Tutorial 10: How to Work with QT Signals in the Services

Download the example scripts here: https://github.com/raemorris/VRED-Python-Tutorial-Examples-Scripts/blob/main/tutorial10.zip?raw=true

Sample Python Code

Here are the accompanying example Python scripts for the *Tutorial 10: How to Work with QT Signals in the Services* video.

- To view the video, visit: https://www.youtube.com/watch?v=xUrMppZIL-I
- To review the video captions, see the *Video Captions* section.

service_signals.py

```
# Example 1.0
# Connecting to service signals
# vrAttachmentService
def annotationCreatedCallback(annotation):
    print("Created annotation", annotation)

def annotationAddedCallback():
    print("Added annotation")

def annotationsDeletedCallback():
    print("Deleted annotation")

def annotationsShowChangedCallback(visible):
    print("Annotations are visible: ", visible)

`vrAnnotationService`.annotationCreated.connect(annotationCreatedCallback)
`vrAnnotationService`.annotationsAdded.connect(annotationAddedCallback)
`vrAnnotationService`.annotationsDeleted.connect(annotationsDeletedCallbac
k)
`vrAnnotationService`.showAnnotationsChanged.connect(annotationsShowChange
dCallback)

# vrNodeService
def nodesChanged(nodes):
```

```
    print("Nodes changed: Rebuilding find cache...")
    vrNodeService.clearFindCache()
    vrNodeService.initFindCache()

vrNodeService.nodesAdded.connect(nodesChanged)
vrNodeService.nodesRemoved.connect(nodesChanged)

# Listening to general object changes
def propertyChanged(obj, propertyName):
   print("object name: {}, property name: {}".format(obj.getName(),
propertyName))

vrObjectService.propertyChanged.connect(propertyChanged)
```

service_signals_sessions.py

```
# https://knowledge.autodesk.com/support/vred-products/learn-
explore/caas/CloudHelp/cloudhelp/2021/ENU/VRED/files/Python-
Documentation/Python-Examples/VRED-Python-Documentation-Python-Examples-
Visualize-user-positions-html-html.html
# This scripts shows an overview of user positions in a collaboration
sesion
class UserMap(vrAEBase):

    def __init__(self):
        vrAEBase.__init__(self)
        self.spheres = {}
        self.addLoop()
        # callback sesson start/stop
        vrSessionService.sessionJoined.connect(self.started)
        vrSessionService.sessionLeft.connect(self.ended)
        # callback user joins/leaves session
         vrSessionService.userArrives.connect(self.userArrived)
         vrSessionService.userLeaves.connect(self.userLeaves)

    def loop(self):
        # this is my local camera position
        myPos =
getTransformNodeTranslation(vrSessionService.getUser().getHeadNode(),True)
        for user in vrSessionService.getRemoteUsers():
```

```
            sphere = self.spheres[user.getUserId()]
            # this is the users head transformation node
            pos = getTransformNodeTranslation(user.getHeadNode(),True)
            # move indicator for user position
            setTransformNodeTranslation(sphere,(pos.x()-
myPos.x())/100,(pos.y()-myPos.y())/100,-500,False)

    def started(self):
        self.group = createNode("Group", "UserMap",
vrCameraService.getActiveCamera())
        self.plane = createCylinder(2, 100, 50, True, True, True, .0, .1,
.0)
        self.setTransparent(self.plane)
        addChilds(self.group,[self.plane])
        color = vrSessionService.getUser().getUserColor()
        sphere = createSphere(3, 2, color.redF(), color.greenF(),
color.blueF())
        addChilds(self.group,[sphere])
        setTransformNodeTranslation(sphere,0,0,-500,False)
        setTransformNodeRotation(self.plane, 90, 0, 0)
        setTransformNodeTranslation(self.plane, 0, 0, -500, False)
        self.setActive(True)

    def ended(self):
        subChilds(self.group,[self.plane])
        subChilds(vrCameraService.getActiveCamera(),[self.group])
        self.setActive(False)

    def userArrived(self,user):
        color = user.getUserColor()
        sphere = createSphere(2, 2, color.redF(), color.greenF(),
color.blueF())
        addChilds(self.group,[sphere])
        self.spheres[user.getUserId()] = sphere

    def userLeaves(self,user):
        sphere = self.spheres[user.getUserId()]
        subChilds(self.group,[sphere])

    def setTransparent(self,node):
        node.getMaterial().fields().setVec3f("seeThrough",.95,.95,.95)
```

```
map = UserMap()
```

