Tutorial 2: How to Write Scripts for VRED with Python

Download the example script here: https://github.com/raemorris/VRED-Python-Tutorial-Examples-Scripts/blob/main/tutorial2.zip?raw=true

Sample Python Code

This is the accompanying example Python scripts for the *Tutorial 2: How to Write Scripts for VRED with Python* video.

- To view the video, visit: https://www.youtube.com/watch?v=K32JqOPBChU
- To review the video caption, see *Video Captions*.

simple_examples.py

```
'''
Run the python examples by copying and pasting the code snippets
into the VRED Script Editor and press run to see the results
'''


groupNode = vrNodeService.findNode("Group Node")
deleteNode(groupNode)



# Example 1.0:
# Finding nodes in the screnegraph
root = vrNodeService.findNode('Root')
camera = vrNodeService.findNode('Perspective')

print(root)
print(camera)
```

```
# Example 1.1:
# Creating new nodes
print()
print('---------------')
groupNode = createNode('Group', 'Group Node', root)
print(groupNode)
updateScenegraph(True)

# Example 1.2:
# Creating geometry and add them to the scenegraph
print()
print('---------------')
planeNode = createPlane(1000, 1000, 1, 1, 1, 1, 1)
planeNode.setName('Plane Geometry')
planeNode.setTranslation(100, 200, 10)
groupNode.addChild(planeNode)

boxNode = createBox(500, 500, 500, 1, 1, 1, 1, 1, 1)
boxNode.setName('Box Geometry')
boxNode.setTranslation(-100, -200, 10)
groupNode.addChild(boxNode)

# Example 1.3:
# Move nodes in the scenegraph
print()
print('---------------')
anotherNode = createNode('Transform', 'Another Node', root)
anotherNode.setTranslation(0, 0, 500)

planeNode = vrNodeService.findNode('Plane Geometry')
boxNode = vrNodeService.findNode('Box Geometry')
```

```
anotherNode.addChild(planeNode)
anotherNode.addChild(boxNode)

# Example 2.0:
# Create a new camera
camera = vrCameraService.createCamera("New Camera")

cameraFrom = QVector3D(2000, 2000, 5000)
cameraAt = QVector3D(0, 0, 0)
cameraUp = QVector3D(0, 0, 1)
fromAtUp = vrCameraFromAtUp(cameraFrom, cameraAt, cameraUp)

camera.setFromAtUp(fromAtUp)
camera.activate()

# Create a camera viewpoint from the current camera
viewpointName = 'New Viewpoint'
viewpoint = vrCameraService.createViewpoint(viewpointName)

# Example 3.0:
# Create a basic phong material
materialA = createMaterial("UPhongMaterial")
materialA.setName("Material A")
materialA.fields().setVec("diffuseColor", [1, 0, 0])

materialB = createMaterial("UPhongMaterial")
materialB.setName("Material B")
materialB.fields().setVec("diffuseColor", [0, 1, 0])

# Example 3.1:
# Create a material switch and add the newly created materials
materialSwitch = createMaterial("SwitchMaterial")
```

```
materialSwitch.setName("Material Switch")
materialSwitch.addMaterial(materialA)
materialSwitch.addMaterial(materialB)

# Example 3.2:
# Apply the material the geometry
plane = findNode("Plane Geometry")
box = findNode("Box Geometry")

materialA = findMaterial("Material A")
materialB = findMaterial("Material B")
plane.setMaterial(materialA)
box.setMaterial(materialB)

# Example 4.0
# Create a variant set
variantSetName = 'New Variant Set'
createVariantSet(variantSetName)
variantSet = getVariantSet(variantSetName)

# Example 4.1
# Add stuff to the variant set
variantSet.addScript("print('Hello World!')")
variantSet.addView(vrCameraService.getViewpoint('New Viewpoint'))
```

Video Captions

Hello and welcome to our tutorials series, Scripting with Python, for VRED Pro. I'm your host Christopher and in this video I will talk about a few Python basics that you need for developing in VRED. I will also show you some examples to get you started with developing your own scripts.

Python is an interpreted programming language that is very popular for plugin development. There are a lot of other CGI tools that offer a Python interface and that is not just a coincidence. Python is said to be a very expressive language and easy to learn. Compared to programming languages like C++, C#, or Java, it does not have to be compiled, so you can just write some code, hit Run and the code immediately does what you told it to do. Python also works nicely together with C++, which is nice because, under the hood, VRED is also using C++. Just to give you something to talk about on parties.

When Autodesk introduced the API version 2, they switched from Python 2.7 to Python 3. This made sense because officials support for Python 2 ended in January 2020. But if you are using scripts that still use Python 2.7 don't worry – VRED has an option to automatically convert Python 2.7 to Python 3.

What else can I say about Python and VRED… well, VRED uses its own integrated Python 3 interpreter. This way they can directly inject the VRED API into its Python environment.

What does this means for us? Python delivers all its functionality in so called modules. They are just a collection of classes and functions. The VRED API is just the same. A collection of modules. Normally, you would have to import all these modules by hand to use them. For example, when you want to use the vrNodeUtils module, you would have to import it by typing "import vrNodeUtils". But VRED already added all these modules for us, so we don't have to do this. This makes scripting inside VRED much more readable and reduces a lot of extra code.

