**UI**

Our project is based on Qt. We use QMainWindow to display our UI. The UI consists of 4 parts: Title, Menu bar, Desktop, Status bar.

**Desktop**

Menu bar: Menu bar is just under the title and above the desktop. There are two widgets in the Menu bar: “System” and “Applications”. System widget contains two actions: “shutdown” and “set wallpaper”. After clicking “shutdown” action, the OS system will shut down and kill all the processes. After clicking “set wallpaper” action, the system will ask the user to select a new image (default in .jpg format) as the wallpaper of the desktop. The default size of the desktop is 1600 x 900 pixels, so if the user wants to fill the whole desktop, the new image should have the same size. Application widget contains 5 applications: Clock, Calculator, File Brower, Memory Monitor and Text Editor.

Status bar: Status bar is under the desktop, which contains the real time in format “dd.MM.yyyy, h:m:s ap” (e.g. 31.05.2020, 1:00:0 pm). The program includes <QTimer> and <QDateTime> to implement the local time display. QDateTime is used to get the current time in the local time zone. And QTimer is used to update the time display at a regular interval (1000ms).

****

**Clock**

First, same as above, we should get the current local time.

Then, the program includes <QPainter> to draw the clock dial and hands.

And same as above, we use <QTimer> to update the clock every second.

Detailed ideas of painter:

1. Store the shape information of each hands in QPoint[] type.

2. Calculate the angle between two scales and use for loop to draw the dial.

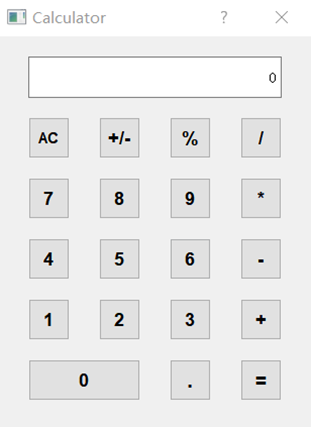
3. Calculate the Angle of rotation of each hand per second.

4. Draw the initial clock with the information of current local time.

5. Update the clock every second with the information of (3).

**Calculator**

The implementation of calculator is the same as the proposal. User can conduct the simple calculation in the simulator. The calculator can implement basic digit arithmetic, including addition, subtraction, multiplication and division. User can reset the initial value at any time. In this calculator, the integer precision is 15 digits, which is set within the program.

****

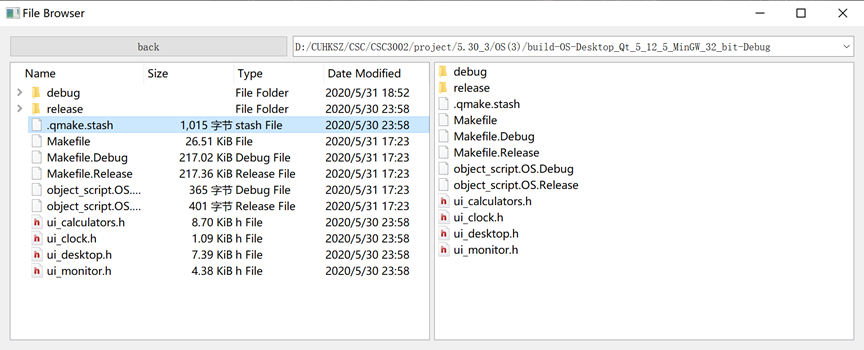
Show the functions of some buttons here. For example, the button “AC” is for set the value to 0, the button “+/-” is for changing the sign of the value, and the button “%” is for multiplying the value by 0.01.

In the procedure of implementation, I get the form of calculator from calculator which comes with computers, and get the necessary resources from online tutorials. The difficult part of my work is to display in-process result on the screen during the procedure of successive operation (i.e. press the digit buttons and operator buttons one after other without pressing “equal” button). In order to overcome this problem, I set global variables to record the times of operation and the kind of last operation, these variables can help to decide whether it is suitable to display memory result on the screen. In details, if the last operator is not “equal”, and the program detects the next operator is pressed, then just show the memory results on the screen.

I learn how to design UI and related buttons in this project, also, connection with certain events and button clicking, operation within multiple windows. In general, the basic implementation of digit arithmetic is the knowledge from class, while the designing of UI, popping up a new window, connecting buttons with events, and operating within multiple windows are new knowledge learned by myself.

