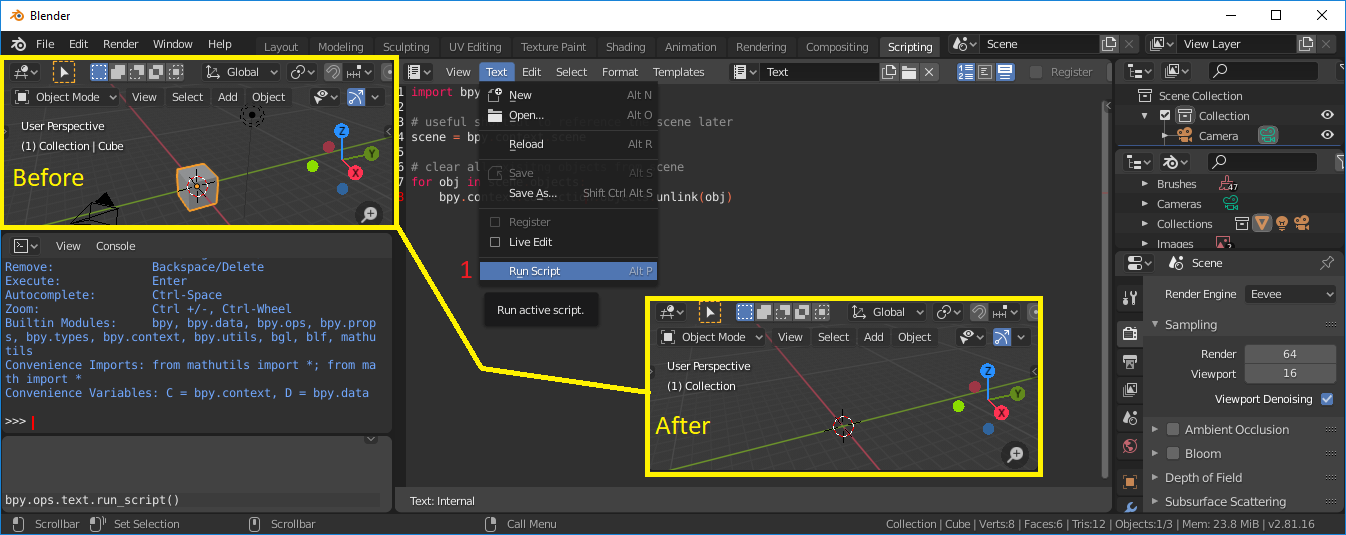
# clear all existing objects from scene

for obj in scene.objects:

bpy.context.collection.objects.unlink(obj)

Blender includes a Python interpreter that always runs. They bpy module can be imported into a script and gives access to Blender classes, data and functions. The python code changes the internal data used by Blender. You can use python scripts to set up and run complicated sequences that would be time consuming to do by hand (e.g. in the Layout/Time line Window).

Select Text – Run Script to run the Script. This script is the equivalent to clear screen and deletes all the objects including the default cube, camera and light.



Now that we have a blank scene we can start adding objects.

First we can add a ball, by including the following code after the code to clear the scene.

#create sphere and make it smooth

bpy.ops.mesh.primitive\_uv\_sphere\_add(location = (0,0,0), radius=1.0)

bpy.ops.object.shade\_smooth()

# Store a reference to it while it is the active object (just created)

sphere = bpy.context.active\_object

# mat\_r is defined be a new material with a red colour

mat\_r = bpy.data.materials.new("mat\_red")

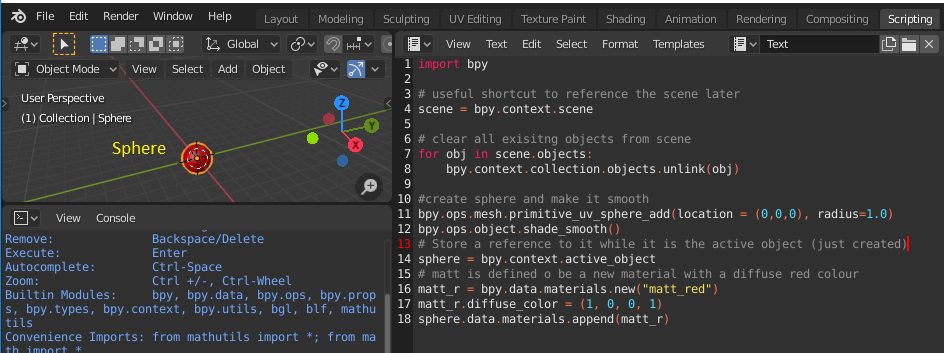
mat\_r.diffuse\_color = (1, 0, 0, 1)

sphere.data.materials.append(mat\_r)

