October 6, 2011

RITK
THE RANGE IMAGING TOOLKIT

RITK Style Guide

Jakob Wasza, Sebastian Bauer, Sven Haase

October 2011

1 Purpose

The following document is a description of the accepted coding style for the Range Imaging Toolkit (RITK). Developers who wish to contribute code to RITK should read and adhere to the standards described here.

2 Document Overview

This document is organized into the following sections

- System Overview & Philosophy - motivation for the guideline
- Copyright - copyright issues
- File Organization - guide to RITK directory structure
- Naming Convention - patterns used to name classes, variables and files
- Namespaces - the use of namespaces

This style guide is not a final ultimate version. Rather, you can send any suggestions to ritk@i5.cs.fau.de and we will revise the issue or add sections to this style guide. If you want to be sure to use the most recent style guide, go to `http://www5.cs.fau.de/ritk` and download the latest version there.

3 Style Guidelines

The following guidelines have been adopted by the RITK community. They will help to standardize the code written for RITK. By developing source code with one single style we can include new enhancements from the community and evaluate bugfixes faster. All in all this provokes a faster evolution and improved quality of the RITK source code. Most of the guidelines are in the style of ITK [ISNC05]. The ITK guide can be found in your ITK directory in *Documentation/Style.pdf*.

3.1 System Overview & Philosophy

The decisions of the following guidelines where established and evolved during the developing process of RITK. These decisions were mostly influenced by the goals RITK needs to fulfil, e.g. real-time processing, generic architecture and portability.

3.1.1 Implementation Language

The core implementation language is `C++`. It was chosen as we decided to build a range image streaming application that is only useful if it can run in real-time and still can handle large amounts of data. RITK uses the full power of `C++` including namespaces, function overloading, inheritance and templates.

3.1.2 Portability

RITK and its dependencies were developed with the intention to compile on different operating systems with different compilers. Nevertheless, only two configurations have been tested yet:

- Windows 7 32bit, MSVC 2008
- Windows 7 64bit, MSVC 2008
- Windows 7 32bit, MSVC 2010
- Linux, GCC

If you can confirm any other combination of operating systems and compilers, please write us an email with additional information.

3.1.3 CMake Build Environment

The RITK build environment is CMake as it is in ITK. For further information go to `http://www.cmake.org`.

3.1.4 Doxygen Documentation System

The Doxygen open-source system is used to generate an on-line documentation. For further information go to `http://www.doxygen.org`.

3.2 Copyright

The license file is included in all downloads. We have not yet put up any copyright header that needs to be included in all files.

3.3 File Organization

Classes are created and organized into a single class per file set. A file set consists of .h header file, .cxx implementation file, and/or a .txx templated implementation file. The files should be placed in the correct directory. Please adapt the already existing system, where plugins are subdivided into open-source and closed-source and within those folders in source-, filter-, and application-plugins. If you modify RITK itself make sure to put your additional source codes in the correct module and keep the logical structure. File names within RITK always have to start with *ritk* (e.g. *ritkRImage.h*).

3.4 Naming Convention

The most important naming rule is to use meaningful names. In addition to that there are several rules to keep the appearance of the source code homogeneous.

3.4.1 Classes

If you create a plugin use the *Plugin Creator*. It will organize and name the code files the right way. For naming additional classes please adapt the structure to the already implemented source files.

3.4.2 Variables

To indicate separate words names are constructed by using case change. For member variables use the prefix $m\_$ (e.g. $m\_MyMemberVariable$). Local variables should start with a capital letter.

3.4.3 Functions

Functions in RITK need to start with a capital letter. When referring to class methods in code, an explicit this→ pointer should be used. This helps clarify exactly which method is being invoked.

3.5 Namespaces

All classes within RITK itself should be placed in the ritk:: namespace. The plugins do not have to use any namespace.

3.6 Code Layout and Indentation

There is only one important point. We DO use tabs in comparison to ITK. For all other layout guidelines please consult the ITK style guide (Chapter 3.7).

References

[ISNC05] L. Ibanez, W. Schroeder, L. Ng, and J. Cates. *The ITK Software Guide*, second edition, 2005.