P7

Sudoku GUI 2 on the Raspberry Pi

In this lab you will learn how to implement event-driven GUI programming using Qt4. You will develop a new GUI for your Sudoku game from labs P1 and P6. In this lab you will build it up of pre-programmed “widgets” from the Qt library and use Qt’s event handling called “signals” and “slots”.

(Note: you do not have to use the QSpinBox widgets shown in the screenshot above; other widgets may be better…)
Schedule

Preparation time : 3 hours
Lab time : 3 hours

Items provided

Tools : None
Components : 8GB Micro-SD card (keep image and code from P6)
Equipment : DVI-D capable monitor
Software : Same Rasparian image file as P6 at
https://secure.ecs.soton.ac.uk/notes_so/elec1204/Labs/P6P7P20/rpi-qt4.img.zip

Items to bring

- Essentials: A full list is available on the Laboratory website at
https://secure.ecs.soton.ac.uk/notes/ellabs/databook/essentials/
- Raspberry Pi
- Raspberry Pi SD card (used in P6)
- Raspberry Pi power supply
- Raspberry Pi HDMI to DVI-D cable
- Raspberry Pi keyboard
- Raspberry Pi mouse

Before you come to the lab, it is essential that you read through this document and complete all of the preparation work in section 2. If possible, prepare for the lab with your usual lab partner. Only preparation which is recorded in your laboratory logbook will contribute towards your mark for this exercise. There is no objection to several students working together on preparation, as long as all understand the results of that work. Before starting your preparation, read through all sections of these notes so that you are fully aware of what you will have to do in the lab.

Academic Integrity – If you undertake the preparation jointly with other students, it is important that you acknowledge this fact in your logbook. Similarly, you may want to use sources from the internet or books to help answer some of the questions. Again, record any sources in your logbook.

Revision History

February 09, 2015 Tristan Aubrey-Jones (taj105) and Mohammed El-Hajjar (meh) First version of this lab created

© Electronics and Computer Science, University of Southampton
1 Aims, Learning Outcomes and Outline
This laboratory exercise aims to:

- Gain experience in event-driven programming.
- Develop the ability to code C++ GUIs using Qt and similar libraries.

Having successfully completed the lab, you will be able to:

- Implement event-driven programming and program GUIs.
- Develop Qt4 applications that use widgets, layouts, dialogs and timers.

The objective of this lab is to create a Sudoku GUI similar to the one in Lab P6, but instead of manual painting and click-handling, uses the Qt4 widget library and proper event-handling, which are called signals and slots in Qt. In addition to the game grid, you will also use file open and close dialogs and a timer to time how long the user takes to solve the puzzle.

A good tutorial for Qt4 can be found at [http://zetcode.com/gui/qt4/](http://zetcode.com/gui/qt4/). Documentation for the classes you will use can also be found at [http://qt-project.org/doc/qt-4.8/](http://qt-project.org/doc/qt-4.8/).

2 Preparation
Read through the course handbook statement on safety and safe working practices, and your copy of the standard operating procedure. Make sure that you understand how to work safely. Read through this document so you are aware of what you will be expected to do in the lab.

2.1 Read Qt tutorial and design GUI
To prepare for this lab you should read the Qt4 tutorial at [http://zetcode.com/gui/qt4/](http://zetcode.com/gui/qt4/) and plan what widgets and layouts you will use to make the Sudoku GUI. You should particularly read up on signals and slots, how they work and how to use them. Your preparation should allow you to answer these questions:

- What should my GUI look like? (Draw wireframe diagrams to design this).
- What widget classes will I use to implement the user interface, and how will I nest them to create a suitable layout? Will this layout scale when the window is resized?
- What signals and slots should I use for the various widgets? How can I cleanly handle the events from all the 81 widgets in the 9x9 grid without writing 81 separate methods?

3 Laboratory Work
During this lab you will develop a Sudoku game GUI using Qt4 on the Raspberry Pi. To start boot up your Raspberry Pi, open “Qt creator” and open the project called “lab7”. You should then implement your GUI by creating suitable widget objects and layouts (in the window constructor), writing methods for all the slots (event-handlers) and linking together the relevant signals (events) and slots (event-handlers) using the “connect” keyword.

3.1 Create the GUI
Copy in any data structures and relevant code from the model part of P6 and code to read games to files on disk. Write code to create all the widgets (buttons, text boxes, spin boxes etc), dialogs and layout in the window’s constructor.
Your GUI should contain:

- A 9x9 grid of suitable widgets.
- Open, close and solve buttons.
- File dialogs that appear, when the open button is clicked.

The layout of these widgets should scale when the window is resized.

### 3.2 Initialize the GUI

Write a method to update the widgets with the values in the current game (the model), so that when a game is opened, this method can be used to display the values on the GUI.

### 3.3 Implement signals and slots

Write slots (i.e. methods used as event-handlers) to respond to users interacting with the GUI. These methods should update the model to reflect the current state of the GUI. Use the “connect” keyword/method in the window constructor to connect signals (events) to the slots (event-handler methods) that should react to them.

### 3.4 React when puzzle is solved

Use code from P6 to recognise when a puzzle is solved and show a message box, when it has been solved. In addition, use a timer to tell the user how long they have taken on the current puzzle so far and when solved, how long they took to solve the puzzle.

### 4 Optional Additional Work

*Marks will only be awarded for this section if you have already completed all of Section 3 to an excellent standard and with excellent understanding.*

- Allow users to save puzzles.
- Add a drop down menu to the top of the window, with open, close and solve buttons.
- Implement an undo button. You can do this using an STL stack<T> of game states, pushing states onto the stack when the user makes changes and popping them off the stack and rolling back to that state, when the user clicks “undo”.