### QT GUI Basics Walkthrough

This walks through setting up a basic GUI that looks like:

![GUI Layout](image)

1) Make a new Chemeketa Basic Project. Open the .pro file and add this line to it (doesn't matter where you add the line as long as you don't break up a line that ends with \ and the line after it).

```cpp
QT += widgets
```

2) Create the application by including QApplication...

```cpp
#include <QApplication>
```

Then modify main so it takes in command line arguments and passes them to a QApplication and executes it. The rest of your code will go between those two lines.

```cpp
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    //REST OF CODE
    return app.exec();
}
```

3) The GUI is best divided into two horizontal rows contained within an invisible box (the QDialog class is a good standard widget for a simple box that will be used as a window).

We'll start by making the top row which consists of a QLabel and a QLineEdit. The row itself will be a QHBoxLayout.

```cpp
//Make a Dialog widget to be top container
QDialog* window = new QDialog;

//Make the top row
QHBoxLayout* hBoxTopRow = new QHBoxLayout;

//Make the two widgets for top row and add to that row
QLabel* lblName = new QLabel("Enter your name: ");
hBoxTopRow->addWidget(lblName);
QLineEdit* lneName = new QLineEdit(" ");
hBoxTopRow->addWidget(lneName);

//Tell the top level window to use that hBox as its layout
window->setLayout(hBoxTopRow);
window->show();
```

4) Add code similar to the following (note you will need to include each Q entity as a separate include... #include <QDialog>, #include <QVBoxLayout>, etc...):

```cpp
    //Make a Dialog widget to be top container
    QDialog* window = new QDialog;

    //Make the top row
    QHBoxLayout* hBoxTopRow = new QHBoxLayout;

    //Make the two widgets for top row and add to that row
    QLabel* lblName = new QLabel("Enter your name: ");
    hBoxTopRow->addWidget(lblName);
    QLineEdit* lneName = new QLineEdit(" ");
    hBoxTopRow->addWidget(lneName);

    //Tell the top level window to use that hBox as its layout
    window->setLayout(hBoxTopRow);
    window->show();
```

5) Run the program. Try resizing the window... which widget(s) expand to fill the space?

Change the order in which you add the two widgets to the hBoxTopRow. What happens to the GUI when you rebuild and run?
6) Time to build the bottom row... delete the last two lines from step 4 (the window-> ones) and replace them with:

```cpp
//Make a hBox to hold bottom row of buttons
QHBoxLayout* hBoxBottomRow = new QHBoxLayout;

QPushButton* btnEnter = new QPushButton("Enter");
hBoxBottomRow->addWidget(btnEnter);

QPushButton* btnCancel = new QPushButton("Cancel");
hBoxBottomRow->addWidget(btnCancel);

//Tell the top level window to use new hBox as layout
window->setLayout(hBoxBottomRow);
window->show();
```

7) Build and run... you should only see the bottom row now. Even though we still have code to make hBoxTopRow, it is not being added to the layout. To get both rows in, we need to make a vertical box to stack them in. Again, delete the last two lines (the window-> ones) and add:

```cpp
//Make a vertical box and add both rows to it
QVBoxLayout* mainLayout = new QVBoxLayout;
mainLayout->addLayout(hBoxTopRow);
mainLayout->addLayout(hBoxBottomRow);

//Tell the top level window to use the vBox as its layout
window->setLayout(mainLayout);
window->show();
```

8) Build and run. Pretty slick. Try resizing the window... yeah, yuck. We don't really want those buttons stretching like that. We need to give that bottom row something that will stretch instead of them. After you create the hBoxBottomRow but before you add the buttons to it, add this line:

```cpp
hBoxBottomRow->addStretch();
```

9) Try it out. Better? Try moving it to after adding the two buttons... how does that behave differently? Try adding a copy both before and after the buttons. How does it behave then? Try changing the before copy to:

```cpp
hBoxBottomRow->addStretch(2);
```

And the after to:

```cpp
hBoxBottomRow->addStretch(1);
```

That gives them proportional weight as they stretch to fill space. Finally, try putting this between the two addWidget calls that are adding the buttons:

```cpp
hBoxBottomRow->addSpacing(200);
```

You should get a large fixed amount of spacing.

Enough fun... probably want to rip out everything you added but the first addStretch
<table>
<thead>
<tr>
<th></th>
<th>Try centering the two rows by doing an addStretch to the mainLayout before and after you do the addLayouts with the two hBoxes. Build and run... looks nicer when you resize, but probably starts out with extra space above and below. To take care of that, do a resize right before showing the dialog:</th>
</tr>
</thead>
</table>
|   |     //Make it be minimum of 300x0...  
|   |     // will automatically grow from there to barely fit everything  
|   |     window->resize(300,0); |
|   | Try looking up some other widgets and creating a GUI using them:  
|   | http://qt-project.org/doc/qt-5.0/qtwidgets/gallery-windowsvista.html |