vr_message_service.py

```
# Utilizing `vrMessageService`
def receivedMessage(message_id, args):
    messages = (
        'VRED_MSG_ARGV',
        'VRED_MSG_CHANGED_CAMERA_UP',
        'VRED_MSG_CHANGED_MATERIAL',
        'VRED_MSG_CHANGED_PB_PARAMETERS',
        'VRED_MSG_CHANGED_SCENEGRAPH',
        'VRED_MSG_CONVERT_OSF_FILE',
        'VRED_MSG_DESELECTED_NODE',
        'VRED_MSG_EXPORTED_FILE',
        'VRED_MSG_IDLE',
        'VRED_MSG_IMPORTED_FILE',
        'VRED_MSG_INIT',
        'VRED_MSG_KEY_PRESSED',
        'VRED_MSG_KEY_RELEASED',
        'VRED_MSG_LOADED_GEOMETRY',
        'VRED_MSG_LOOP',
        'VRED_MSG_NEW_SCENE',
        'VRED_MSG_NONE',
        'VRED_MSG_PRENEW_SCENE',
        'VRED_MSG_PRE_QUIT',
        'VRED_MSG_PROJECT',
        'VRED_MSG_PROJECT_LOADED',
        'VRED_MSG_PROJECT_MERGED',
        'VRED_MSG_SAVED_GEOMETRY',
        'VRED_MSG_SELECTED_CAMERA',
        'VRED_MSG_SELECTED_LIGHT',
        'VRED_MSG_SELECTED_MATERIAL',
        'VRED_MSG_SELECTED_NODE',
        'VRED_MSG_SWITCH_MATERIAL_CHANGED',
        'VRED_MSG_UPDATE_UI',
        'VRED_MSG_USER',
        )
```

```
    # Print the message that was signaled
    for message in messages:
        if message_id == getattr(vrController, message):
            print(message)

    # Listen specifically to the SELECTED CAMERA message
    if message_id == vrController.VRED_MSG_SELECTED_CAMERA:
        print("Camera selected!")

`vrMessageService`.message.connect(receivedMessage)
```

Video Captions

Hello and welcome to this Python scripting tutorial for VRED Pro. My name is Christopher and today I will show you how you can use Qt Signals to react to events in VRED.

So, what are Qt Signals? Qt Signals are implemented in the Qt Framework that is also used by VRED. Signals are a way to communicate between different parts of the software, so, for example, a user interface can communicate with the logic layer of a program and exchange information. In VRED, the same method is used to signal to a Python script if any data has changed – or more generally - if an event has occurred that a user might want to react to. When writing a script, you can connect to signals of VRED services and can perform an action if that signal is triggered.

What are common uses cases for this? With the API version 2, Autodesk introduced lots of signals that can be used for all kinds of things. There are, for example, signals in the annotation service that notify the script when an annotation was created or deleted. There are signals that tell you when nodes are added or removed from the Scene Graph. This can be helpful when you automatically want to rebuild the find cache when the Scene Graph has changed. There are even signals that tell you when all lights have been validated.

The VR Session uses signals to tell when users join or leave a session. So, one could show that information in a custom user interface or show connected VR users in the 3D space. This example is from the official Autodesk examples and shows how they use signals to update the overview map for a VR Session and place the users in the 3D space.

Let's see how we can use signals in our own scripts. Signals are a part of the API version 2 services, so you can scan the documentation to find any services that offer signals. For example, the `vrAnnotationService` documentation has a subsection where the signals are listed with their parameters.

In this example, I'm connecting to these signals in the annotation service. At first, we create the callback functions for each signal. These are the functions that are called whenever this signal is

triggered. In this case, we have callback functions for when an annotation is created, added, deleted and when the visibility of annotations changed.

The documentation tells us what data is passed by the signal. For example, the `createAnnotation` signal passes the annotation that was created, and the visibility parameter tells us whether the annotations are set to visible or not. In the next lines, we connect to these signals using the connect function of the signal, and we use the name of the callback function as the parameter. When we execute this code and add an annotation, the signal should be triggered and print a message in the terminal. Which it does.

Next, I want to show you how you can listen to changes in the Scene Graph to automatically trigger a rebuild of the find cache. The find cache is used to speed up find operations of nodes and can speed up the performance, if you have scripts that rely on dynamically finding nodes in the Scene Graph. In this example, we create a callback function "nodes changed" with the input parameter "nodes" that is a list of all the nodes that have changed.

In this function, we first clear the cache and then rebuild it. In the next lines, we connect to the signals that tell us if any nodes have been added to or have been removed from the Scene Graph. Namely, the "nodes added" and "nodes removed" signals. And now, when we execute the code and change nodes in the Scene Graph, it will always automatically rebuild the find cache for us.

Apart from the signals in the services, there are other useful signals in VRED that can be used to listen to system messages. For example, you can react when a new scene was created or before a scene is saved. In the last example, we connect to the so called `vrMessageService` to listen to all kinds of system messages. A full list of these messages can be found in the documentation of the `vrController` module.

The `vrMessageService` is sending a code each time a system event happened, which translate to one of these system messages. In this code example, whenever the `vrMessageService` triggers a signal, the corresponding message code is printed in the terminal. For example, if we select a camera the "SELECTED_CAMERA" message is triggered.

Okay! That's it for today! I hope you are now ready to use Qt signals in VRED. Thanks for joining me and see you next time!