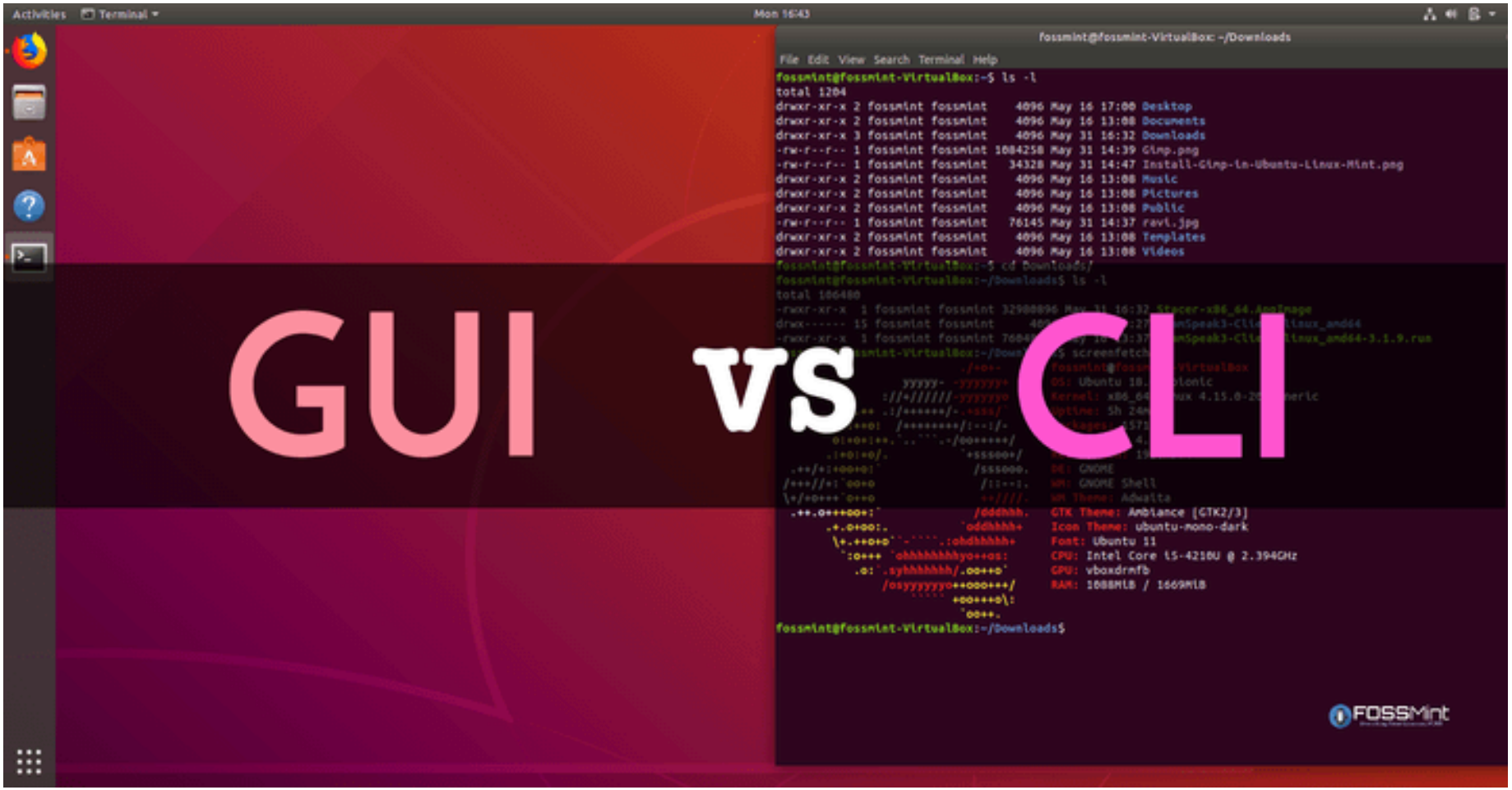
GUI

Modern cpp Programming lecture 7

GUI

- Graphical User Interface
  - A system of interactive visual components for computer software
  - ***A form of user interface that allows users to interact with devices through graphical icons and audio indicator*** *(Wikipedia)*
  - Users can manipulate the program by various interaction devices
    - Keyboard, Mouse (Desktop computer)
    - Touchpad, Trackpad (Laptop)
    - Multi-touch screen, virtual keyboard (Smartphone, Tablet PC)

GUI vs CLI

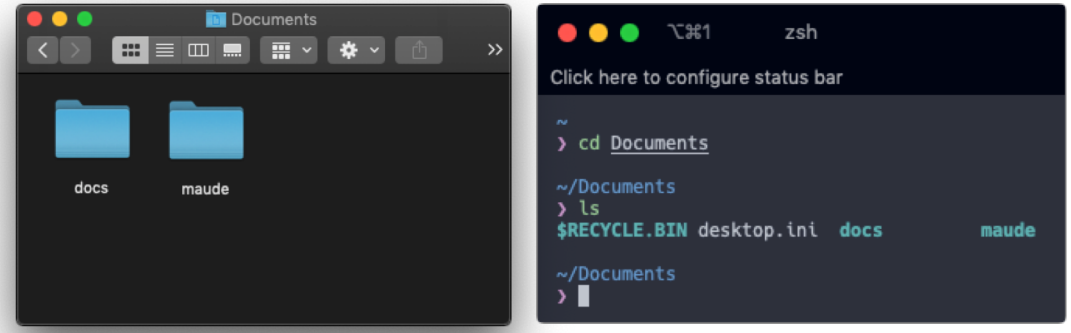

GUI
vs
CLI
Activities
Terminal
Mon 16:43
fossmint@fossmint-VirtualBox: ~/Downloads
File Edit View Search Terminal Help
fossmint@fossmint-VirtualBox:~$ ls -l
total 1204
drwxr-xr-x 2 fossmint fossmint 4096 May 16 17:00 Desktop
drwxr-xr-x 2 fossmint fossmint 4096 May 16 13:08 Documents
drwxr-xr-x 3 fossmint fossmint 4096 May 31 16:32 Downloads
-rw-r--r-- 1 fossmint fossmint 1084258 May 31 14:39 Gimp.png
-rw-r--r-- 1 fossmint fossmint 34328 May 31 14:47 Install-Gimp-in-Ubuntu-Linux-Mint.png
drwxr-xr-x 2 fossmint fossmint 4096 May 16 13:08 Music
drwxr-xr-x 2 fossmint fossmint 4096 May 16 13:08 Pictures
drwxr-xr-x 2 fossmint fossmint 4096 May 16 13:08 Public
-rw-r--r-- 1 fossmint fossmint 76145 May 31 14:37 ravi.jpg
drwxr-xr-x 2 fossmint fossmint 4096 May 16 13:08 Templates
drwxr-xr-x 2 fossmint fossmint 4096 May 16 13:08 Videos
fossmint@fossmint-VirtualBox:~$ cd Downloads/
fossmint@fossmint-VirtualBox:~/Downloads$ ls -l
total 106480
fossmint@fossmint-VirtualBox:~/Downloads$ screenfetch
DE: GNOME
WM: GNOME Shell
WM Theme: Adwaita
GTK Theme: Ambiance [GTK2/3]
Icon Theme: ubuntu-mono-dark
Font: Ubuntu 11
CPU: Intel Core i5-4210U @ 2.394GHz
GPU: vboxdrmfb
RAM: 1088MiB / 1669MiB
fossmint@fossmint-VirtualBox:~/Downloads$
FOSSMint

GUI vs CLI

GUI	CLI
Graphical User interface	Command Line interface
Even a beginner can easily handle	User should have good knowledge of commands (Better for the experts)
Requires more memory as it contains a lot of graphical components	Does not require much memory
Slower	Faster
More flexible	Not much flexible

GUI vs CLI

- For example…
  - Most OSs provides both Graphical & Command-line ways to manage file system
  - macOS & Linux
    - Finder vs. Terminal



  - Windows
    - Window File explorer vs. Command Prompt

GUI

- Most of the programs you use…
- are GUI program!!

Office program

Game

Messenger

Online conferencing program

- Therefore you must learn how to implement GUI!!

Implementing GUI program

- How?
- Almost all of modern programming languages provides GUI Library
  - Java
    - Swing, Applet…
  - cpp
    - **Qt**, WinAPI, wxWidgets
  - C#
    - .Net (dot net)

Implementing GUI Program

- In nowadays…
  - Using the combination of html / css / javascript is recommended
  - Why?
    - Cross-platform
    - Lots of packages / libraries
    - Easy to implement functionalities
    - Easy to design your program
- However, in this class…
  - We’re learn the GUI framework for cpp!!

Qt

- Open source GUI framework for cpp
- Developed in 1990 by *Eirik Chambe-Eng & Haavard Nord*
- Cross-platform
  - Windows, macOS, Linux, Android, iOS…
- Many famous programs were developed using Qt
  - Google Earth, Mathematica, Bitcoin Core, Telegram…
- Also provides Qt for python (not that useful…)

Qt development

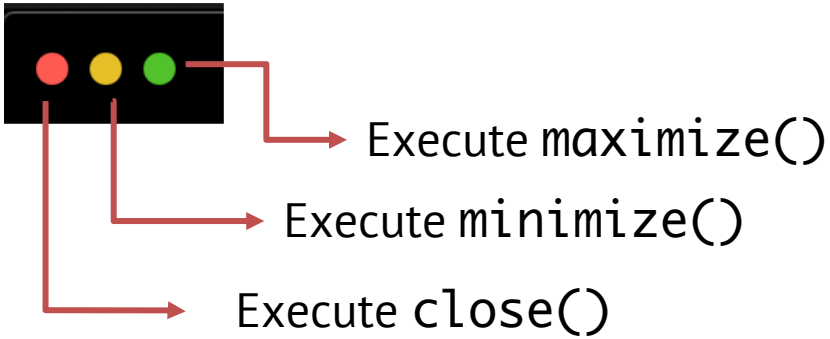
- Qt creator
  - A cross-platform IDE (integrated development environment)
    - macOS, Linux, Windows
  - Supports convenient Qt program development
  - Includes *Qt Designer*
    - Integrated GUI layout builder
    - Users can rapidly design and build widgets and dialogs
    - Previewed immediately (convenient!!)

Qt Classes

- Qt is implemented using OOP
  - Every module in Qt is *Class*
  - Each module can interact with other module by using *methods*
  - You can find Qt Classes info in the doc
  - ***Signals & Slots***
    - The way of sending & receiving messages btw Qt objects (modules)
    - Central feature of Qt

Signals & Slots

- Communications btw objects
- For example…

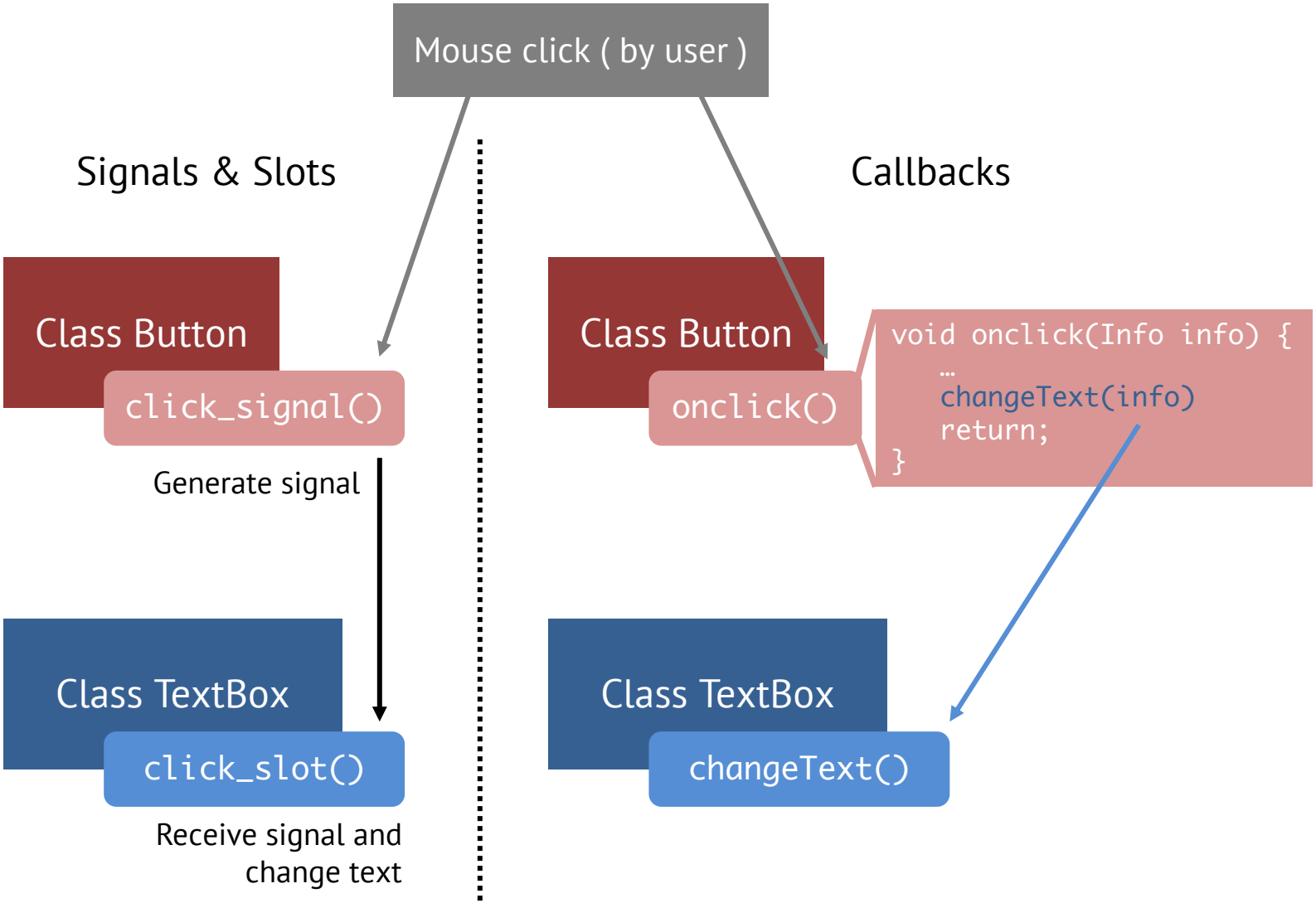



```
Class Window {
private:
    int tab_num
    string current_url
    …

public:
    maximize();
    minimize();
    close();

};
```

