NHERI SIMCENTER PROGRAMMING BOOTCAMP

JULY 30 THROUGH AUGUST 3, 2018, AT UC BERKELEY'S RICHMOND FIELD STATION

GUI Development





OUTLINE

- GUI Design Fundamentals
- The Qt Framework
 - Common Data Types/Classes
 - Building the UI
 - Layout Management
 - Signals and Slots
 - Model View Controller Concept
 - Helper Widgets
- Quite a few Exercise Sessions

GUI FUNDAMENTALS

What is a WINDOW?



GUI FUNDAMENTALS

What is a WINDOW?

- "A rectangular area on your screen"
- "Any rectangular area on your screen"

MultiBase Calculator					
<u>F</u> ile					
Cls	Bck				
input >	bin	oct	dec	hex	
	d	e	f	mod	
2bin	a	b	c	1	
2oct	7	8	9	*	
2dec	4	5	6	-	
2hex	1	2	3	+	
Close	0	()	=	

GUI FUNDAMENTALS

Characteristics of an Application with a GUI

- Arbitrary sequence of execution
- May change shape/size
- May be (partially) covered
- Can be active or inactive

MultiBase Calculator									
<u>F</u> ile	<u>F</u> ile								
Cls	Bck								
input >	bin	oct	dec	hex					
	d	е	f	mod					
2bin	а	b	с	1					
2oct	7	8	9	*					
2dec	4	5	6	-					
2hex	1	2	3	+					
Close	0	()	=					

I. CLOSE YOUR LAPTOP/WALK AWAY FROM YOUR COMPUTER !

- 2. Define target requirements write them down !
 - Basic functionality
 - Available/required input
 - Desired outcome/output
- 3. Develop User Interface (UI)
 - I. Sketch on paper/whiteboard/napkin/BART ticket/etc.
 - 2. Redo a few times till you like it; Draw a large sketch of the final version
 - 3. Identify all objects by type and functionality
 - 4. Play use-scenarios on paper
 - 5. Update your design as needed

Pile Group Tool					
3	Pile Group	Application			
			Spotnersen (Stare Plane)		
Pile Properties		Layer Properties	Lover 1 Lover 2 Lover 3		
Pile Weight		Soil Thickness			
Pile Angle		Gamma	 ²⁰		
Pile Length		phi	⁽²⁾		
Pile Head	Fixed •	Gsoil	²²		
Pile Cross Section	Fixed •	pu	— (23)		
(12) E		ĸ	⁽⁴⁾		
(13) Damping		gwt	 25		
Displacement Profile		Domping	 26		
Applied Loads					
Moment Force					
Vertical Force					
			But 27		
		28			
	Some Foo	oter Content			
			•		

- On the way to Version 0.1 of the PileGroupTool
 - First idea
 - Rough sketch of elements and layout



Element ID	Element	Description	Category	Action & Events	Else
1	Problem definition area		container		
2	Parameter definition area		notebook		
3	result visualization area		notebook		use instances of QCP



Element ID	Element	Description	Category	Action & Events	Else
1	Problem definition area		container		
1a	applied horizontal force	textinput		store info and adjust plot in section 1	
1b	layer #1 thickness	textinput		store info and adjust plot in section 1	
1c	layer #2 thickness	textinput		store info and adjust plot in section 1	
1d	visualization/pile	graphic		double-click activates property seaction 2	
2	Parameter definition area		notebook		
2a	specific weight	textinput		update property variable upon change	
2b	friction angle	textinput		update property variable upon change	
2c	shear modulus	textinput		update property variable upon change	
2d	pu (ultimate pressure)	textinput		update property variable upon change	
2e	k-parameter	textinput		update property variable upon change	
2f	ground water table	combo box: above below		update property variable upon change	defines whether we deal with saturated or wet soil
3	result visualization area		notebook		use instances of QCP
3a	displacement graph selector	visualize computed displacements	tab	change page in notebook to show respective result	
3b	moment graph selector		tab	change page in notebook to show respective result	
3c	shear graph selector		tab	change page in notebook to show respective result	
3d	pile position axis		QCP	allow to zoom in/out	measured from top down
3e	result value axis	adjust to max value	QCP		

EXERCISE #I: GUI DESIGN

- Design a UI for an application that collects a person's information
 - First and last name
 - Address, city, state, ZIP
 - Date of birth
- Create a table listing each element

ID	Туре	Action	Widget	notes
I	Text input	none	???	Check for valid name?
2				

Share with neighbor, discuss options, revise your design as appears useful

QT FRAMEWORK

What is Qt?

A framework to

- Create platform-independent applications
 - Desktop:Windows, Mac, Linux
 - Mobile devices: iOS, Android
 - Cars, Medical devices, ...
- Provide a large number of very useful data representation classes

IT IS NOT FREE !!!!

