Adding New Pages to QGC

How can we add our own .qml file to QGC? What can we do after adding it? We will work on it. Now let's go to where it all started.. The world was a cloud of dust.. Nope! I don't think you need to go that far. Ehm, yes, let's focus on our work.

Our base page where interface elements are defined in QGC is **"MainRootWindow.qml"**. An ideal place to start customizing QGC. When you start or compile the program you will see a QGC Logo in the top left. It's like a hamburger menu. When you click on the logo, a popup appears that will direct you to different pages to **adjust the settings.** From here you can go to the relevant settings page. We will also add to these buttons. Maybe you want to change the interface hierarchy. You might want a separate page where you can create the C++ backend, add your own sensors and test it. Now let's play with these buttons a bit.

Now let's come to the codes, less words more code!
As you can see here, we have a **showTool** function created to open new pages in the mainwindow. The function gets 3 parameters; **title, our page and icon.**

```
function showTool(toolTitle, toolSource, toolIcon) {
       toolDrawer.backIcon     = flightView.visible ? "/qmlimages/PaperPlane.svg" :
"/qmlimages/Plan.svg"
       toolDrawer.toolTitle    = toolTitle
       toolDrawer.toolSource   = toolSource
       toolDrawer.toolIcon     = toolIcon
       toolDrawer.visible      = true
   }
```

Just below this function, 3 more are defined.

```
 function showAnalyzeTool() {
       showTool(qsTr("Analyze Tools"), "AnalyzeView.qml", "/qmlimages/Analyze.svg")
   }

   function showSetupTool() {
       showTool(qsTr("Vehicle Setup"), "SetupView.qml", "/qmlimages/Gears.svg")
   }

   function showSettingsTool() {
       showTool(qsTr("Application Settings"), "AppSettings.qml", "/res/QGCLogoWhite")
    }
```

These functions trigger the **showTool function to open pages.** I heard a question like this; so where do we call these functions? Before answering that, let's create our own function. Maybe there's a happy little function around here... Yes, when I was a kid, I looked forward to watching Bob Ross.

```
 function showCustomPage(){
        showTool(qsTr("Custom Page"), "AddNewPage.qml",  "/res/QGCLogoWhite")
```

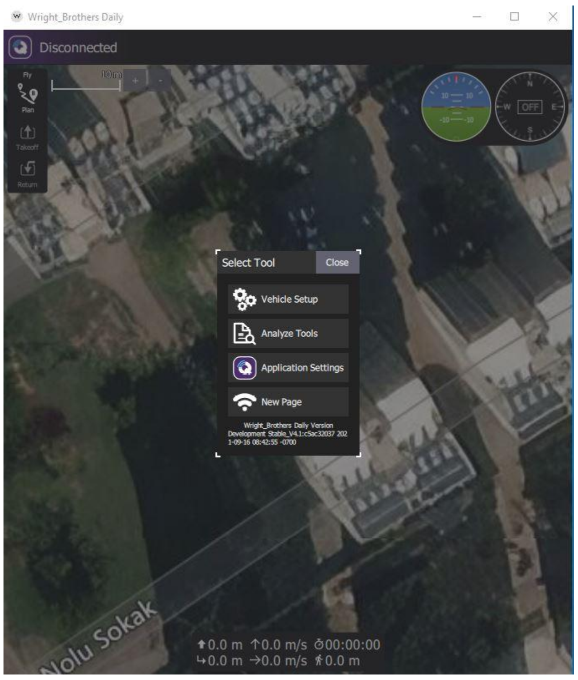
```
   }
```

Here we have defined a **name, filename and an icon** for our own .qml page. You can add the icon using the icon set in QGC or add it yourself. When I want to add custom icons, I download free icon sets and edit them in InkSpace or Adobe Illustrate.

After we create our function, let's see where to call it. In MainRootWindow.qml we find **toolSelectDialogComponent** and add a new SubMenuButton.

```
 Component {
       id: toolSelectDialogComponent
          .
          .
          .
 SubMenuButton {
                      id:                   newpagebutton
                      height:               _toolButtonHeight
                      Layout.fillWidth:     true
                      text:                 qsTr("New Page")
                      imageResource:        "qrc:/qmlimages/wifi.svg"
                      imageColor:           "transparent"
                      visible:
!QGroundControl.corePlugin.options.combineSettingsAndSetup
                      onClicked: {
                          if (!mainWindow.preventViewSwitch()) {
                              toolSelectDialog.hideDialog()
                              mainWindow.showCustomPage()
                          }
                      }
                  }
```

And this is the final version of our menu. Now let's move on to the codes we wrote on the page we called.



Wright_Brothers Daily Version
Development Stable_V4.1:c5ac32037 2021-09-16 08:42:55 -0700

0.0 m 0.0 m/s 00:00:00
0.0 m 0.0 m/s 0.0 m

The order will be as follows, first we will put a page for the background, then two buttons and add a feature to them. These parts are pretty easy.