- Signals & Slots is one of the ways to implement the connection
  - Different to other frameworks that usually use *callbacks*

Signals & Slots

- *Signals & Slots* vs. *Callback*



Signals & Slots

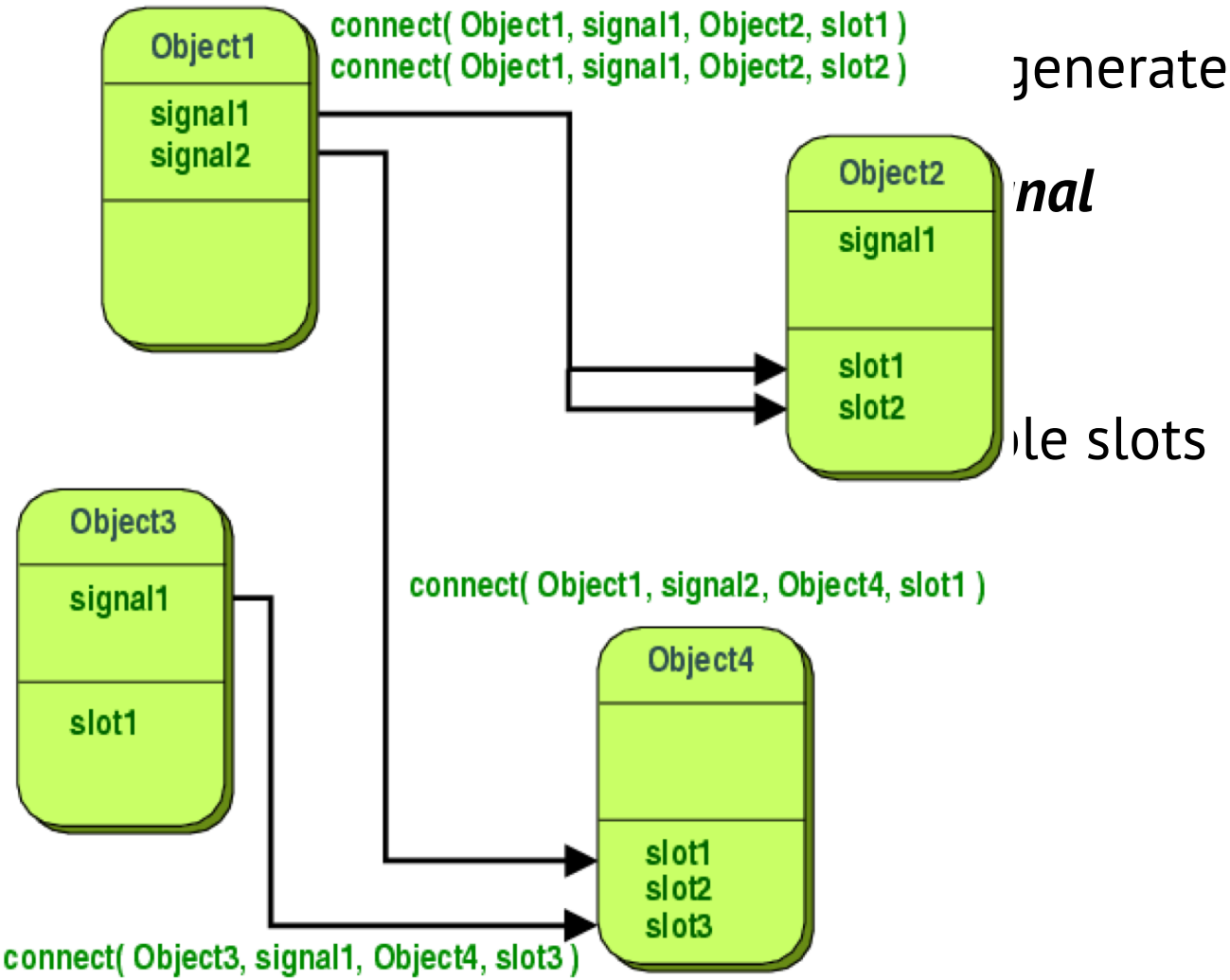
- Event-Driven
- If an event occurs in a module, the module generates ***signal***
- Then the target ***slot(s)*** response(s) to the ***signal***
- You should *connect* signals & slots
- multiple signals can be connected to multiple slots

Signals & Slots

- Event-Driven
- If an event occurs in a module, the module generates ***signal***
- Then the target ***slot(s)*** response(s) to the ***signal***
- You should *connect* signals & slots
- multiple signals can be connected to multiple slots

Signals & Slots

- Event-Driven
- If an
- Ther
- You
- mult

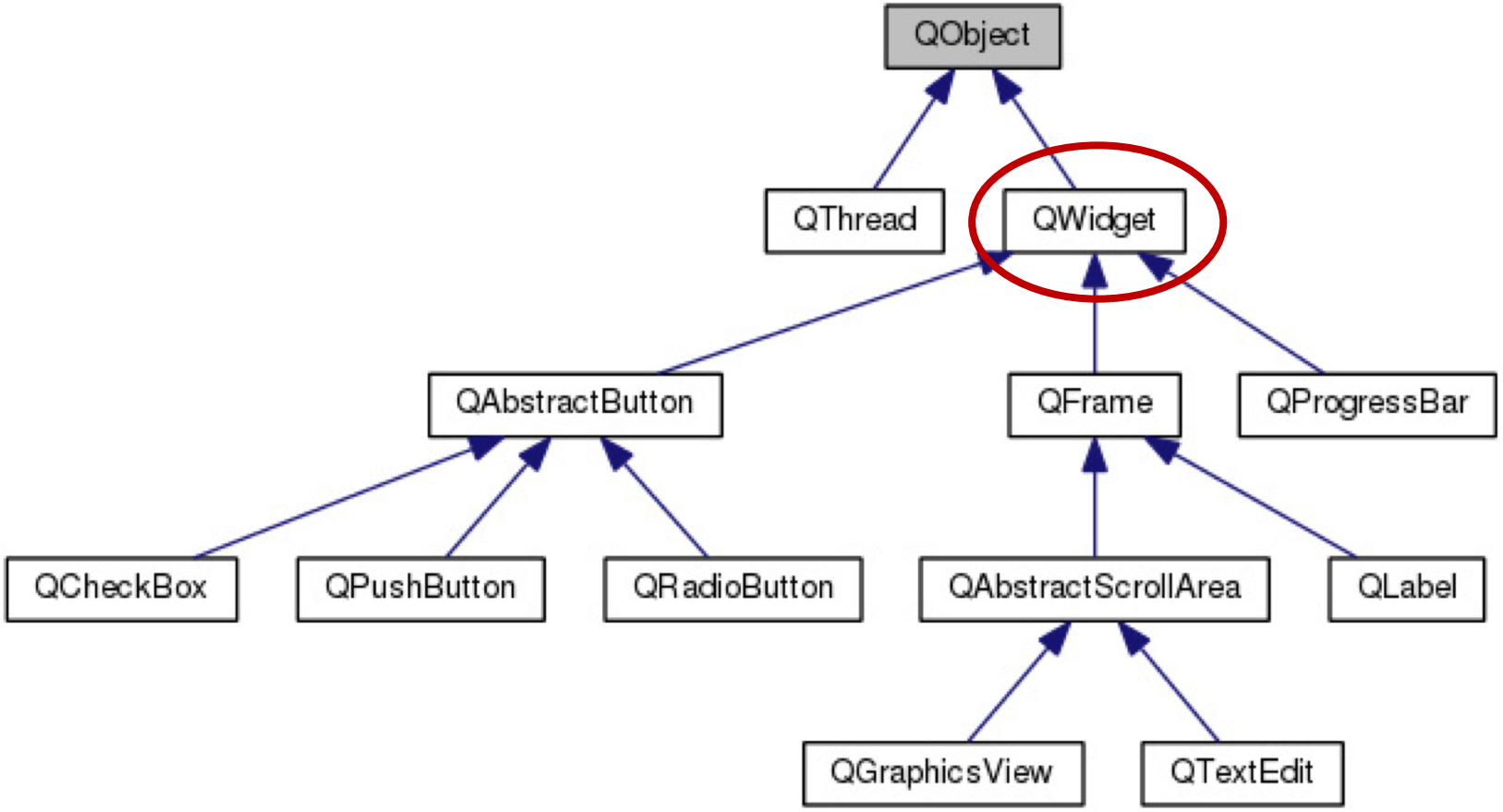


QObject class

- The heart of the Qt Object Model
  - The base class of almost all Qt classes & widgets
  - Defines the functions for basic Qt features & ***inherit them***
    - `destroyed()`
    - Basic functions to support signals & slots
    - `connect()` function for signals & slots
    - And else…

QObject class

- QObject hierarchy



QWidget class

- The core of Qt user interface
- Receives user input
  - mouse, keyboard, and other…
- Paints a representation of itself on the screen
- Locating widgets
  - Clipped by its parent & by the widgets in front of it

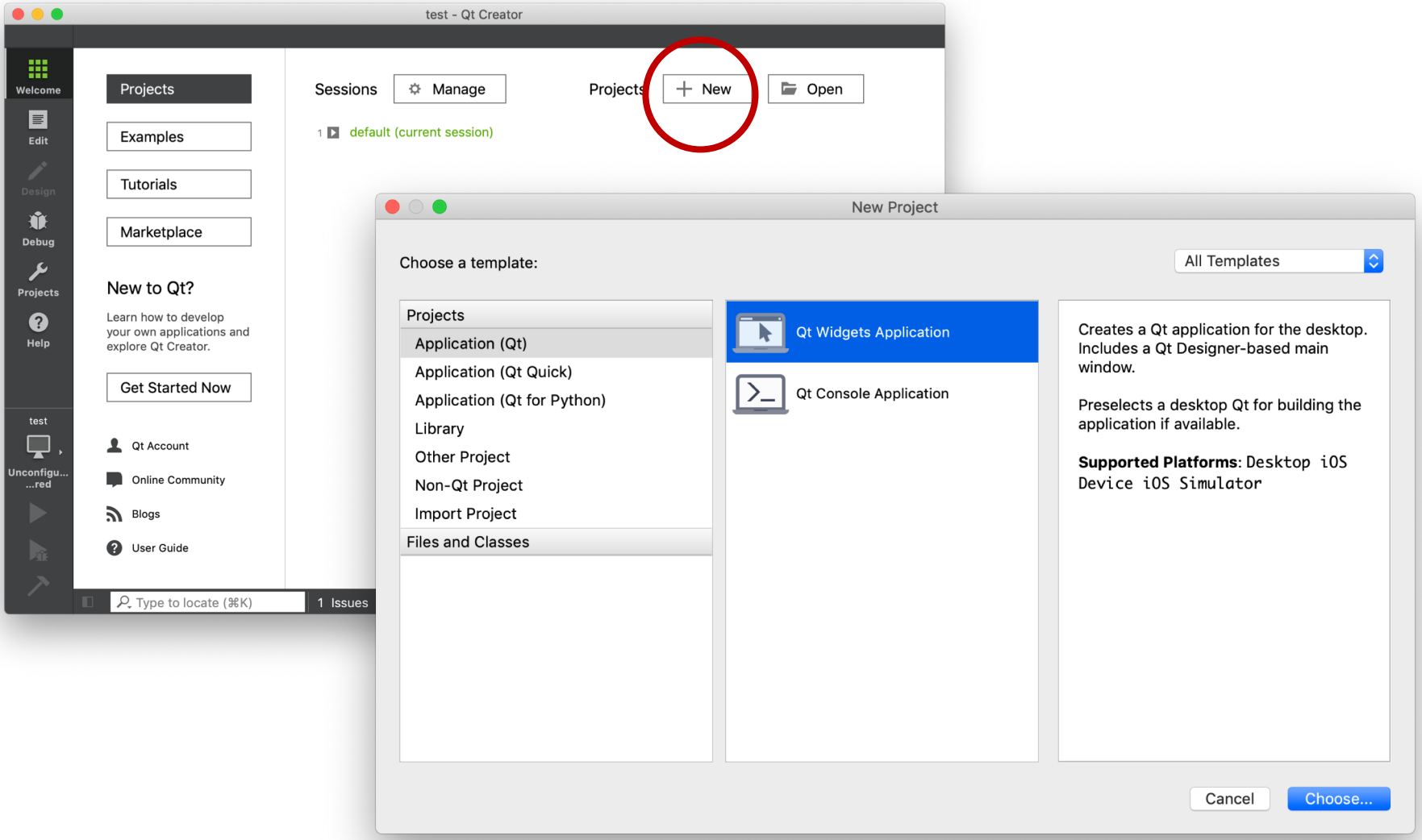
Qt Tutorial

Preparation

- Install Qt & Qt Creator
  - We'll use Qt 5.12.9.
  - Refer the manual for the installation

