Nebraska Intelligent MoBile Unmanned Systems Lab

Beginning C++ GUI Development in ROS Edit

This entry was posted in Educational Material * Procedures * *on February 27, 2017 by Adam Plowcha.*

Disclaimer: Part 1 – This tutorial is designed to get you started with developing a GUI for your ROS project using C++ and Qt. If you'd like to use Python, you should check out the official ROS tutorial here. Also, the assumption is that you have a basic understanding of creating/building a ROS application in C++ (to include editing the CMakeLists.txt and package.xml files) and know how pull a package from the Nimbus SVN.

Part 2 – This is not the only way to develop a Qt GUI in ROS. It is a very iterative approach that should help you understand how ROS uses rqt plugins, at least at a basic level. Also, it allows you to make sure that you've got working code at certain points before moving on to the next phase of development.

Part 3 – This is not a tutorial on the ins and outs of Qt Creator, but it should at least get you to the point where you can get the IDE up and running.

Part 4 – The below has been tested with both Ubuntu 14/ROS Indigo and Ubuntu 16/ROS Kinetic.

==========

There are two ways to configure your project. You can use the "manual" method, or you can use the "automatic" method provided by way of the create_gui.py script included in the ros_gui_template project located in the Nimbus SVN. Both methods are described

below.

Manual Method

==========

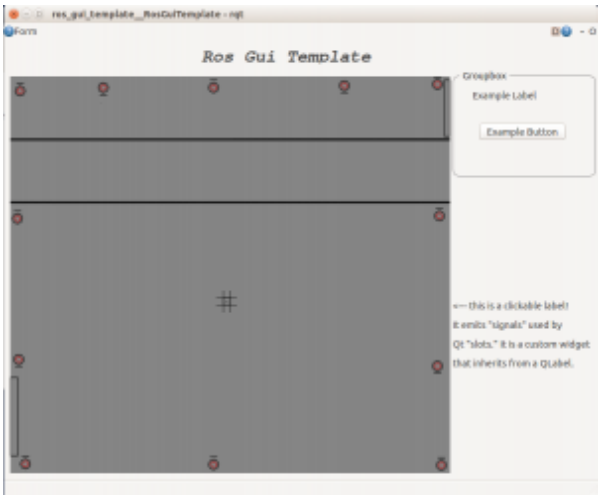
Step 1: Get the ros_gui_template package from the Nimbus SVN. It's located in var/www/svn/UAV/ros-workspaces/ros_gui_template. You can check it out directly into whatever project you're working on, or you can put it in it's own ROS workspace for later use. Remember, the files in the ros_gui_template project must be somewhere under the ***src*** directory of your catkin workspace – either directly or within a subdirectory.

Step 2: Put the ros_gui_template folder in the ***src*** folder of whatever package you want to add a gui to. For instance, you could put it inside of ~/nimbus_asctec /src/nimbus_asctec/

Step 3: Now go back to ~/nimbus_asctec (or whatever your root catkin workspace is) and run `catkin_make`. If you've already built you project once, then you'll probably just see it build the ros_gui_template. Once that's done, you should be able to run the following:

```
roslaunch ros_gui_template ros_gui_template.launch
```

You should eventually see the following:



If you do, then success! Now the fun begins.

Step 4: Obviously, you want your code in there with your file names and so on. That means you'll need to make some changes to the files in your project. Basically, you're going to replace any instances of "ros_gui_template" and "RosGuiTemplate" with your own project name. For tutorial purposes, we'll use "your_project" and "YourProject."

Start with the folder that you just placed inside you project. Rename it to "your_project." Now go into the base directory of "your_project" (where you see CMakeLists.txt, package.xml, plugin.xml, and so on) and make the the following changes to the following files:

CMakeLists.txt

==========

-Find and replace all instances of “ros_gui_template” with “your_project”

package.xml

==========

-Find and replace all instances of “ros_gui_template” with “your_project”
-Change the other tags as/if required. Remember, ROS stuff will probably change here as your project progresses.

plugin.xml

==========

-Change path=”lib/libros_gui_template” to path=”lib/libyour_project”
-Change class name=”ros_gui_template/RosGuiTemplate” to class name=”your_project/YourProject”
-Change type=”ros_gui_template::RosGuiTemplate” to type=”your_project::YourProject”
-Change the other stuff as/if needed

Rename ros_gui_template.launch to whatever-you-want.launch

whatever-you-want.launch

==========

-Find and replace all instances of “ros_gui_template” with “your_project”

*In the **include** folder, rename “ros_gui_template.h” to “your_project.h”*

your_project.h

==========

-Change the #ifndef and #define lines as you see fit
-Find and replace all instances of “ros_gui_template” with “your_project”
-Find and replace all instances of “RosGuiTemplate” with “YourProject”

*In the **src** folder, rename “ros_gui_template.cpp” to “your_project.cpp”*

your_project.cpp

==========

-Find and replace all instances of “ros_gui_template” with “your_project”
-Find and replace all instances of “RosGuiTemplate” with “YourProject”

*In the **resource** folder, rename “ros_gui_template.ui” to “your_project.ui”*

your_project.ui

==========

-Find and replace all instances of “RosGuiTemplate” with “YourProject”