- Free for OpenSource
- Free for personal use

COMMON DATA CLASSES

QString

#include <Qstring.h>

QString mString;

- A smart string object
- No worries about '\0' (which is a pain even for experienced C-programmers, honestly)

Has formatting tools

mString = "this is process {} of {}";

mString.arg(proc).arg(numProcs);

Has Unicode support (Asian fonts, European fonts)

COMMON DATA CLASSES

QVector<TYPE>

QVector<double> array1;

```
QVector<double> *array2 = new QVector<double>();
```

```
QVector<QVector<double> *> array3;
```

```
array I.append(42.0);
```

```
int n = array2->size();
```

```
double x = array1[2]; array1[1] = array1[2]; array1[1] = x;
```

```
array3[2] = new QVector<double>();
```

COMMON DATA CLASSES

QList<TYPE>

QList<QString> stringList1;

QStringList stringList2;

Looping made simple:

#include <iostream.h>
#include <Qstring.h>
#include <QStringList.h>

foreach (QString s, stringList1) {
 // do something with string s
 std::cout << s << std::endl;</pre>

BUILDING THE GUI

Option #1:

- Directly in code
- Check out <u>http://zetcode.com/gui/qt5/</u> (THESE GUYS ROCK !)

Option #2:

- Using Qt Designer (built into Qt Creator)
- Let's switch and build your app together (Live Demo)

DEVELOPER TOOL FOR QT

Qt Creator

	[master] - CWE-Simulation-Tool - Qt Creator				
Welcome	Projects	+ New Project	🗁 Open Project		
Edit	Examples	Sessions	Recent Projects		
Design	Tutorials	1 D default (current session)	1 CWE-Simulation-Tool ~/Development/SimCenter/CWE-Simulation-Tool/CWE-Simulation-Tool.pro		
e bug			2 Example1 ~/Development/SimCenter/SimCenterBootcamp/Code/Qt/Example1/Example1.pro		
J Projects	New to Qt?		3 Example2 ~/Development/SimCenter/SimCenterBootcamp/Code/Qt/Example2/Example2.pro		
0	Learn how to develop your own applications and explore Qt Creator.		4 PileGroupToolwithQCP ~/Development/SimCenter/PileGroupTool/PileGroupToolwithQCP.pro		
неір	Get Started Now		5 PileGroupTool ~/Development/SimCenter/PileGroupTool/PileGroupTool.pro		
			6 PiParallel ~/Development/BootCampExercises/PiParallel/PiParallel.pro		
			7 Pi ~/Development/BootCampExercises/Pi/Pi.pro		
			8 WorkshopQtExample2 ~/Development/SimCenter/WorkshopQtExamples/Example2/WorkshopQtExample2.pro		
			 BridgSteelUI ~/Development/WSDOT/QTBridg/Dev/BRIDG/Steel/QTProj/BRIDG_Steel_UI/BridgSteelUI.pro 		
CWETool			BridgConcreteUI ~/Development/WSDOT/QTBridg/Dev/BRIDG/Concrete/QTProj/BRIDG_Concrete_UI/BridgConcreteUI.pro		
Debug	L Qt Account		SchemaCompiler ~/Development/WSDOT/QTBridg/Dev/BRIDG/BSDK/SchemaCompiler/SchemaCompiler.pro		
	Online Community Blogs		ProcessText ~/Development/WSDOT/QTBridg/Dev/BRIDG/Concrete/ProcessText/ProcessText/ProcessText.pro		
N R	? User Guide		LevelSetTool ~/Development/UW/LevelSetTool/LevelSetTool.pro		
>	□ P. Type to locate (ℋK)	1 Issues 2 2 Search Results 3	Application Output 4 Compile Output 5 Debugger Console 8 Test Results 🗧 🔹 🗖		



	Qt Widgets Application					
🔷 Location	tion Introduction and Project Location					
Kits Details Summary	This wizard generates a Qt Widgets Application project. The application derives by default from QApplication and includes an empty widget.					
	Name: Example					
	Create in: /Users/pmackenz/Development/SimCenter/SimCenterBootcamp/Code/Qt Choose Use as default project location					
Cancel	Continue					

Location	Kit Selection								
> Kits	The following kits can be used for project Example0 :	The following kits can be used for project Example0 :							
Details	Type to filter kits by name								
Summary	Select all kits								
· //	🗹 🖵 Desktop Qt 5.11.1 clang 64bit De	atails							
	Debug nz/Development/SimCenter/build-Example0-Desktop_Qt_5_11_1_clang_6 Choose	;e							
	Warning: The build directory needs to be at the same level as the source directory.								
• /	 Release /Users/pmackenz/Development/SimCenter/build-Example0-Desktop_Qt_ Choose Warning: The build directory needs to be at the same level as the source directory. 								
	✓ Profile /Users/pmackenz/Development/SimCenter/build-Example0-Desktop_Qt_ Choos	e							
	Warning: The build directory needs to be at the same level as the source directory.								
Cancel	Go Back	Contir							