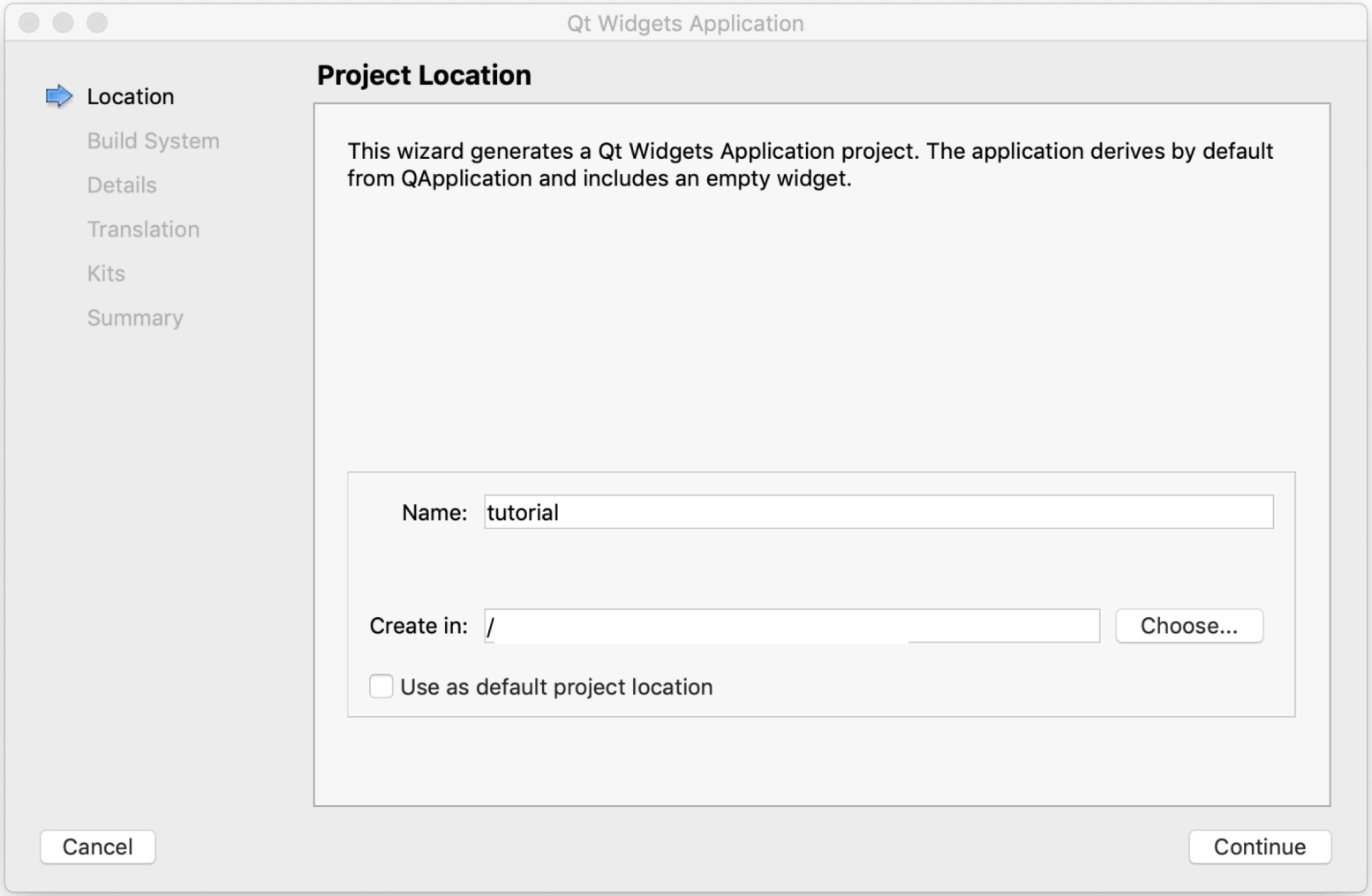
Create new project

- Click ***+New*** project button and select ***Qt Widgets Application***



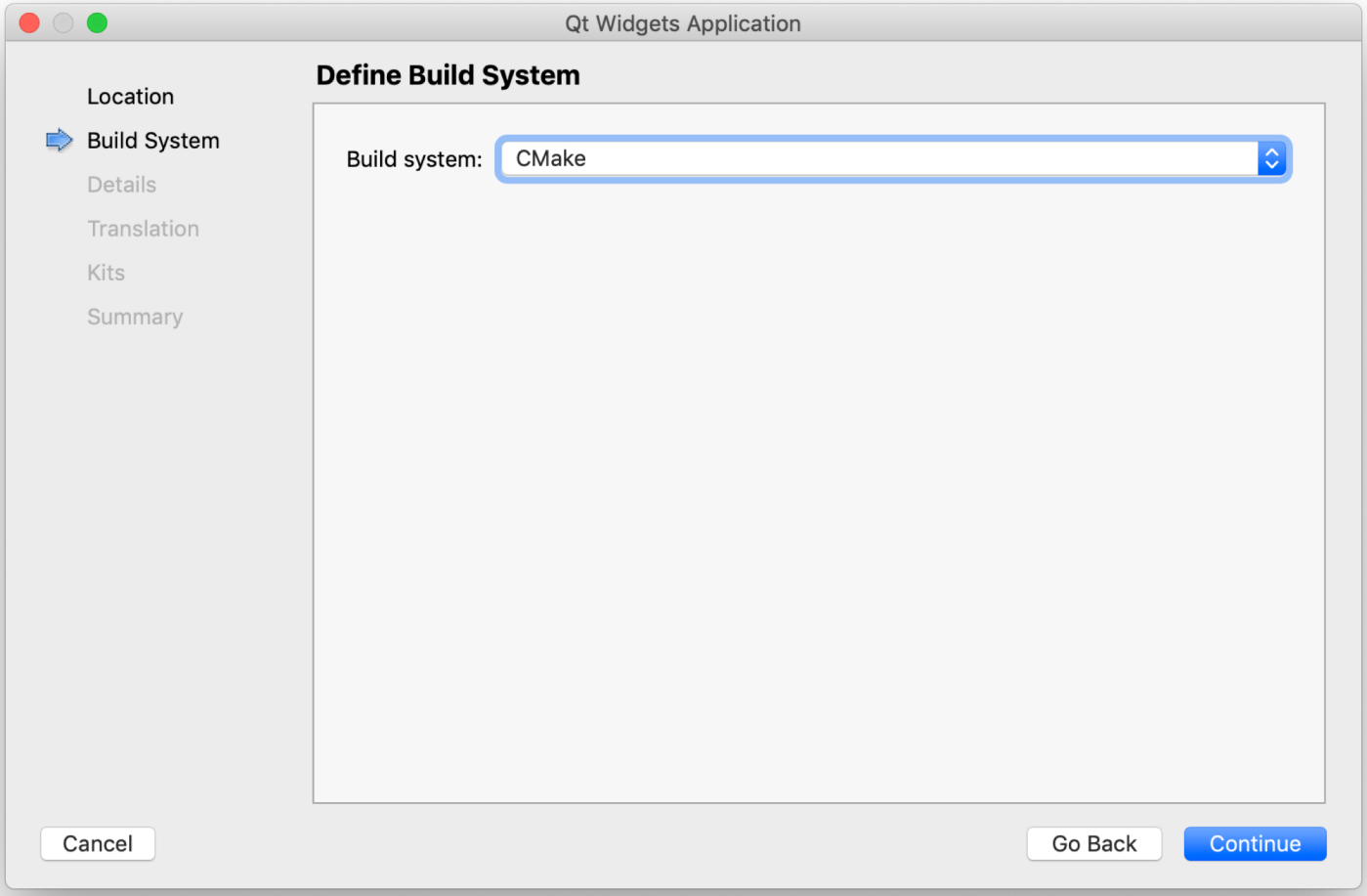
Create new project

- Set proper project path (location) and continue



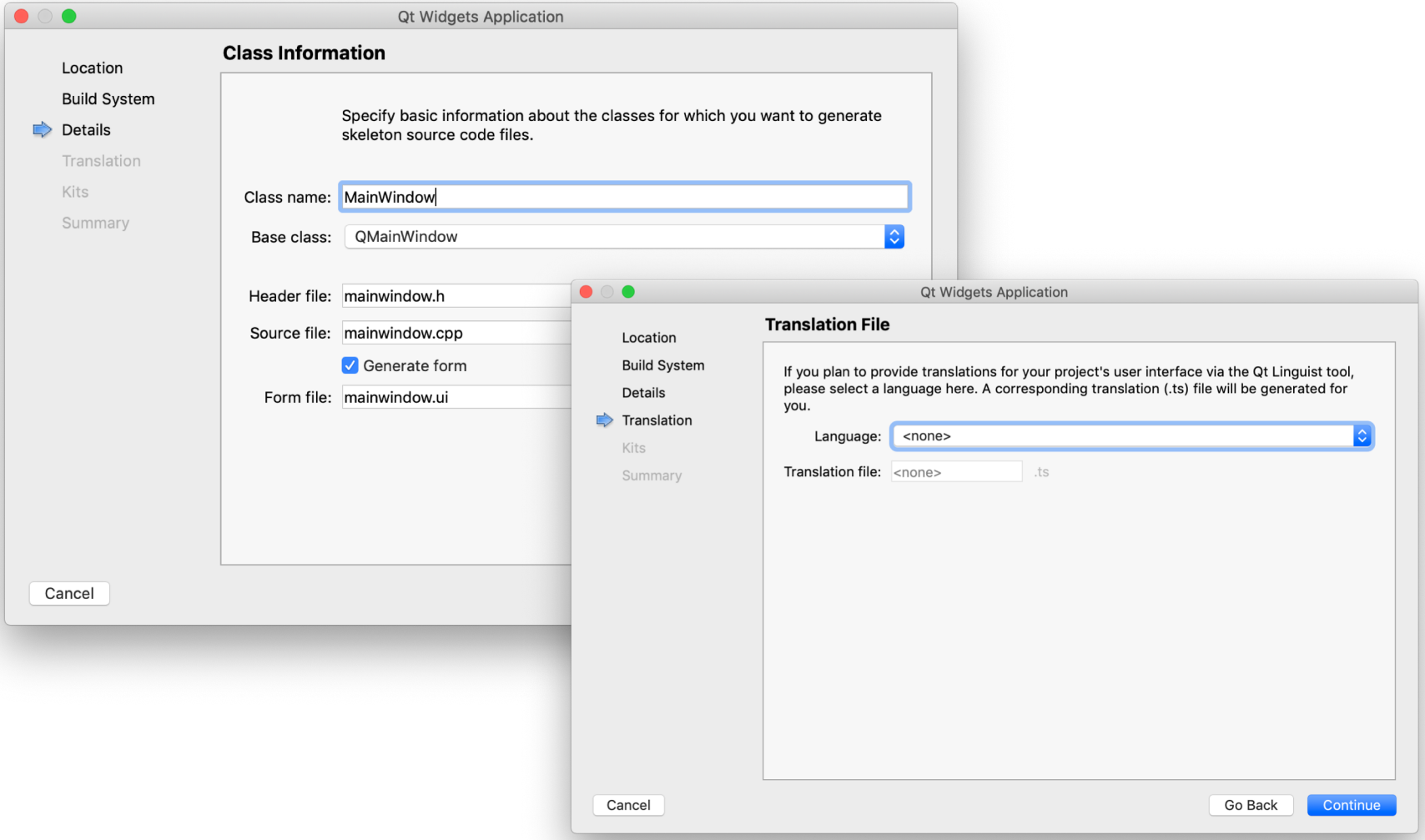
Create new project

- Select *cmake* as build system (maybe different in Windows)



