Overview

Lexocad uses the project PySide2 as a library to create graphical user interfaces (GUIs).

PySide2 is a Python binding to the Qt framework, one of the most powerful GUI libraries.

The provided copy of the software Qt Designer ("*designer.exe*") allows you to graphically compose your own dialogs. If you receive a message complaining about missing dependencies then you have to copy "*designer.exe*" to the Lexocad root folder, where it will find all required Qt libraries.

Quick Start

Below is a very simple script that shows a dialog and, when confirmed, creates a cube with the parameters coming from the dialog itself. The easiest way to test and get it working:

- Go*: Extra > Python > Python Scripts*. Click *Gui* and then *QtExample*.

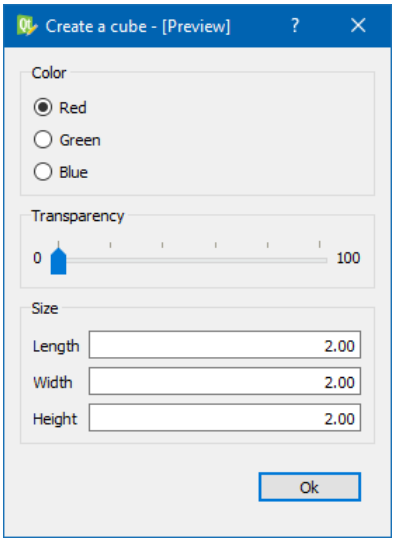
```
# Import Lexocad libraries
import Base
import OpenLxApp as lx
# Import the Qt libraries and the dialog
from PySide2.QtWidgets import *
from QtExample.Dialog import *


# Setup the dialog
class Dialog(QDialog, Ui_Dialog):
    def __init__(self, parent=None):
        super(Dialog, self).__init__(parent)
        self.setupUi(self)


if __name__ == '__main__':
    # Execute the dialog
    dialog = Dialog()
    result = dialog.exec_()

    if result == QDialog.Accepted:
        # Get the color from the dialog
        Color = Base.Color(255, 0, 0) if dialog.Red.isChecked() \
            else Base.Color(0, 255, 0) if dialog.Green.isChecked() \
            else Base.Color(0, 0, 255)

        # Get the transparency from the dialog
        Ratio = dialog.TransparencyRatio.value()

        # Get the measures from the dialog
        L = dialog.L.value()
        W = dialog.W.value()
        H = dialog.H.value()

        # Create the geometry in the document
        doc = lx.Application.getInstance().getActiveDocument()
        block = lx.Block.createIn(doc)

        # Assign the properties to the geometry, as set in the dialog
        block.setXLength(L)
        block.setYLength(W)
        block.setZLength(H)

        # Create the element in the document, assign the geometry to it
        elem = lx.Element.createIn(doc)
        elem.setGeometry(block)

        # Assign the properties to the element, as set in the dialog
        elem.setDiffuseColor(Color)
        elem.setTransparency(Ratio)

        # Recompute the document
        doc.recompute()
```

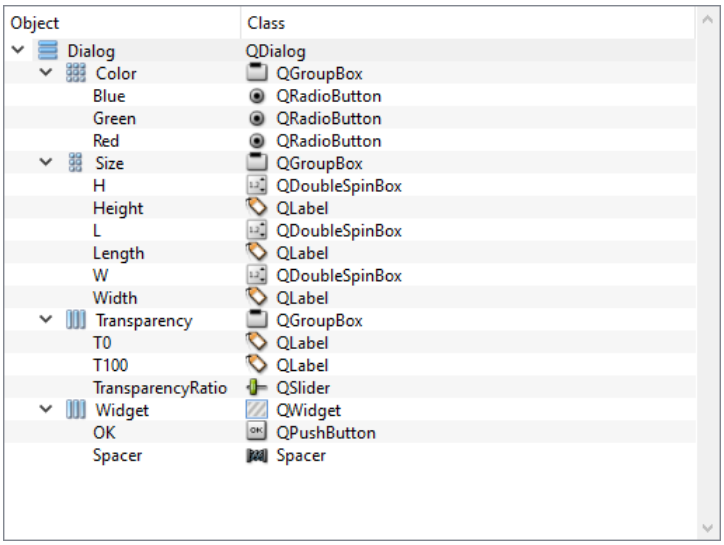
Qt Designer

With the software Qt Designer you can compose and customize your dialogs in a what-you-see-is-what-you-get (WYSIWYG) manner.

If you are new to Qt Designer you can follow the subsequent steps, summarized from the official Qt tutorial: http://doc.qt.io/qt-5/qtdesigner-manual.html

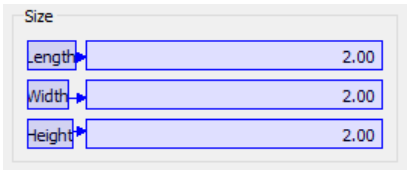
1. Widget Editing
   - Design your dialog the way you like using the drag and drop interface. Assign meaningful names especially to the widgets that you are going to interact with from your Python script: it will make your life easier.



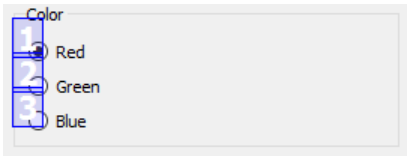


2. Signals and Slots Editing
   - Connect the widgets that interact with each other.

3. Buddy Editing
   - Link the labels to their buddy widgets: a buddy widget accepts the input focus on behalf of a label, when the user types the label's shortcut key combination.



