Tutorial 8: How to Combine the Old V1 and the New V2 Python Interface

The *Tutorial 8: How to Combine the Old V1 and the New V2 Python Interface* video covers examples of interoperability between the old vrNodePtr and new vrdNode object.

Download the example scripts here: https://github.com/raemorris/VRED-Python-Tutorial-Examples-Scripts/blob/main/tutorial8.zip?raw=true

Sample Python Code

Here are the accompanying example Python scripts for the *Tutorial 8: How to Combine the Old V1 and the New V2 Python Interface* video.

- To view the video, visit: *Coming soon*
- To review the video caption, see *Video Caption*.

deprecated_examples.py

```
# Deprecated API v1 functions


# vrCamera
addViewPoint("New Viewpoint")


vrCameraService.createViewpoint('New Viewpoint')


# vrOSGWidget
createBackplate('filePath')
setBackplate('filePath')
deleteBackplate()


vrSceneplateService.createNode()  # TODO Parameter prüfen
```

```
# vrRenerSettings
setRaytracingAAThreshold()
```

difference_nodes_v1_v2.py

```python
# Node pointers with API v1

# Find node "EXT" with API v1
extNodePtr = findNode("EXT")

# set translation
extNodePtr.setTranslation(0,0,38)

# get translation
translation = extNodePtr.getTranslation()
print("Translation:", translation)

# set name
extNodePtr.setName("Another EXT")

# Node pointers with API v2

# Find node "EXT" with API v2
ext`VrdNode` = `vrNodeService.findNode`("EXT")

# set translation
translationVector = QVector3D(0,0,38)
ext`VrdNode`.setTranslation(translationVector)

# get translation
```

```
translation = ext`VrdNode`.getTranslation()
print("Translation:", translation)

# set name
ext`VrdNode`.setName("Another EXT")
```

node_examples.py

```python
# Example 1.0
# Finding Nodes

# Finding nodes with API v1 and v2
study_NodePtr = findNode('Study_Transform')
study_`vrdNode` = `vrNodeService.findNode`('Study_Transform')

# Example 2.0
# Using API v1 functions with `vrdNode`s
createNode("Group", "Interieur_nodePtr", study_NodePtr)
print("Created new group with nodePtr")

createNode("Group", "Interieur_`vrdNode`", study_`vrdNode`)
print("Created new group with `vrdNode`")

updateScenegraph(True)


# Using API v2 functions with `vrNodePtr`s
interieur_NodePtr = findNode('Interieur_nodePtr')

childIndex = study_`vrdNode`.getChildIndex(interieur_NodePtr)
```

```
print("Interieur Node is at index:", childIndex)



# Example 3.0

# Convert from nodePtr to `vrdNode`

targetNodePtr = findNode("Target")

targetNode = `vrNodeService`.`getNodeFromId`(targetNodePtr.getID())


# Convert from `vrdNode` to nodePtr

targetNodePtr = toNode(targetNode.getObjectId())
```

Video Caption

Hello and welcome to this new Python tutorial for VRED Pro. My name is Christopher and today I will talk about how you can combine the different data types and functions from the Python API version 1 and version 2.

The API version 2 release introduced some huge changes in the way we interact with data in VRED. Instead of using modules, the functionality is now combined in so called services. For example, the vrCameraService or the vrNodeService now contain all the methods that were previously available in vrCamera and vrNodePtr. These services replace most of the functions from API version 1, but not all of them. There still is some functionality that is only available with API version 1. There also may be plugins and scripts that use the old API, with interfaces that expect data compatible to API version 1. So, there are situations where we have to combine functions and data types from both APIs to make sure old scripts and plugins work well with new scripts we are developing.

Let's start with the biggest difference. Dealing with nodes. Version 1 used the so-called node pointers to interact with nodes, like transform nodes, group nodes, geometry nodes and so on. You could find nodes in your scene with the function from the vrScenegraph module findNodes, which returns a vrNodePointer. With this node pointer, we can call any function that is part of the vrNodePointer class that we can see in the documentation. So, for example, all transform operations, setting or getting a material, adding children to the Scene Graph, and so on.

With the introduction of API version 2, Autodesk now introduced a so called vrdNodeType that aims to replace the old vrNodePointer to some extend and a vrNodeService class to help you working with it. When called from the vrNodeService, the findNode operation returns a vrdNode and can be called with vrNodeService.findNode. This way, you cannot confuse these similar functions. With the vrdNode that is returned by the service, you can now call any function that is defined in the vrdNode class. Because vrdNode inherits if functions from vrdTransformNode, you also can call any transform operations from there.

There are still enough functions from API version 1 and 2 that will get used in the same script and they will either use the “node pointer” class as parameters or the new vrdNodes. Normally, this would be really confusing, fortunately, the Autodesk engineers changed both APIs, so that they accept both types of “nodes” in a function. It’s perfectly fine to use “node pointer” in a function that normally expects a vrdNode and the other way round. This way we don’t have to worry if we have the correct node. In this example, we can create a new node and specify its parent node. We can see that we can use both types of node as a parameter without throwing an error.

Of course, it’s also no problem to convert a “node pointer” to a vrdNode.

The documentation points out that there are two functions to do this, but they are not the easiest to remember. When converting from a new vrdNode to an old “node pointer” you will use the toNode function with the object ID of the vrdNode as a parameter. This will convert your vrdNode to a “node pointer”. The other way round is done by calling the getNodeFromId function from the vrNodeService class. This is also done with the “node id” as parameter. But this time, we get the ID with a different function. Well, if you need this conversion, just have a look in the documentation. No one really expects that you know them by heart and peeking is allowed.

The API version two also introduced new data types that help when developing software. For example, the “transform operations” now all use the “QT vector” types. Using them is not really difficult. Most of the time, it's just putting the values you would have otherwise used separately into the new data types, like in this example of a new vector 3D type.

Color types have a static function that help you create an object and also offer different color formats, like RGB or HSV. You don’t have to import these types because they are already added to the VRED namespace by default. We have a whole tutorial covering the new data types so, be sure to check it out to get more information on that topic.

As of now, both the vrNodePointer and vrdNode are perfectly valid in your scripts. There are use cases where you want to use one over the other. Either because it is just more convienient to do so or because the new API just isn’t ready yet to replace all functionality of vrdNodePointers.

A good example is when dealing with materials. You only can set and get materials with the vrNodePointer and not the vrdNode. But I guess we will see something like a vrMaterialService in an upcoming release. With time, the focus will definitely shift from the old to the new API, so you should use the API version 2 whenever possible.

That’s it for today, about the old API version 1 and the new API version 2. You should now have a good understanding on how to combine the two APIs to write efficient and future proof scripts. Thanks for watching and see you next time.