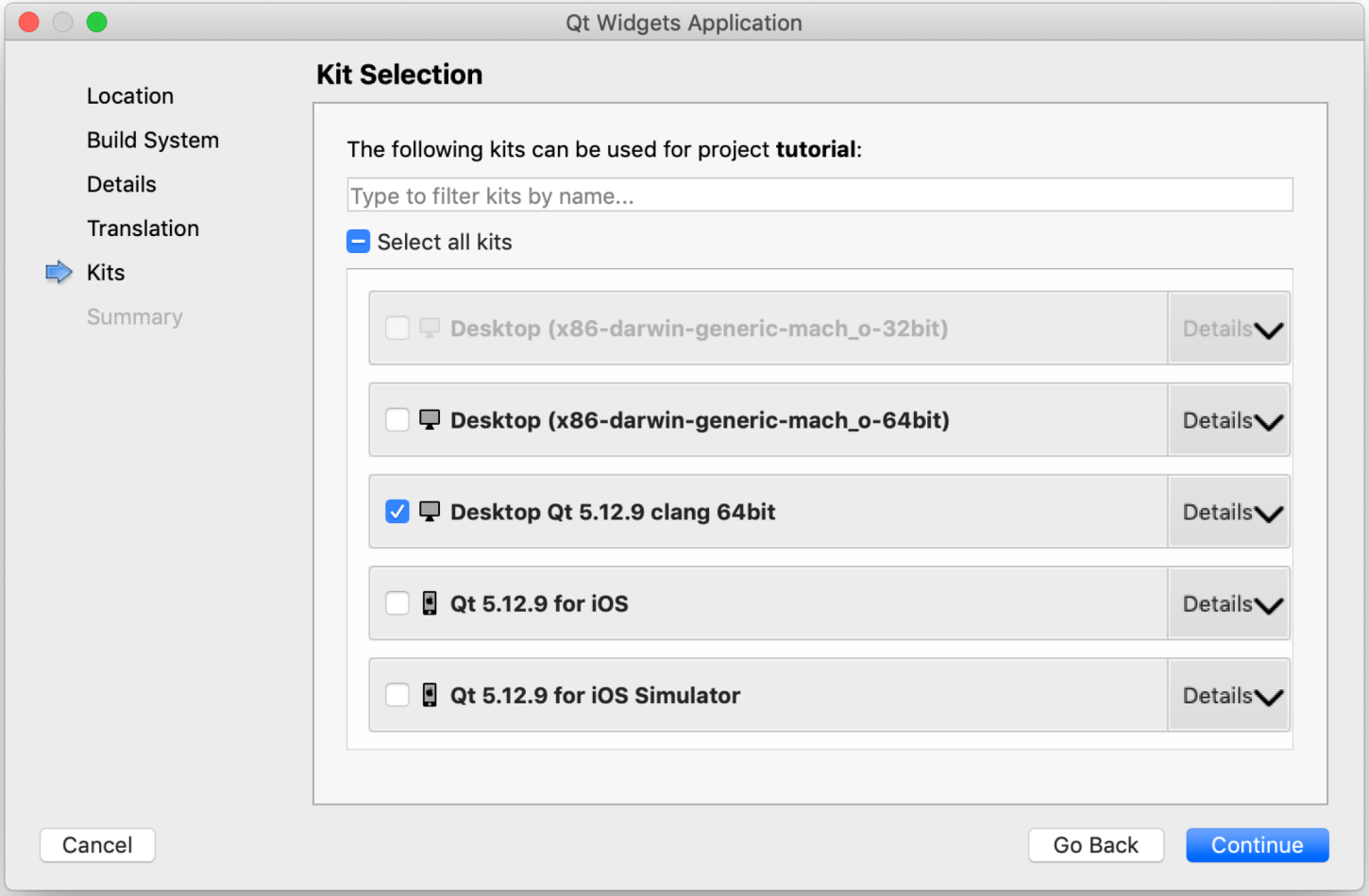
Create new project

- Use default Class information & Translation settings



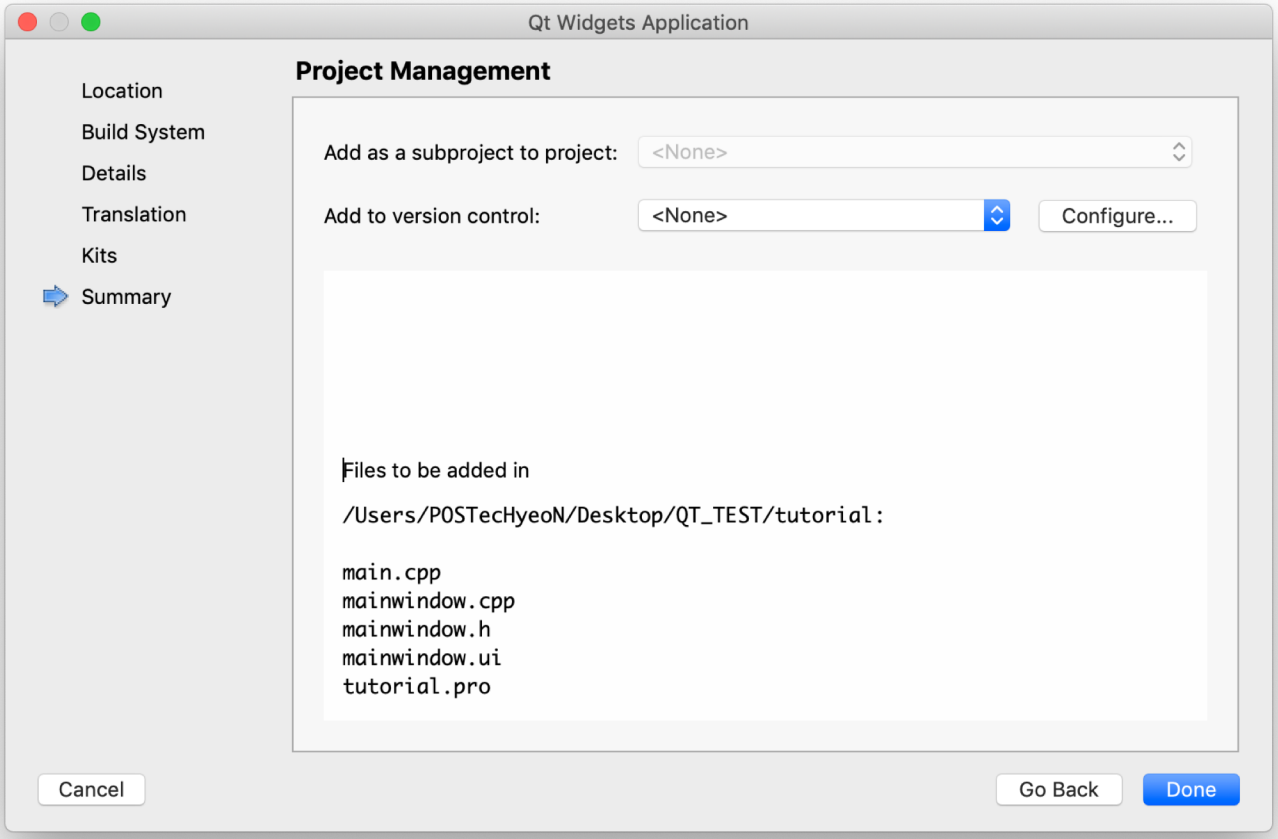
Create new project

- Select proper kit for the **Desktop & your compiler** and continue



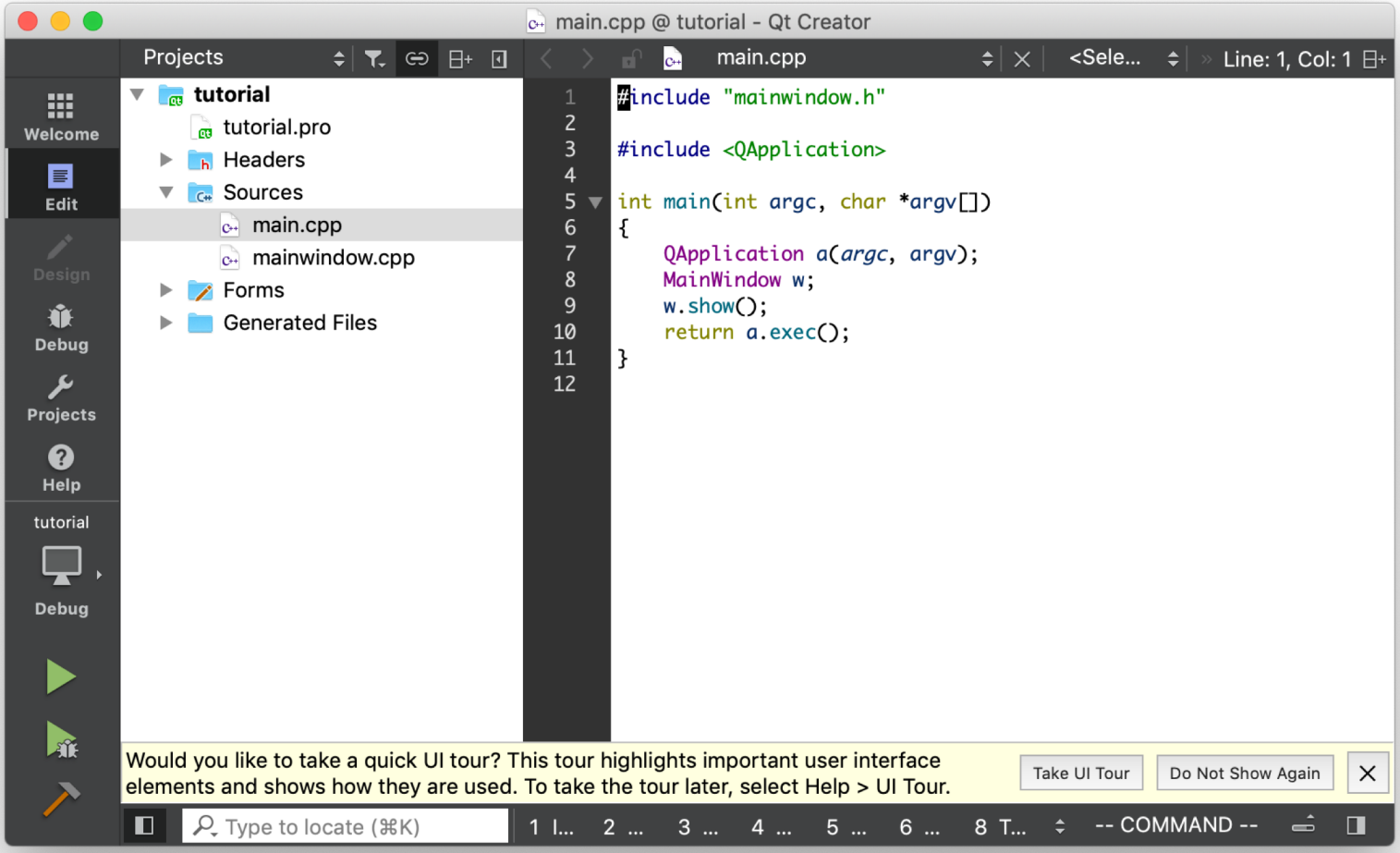
Create new project

- Use default project management setting



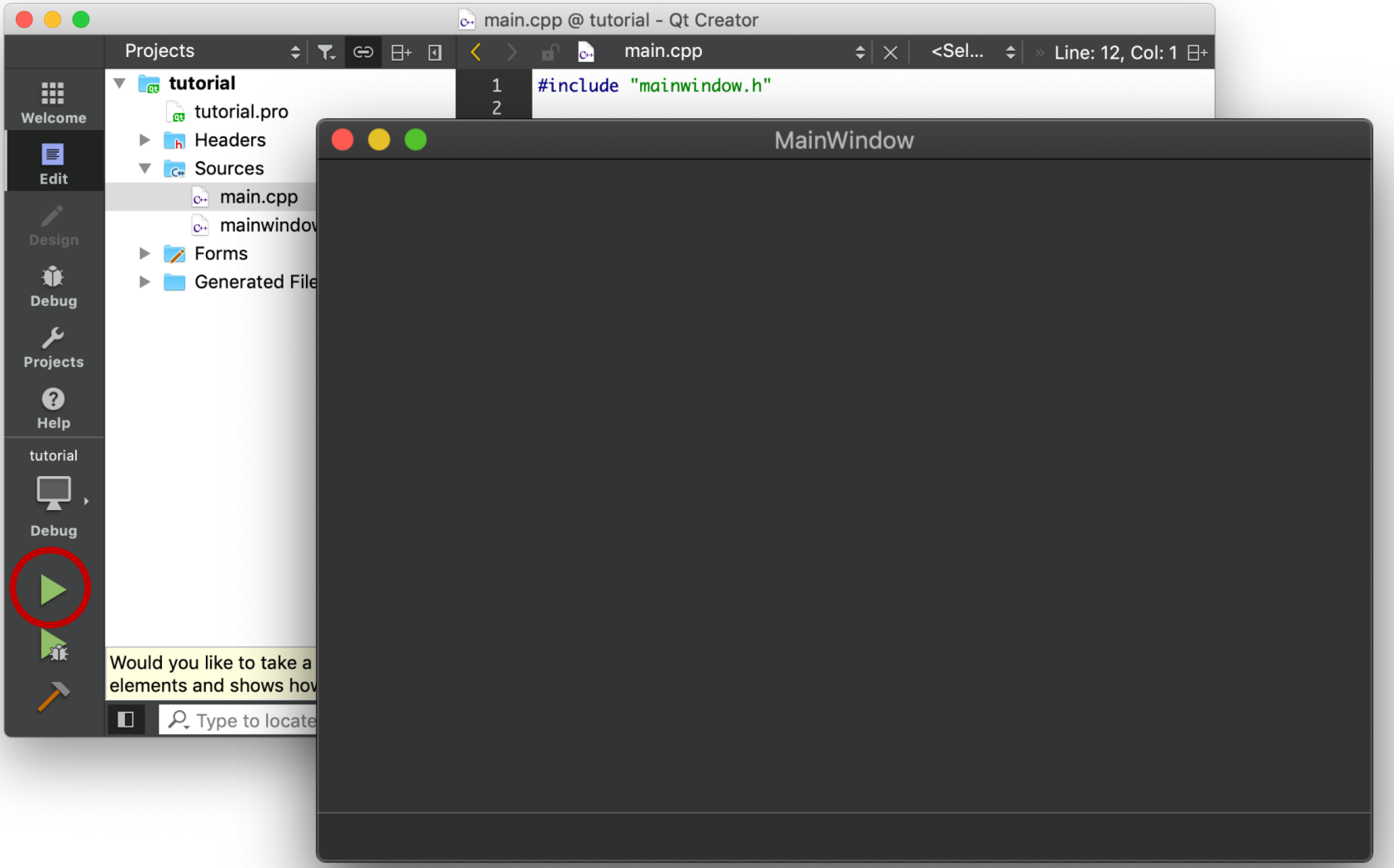
Create new project

- Qt creator should look like this:

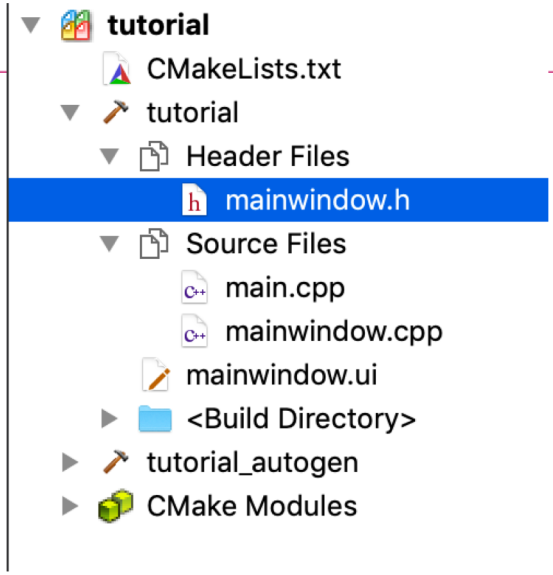


Create new project

- Press compiler button: then MainWindow appears



Create new project

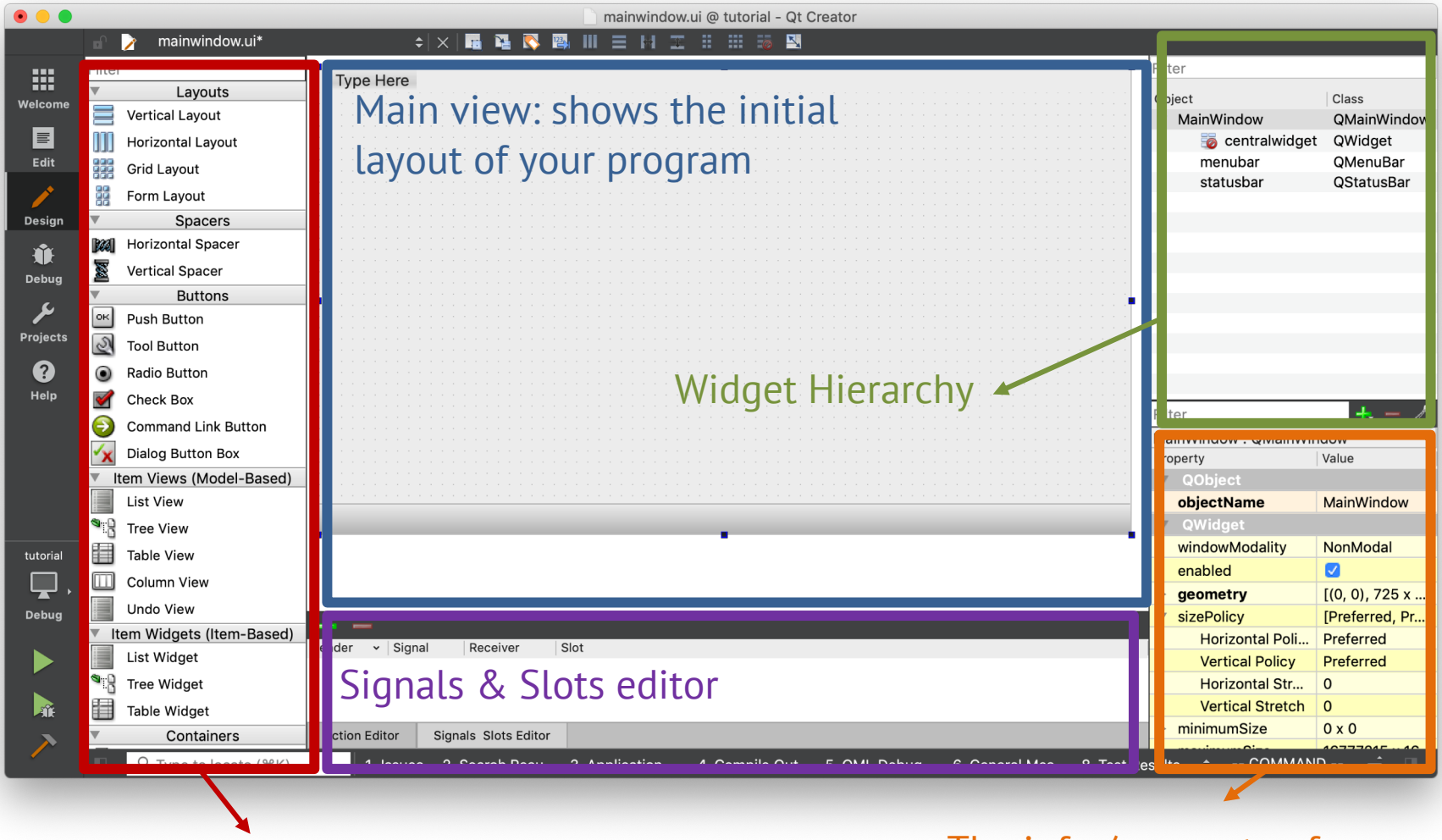


- File description
  - CMakeLists.txt
    - Project configuration file
  - mainwindow.h
  - mainwindow.cpp
    - Defines the behavior & characteristic of mainWindow
    - Inherits QMainWindow **(which inherits QWidget)**
  - mainwindow.ui
    - Defines the layout of mainWindow
    - You can modify it in ***Design*** tab
  - main.cpp
    - Open / close mainWindow / Application

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

UI modification

- Enter Design mode by double-clicking mainwindow.ui

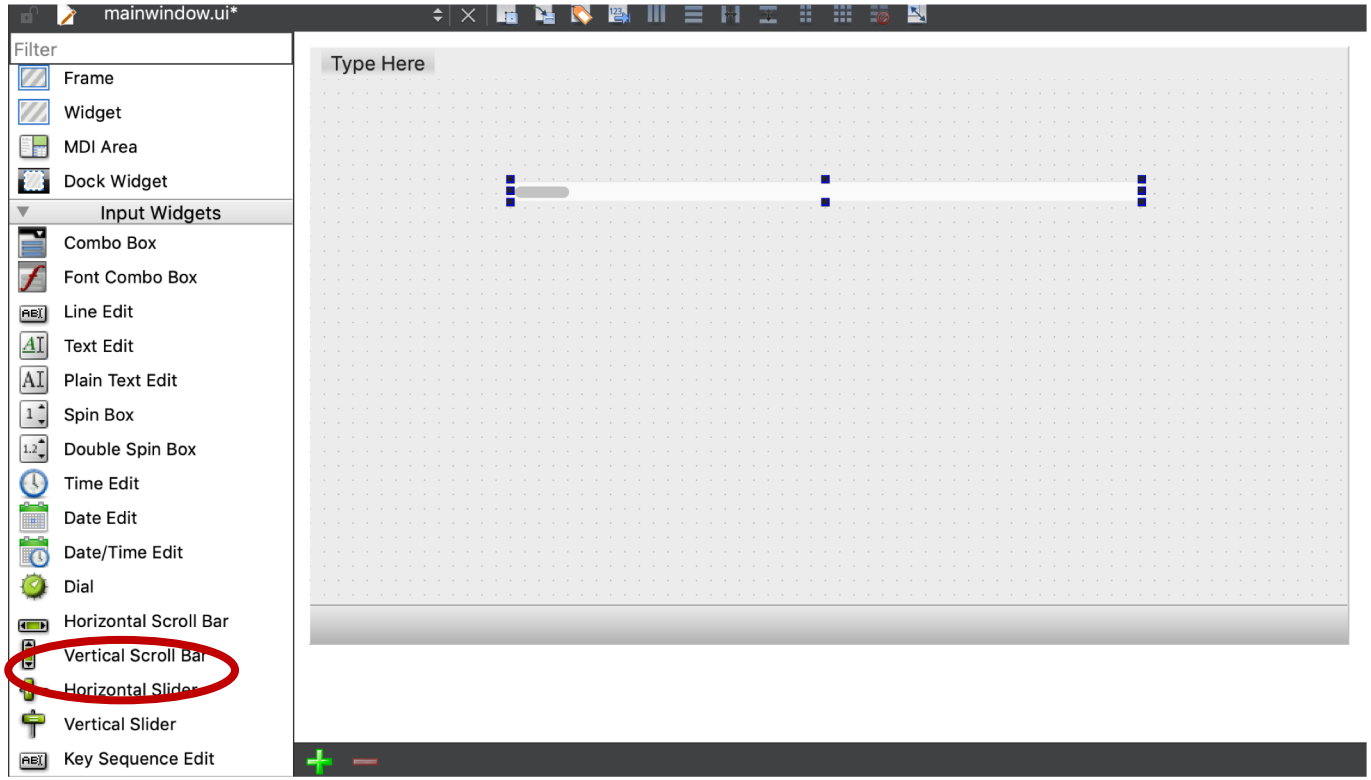


QWidget Library: You can drag & drop any widget from this menu

The info / property of currently selected widget

UI modification

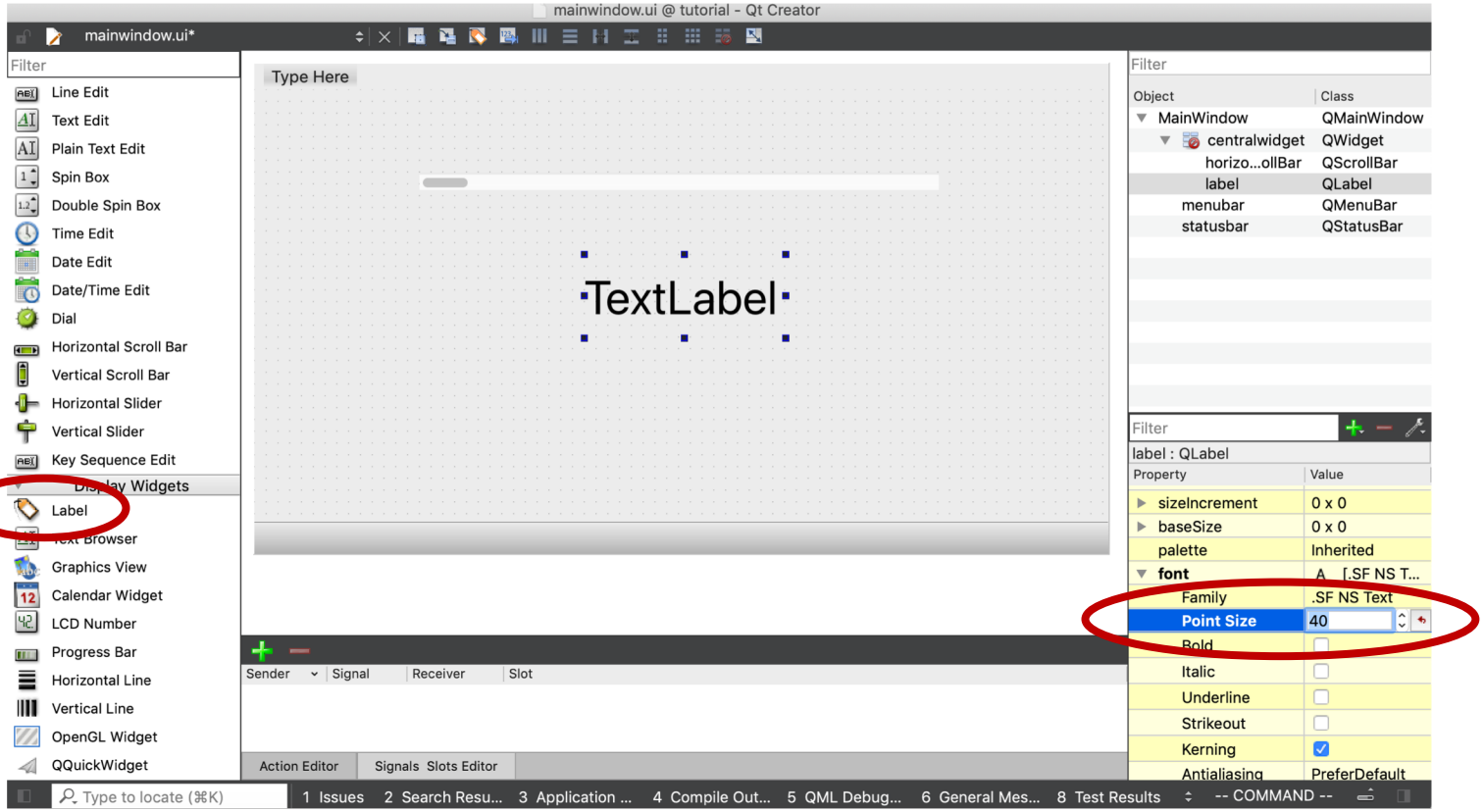
- Add Horizontal slide (Drag & Drop)
  - Also modify the location & size