**File browser**

In the file browser, we attempt to list all the file in different ways and let the user to manage the files including reading the file, opening the file returning to the upper level of the path. Our browser window follows the concept of simplification. We only show the most significant information to the uses. It only contains a back button, a ComboBox to show the exact path and one tree on the left one table on the right to show the files in the system. User can use right click to the file for further operational features, for example, renaming, opening, deleting and copying. I have designed a window to browsing the system adding features like right click reaction, file layout in tree and file listing in table.

****

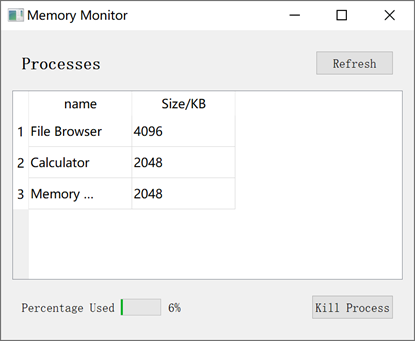
I search from the Internet seeking for the tutorial of the implementation of the file system in qt creator. I found that there exist several totally different ways to implement this. But each of the tutorial can offer me some thinking of how to run the program.

In the end, I use QTreeView and QListView to implement the file browser. The knowledge I learned from the websites are more likely to be fragmented knowledge. I found some tutorial in CSDN. One is about searching a file in the computer, that gives me the idea of using the widgets. Another tutorial is about the splitter, so I use splitter to separate the windows as well.

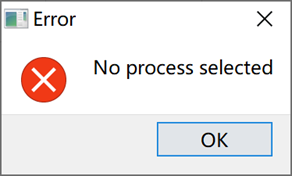
**Memory Monitor**

To implement this part. We first implement two classes named MyWidget and MyMainWindow inherit QWidget and QMainWindow respectively. In these two classes, we overwrite the close event, add a new Boolean variable and two new methods. The Boolean variable “isOpened” is used to check whether the window is closed or not and it is set to be true when an object is created. getName method return the window title of the app. getSize is a virtual method which returns the memory used of the app. In the close event, we set isOpened to be false if the event is accepted.

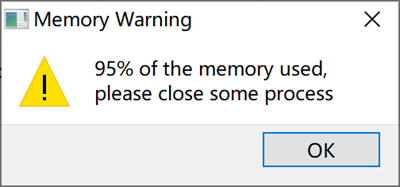
We use two std::vector called WidgetList and MainWindowList to store the pointers to the apps. When the memory monitor is created, it will load all the apps, then display their name and memory occupation in a QTableWidget as showed in the next graph.

****

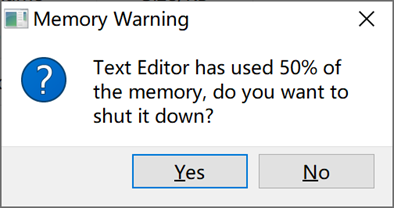
We can select certain cells, and clicked kill process button to shutdown the app selected. The refresh button is used to refresh the content in the QTableWidget. When an app is closed, we can click the refresh button to remove the corresponding row in the QTableWidget. Also, the monitor will auto refresh once per second. The row selected is set default to -1 which means no row is selected. When we click the kill process button under this circumstance, the monitor will raise an error. Also, after each click of the kill process button, the row selected will be reset to -1.



When the total memory used exceeds 95%, the monitor will raise a warning to remind the user to close some apps.



When one app occupies 50% of the total memory, the system will ask the user whether to shut it down or not.

