Tutorial 9: How to Use the New Data Types in V2

The *Tutorial 9: How to Use the New Data Types in V2* video covers examples of math operations with QMatrix4x4 and QVector3D.

Download the example scripts

Jump to script examples

Sample Python Code

Here are the accompanying example Python scripts for the *Tutorial 9: How to Use the New Data Types in V2* video.

math_operations.py

```
# Calculating with QtVectors
camera = vrCameraService.getActiveCamera()
cameraTranslation = camera.getTranslation()

print("Distance to origin: ", cameraTranslation.length())
print("Distance to origin (point): ",
cameraTranslation.distanceToPoint(QVector3D(0,0,0)))

distanceToZAxis = cameraTranslation.distanceToLine(QVector3D(0,0,0),
QVector3D(0,0,1))
print("Distance to z-axis", distanceToZAxis)

# Example 2.0
# Using vrMathService
boxPtr = vrNodeUtils.createBox(1000, 1000, 1000, 1, 1, 1, 1, 1, 1)
boxPtr.setName("Box")
boxPtr.setTranslation(100, 200, 300)
boxPtr.setRotation(23, 34, 45)
boxPtr.setScale(1.2, 1.3, 1.3)

box = vrNodeService.findNode("Box")
transform = box.getWorldTransform()
```

```
translation = vrMathService.getTranslation(transform)
rotation = vrMathService.getRotation(transform)
scale = vrMathService.getScaleFactor(transform)
scaleOrientation = vrMathService.getScaleOrientation(transform)

print("Translation", translation)
print("Rotation", rotation.toEulerAngles())
print("Scale", scale)
print("Scale Orientation", scaleOrientation.toEulerAngles())
```

new_data_types.py

```
# Example 1
# QColor:
colorRed = QColor(255, 0, 0)
colorBlue = QColor(0, 255, 0)
colorGreen = QColor(0, 0, 255)

fontcolor = QColor.fromRgb(127, 24, 78)
backgroundColor = QColor.fromHsv(180, 255, 255)

annotation = vrAnnotationService.createAnnotation("New Annotation")
annotation.setFontColor(fontcolor)
annotation.setBackgroundColor(backgroundColor)
annotation.setText("I like it!")

x = 0
y = 0
z = 300
position = QVector3D(x, y, z)
matAnnotation = vrAnnotationService.findAnnotation("MaterialAnnotation")
matAnnotation.setPosition(position)

# Example 2:
# QVector4D, QVector3D, QVector2D

# QVector4D
camera = vrCameraService.getActiveCamera()
viewFrustum = camera.getFrustum()
print("Frustum as QVector4D:", viewFrustum)
```

```python
# QVector3D
camera = vrCameraService.getActiveCamera()
fromAtUp = vrCameraService.getActiveCamera().getFromAtUp()
fromVector = fromAtUp.getFrom()
atVector = fromAtUp.getAt()
upVector = fromAtUp.getUp()

print("From Vector:", fromVector)
print("At Vector:", atVector)
print("Up Vector:", upVector)

cameraTangent = QVector3D.crossProduct((atVector - fromVector), upVector)
cameraTangent.normalize()
print("Cross product", cameraTangent)

x = 0
y = 0
z = -300
position = QVector3D(x, y, z)
matAnnotation = vrAnnotationService.findAnnotation("MaterialAnnotation")
matAnnotation.setPosition(position)

# QVector2D
camera = vrCameraService.getActiveCamera()
camera.setSensorSize(QVector2D(10, 10))
camera.updateFromPerspectiveMatch()

#Example 3:
# QMatrix4x4
camera = vrCameraService.getActiveCamera()
projectionMat = camera.getCustomProjectionMatrix()
print(projectionMat)

# Example:
# Math operations
```

old_data_types.py

```
# Example 1
# Color
blue = 127
red = 255
green = 127

vrRenderSettings.setRenderBackgroundColor(blue, red, green)
renderBackground = vrRenderSettings.getRenderBackgroundColor()
print(type(renderBackground), renderBackground.x(), renderBackground.y(),
renderBackground.z())

# Example 2
# Vectors
camera = vrCamera.getActiveCameraNode()
cameraPosition = camera.getTranslation()
print(type(cameraPosition), cameraPosition)

# Example 3
# Math operations on vectors
v1 = vr
```

Video Captions

Hello and welcome to our final video in our Python scripting tutorials for VRED Pro. My name is Christopher and today I will introduce the new data types in the new API version 2, talk about their background, show you how you can work with them.

When we look at the Python API version 1, there aren't really any data types to work with. All parameters are either passed by their single values, or as a list, like the components of a vector or a color. Also, the functions return values, most of the time, are either a single value, a tuple, or a list.

From a Python point of view, this is ok. This is called Pythonic – so, the way this is done in Python. But, this way of working has its disadvantages. For example, when we are working with vectors, we always have to convert the Pythonic representation in a more convenient representation to perform mathematical operations.

The Python API version 2, therefore, introduces new data types that tackle the previous issues. It introduces an object-oriented way of working with data. That means that values like the components of a vector are combined in a single object rather than being treated individually. For example, input parameters and return values for geometrical functions that deal with vectors now use the new data types. These data types have been borrowed from the Qt Framework that VRED uses internally and each data type comes with exactly the functions you would expect from it.

For 3D graphics programming, we have the new types “Q Vector 2D”, “3D” and “4D”, and also a Matrix data type. We have a new color type that can create colors from hex values, RGB, or other color formats. And, we have some types that help us building a graphical user interface, like QIcon, QSize, or QImage.

When you want to work with these types, you do not have to import them by hand. VRED already added them for you and you can just start typing a “large Q” in the Script Editor, hit SPACE, while holding the control key, and you get a list of all the new data types you can use.

So, with that said. Let’s look at some operations we can do with the new data types. In the first example, we create a new annotation and set its color. For this, we can use the new “Q Color” type that allows us to create colors for example from RGB values, but also from HSV or other color spaces. In the next step, we can move the annotation a bit higher to actually place it onto the hood of the car. To do this we simply create a new 3D Vector and put it in the ”set position” method of our annotation.

For our next example, we want to calculate a vector that points to the right from our current camera. So, actually, we want the vector that is described by the cross-product of the camera viewing direction and the up vector. Calculating a ”cross-product” or “dot product” are basic operations when doing 3D programming, but with the new data types, it's finally really easy to do this.

In the script, we first get the up, from, and at values from the current camera that basically describes the up direction, the from direction from where the camera is looking, and the direction where the camera is looking at. With these values, we can basically calculate the cross-product in one line, by calculating a vector from the camera to the target and the up vector. In the last set, we just normalize the result and then we are done. This is the advantage of these new data types. We have all the functionality without having to import any external modules or implementing our own types.

Another example: We want to calculate the distance from the camera to the origin and the distance to the z-axis of the coordinate system. This also very easy because the vector classes already offer such methods. In this case, it's the method distance to point and distance to line we are using here. So, you see, there is really no need to use an external library or write your own vector implementation. The data types should contain most of the functions you will ever need when doing 3D programming.

There are so much more useful data types we could talk about, like the “Q Time” data type for measuring times or “Q Size” for representing areas. But the vector types are definitely the most useful ones. To learn about all the other types, I can really recommend the official Qt documentation. They also have some great examples here. Just remember to use the auto-completion feature of the Script Editor with CTRL SPACE that shows you available functions and types and use online search to find the documentation of the Qt types.

Okay. That’s it for today about all the new data types in VRED. I hope you enjoyed it! Thank you for joining me and see you next time