  - Compile it and check the slider…nothing happens while moving it

UI modification

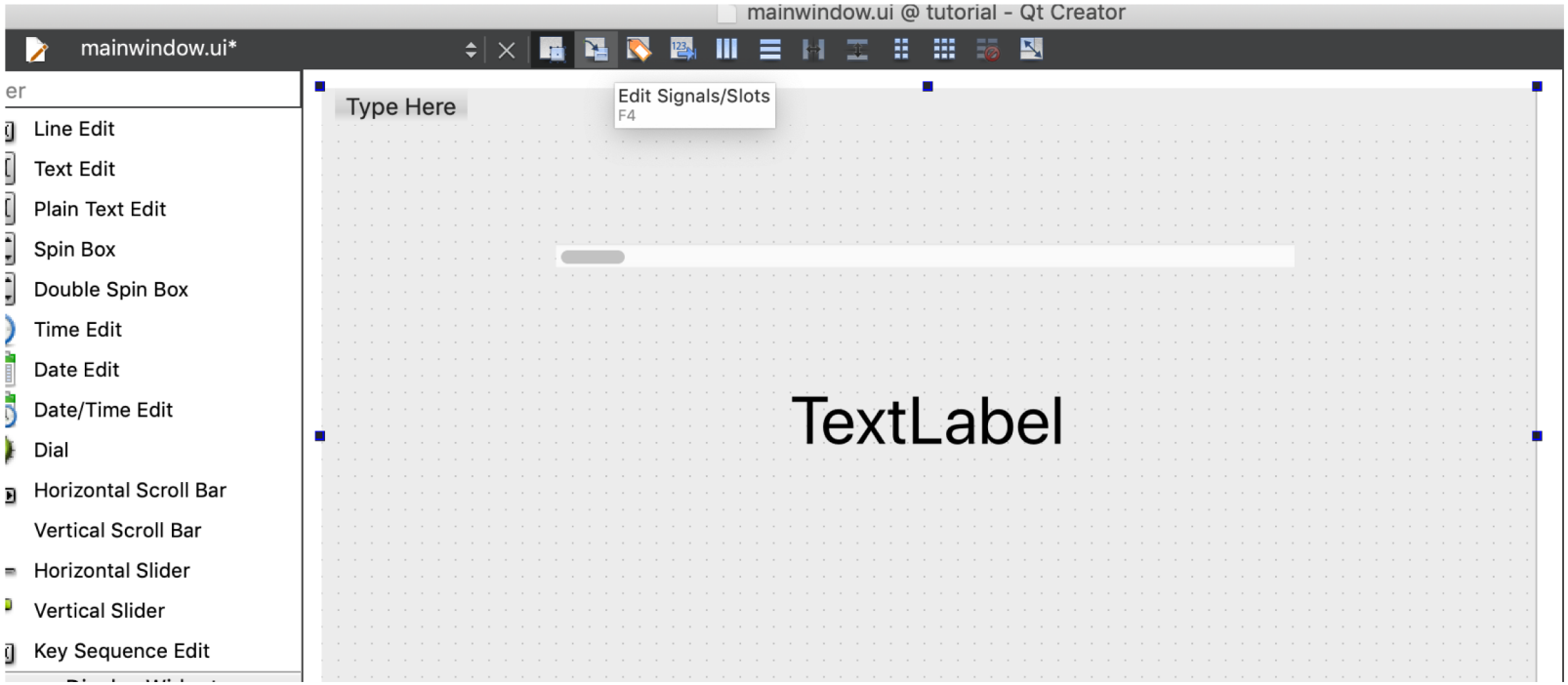
- Add text label (Label) and set its font size to 40



- Compile and check the difference!!