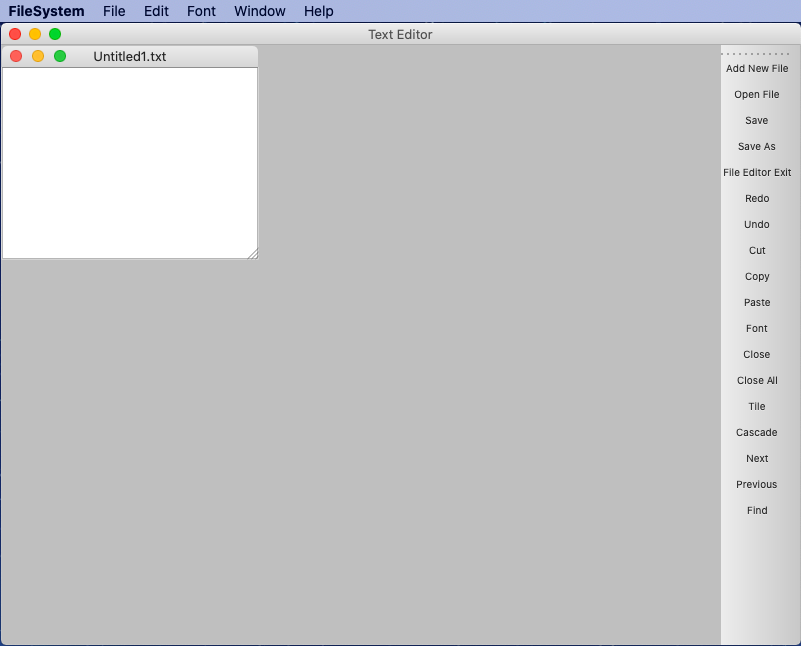


When we use the “new” method to create an app, the system will add the pointer to one of the two vectors. If we have already created the memory monitor, it will calls the loading functions of the monitor. We only allow one monitor to be created. It can be showed or closed but cannot be destroyed until the system shutdown.

When we use the click shutdown on the desktop or close the desktop window, the system will traverse all the pointers in the two vector and delete the object they point to.

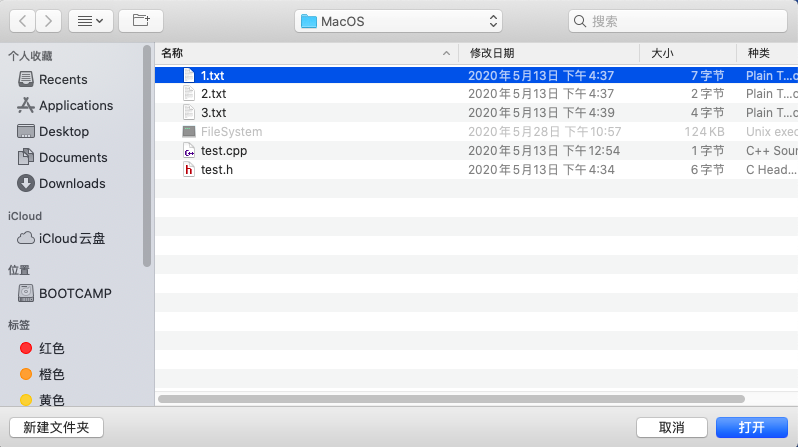
**Text Editor**

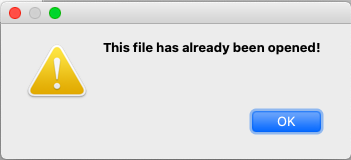
This description is mainly based on Mac OS system. In Windows system, due to the facts that some build-in functions of Qt works differently in different systems and each system may have different resolution ratios, the interface may seem a little bit strange but the basic functions of this text editor work well in both systems.

The figure below shows the interface of this text editor. It consists of three components: a menu bar, a toolbar and a main window. There are five sections in the menu bar: file, edit, font, window and help. Each of them has some functions inside. And these functions are also displayed at the right side of the interface for the ease of use. All the changes will show in the main window.

Then, let me introduce these functions one by one. If user clicks “Add New File”, a new blank text box will be displayed in the main window. If it is the first text box user creates by clicking “Add new file”, the filename will be “Untitled1.txt”. If it is the second text box user creates by clicking “Add new file”, the filename will be “Untitled2.txt” and so on. The filename shows at the title bar of the corresponding text box.

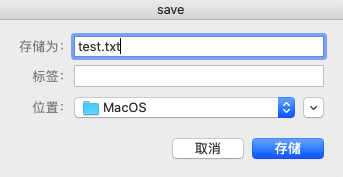
If user clicks “Open File”, a dialog window will pop up which shows below. It contains all the files user has saved in this operation system. User can choose and open files with “.txt”, “.cpp” and “.h” suffix. If the file has already been opened in the file editor, a warning prompts will show to remind user.