**Interpretation of script**

bpy [blender python import] – ops [operators] – mesh [mesh operator] – primitive\_uv\_sphere\_add [add a sphere to the scene] followed by (execution arguments or parameters)

Again run the script to see if the ball appears.



We can now add lighting as follows

# Create a light source and configure it

light\_data = bpy.data.lights.new(name="light\_1", type='POINT')

light\_data.energy = 30

# create new object with our light data block

light\_object = bpy.data.objects.new(name="light\_1", object\_data=light\_data)

# link light object

bpy.context.collection.objects.link(light\_object)

# make it active

bpy.context.view\_layer.objects.active = light\_object

#change location

light\_object.location = (-6, 0, 12)

Add this code and run the complete script to see the light shine on the side of the sphere. Make sure to Press “z” in the 3D Viewport window (top right) and select “Rendered”. Note: not all renderers will support the different surface finishes in the same way. So if you end up doing this type of work in more detail you will need to set the default shader node for the object/material to principled bsdf. However, what we have is good enough for now.

Add a camera to the scene using the following code snippet.

# Create a camera

cam\_data = bpy.data.cameras.new(name="cam")

cam\_object = bpy.data.objects.new(name="Camera", object\_data=cam\_data)

bpy.context.collection.objects.link(cam\_object)

scene.camera = cam\_object

# Configure the camera

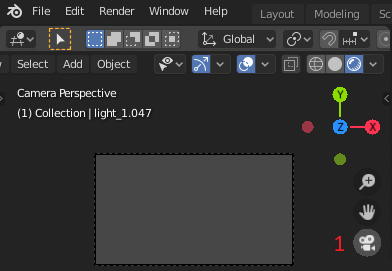
cam\_object.location = (0, -4, 0)

cam\_object.rotation\_euler = (0,0,0)

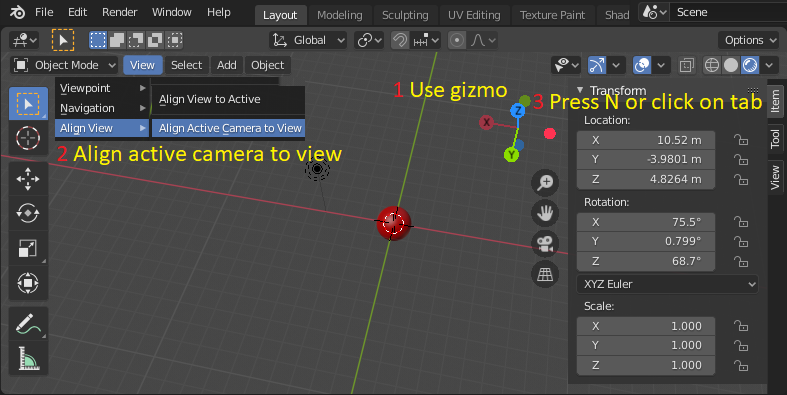
cam = bpy.data.cameras[cam\_data.name]

cam.lens = 30

When you run the script and click on the render view, you will notice that the camera is pointing the wrong way, press the camera button to see the camera view.



To find a good view go to the 3D Viewport window (you can do this in Layout rather than Scripting as it is bigger), use the gizmo to get a good view of the sphere and then press View – Align view – Align camera to view.



To find the camera transform click on the small tab at the top right of the 3D viewport or press N. Note the position (e.g. 10.5, -4.0, 4.8) and the rotation converted from degrees to radians (e.g. 1.32, 0, 1.12). Note to convert from degrees to radians multiply by PI/180.