	Qt Widgets Application
Location	Class Information
Kits	Specify basic information about the classes for which you want to generate skeleton source code files.
Summary	Class name: MainWindow
	Base class: QWidget ODialog
	Header file: mainwindow.h
	Source file: mainwindow.cpp
VI. /	Generate form: 🗹
	Form file: mainwindow.ui
Cancel	Go Back Continue







Step #I

Build	Debug	Analyze	Tools	Windo				
Build Build	Build All							
Run Depl	<mark>qmake</mark> oy All			-				
Depl	oy Projec	t "Example(0"	-				
Rebu Rebu Clea Clea	Rebuild All Rebuild Project "Example0" Clean All Clean Project "Example0"							
Cano	el Build			^ 🛛				
Run Run	Without D)eployment	olootor	ЖR				
Ope	Step ;	#1:only r chang	needeo ged	l if Ul				

Step #2

Build	Debug	Analyze	Tools	Wind
Build	IAI		ſ	ን ℋB
Builc	Project "	Example0"		ЖB
Run Depl Depl	qmake oy All oy Project	t "Example	0"	
Rebu Rebu Clea Clea	uild All uild Projec n All n Project	t "Example" "Example0	90" "	
Cano	el Build			^ (X
Run Run Oper	Without D n Build an	eployment d Run Kit S	: elector	₩R

RUN !	MainWindow
	Hello World!

EXERCISE #2: CREATING YOUR GUI

- Let's return to your GUI design from Exercise #1
- I. Create a new Qt Widget Application project using Qt Creator
- 2. Open Forms => MainWindow.ui
- 3. Create your GUI as close to your design as possible
- 4. Go through all the objects and assign them a more descriptive name like:
 - TB_firstName
 - CB_theState
 - ✤ Etc.
- 5. Run qmake, build the app, and run it

This one should be surprisingly easy \odot



A SIMPLE APPLICATION





A SIMPLE APPLICATION USING LAYOUTS

New Demo Application	
Reset View	
	New Dame Applicati
	Reset View





EXERCISE #3A: LAYOUTS





EXERCISE #4: CREATING YOUR GUI

- Let's return to your GUI design from Exercise #2
- 1. BEFORE doing anything, think about layout for your app.
 - How do you want each field to line up?
 - How shall each field grow relative to each other?
 - How can you achieve that with the least of layouts?
- 2. Move on and implement your layout
 - I. Select container object
 - 2. Right-click and select layout
 - 3. Choose the desired layout

This one is usually harder but VERY IMPORTANT

SIGNALS AND SLOTS

How does a GUI work?

- Create the graphics
 - Instance of QMainWindow
 - Add child widgets
 - QFrame
 - QPushButton
 - etc.
- Emit signals for events
- Connect signals to slots
- Run the Event loop

	MultiBase Calculator					
7	<u>F</u> ile		~			
	213	Bck				
	nput >	bin	oct	dec	hex	
		A	e	f	mod	
	2bin	a	ь	c	1	
	2oct	7	8	9	*	
	2dec	4	5	6	-	
	2hex	1	2	3	+	
	Close	0	()	=	

SIGNALS AND SLOTS



OPTION I: OVERLOADING DEFAULT SLOTS

Each Widget emits signals on specific events

Signals

void	<pre>clicked(bool checked = false)</pre>
void	pressed()
void	released()
void	toggled(bool checked)

- > 3 signals inherited from QWidget
- > 2 signals inherited from QObject

- Each widget has a unique name
- **Example:**
 - > Widget name: run_button
- Event clicked connects to default slot:
 - > on_run_button_clicked()
 - You can overload that slot in your application
- Implementation made easy:
 - > Qt Creator
 - Right click => go to slot => clicked

