

This PDF is available for free; feel free to share it!

[Subscribe to my YT channel](#) & turn on notifications to get updates on new videos!

PART 1 --- INTRODUCTION

Hi!

In this tutorial, we will see how to use the ***UV export layout Python function*** in Blender to export the **UV maps** (or: layouts) of all the MESH objects available in a scene.

This script can be helpful if you need to provide UV maps as images, such as when publishing your 3D scene on 3D stock websites.

Before we go on, I remind you that - as it happens with most of my tutorials - the text version of this videotutorial is available for free download at the link provided in the description.

This tutorial was made using Blender 3.3; sometimes, the Blender developers make changes to the **Blender Python APIs** (*BPY* for short), but this script should work from version 3 onwards.

The ***bpy.ops*** function we are going to use is pretty simple: it only requires two parameters, namely the filepath of the image to be created and the **UV size** in pixels; however, since we have to use it for all available MESH objects in the scene, we will use a ***for*** statement on the MESH objects.

Additionally, I will specify the **file extension**, which is **PNG** by default. Actually, I will export the layouts as PNGs, but I want to show you how to specify it because you can also export the UV layouts as **SVG** or **EPS** files.

PART 2 --- THE TUTORIAL

Okay, let's get started!

Open a BLEND file.

You can start with a brand-new file, too, but **you will have to save it** somewhere because we will put the PNG files in the same folder as the project.

Let's do some basic imports:

```
import bpy
from bpy import *
```

As I said before, the **`export_layout` function requires two parameters**, so let's define two variables (we could write the values in the function itself, but I prefer to define them here for clarity's sake):

```
UVpath = bpy.path.abspath("//") + "UV-LAYOUT---"
UVsize = (2048, 2048)
```

As you can see, **`bpy.path.abspath`** will yield the absolute path on disk of the BLEND file. We are going to add a prefix ("**`UV-LAYOUT---`**") to all the PNG files, but you can change or remove that.

The **`UVsize`** parameter is a **tuple** of two numerical values, which are the dimensions - in pixels - of the images to be created.

I'm writing 2048, thus creating a 2k image.

In order to execute the function on all the MESH objects available in a scene, we have to create a ***for* statement**. So we can write:

```
for o in bpy.data.objects:
```

then, inside the block of the ***for* statement** (so don't forget to indent the text), we can add a "filter" on the type of the object:

```
    if(o.type=="MESH"):
```

The condition is clear: the type of the current `bpy.data.object` must be MESH.

The following statements must be put inside the ***if***, so - again - don't forget to indent the text!

```
        bpy.ops.objects.select_all(action='DESELECT');
```

we have to do this every time **to deselect all the other objects**, otherwise we

would get an image with two or more UV maps overlapped.

`bpy.context.view_layer.objects.active = o`

This assignment **will set the active object of the scene**, assigning the current mesh to `context.view_layer.objects.active`.

We have to do this because the following **OPS (Blender Python operation)** will be performed on the one and only active object (remember: you can select multiple objects, but there's only one active object at a time).

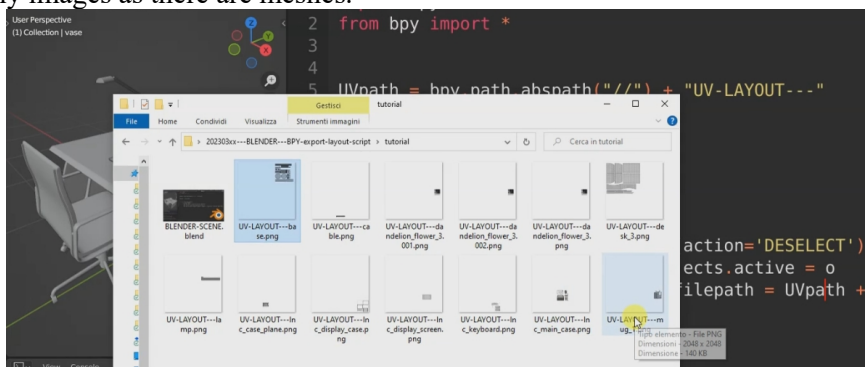
`bpy.ops.uv.export_layout(filepath = UVpath + o.name, size = UVsize, mode='PNG')`

The `ops` statement will **export the layout of the active object** to the given filepath (made by the UVpath variable, with the "UV-LAYOUT---" prefix, plus the name of the active object) and with the given size, in pixels.

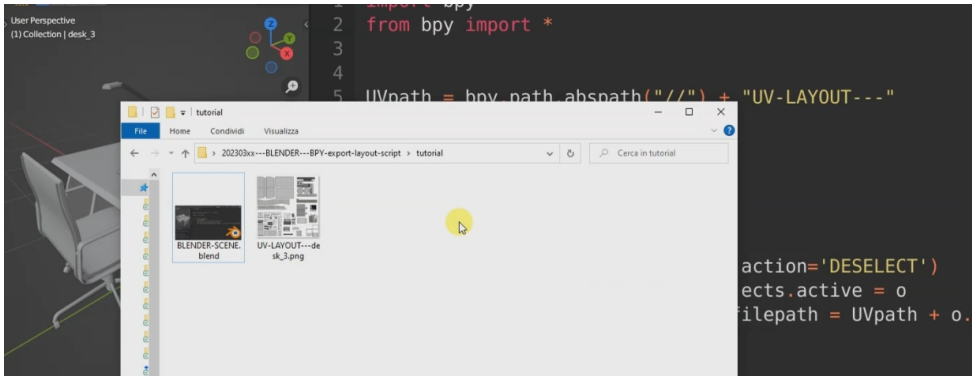
```
1 import bpy
2 from bpy import *
3
4
5 UVpath = bpy.path.abspath("//") + "UV-LAYOUT---"
6
7 UVsize = (2048,2048)
8
9
10 for o in bpy.data.objects:
11     if(o.type=="MESH"):
12         bpy.ops.object.select_all(action='DESELECT')
13         bpy.context.view_layer.objects.active = o
14         bpy.ops.uv.export_layout(filepath = UVpath + o.name, size = UVsize, mode='PNG')
```

As I told you before, you can export the UV layouts as **PNGs, EPSs, or SVGs**. You can specify the output format with the mode parameter, as I just showed.

If we take a look at the output folder, we can see that the script has created as many images as there are meshes.



Since the meshes in my "Desk set 3" 3D model share the same layout with non-overlapping UV islands, I can join all the objects together, run the script again, and get just one non-overlapping UV layout map of the asset.



Before closing this tutorial, I'm going to give you an assignment: edit this script to export both the 2k and 4k UV layouts of the meshes available in your scenes, providing the images with two different prefixes ("*UV-2K---*" and "*UV-4K---*", for example).

That's all for this tutorial!
See you soon!