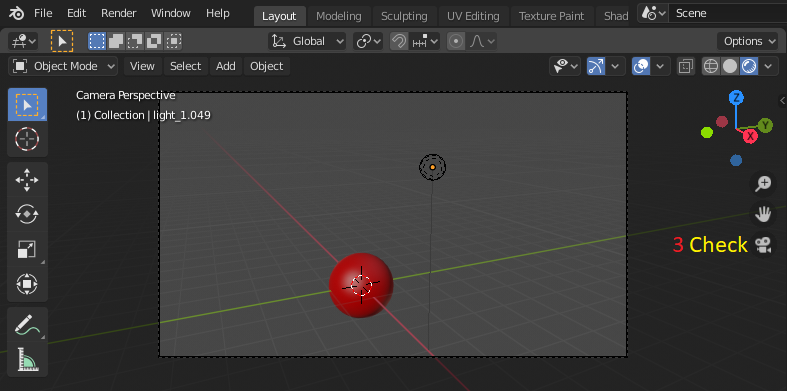
If this is a challenge, then you could use the following lines to replace the existing ones.

# Configure the camera

cam\_object.location = (10.5, -4.0, 4.8)

cam\_object.rotation\_euler = (1.32, 0, 1.12)

Run the script and press the camera view. Check that the view has changed when you press the camera button to see the rendered view.



Add the following code snippet to create key-frames so as to create an animation and run the script.

# animation

positions = (-2,-2,0),(-2,2,0),(2,2,0),(2,-2,0),(-2,-2,0)

# start on frame 0

number\_of\_frame = 0

for position in positions:

# create a frame using number\_of\_frame

scene.frame\_set(number\_of\_frame)

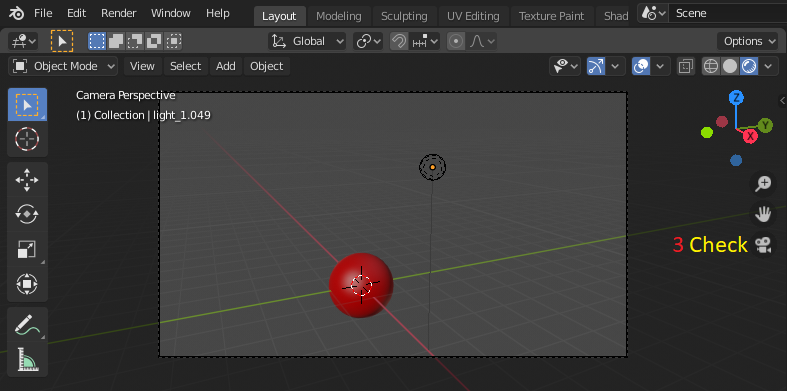
sphere.location = position

sphere.keyframe\_insert(data\_path="location", index=-1)

# move forward 10 frames, blender estimates positions between keyframes

number\_of\_frame += 10

Switch to layout and then adjust the position of the playback slider on the Timeline editor at the bottom of the screen. You should see the sphere move in a square.



To set the start and end frame you can add the following two lines.

Aside: You should use the opportunity to try using the auto-complete function. Type the second line until you get to the “f” following the “.” and press <tab> to give a list of auto complete options. If you press the <tab> on the “.” it will list all the methods/variables available – very useful.

# Set start and end frames

scene.frame\_start = 1

scene.frame\_end = 40

You could now create an animation from the script by rendering it as a video (see assignment 1 for a reminder on how t do this).

The following is useful but unnecessary to complete the assignment.

Camera look at function: A “look at” method is not fundamental to the assignment so don’t do it if the install required is a challenge. The instructions for installing this utility without installing Visual Studio were included on the first handout and are repeated at the end of this handout. With the maths utility and our knowledge of quaternions we can create a “look at” method that will position the camera looking towards a point of interest. Try adding the following code to you program. You will need to comment out the existing lines of code that positions the camera and put the code to import from *mathutils* after the import bpy. The camera will now look at the sphere, wherever it is positioned.