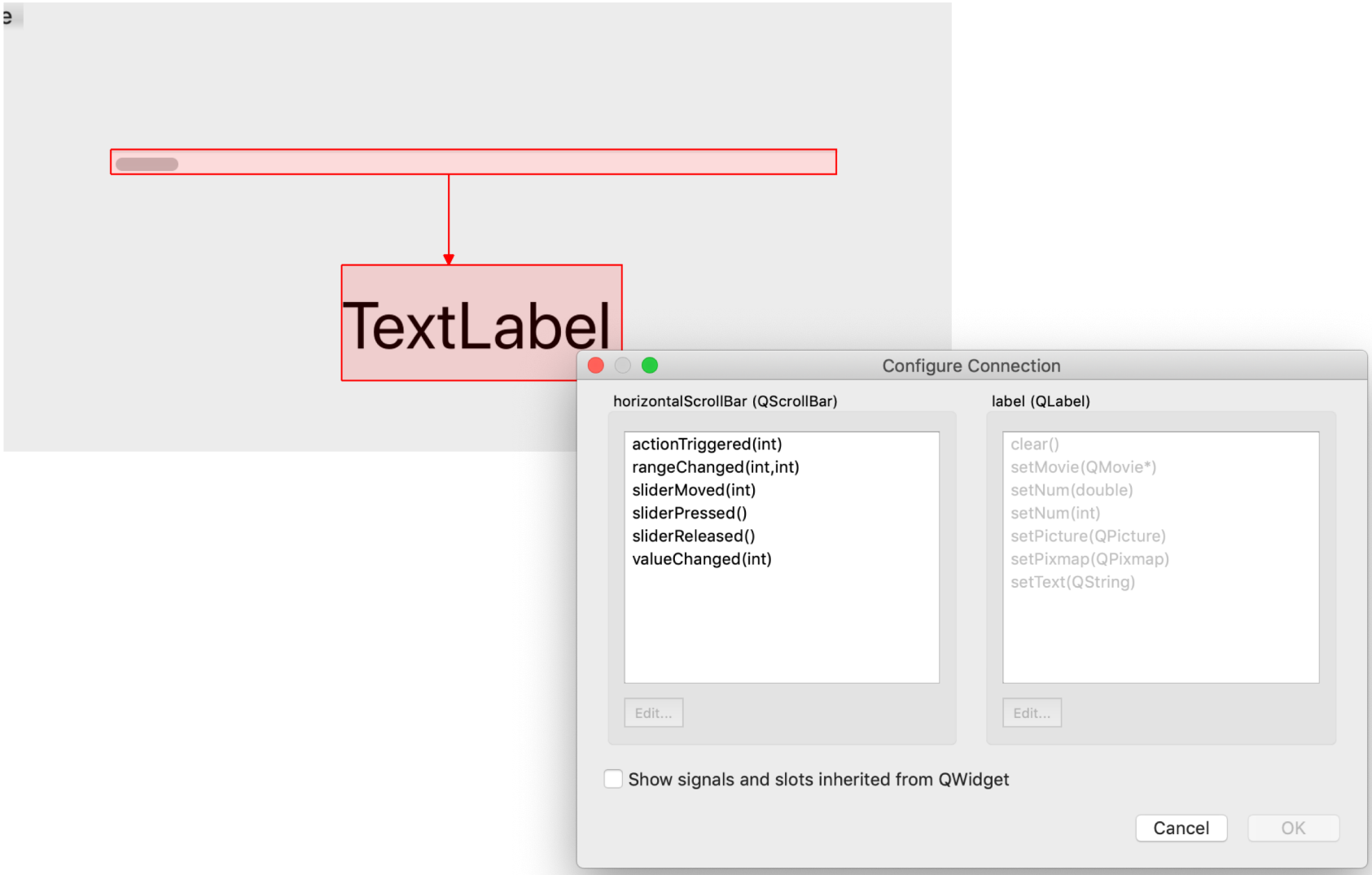
Add Signals & Slots

- Select ***Edit Signals/Slots*** button to enter editing mode



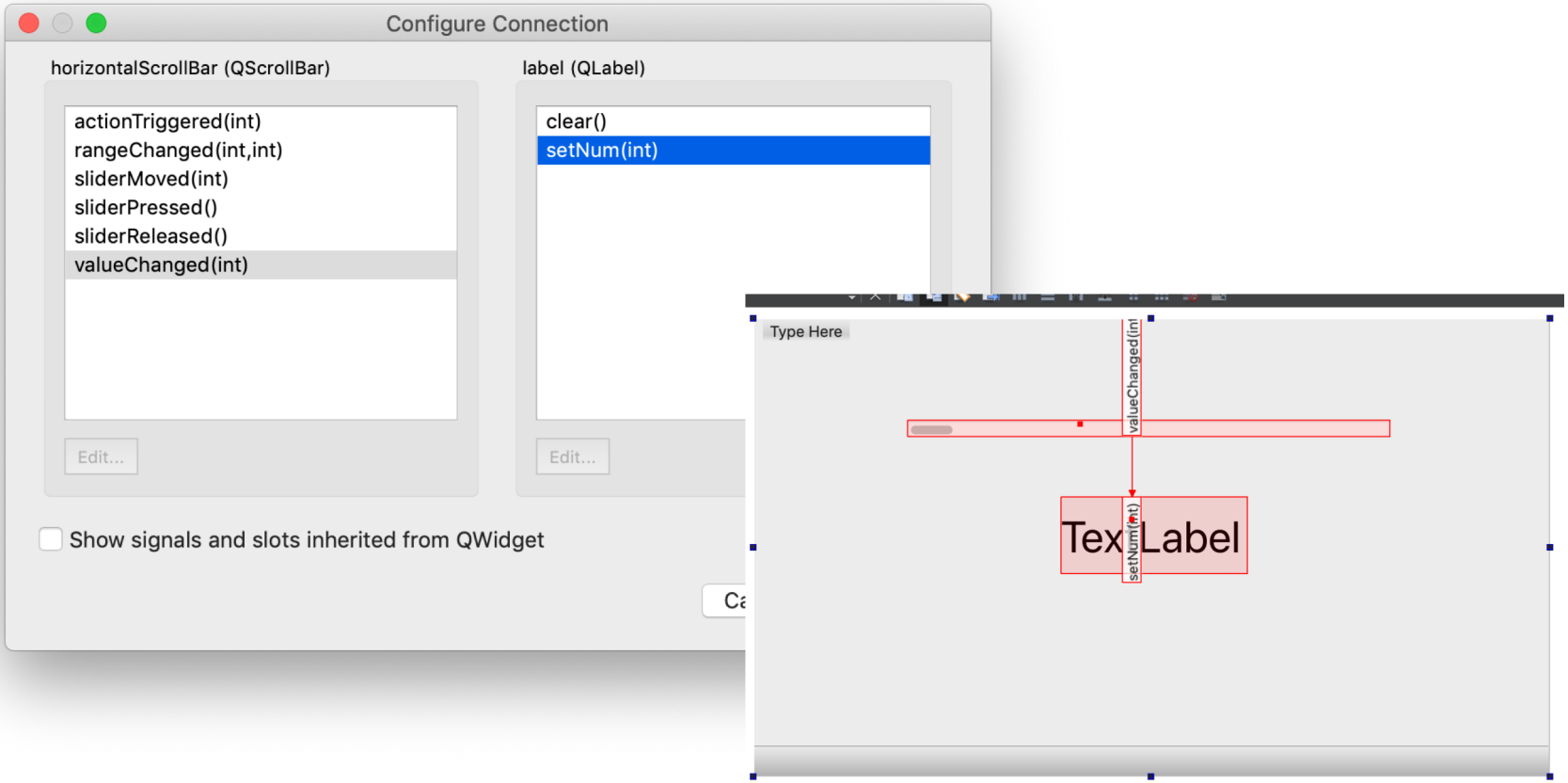
Add Signals & Slots

- Click **slider** and drag & drop it to **TextLabel**



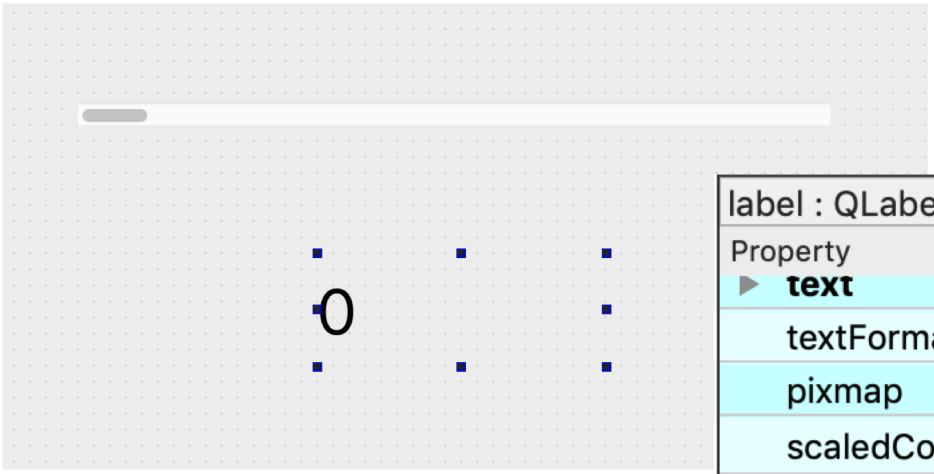
Add Signals & Slots

- Select proper signal & slot
  - signal : `valueChanged()`
  - slot: `setNum()`



Add Signals & Slots

- Set TextLabel's initial value to "0" and set center alignment

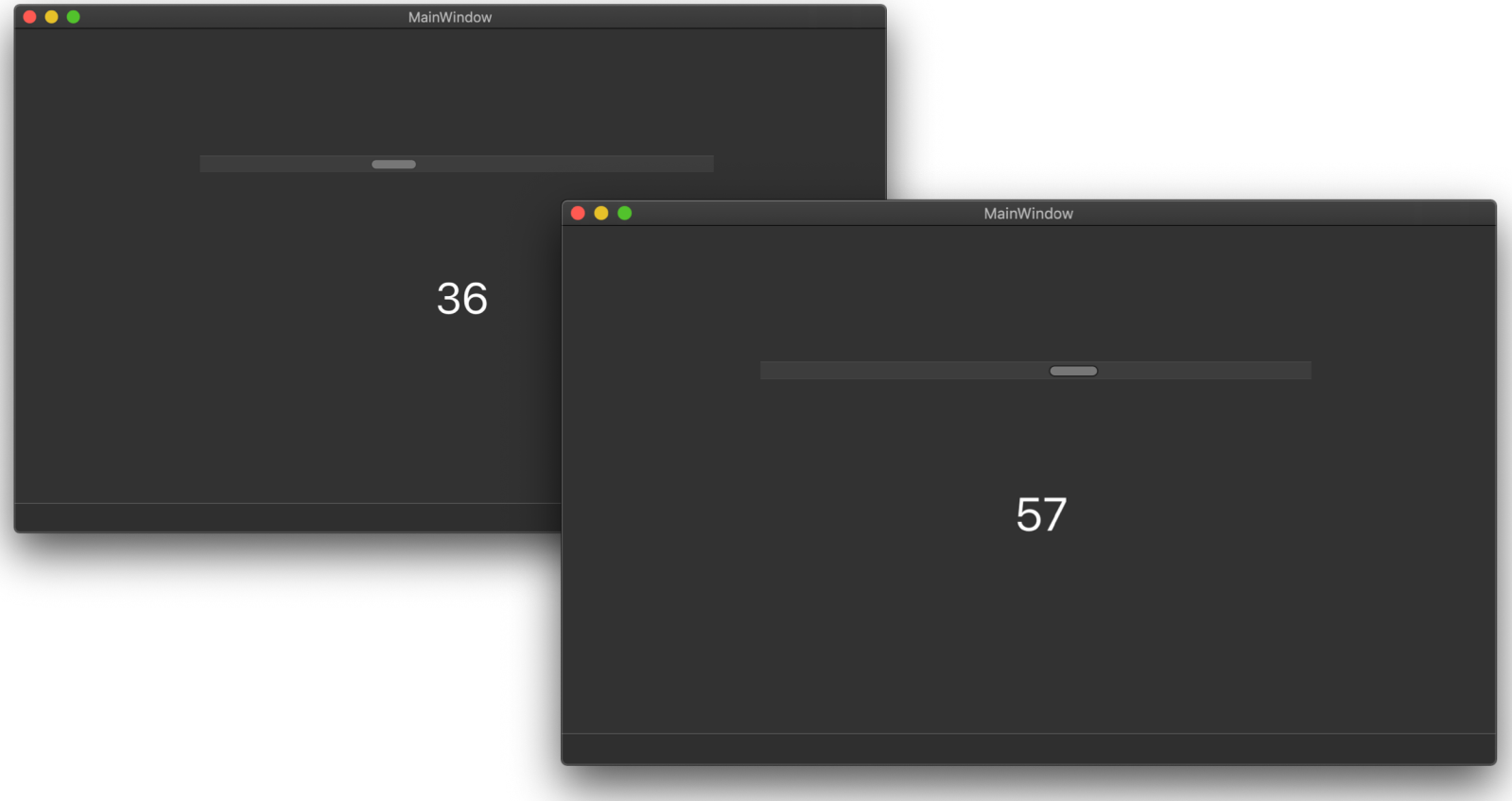


label : QLabel

Property	Value
text	0
textFormat	AutoText
pixmap	
scaledContents	☐
alignment	AlignHCenter, AlignVCenter
Horizontal	AlignHCenter
Vertical	AlignVCenter
wordWrap	☐
margin	0
indent	-1
openExternalLinks	☐
textInteractionFla...	LinksAccessibleByMouse

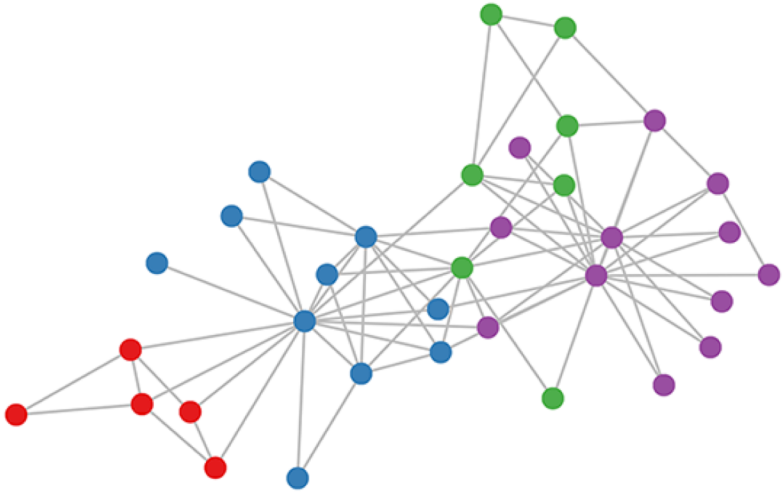
Add Signals & Slots

- Compile it!!



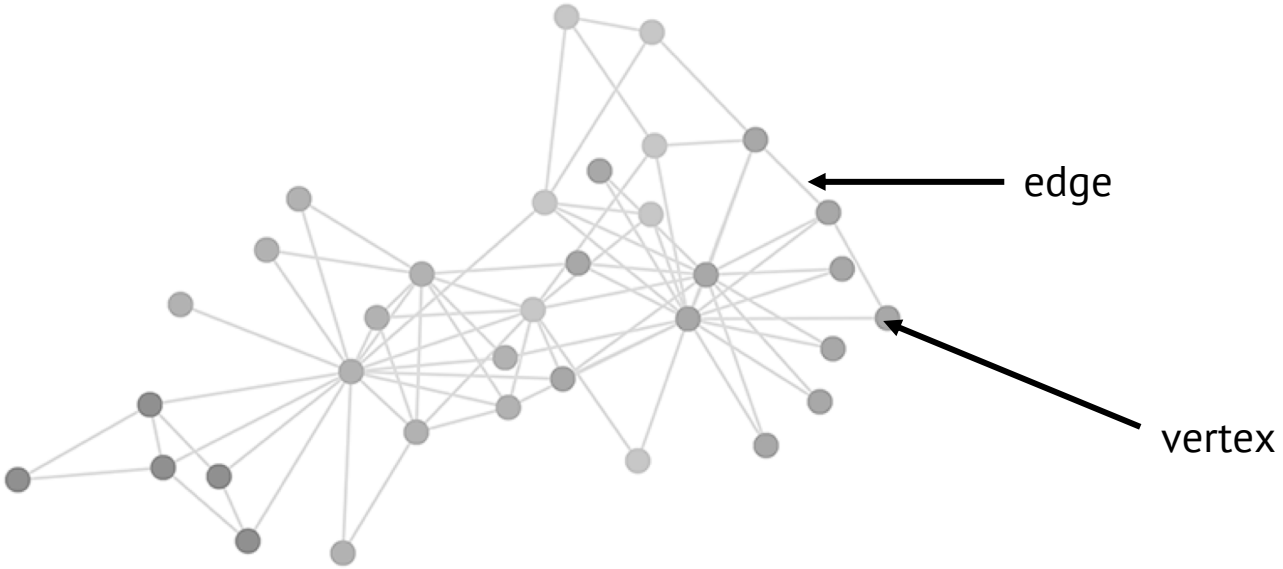
ASSN4

Graph Simulator



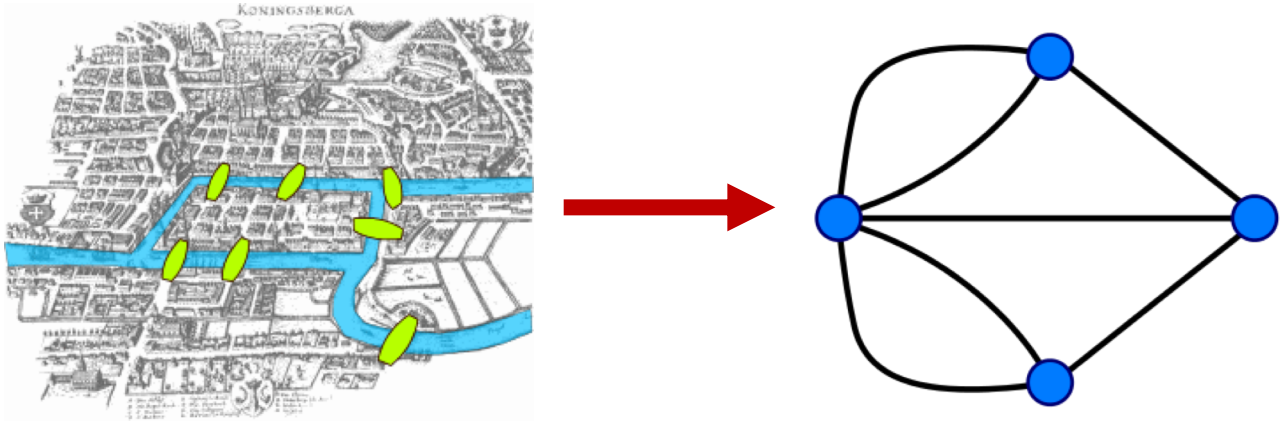
Graph Simulator

- Graph
  - A structure amounting to a set of objects in which some pairs of the objects are *related*
  - consists of ***vertices*** and ***edges***
  - *edges* connects related *vertices* (points)



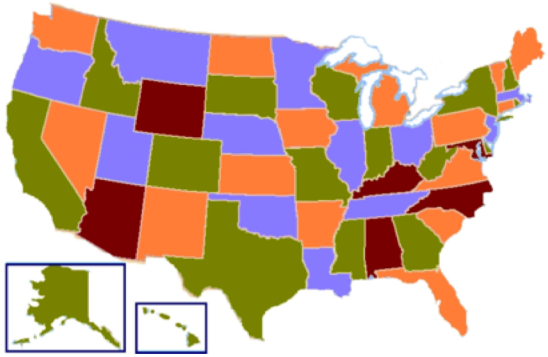
Graph Simulator

- Graph problem example
  - Seven Bridges of Königsberg
    - Can you take a walk through the town, visiting each part of the town and crossing each bridge only once?
    - town : vertices
    - bridge : edges
    - *Disproof by Euler* -> ***The birth of Graph Theory!!***



Graph Simulator

- Analyzing graph is very important in various fields
  - computer network
  - transportation
  - four color theorem
  - software analysis



Graph Simulator

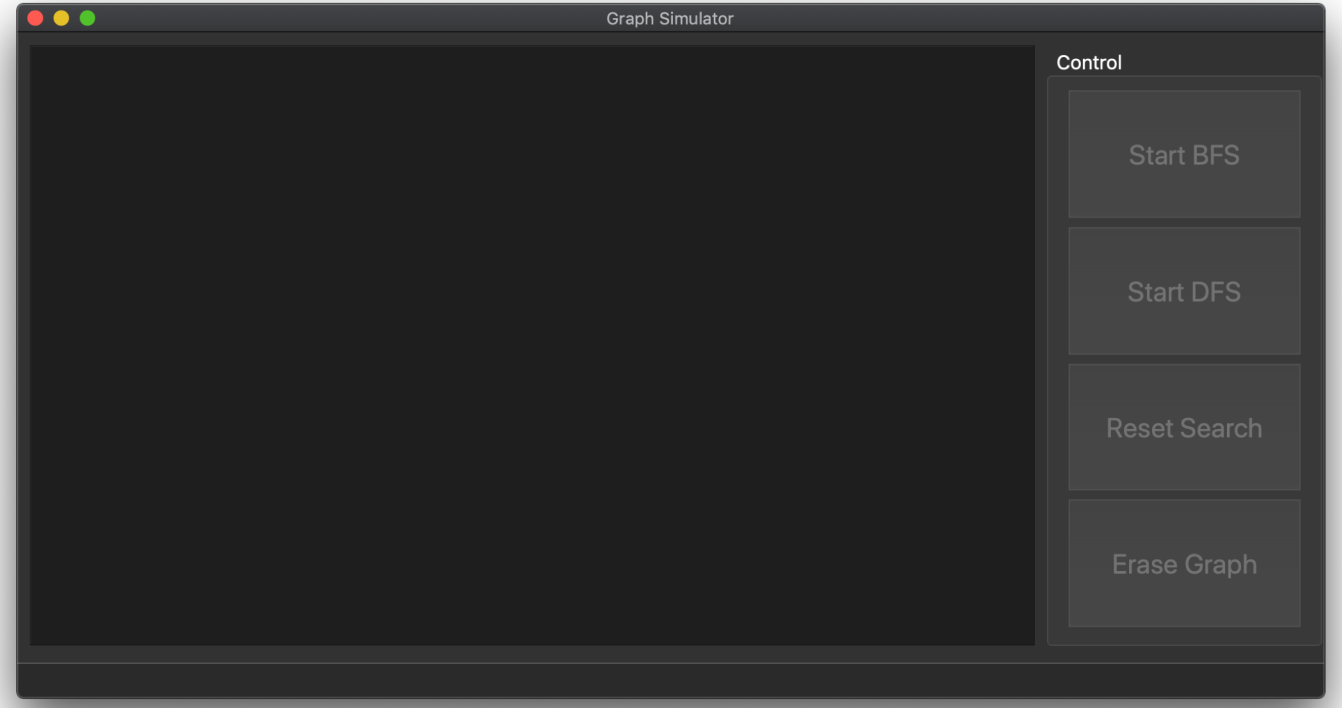
- You'll implement GUI program to help people analyze graphs
- Main Functionality
  - Graph construction (drawing)
  - Performing Depth-First Search (DFS)
  - Performing Breadth-First Search (BFS)
- Main goal of the assignment
  - understanding the basic graph structure & algorithm
  - experiencing GUI programming

Graph Simulator

- Basic Graph knowledge / DFS / BFS
  - Lots of materials on the Internet
  - Recommendation
    - Youtube
    - Korean material:
      - 1
      - 2
      - 3

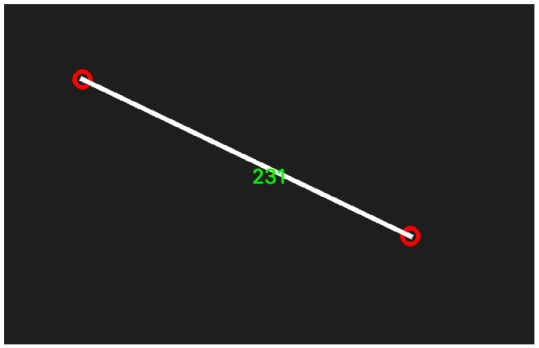
Graph Simulator

- UI
  - Main Graphics View
  - Control tab consists of Four buttons (which are initially disabled)
  - Highly recommend to *copy* the UI



