How to write debug text to file in Qt

Introduction

In this article we will take a look on how can we create a debugger in Qt. When we print debug information in Symbian phone we use `RFileLogger::WriteFormat`. We have some similar function that can be used for debugging purposes when we write Qt application for Symbian phone. The debug message is printed to the standard output under X11 or to the debugger under Windows. When we run our application in Symbian phone then we can print it in a file. The file can be taken out from phone and can be analyzed.

qInstallMsgHandler

Qt allows us to divert debug messages to wherever we want by using so-called ***message handlers***. The message handler is a function that prints out debug messages, warnings, critical and fatal error messages. The Qt library (debug version) contains hundreds of warning messages that are printed when internal errors (usually invalid function arguments) occur. If we implement our own message handler, we get total control of these messages. Only one message handler can be defined, since this is usually done on an application-wide basis to control debug output. We can install the message handler by calling `qInstallMsgHandler(handler_function)`. Where handler_function is a function pointer that takes the following parameters `void handler_function(QtMsgType, const char *);`

```
    int main(int argc, char **argv)
    {
        qInstallMsgHandler(debugOutput);
        QApplication app(argc, argv);
        ...
        return app.exec();
    }
```

MsgHandler function

Here we can handle we can print different kind of message in file. For example `QtDebugMsg, QtWarningMsg` etc. We can constantly open a file and print to it and close it. Or we can open it in constructor of a class and flush/close in the destructor of the class. If we want to execute the application and want to get the debug information out then the first one is better (since file will be released). If we want to see the debug information after closing the app then second one should be OK.

```
void debugOutput(QtMsgType type, const char *msg)
 {
  QFile debugfile("C:\\Data\\debug.txt");
  Q_ASSERT(debugfile.open(QIODevice::WriteOnly | QIODevice::Text  | QIODevice::Append));

     switch (type)
     {
         case QtDebugMsg:
          {
          QTextStream out(&debugfile);
          out << msg;
          }
          break;


         default :
          break;
```

```
    }
  debugfile.flush();
  debugfile.close();
 }

#define TRACE_FUNC_ENTRY qDebug()<< __PRETTY_FUNCTION__ << ">" << endl ;
#define TRACE_FUNC_EXIT qDebug()<< __PRETTY_FUNCTION__ << "<" << endl ;
```

Example Applications

lightmaps example was modified and tested with N97. The modified version can be found from following link: File:Lightmaps.zip

Related thread in Discussion Board can be found here