import bpy

# Use pip install mathutils if this line returns error (Python 3.6.1)

from mathutils import Vector

# Configure the camera (comment out or removed)

# cam\_object.location = (10.5, -4.0, 4.8)

# cam\_object.rotation\_euler = (1.32, 0, 1.12)

# cam = bpy.data.cameras[cam\_data.name]

# cam.lens = 30

# Configure the camera and face it towards the sphere

cam\_object.location = (10.5, -4.0, 4.8)

direction = sphere.location - cam\_object.location

rot\_quat = direction.to\_track\_quat('-Z', 'Y')

cam\_object.rotation\_euler = rot\_quat.to\_euler()

cam = bpy.data.cameras[cam\_data.name]

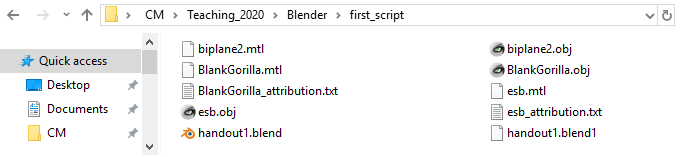
cam.lens = 30

**Assignment:** Create a short low resolution animated movie using a blender script of your own making (<100Mbytes at run time). If you wish you can do the King Kong example (yourself) but do something different or modify the Kong example so that I don’t have to watch 70 identical videos. You can, if you wish, create a blend swap account to look for suitable assets to enhance your own work (or other sites with open source models), to do this go to.

<https://www.blendswap.com/> (certificate issue at time of writing, don’t use until this is resolved)

Always include the attribution to acknowledge the work of others.

**Example:** try to recreate the scene where the biplane flies around the Empire State Building as King Kong climbs (or just holds onto) the side of the building. The assets to do this are in a zip file on moodle (obtained from Blendswap and exported as Wavefront Objects (.obj) models), download, unzip and place them in the same folder as the project *.blend* file.



King Kong 1933



To load the biplane, Gorilla and Empire State Building use the following snippet of code, change the path to the correct file path for your machine.

# Load the model of the Empire State Building (and center it)

bpy.ops.import\_scene.obj(filepath="G:\\Desktop\\Maynooth\_University\\CS426\\assignment3\\Content\_new\\Content\\esb.obj", filter\_glob="\*.obj;\*.mtl", axis\_forward='-Z', axis\_up='Y', global\_clight\_size=0.0)

bpy.ops.transform.translate(value=(0.0, 1.45, 0.0))

# Load the biplane model (and resize it)

bpy.ops.object.select\_all(action='DESELECT')

bpy.ops.import\_scene.obj(filepath="G:\\Desktop\\Maynooth\_University\\CS426\\assignment3\\Content\_new\\Content\\biplane2.obj",

filter\_glob="\*.obj;\*.mtl", axis\_forward='-Z', axis\_up='Y')

bpy.ops.transform.translate(value=(0.0, 0.0, 0.0))

bpy.ops.transform.resize(value=(0.2, 0.2, 0.2))

biplane=bpy.context.selected\_objects[0]

# Load the Gorilla model (and resize it)

bpy.ops.object.select\_all(action='DESELECT')

bpy.ops.import\_scene.obj(filepath="G:\\Desktop\\Maynooth\_University\\CS426\\assignment3\\Content\_new\\Content\\Gorilla.obj",

filter\_glob="\*.obj;\*.mtl", axis\_forward='-Z', axis\_up='Y')

bpy.ops.transform. rotate ( value=-1.1 , orient\_axis='X')

bpy.ops.transform.translate(value=(0.0, -0.75, 8))

bpy.ops.transform.resize(value=(0.8, 0.8, 0.8))

The scene may look better if seated on a plane (green surface).

# create a flat plane on which to build the scene

bpy.ops.mesh.primitive\_plane\_add(location=(0,0,0.01))

plane = bpy.context.active\_object

plane.dimensions = (100,100,0)

mat = bpy.data.materials.new("mat\_plane")

mat.diffuse\_color = (0, 1, 0, 1)

plane.data.materials.append(mat)

If you wish to rotate the plane as it flies around the building, then the following lines may be of help. The plane is being located at positions set using points on the unit circle identified using (cos() and sin()). The plane also needs rotating about its own axis, so it appears to fly forward. The code makes use of the python maths library.

import bpy

import math

….