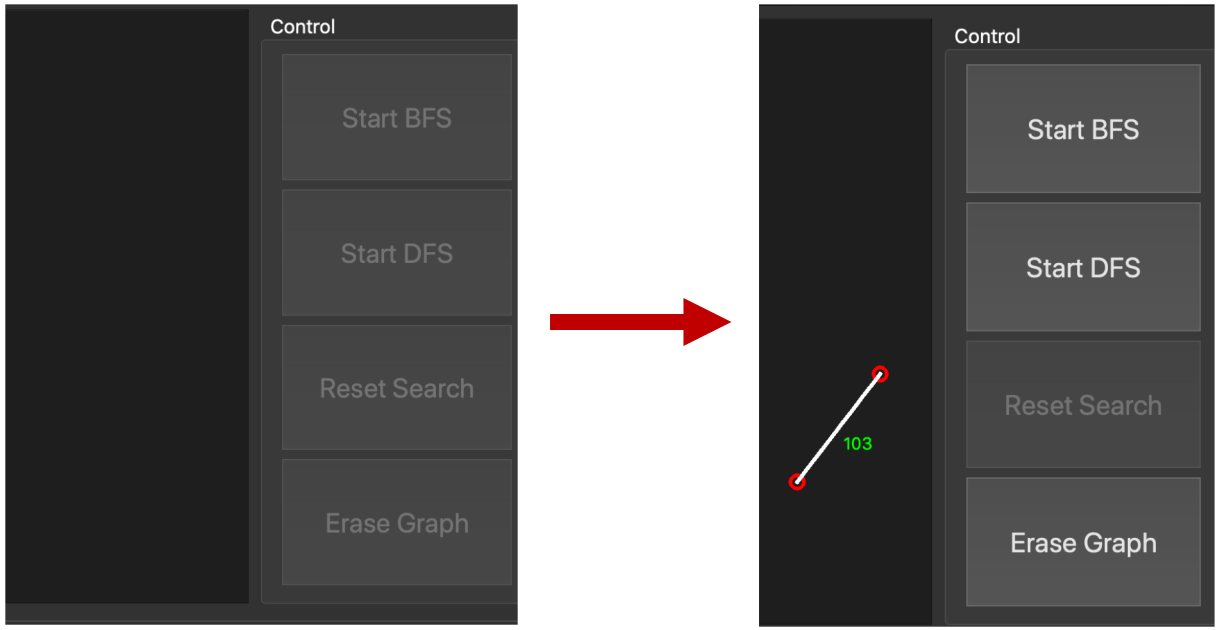
Graph Simulator

- Drawing Graph
  - create vertex by double-clicking main view
  - connect vertices (create an edge) by drag & drop
    - edges are located only b/w vertices
    - edges should not be created in unreasonable positions
    - the integer distance b/w two vertices connected by edge also should be identified (recommend to locate it in middle)



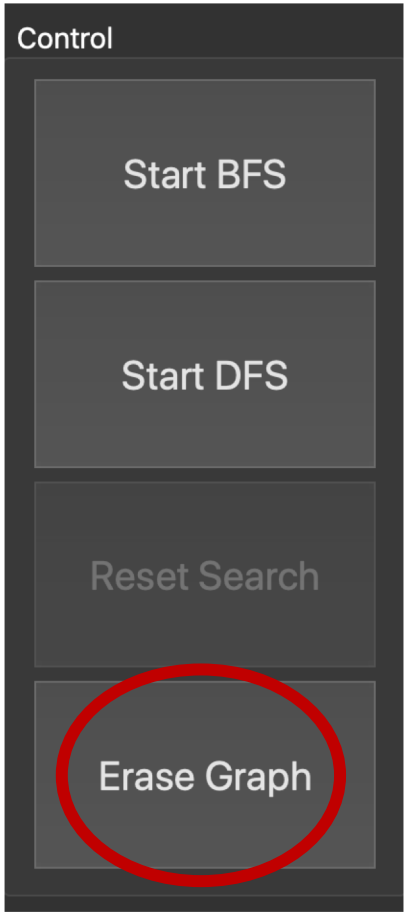
Graph Simulator

- Drawing Graph
  - initially, all buttons are disabled
  - when the user starts drawing graph, 3 buttons:
    - bfs, dfs, erase
    - become enabled



Graph Simulator

- Drawing Graph
  - Users can erase entire graph by clicking *erase* button
  - The button resets the program
  - (turns to initial status)



Graph Simulator

- Performing DFS
  - If the user clicks *Start DFS* button,
    - the program enters ***DFS mode***
    - DFS starts on arbitrary vertex (any vertex is okay!)
    - and the button label becomes *Progress DFS*
    - DFS progress is shown by changing vertex color
  - If the user clicks *Progress DFS* button,
    - if (DFS is not finished) : change the color of next vertex in DFS
    - else
      - go back to initial state (right before DFS)

Graph Simulator

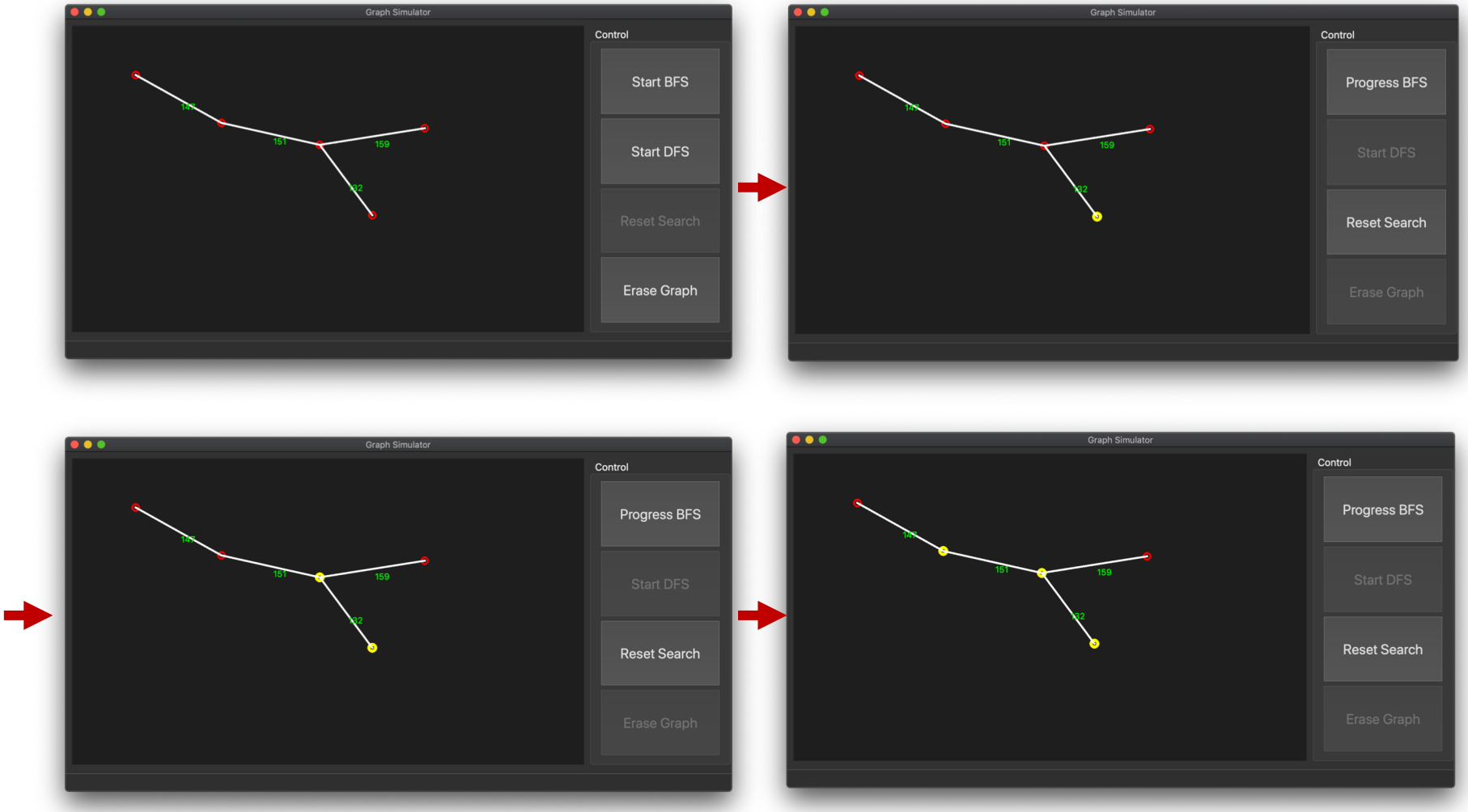
- Performing DFS
  - During ***DFS mode***, following functionalities need to be forbidded
    - entering *BFS mode*
    - drawing new element
    - erasing graph
    - Therefore, the program should disable…
      - *Start DFS* button
      - *Erase Graph* button

Graph Simulator

- Performing DFS
  - while performing DFS, the user can *reset* the progress
  - Therefore, when the program enters ***DFS mode***,
    - *Reset button* should be enabled
  - If the user clicks *Reset* button
    - The program ends the progress
    - and go back to initial state(right before the progress)

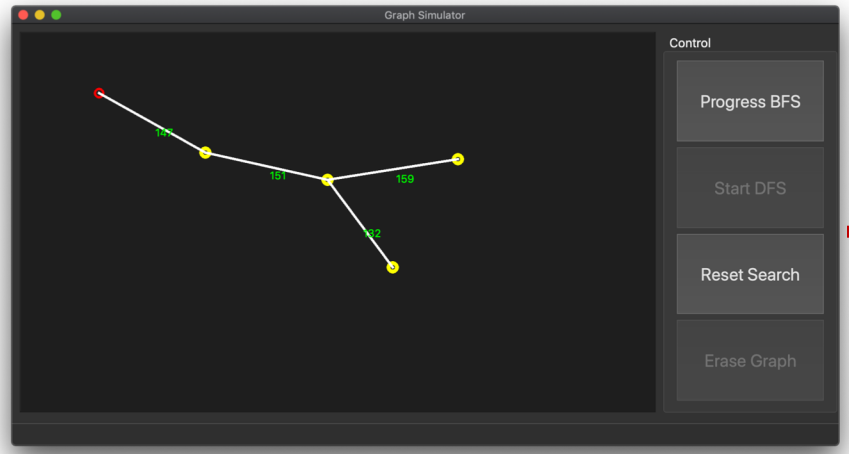
Graph Simulator

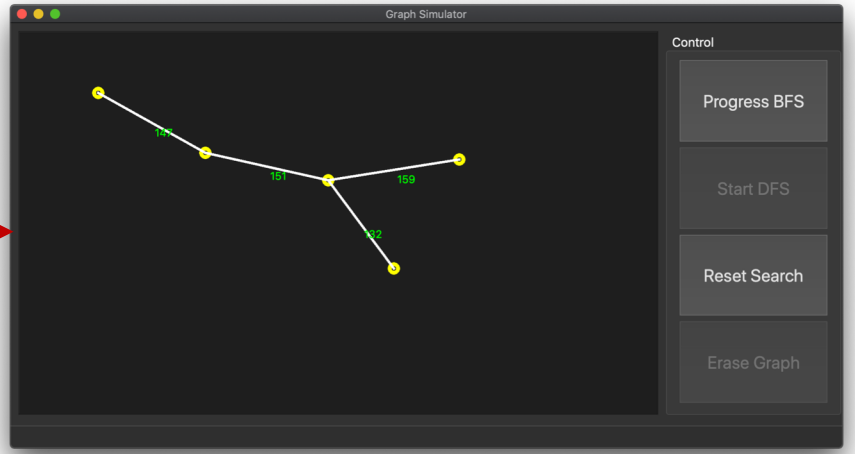
- DFS progress

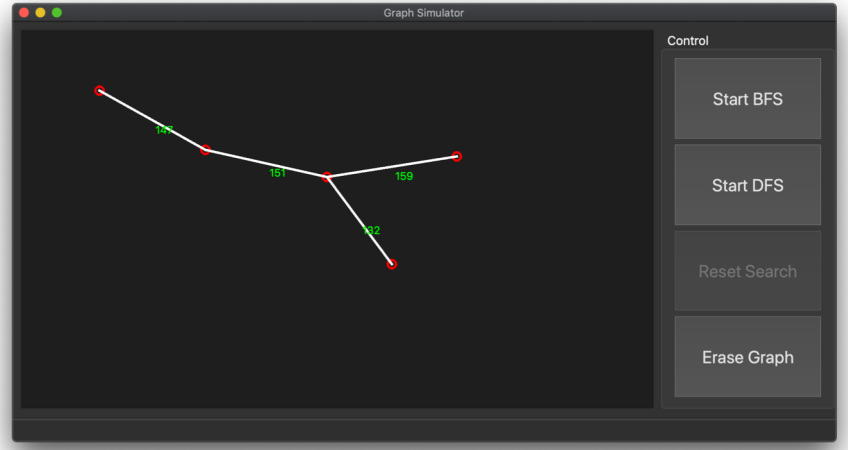


Graph Simulator

- DFS progress







Graph Simulator

- Performing BFS
  - Enter by clicking *Start BFS* button
  - almost same to DFS, except that the progress follows BFS
  - should use different vertex color to identify BFS progress

Graph Simulator

- Demo will help you!!
  - YouTube

Thank you!!

contact: hj@hcil.snu.ac.kr