I created a gradient rectangle called **MyCustomView.qml.**

```
Item {
       Rectangle{
       id:mybackground
        anchors.fill: parent
        gradient: Gradient{
            GradientStop {
                position: 0
                color: "#e0c3fc"
              }
            GradientStop {
                position: 1
                color: "#8ec5fc"
              }
          }
       }
    }
```

Then I created another file named **CustomButton.qml.**

```
import QtQuick 2.0
Item {
   width: 200
   height: 200
   property alias myCustomButtonColor: myCustomBtn.color
   property alias contentText: content.text

   Rectangle{
   id: myCustomBtn
   color: "red"
   radius: width*0.5
   anchors.fill: parent
       Text {
           id: content
           text: qsTr("Text")
           font.bold: true
           anchors.horizontalCenter: parent.horizontalCenter
           anchors.verticalCenter: parent.verticalCenter
           font.pixelSize: 25
           font.family: "Garamond"
       }
   }
}
```

Here I have assigned property aliases to access the properties of the Rectangle. And I set the properties of the rectangle and put a text in it.

Then add two more files named **Sender.qml** and **Receiver.qml.** I called my CustomButton object.

Receiver.qml

```
CustomButton{
id: receiver
   function receive(val){
   contentText=val
   }

   SequentialAnimation on myCustomButtonColor{
   id:btnnotice
   running: false
       ColorAnimation {
           from: "#FFFADE"
           to: "#AB7AE2"
           duration: 200
       }

       ColorAnimation {
           from: "#AB7AE2"
           to: "#FFFADE"
           duration: 200
       }
    }
}
```

Sender.qml

```
CustomButton{
id: sender
property int counter: 0
signal send(string value)
property Receiver target: null

       onTargetChanged: {
       send.connect(target.receive);
       }
           MouseArea{
           anchors.fill: parent
           onClicked: {
           sender.counter++;
               sender.send(counter);
           }
           onPressed: {
           sender.myCustomButtonColor="#FEB5A1"
           }
           onReleased: {
           sender.myCustomButtonColor="#C4806E"
           }
       }
}
```

Here we created a signal to Sender and decided what would happen when clicked in MouseArea. To access Receiver in click method we defined **"property Receiver target: null"** and as we click, the number in the Receiver increases one by one!

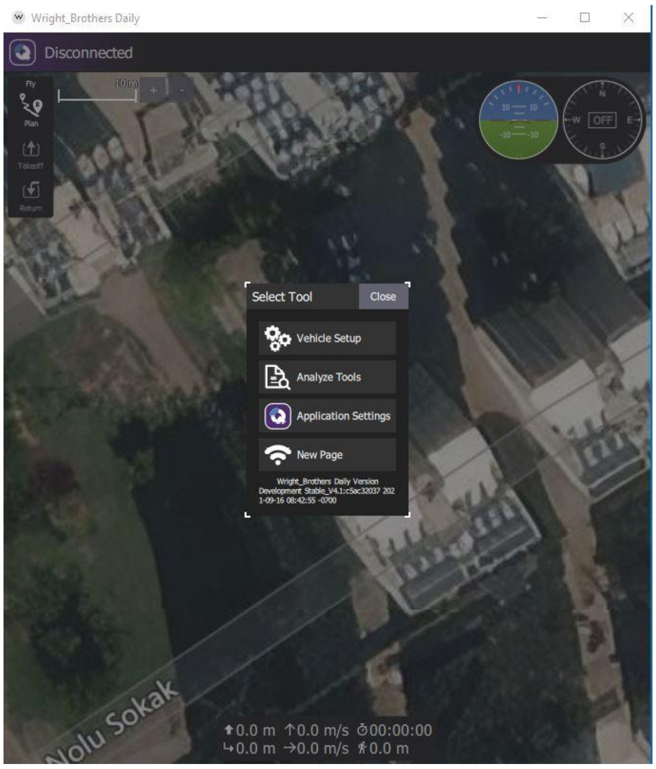
In addition, we decided what the colors would be with **“pressed”** and **“released”**. So how do we access these colors with **"myCustomButtonColor"?** As I mentioned before; while creating CustomButton.qml, we used **"property alias"** to access the properties of this object from another page.

Finally, we create our file named Add New Page.qml, which we call with mainWindow.showCustomPage() and let the show begin!