OPTION 2: CREATING YOUR OWN SLOTS



OPTION 2: CREATING YOUR OWN SLOTS

<pre>striggered(); triggered();triggered(); n_triggered(); n_triggered(); t_triggered(); t_triggered(); le_manager::linkMainWindow</pre>	463 464 ▼ 465 466 467 468 469 470 471 ▼ 472 473	<pre>void MainWindow::bn_actionExit_triggered() { this->close(); } /* ***** check box status changes ***** */ void MainWindow::on_chkBox_assume_rigid_cap_clicked(bool checked { assumeRigidPileHeadConnection = checked; nWindow *theMainWin)</pre>
<pre>striggered(); triggered();triggered(); n_triggered(); rt_to_OpenSees_triggered(); t_triggered(); le_manager::linkMainWindow</pre>	465 466 467 468 469 470 471 ▼ 472 473	<pre>{ this->close(); } /* ***** check box status changes ***** */ void MainWindow::on_chkBox_assume_rigid_cap_clicked(bool checked { assumeRigidPileHeadConnection = checked; nWindow *theMainWin)</pre>
<pre>_triggered(); triggered(); _triggered(); n_triggered(); rt_to_OpenSees_triggered(); t_triggered(); le_manager::linkMainWindow</pre>	467 468 469 470 471 ▼ 472 473	<pre> } /* ***** check box status changes ***** */ void MainWindow::on_chkBox_assume_rigid_cap_clicked(bool checked { assumeRigidPileHeadConnection = checked; nWindow *theMainWin) </pre>
<pre>_triggered(); triggered(); _triggered(); n_triggered(); rt_to_OpenSees_triggered(); t_triggered(); le_manager::linkMainWindow</pre>	468 469 470 471 ▼ 472 473	<pre>/* ***** check box status changes ***** */ void MainWindow::on_chkBox_assume_rigid_cap_clicked(bool checked { assumeRigidPileHeadConnection = checked; nWindow *theMainWin)</pre>
<pre>Langgered(); n_triggered(); rt_to_OpenSees_triggered(); t_triggered(); le_manager::linkMainWindow</pre>	470 471 ▼ 472 473 (CWE_Mai	<pre>void MainWindow::on_chkBox_assume_rigid_cap_clicked(bool checked { assumeRigidPileHeadConnection = checked; nWindow *theMainWin)</pre>
<pre>rt_to_OpenSees_triggered(); t_triggered(); le_manager::linkMainWindow</pre>	472 473 (CWE_Mai	<pre>{ assumeRigidPileHeadConnection = checked; nWindow *theMainWin)</pre>
<pre>le_manager::linkMainWindow</pre>	(CWE_Mai	nWindow *theMainWin)
<pre>>remoteTreeView->setModeLL oject::connect(ui->remoteTre this, SLOT(controlsEnabled(true);</pre>	<pre>ink(theM ink(theM eeView, ustomFil :get_fil :get_fil :get_fil :get_fil :get_fil</pre>	<pre>lineMode()) ainWindow->getFileModel()); SIGNAL(customContextMenuRequested(QPoint)), eMenu(QPoint))); e_handle(), SIGNAL(fileOpDone(RequestState,QString)), one(RequestState,QString))); e_handle(), SIGNAL(fileOpStarted()), tarted()));</pre>
	<pre>e_globals::get_CWE_Driver(>remoteTreeView->setModelL .ject::connect(ui->remoteTr</pre>	<pre>e_globals::get_CWE_Driver()->inOff >remoteTreeView->setModelLink(theM 'ject::connect(ui->remoteTreeView,</pre>

DEBUGGING WITH SIGNALS AND SLOTS

- Clean way
- 1. Set breakpoints at entries to slot implementation(s)
- 2. Start ("run") application
- 3. Don't stop at first occurrence but continue till app accepts new user input.
- Brute-force method:
 - Write debug output at start of slot implementation(s)

```
#include <QDebug.h>
```

```
void MyClass::MySlot(int arg1) {
    qDebug() << "Entering MySlot";
    // your code here</pre>
```

EXERCISE #5: ADDING CALLBACK FUNCTIONS

- Let's add some functionality to your GUI
- 1. Create a class method (function) that collects the information from the UI and stores it in a private structure like this one:

- 2. Create a slot that writes out a formatted address label
- 3. Create a button (if you don't have one yet) labeled "Print Address Label"
- 4. Connect this button's clicked signal to your slot
- 5. Qmake => build => run

DESIGN CONSIDERATIONS

VIEW – CONTROLLER – DATA model

- VIEW
 - Visual parts, display classes
- - Registers user requests
 - Manages actions in analysis models
 - Controls flow of data
- DATA
 - All kinds: text, floats, arrays, class objects, ...

This is the image of your app

This represents the smarts of your app

This is what only Excel users care to look at

MODEL – VIEW CONCEPT

QAbstractItemView

- QTreeView
- QTableView

QListView

The Display Widget

QTreeView mView;

QAbstractItemModel

QAbstractItem

The data to be displayed

QAbstractItemModel *model = new QAbstractItemModel();

Connecting data and view:

mView.setModel(model);

Note: this is just a pointer to the model, NOT a copy.

USEFUL HELPER WIDGETS

QDialog

- QFileDialog
- QMessageDialog
- QColorDialog
- QFontDialog
- •••

QDir ... all the help you need dealing with paths across different platforms

QDateTime ... dealing with time formats, date formats, calculating number of days, elapsed time, time zones

EXERCISE #6: CREATE A NICE ADDRESS LABEL

- Update your slot for create_label_button_clicked() (or add another one) such that is
 - Pops open a dialog showing a nicely formatted address label in a QTextBrowser widget