# start on frame 0

number\_of\_frame = 0

# Set start and end frames

scene.frame\_start = 0

scene.frame\_end = 40

for number\_of\_frame in range(scene.frame\_start,scene.frame\_end):

# create a frame using number\_of\_frame

scene.frame\_set(number\_of\_frame)

biplane.location = (5\*math.cos((2\*3.1415)\*number\_of\_frame/40),5\*math.sin((2\*3.1415)\*number\_of\_frame/40),8)

biplane.keyframe\_insert(data\_path="location", index=-1)

biplane.rotation\_mode = 'XYZ'

biplane.rotation\_euler= [-3.1415/2,3.1415,(2\*3.1415)\*number\_of\_frame/40]

biplane.keyframe\_insert("rotation\_euler", index=-1)

You should submit the final video for grading, in addition you should also submit a zip file of the blender project (including assets or reference to them if very big) and the script that you created. If you decide to do the King Kong example, then marks will depend on some novelty being added to the script or rendered scene.

[3 marks the movie, 3 marks the blend file containing functioning script, 4 marks some novelty in the script and/or rendered scene] Moodle will accept both files in one submission.

Other video sequences

[5 marks for the movie, 5 marks for the blend containing a functioning script]

Finally: This is enough for now, but it is possible to add new options in drop down menus in Blender using python. You could add a “Add Movie Scene” button made available as a plugin in Blender. We will leave that for another day, what you have done is a good start towards this.

See the following for more details.

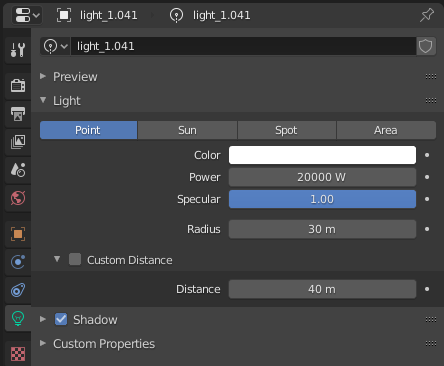
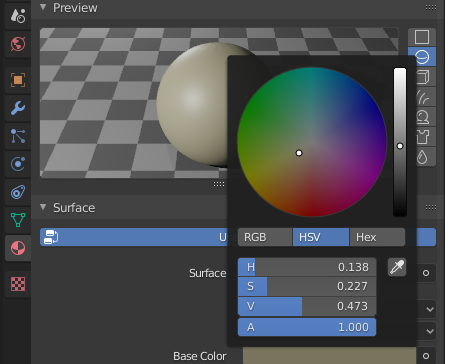
https://docs.blender.org/manual/en/latest/advanced/scripting/addon\_tutorial.html

Below is a frame from a video created using the approach described above,

A picture containing text

Description automatically generated

Note: the light needed adjusting (20,000W, point, radius 30m) and a texture of the gorilla (dark brown material) was added to the gorilla, after running the script. It could have been added in code.

CM Feb2022 V1.4

**MathUtils:** To extend this tutorial you would need to add the maths utility package for python – it makes moving the camera much easier as it allows you to “look at” a point of interest. That said if you can’t get it to install keep going as the assignment can be done without them.

If you have Blender and VisualStudio 2019 on your PC already then all you will need to do is type the following command from the command line. Click on the windows icon – type cmd in the white space and then select command prompt. You may get warning about float to double conversion, but it should run.

python -m pip install mathutils

Text

Description automatically generated

Blender installs the python it needs however you may wish to do a full install of the relevant version of Python to have all the libraries if you are doing significant amounts of scripting.

Text

Description automatically generated

If you do not have Visual Studio on your machine, then you will need to install MS Build Tools so that mathutils can be compiled by pip, again this can wait if you want to get started straight away. It may be better to have the full VS2019 up and running now.

<https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=BuildTools&rel=16>

[https://docs.blender.org/api/current/mathutils.html - mathutils.Quaternion](https://docs.blender.org/api/current/mathutils.html#mathutils.Quaternion)

Graphical user interface, application

Description automatically generated