4. Tab Order Editing
   - Set the “Tab order” of your widgets to allow users to navigate between widgets and controls using only the keyboard.



When you are done, remember to save you dialog as a “`.ui`” (user interface) file and take note of its path.

Converting your dialog into a true Python script

The "`.ui`" (user interface) file is not intelligible by Python and thus it needs to be converted into a script: Lexocad will do this task for you.

- Go: *Extra > Python > Python Console*. Copy the simple script below and paste it inside the interactive console, eventually press the [Enter] key.

```
import pyside2uic
pyside2uic.compileUiDir("D:/Scripts/QtExample")
```

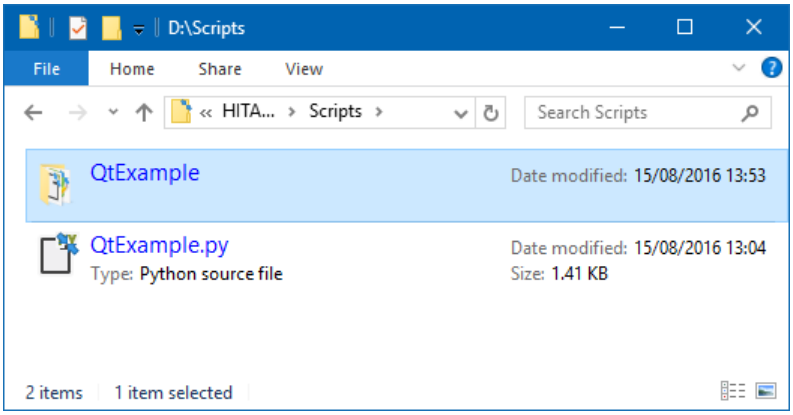
The above script searches inside the folder "`D:/Scripts/QtExample`" for "`.ui`" files and will convert them into Python scripts.

You are required to adjust the above path to your needs.

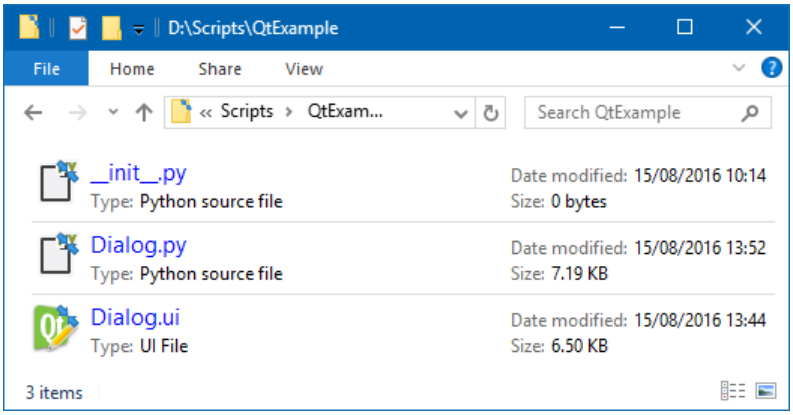
Organizing your files

To import the dialog into your main Python script you need to organize your files in an appropriate way.

- Create a folder with the same name of your main script (without extension) and enter it.



- Create an empty text file named "`__init__.py`".
- Put your dialog – with the "`.py`" extension – inside of this folder.
- Possibly put your dialog – with the "`.ui`" extension – here as well.



A closer look at the script

Let us examine the example a bit closer.

```
# Import Lexocad libraries
import Base
import OpenLxApp as lx
# Import the Qt libraries and the dialog
from PySide2.QtWidgets import *
from QtExample.Dialog import *
```

The import statements load the required Lexocad modules, Qt modules (“`QtWidgets`”) and the dialog that you have previously designed (“`QtExample`” is the folder that you have created previously, and “`Dialog`” is the converted file “`.ui → .py`”).

```
# Setup the dialog
class Dialog(QDialog, Ui_Dialog):
    def __init__(self, parent=None):
        super(Dialog, self).__init__(parent)
        self.setupUi(self)
```

The dialog is assigned to a `class` and constructed with just a few lines of code.

Note that the name “`Ui_Dialog`” may change, depending on how you named your dialog. To be sure, open the converted file “`.ui → .py`” and look for the `class` name close to the beginning of it.

```
if __name__ == '__main__':
    # Execute the dialog
    dialog = Dialog()
    result = dialog.exec_()
```

The dialog is “executed”, i.e. it is shown and the script stops until the dialog is closed (confirmed/cancelled).

```
    if result == QDialog.Accepted:
        # Get the color from the dialog
        Color = Base.Color(255, 0, 0) if dialog.Red.isChecked() \
            else Base.Color(0, 255, 0) if dialog.Green.isChecked() \
            else Base.Color(0, 0, 255)

        # Get the transparency from the dialog
        Ratio = dialog.TransparencyRatio.value()

        # Get the measures from the dialog
        L = dialog.L.value()
        W = dialog.W.value()
        H = dialog.H.value()
```

If the dialog is “accepted” then the values of the widgets are taken and assigned to distinct variables.

```
        # Create the geometry in the document
        doc = lx.Application.getInstance().getActiveDocument()
        block = lx.Block.createIn(doc)
        …
```

The rest of the code is related to Lexocad.

For more information see the documentation in “`Quickstart (Python).pfd`” or take a look at the provided examples.

Qt Designer license

https://doc.qt.io/qt-5/designer-license-information.html