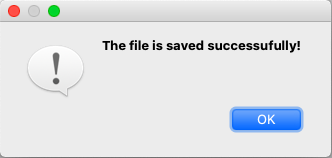




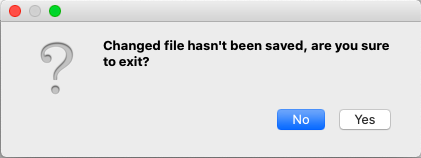
If user clicks “Save”, there are two possibilities. If the selected file is newly added, it hasn’t been saved before so it does not have a corresponding path. It will have the same effect as clicking “Save As”. If the file is opened but not newly added, it already has its corresponding path. After clicking “Save”, next time when user opens this file, all the changes are saved.

If user clicks “Save As”, a dialog window will pop up which shows below. User should enter the filename in the first line and click “Save”. Then a new file will be saved in this operation system and can be found in the file browser. The filename which shows at the title bar of this text box will be changed to the filename user enters. If the file is saved successfully, a prompt will show to remind user.



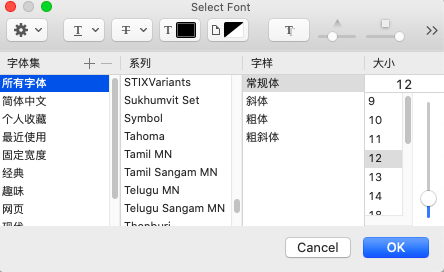


If the user clicks “File Editor Exit”, if there is no file or all the files are saved, the file editor exits. If there is one file that hasn’t been saved, a prompt will show to remind user. The same situation will happen if user clicks the close button.

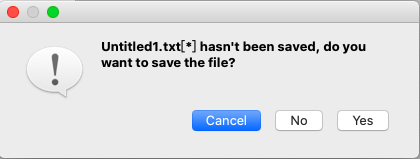


For “Redo”, “Undo”, “Cut”, “Copy” and “Paste”, the usage of them are clear and there is no need to explain them.

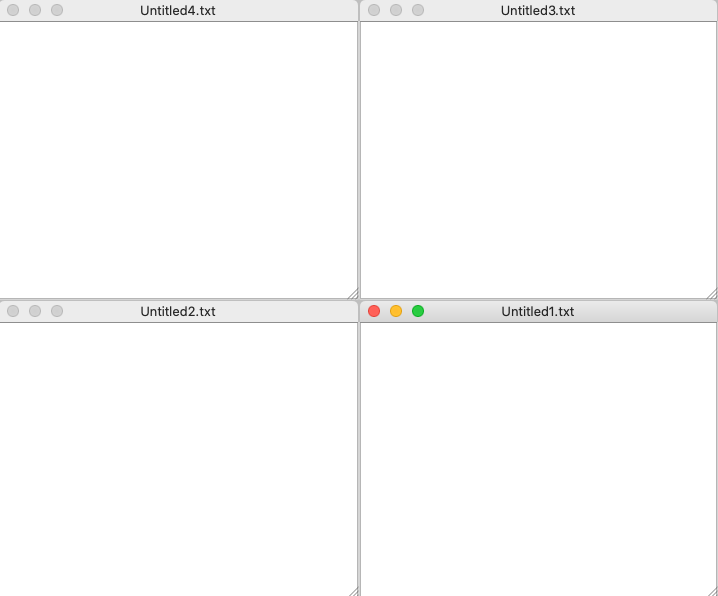
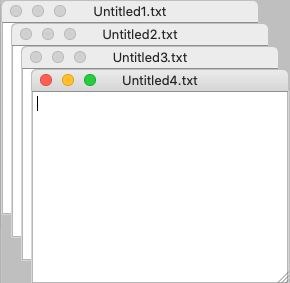
If the user clicks “Font”, a dialog window will pop up which shows below. User can select font type, font size and whether to be bold formatting or Italic formatting. After clicking “OK”, if the user chooses some words, these words will change. If the user doesn’t choose any words, the words user inputs later after the cursor will change to the format the user sets.