When developing external modules or script plugins, you have to be careful. Here you do have to explicitly import the VRED API, but only the API version 1. I will talk about it a little bit more in an upcoming video.

So, let's look at some real examples. They won't be too complex, but this is just to warm up a little bit.

In most scripts, you will have to access your Scene Graph in some way. The Scene Graph is the central part of 3D graphics and holds all the information about geometry, lights, transformations, and so on. It is a hierarchical structure that starts with the root node and divides your scene into transform nodes, groups nodes, or switch nodes down to the geometry itself. In a typical script you will either add nodes or geometry to a Scene Graph, move nodes from one place to another, or you want to find a node in the Scene Graph to manipulate it.

In our first example, we start by searching nodes in our Scene Graph. We open the Script Editor that is located under Edit in the menu bar and enter our script. Here we start by searching for the root of our scene and the camera node using the vrnodeservice. The camera node is called Perspective, as you can see in the Scene Graph.

Nodes are typically refered to by their name in the Scene Graph. We store the result of this operation as a variable, which now hold an object of type vrdNode. vrdNodes are defined in the API version 2. We can execute the script by hitting Run and see the result in our print message in the terminal. This message tells us what type of objects the two variables now hold.

Next, we want to create new nodes and add it to the Scene Graph. Here we are creating a node of type Group. You can look at the documentation to find all node types you can create. To do this, we use the createNode function with the first parameter, defining the node type, the second parameter setting, its name, the third parameter, telling VRED which node should be the parent of our new node. In this case, we want our new node to be located under the root of our scene. After this, we want to call the updateSceneGraph function from the vrSceneGraph module. This command will directly show us the changes in the Scene Graph.

Let’s create some geometry and add it to our new group node. This can be done by using a set of functions that will create primitive geometry for us. Here we are using the createPlane function and the createBox function.

Notice how the editor already suggests functions we can use when we start typing. This is very helpful when writing scripts. You can just start typing and see what functions are available. Unfortunately, this feature does not tell you what parameters you have to use, so there is now way around reading the documentation. Sorry.

The first few parameters describe the size of the plane and the box, the next three parameters define the resolution, and the last three numbers defined the color, which is just 100 per cent white. I also added two functions to remove the old group node, whenever I execute the script. Otherwise, our Scene Graph would get cluttered with old group nodes.

Next, we add a name for each geometry and set their translation in the scene with the setTranslation function. This function receives an x, y, and z component that describes the position in the scene.

Please note that this setTranslation function is from the API version and uses the old set of parameters, instead of the vector representation.

In the last step, we add these two geometries to our group node. Otherwise, the geometry would float somewhere in our memory, but would never be attached to our Scene Graph. When we hit Run, we now can see our new geometries in the scene.

We can also move nodes from their parent to another node. Let’s start with creating another node called “Another node”. This time, we use the node typeTransform. This way, we can translate the node after we created it, using the setTranslation function.

As in the first example, we find our two geometries using their name. Then, we use the addChild function of our new transform node. This way, we remove the nodes from the group node and add it to the transform node. You also see that the geometry now has moved a little bit up.

Next, we want to create a new camera. Here we can use the vrCameraService. We also want to set its location in the scene and define where it should look. Here, we use the setFromAtUp function of the camera that expects an object of type vrCameraFromAtUp. We can create this object by using its constructor and input three vectors that define the position, where it should look at, and a vector that defines which side is “up” in the scene. At last, we activate the new camera and see that our camera in the scene has changed.

It’s also quite easy to create viewpoints from our current camera position. This is also done with the vrCameraService. It has a createViewpoint function that just expects a name as an input. This way we can move our camera in the scene and capture viewpoints that we can access in our camera menu.

In the next example, we want to create our own materials. Let’s create two new materials with the createMaterial function. We have to specify a material type, which is described in the documentation.

Here, we are creating a basic phong material. To set the color of our materials, we have to use fieldAccess. This is a special kind of module, which allows us to manipulate and read basically all data that is stored in nodes.

I will talk about fieldAccess in another tutorial more in depth. Just notice that we use it here to define a red and a green material. When we run the script and open the Material Editor, we see that we now have two new materials. Maybe you have to close the Material Editor and open it again for the changes to show.

Now, we also want to add these material to our geometry. So again, we find our geometry in the scene and then use the setMaterial function to change their material.

For our last example, we want to create a new variant set. This can be done with the createVariantSet function that expects the name of the new variant set as an input. We also can find a variant set with the getVariantSet function, using the name of the variant set as the parameter. Of course, we can also add stuff to our variant set. Here is just a small example of how you can add a Python script to our new variant set.

This was just a fraction of the possibilities you have when scripting in Python. There is so much more to show you, like managing sceneplates, creating animations, and so on. This was just a few examples to get you interested.

Remember that there are a lot of examples in the example section of VRED, where you can search for solutions to your particular task. For example, if you want to add a menu to your VRED scene, you can have a look at this menu example that tells you all the basics for this.

Okay, that’s it for today. I hope you enjoyed our todays video on basic Python scripting in VRED. Thanks for joining and see you next time.