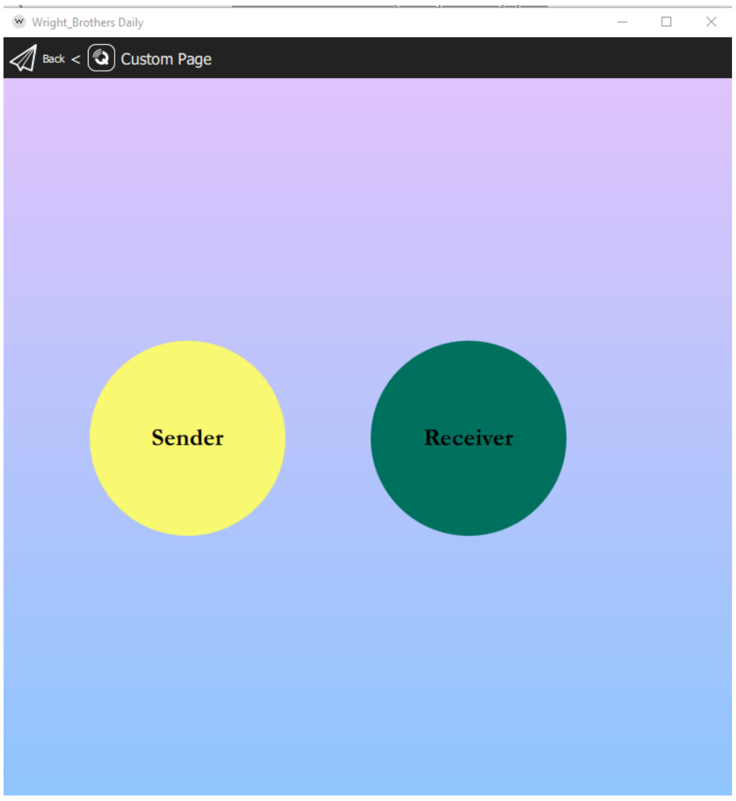
```
import QtQuick 2.0
import QtQuick.Controls 2.0
import QtQuick.Layouts 1.12
import QGroundControl 1.0
import QGroundControl.Palette 1.0
import QGroundControl.Controls 1.0
import QGroundControl.ScreenTools 1.0
   Rectangle{
   id: mypage
   anchors.fill: parent
   z: QGroundControl.zOrderTopMost
           MyCustomView{
           id:addbackground
           anchors.fill: parent
               Sender{
               id: sender
               target: receiver
               x:88
               y:140
               myCustomButtonColor: "#F9F871"
               anchors.verticalCenter: parent.verticalCenter
               contentText: "Sender"
                 }
               Receiver{
                   id: receiver
                   x:375
                   y:140
                   myCustomButtonColor: "#00705E"
                   anchors.verticalCenter: parent.verticalCenter
                   contentText: "Receiver"
               }
           }
   }
```

Here we define the positions of Receiver and Sender by specifying X and Y properties. We did this to be quick. However, if you want a developable and readable code structure, you should use layout.

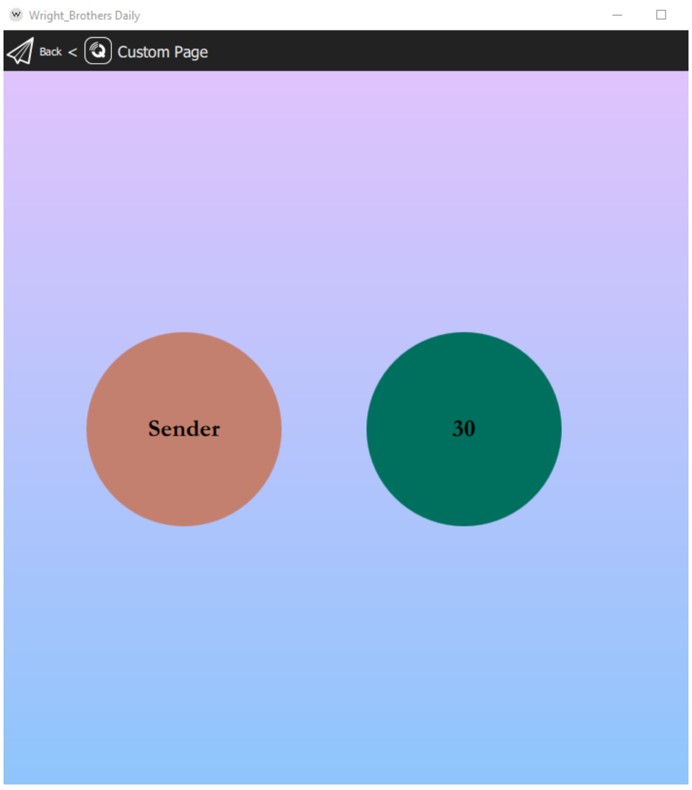
We could also do this by adding fewer pages. However, we tried this method to practice creating objects and adding pages. Is it wise to do it with fewer pages or do it this way? What will happen when conditions change and codes multiply? To understand this, maybe we will also look at how to test performance analysis with QML Profiler.



Wright_Brothers Daily
Disconnected
Fly
Plan
Takeoff
Return
Select Tool
Close
Vehicle Setup
Analyze Tools
Application Settings
New Page
Wright_Brothers Daily Version Development Stable_V4.1:c5ac32037 2021-09-16 08:42:55 -0700
Nolu Sokak
0.0 m 0.0 m/s 00:00:00
0.0 m 0.0 m/s 0.0 m



Wright_Brothers Daily
Back < Custom Page
Sender
Receiver



Wright_Brothers Daily
Back < Custom Page
Sender
30