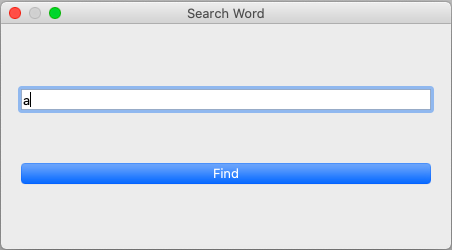
If the user clicks “Close”, the current active text box will be closed. If the current active text box is not saved, a prompt will show to remind user. If user clicks “No”, it will not save the text and the text box closes. If user clicks “Yes”, it will save the text and the text box closes. If the user clicks “Close All”, it is the same as clicking “Close” to all the opened text box one by one.

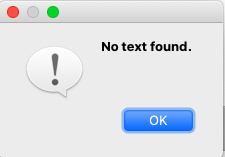


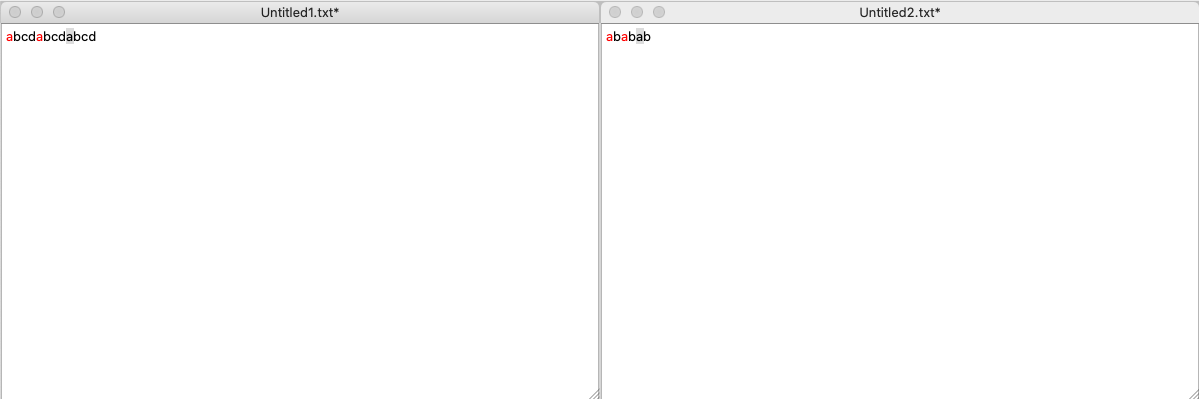
If the user clicks “Tile”, the way opened files being displayed shows below. If the user clicks “Cascade”, the way opened files being displayed shows below. If the user clicks “Next”, the active text box will be the next one. If the user clicks “Previous”, the active text box will be the previous one.



If the user clicks “Find”, a window will pop up which shows below. User can enter the character or the word he/she wants to find and click “Find”. If there is not such a character or word in any of the text box, a prompt will show to remind user. If there is, all the corresponding character or word will be marked red except the last occurrence of the character or word in each text box will be shaded.







Lastly, if the user makes some changes to one text box, the filename which shows at the title bar of this text box will be marked a “\*”. If the text box is saved, the “\*” will disappear.

I finish all the functions I mentioned in the proposal with some extensions. This part is totally new to me so I learn a lot from some videos. For example, <https://www.bilibili.com/video/BV1g4411H78N?from=search&seid=13110346917756597501>. I think my work is great because I implement many functions and the text editor is powerful. I learnt a lot from this project, some of the stuff are learnt outside, for example signals, slots and many build-in functions of Qt. Some of the stuff are learnt in the classroom, for example for loop and while loop, map and pointer.

During this process, I encounter many difficulties. At first, I don’t have any ideas about how to implement it. I watched a lot of videos and know how to use signals and slots, “QMainWindow”, “QMdiArea”, “QTextEdit”, “QMdiSubWindow”, “QDialog” and so on. Later, I also encounter many difficulties. For example, in Qt, I can only use the function activeSubWindow() to get the active QMdiSubWindow, but there is no function to get the active QTextEdit. I overcame it by using a map, QMdiSubWindow is key, QTextEdit is value. In “Add New File” and “Open File”, the program inserts a pair to the map. In other functions, use get() function to get the active QTextEdit. For example, in “Find” function, the build-in find function can only show the last occurrence of the character or word, but I want to show all the occurrences. I overcame it by using a text cursor. Once it finds a corresponding character or word, WordUnderCursor function can let me do operations to the character or word.