*In the **scripts** folder, rename “ros_gui_template.py” to “your_project.py” (make sure this file stays executable)*

your_project.py

==========

-Find and replace all instances of “ros_gui_template” with “your_project”

Step 5: ROS has a tendency to cache certain things, and sometimes this can pose a problem, especially after you change file/folder names within a project. In order to ensure you’ll be able to run your new GUI, you’ll have to clear your rqt_gui cache with

the following command: `rm ~/.config/ros.org/rqt_gui.ini`

Step 6: Now go back to the root of your workspace and run `catkin_make`. Hopefully, everything will build and you'll be able to test your project with `roslaunch your_project whatever-you-want.launch`

At this point you should see the exact same window as before.

Step 7: After you get that working, you'll be ready to use Qt Creator to get in there and edit your UI and whatever else you need to do. If you don't have Qt Creator, you can get it with `sudo apt-get install qtcreator`. ***Remember, you need to source your devel/setup.bash and THEN launch qtcreator from that same command line!*** From there, you open a project and choose the CMakeLists.txt file in your_project's directory (where package.xml is and so on). You will be then asked to configure your project by picking a default build directory. You can safely accept the default presented as you will not be using Qt Creator to build your project – see the warning below.

WARNING

****

Qt Creator is an excellent IDE that you can use to edit your ROS C++ code as well as graphically create/edit your user interface; HOWEVER, it uses the QMake utility to build and manage its projects. QMake does not play well with ROS and catkin_make. Therefore, DO NOT USE Qt Creator's "build" button on your project. ALWAYS use catkin_make from the command line!

****

Step 8: If you're not a Qt expert, there are plenty of awesome tutorials on youtube that will take you through the basics of creating buttons, assigning functions to those buttons and so on. It's a fairly simply process, but it's not quite as easy as Visual Studio development. VoidRealms has an excellent set of 107 videos on YouTube that can help you gain some understanding of Qt and Qt programming. Be advised, it's only about Qt and C++ – you'll have to bring your own understanding of ROS to bear as needed.

Automatic Method

==========

Step 1: Get the ros_gui_template package from the Nimbus SVN. It's located in var/www/svn/UAV/ros-workspaces/ros_gui_template. You can check it out directly into whatever project you're working on, or you can put it in it's own ROS workspace for later use. Remember, the files in the ros_gui_template project must be somewhere under the ***src*** directory of your catkin workspace – either directly or within a subdirectory.

Step 2: Navigate to the directory where you placed the ros_gui_template files. You should see CMakeLists.txt, package.xml, create_gui.py, and others.

Step 3: Run the create_gui.py script with two arguments. The first argument is the

name of your package, and the second is the name of the class that defines your gui. Typically, these are an underscored version and a CamelCased version of the same word as specified in the ROS C++ Style Guide. See the below example.

```
./create_gui.py your_project YourProject
```

You'll see some output as the script goes through all the files that were downloaded with the ros_gui_template and makes all the needed changes as described in the "manual" section of this tutorial.

The final thing the script does is to remove the SVN info from the downloaded files so that you won't inadvertently change the repository master.

Step 4: Now go back to the root of your workspace and run `catkin_make`. Hopefully, everything will build and you'll be able to test your project with `roslaunch your_project your_project.launch`

At this point you should see a gui that looks like the one shown in Step 3 of the "manual" section of this tutorial.

Step 5: After you get that working, you'll be ready to use Qt Creator to get in there and edit your UI and whatever else you need to do. If you don't have Qt Creator, you can get it with `sudo apt-get install qtcreator`. ***Remember, you need to source your devel/setup.bash and THEN launch qtcreator from that same command line!*** From there, you open a project and choose the CMakeLists.txt file in your_project's directory (where package.xml is and so on). You will be then asked to configure your project by picking a default build directory. You can safely accept the default presented as you will not be using Qt Creator to build your project – see the warning below.

WARNING

****

Qt Creator is an excellent IDE that you can use to edit your ROS C++ code as well as graphically create/edit your user interface; HOWEVER, it uses the QMake utility to build and manage its projects. QMake does not play well with ROS and catkin_make. Therefore, DO NOT USE Qt Creator's "build" button on your project. ALWAYS use catkin_make from the command line!

****

Step 6: If you're not a Qt expert, there are plenty of awesome tutorials on youtube that will take you through the basics of creating buttons, assigning functions to those buttons and so on. It's a fairly simply process, but it's not quite as easy as Visual Studio development. VoidRealms has an excellent set of 107 videos on YouTube that can help you gain some understanding of Qt and Qt programming. Be advised, it's only about Qt and C++ – you'll have to bring your own understanding of ROS to bear as needed.

If you have comments or questions, don't hesitate to let me know.

–Adam

Leave a Comment

Logged in as Adam Plowcha. Log out?

Comment

Post Comment

Post navigation

← Getting started with nimbus_asctec

Search

Search for:

Search

Archives

Archives

Select Month

Members

- Site Admin
- Log out
- Entries RSS
- Comments RSS
- WordPress.org

Back to top