**QT Coding Convention**

**Basic coding convention from Nokia and QT official web**



**Additional conventions which are not mentioned by Nokia and QT**

**1：Add file and version information.**

At the beginning of header files and .cpp files, need to use comment make simple notes about Copyright, Function description ,Version, Author,ect.

／\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*　Copyright(c)2010 Teleca company.

\*　All rights reserved.

\*

\*　Name：filename.h/ filename.cpp

\*　Function：Brief description files contents and features.

\*

\*　Version：1.1

\*　Author： Johnny Liu

\*　Date： June 28, 2010

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**2. Fucntion header comments: Features, Parameters and Return value**

Every program shoud begin with a brief comment to explain the features of function. And all of parameters and return values should add comments for brief description too. E.g:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Name：\*\*\*

\* Function：Briefly describe the features of function

\* Parameters list：param1——Description；

\* param2——Description；

\* param3——Description；

\* Return value：Briefly describe the return value

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**3. Variabless definition and comment.**

1)The definition of variabless use camel style(The compound words or phrases in which the elements are joined without spaces, the first letter is use low case and the words behind are all start with upper case). The name should use “Noun” or “Adj. + Noun”. E.g.

float value;

float oldValue;

float newValue;

All static variables and global variables are need comments. And important local variables also need commments.

2)The usual controls variables, name are use “controls abbreviation”, “Verb + controls abbreviation” or “Adj + controls abbreviation”.

The usually used controls abbreviation form are **in Appendix\_2 at the end**.

**4. The principle of variables and function names**

1)Variables name are use Hungarian notation, the details of Hungarian notation **please check the Appendix\_1 at the end.**

2)Global variables and class data members use long name, local variables use short name. The name of class data members beginning with m\_, static variables beginning with s\_, global variables beginning with g\_.

* The prefix of class data members start with m\_(‘m’ means member) can avoid confuse with another variables. E.g.

void Object::SetValue(int width, int height)

{

m\_width = width;

m\_height = height;

}

* The prefix of static variables is s\_(‘s’means static), E.g.

void Init(…)

{

static int s\_initValue; // static variables

…

}

* If we have to use global variables, the prefix of it is g\_ (means global), E.g.

int g\_howManyPeople; // global variables

int g\_howMuchMoney; // global variables

**Notes:** We should try to avoid use global variables as we can.

3)Local variables should simple and easy to understand, use common variables, E.g. nCount，i，j，k，len, pos, ect.

4)In program, if two or more classes have same abbreviation, like QToolBar and QToolButton, both abbreviation are tb, then we need to change the abbreviation in one of them, the change principle is avoid conflict and can express the mean of class. E.g. The abbreviation of QToolBar can be change to ‘tbar’, but QToolButton still use ‘tb’ for arrreviation.

5) Class name start with the combination of words that all start with upper case, but the function name use camel style. E.g.

class Node; // class name

class LeafNode; // class name

void drawRect(void); // function name

void setValue(int value); // function name

**5. Header files structure and class declaration arrangement.**

1) The header files consist with three parts.

<1>The version declaration at the beginning of header files.（Reference the first rule）.

<2> Preprocessing block.

<3>Function and class declaration, ect.

In it, the header files start with**(#define\*\*\*, #ifndef\*\*\***)，end with**(#endif //\*\*\***).

2)In class declaration, the order is: **Q\_OBJECT-> public-> siganls-> slots->protected-> priavte**. If need to declaration some another data types(Structure,Enumerate,etc.), should put those declaration before the data members and function members. If in same class, both data variables and function declaration are use same type declaration , split both of them. E.g.

**private：**

void function（）；

…

**private：**

int m\_num；

…

3)At usual, if we use the Signals/Slots, the first sentence of class declaration is Q\_OBJECT.

4)Suggest avoid to use protected type function members or data members, because in Qt libraries, most of event handle(slots) are use protected type.

**6:UI layout principle.**

1)When make Qt UI, use QLayout as more as you can for layout management, try to avoid use absolute coordinates, unless you are certain that the UI won’t changed it’s coordinate forever.

2)If one area have many controls, try to put those controls in a window box(e.g. QWidget, QFrame,QGroupBox,etc.), then put those window boxes in the UI.

**7.Debug information**

At first, in order to debug our program, we should add debug information in our codes. We should use the macro QT\_NO\_DEBUG\_OUTPUT control it, E.g.

#ifndef QT\_NO\_DEBUG\_OUTPUT

qDebug(“debug message”);

QMessageBox::warning(0, “warning”, “message”);

#endif // QT\_NO\_DEBUG\_OUTPUT

**Appendix:**

* 1. Appendix\_1: Hungarian notation 
  2. Appendix\_2: Contorls abbreviation 