MITO v2.0 – How to compile

Compiling VTK

1 – Download **VTK 5.8.0** and unpack the sources into a folder of your choice (i.e.: `C:\Libs\VTK-5.8.0`)

2 – Run CMake and fill the “Where is the source code” field with the path of the library (i.e.: `C:\Libs\VTK-5.8.0`)

3 – Fill the “Where to build the binaries” field with the path where the binaries should be built (i.e.: `C:\Libs\VTK-5.8.0\bin`)

4 – Click on “Configure”

5 – Choose Yes when CMake asks for the permission to create the binaries folder

6 - Choose the compiler for which CMake has to generate the binaries from the dropdown list (i.e.: Microsoft Visual Studio 9 - 64 Bit)

7 – Check the `VTK_USE_PARALLEL` option under the VTK leaf and uncheck `BUILD_TESTING`, `BUILD_SHARED_LIBS`, `BUILD_DOCUMENTATION` and `BUILD_EXAMPLES` under the Build leaf.

8 – Click on “Configure” and then on “Generate”

9 – If there were no problems, you should now be able to open the generated binaries and compile VTK

Compiling ITK

1 – Download **ITK 3.20.0** and unpack the sources into a folder of your choice (i.e.: `C:\Libs\ITK-3.20.0`)

2 – Run CMake and fill the “Where is the source code” field with the path of the library (i.e.: `C:\Libs\ITK-3.20.0`)

3 – Fill the “Where to build the binaries” field with the path where the binaries should be built (i.e.: `C:\Libs\ITK-3.20.0\bin`)

4 – Click on “Configure”

5 – Choose Yes when CMake asks for the permission to create the binaries folder

6 - Choose the compiler for which CMake has to generate the binaries from the dropdown list (i.e.: Microsoft Visual Studio 9 - 64 Bit)

7 – Uncheck `BUILD_DOXYGEN`, `BUILD_EXAMPLES`, `BUILD_TESTING` and `BUILD_SHARED_LIBS` under the Build leaf

8 - Click on “Configure” and then on “Generate”

9 – If there were no problems, you should now be able to open the generated binaries and compile ITK

Compiling CxImage

1 – Download **CxImage 7.01** and unpack the sources into a folder of your choice (i.e.: `C:\Libs\cximage701_full`)

2 – Open the project file inside the sources folder (i.e.: `C:\Libs\cximage701_full\CxImgLib.sln`) and proceed with the compilation

Compiling DCMTK

1 – Download **DCMTK 3.5.4** from http://dicom.offis.de/download/dcmtk/dcmtk354/ and unpack the sources into a folder of your choice (i.e.: `C:\Libs\DCMTK-3.5.4`)

2 – Run CMake and fill the “Where is the source code” field with the path of the library (i.e.: `C:\Libs\DCMTK-3.5.4`)

3 – Fill the “Where to build the binaries” field with the path where the binaries should be built (i.e.: `C:\Libs\DCMTK-3.5.4\bin`)

4 – Click on “Configure”

5 – Choose Yes when CMake asks for the permission to create the binaries folder.

6 - Choose the compiler for which CMake has to generate the binaries from the dropdown list (i.e.: Microsoft Visual Studio 9 - 64 Bit)

7 – Click on “Configure” and then on “Generate”

8 – If there were no problems, you should now be able to open the generated binaries

9 – Open the file
`dcmtk-3.5.4\config\include\dcmtk\config\cfwin32.h`
and replace the code around line 353:

```
/* Define `size_t' to `unsigned' if <sys/types.h> does not define.
*/
/* #undef HAVE_NO_TYPEDEF_SIZE_T */
#ifdef HAVE_NO_TYPEDEF_SIZE_T
typedef unsigned size_t;
#endif

/* Define `ssize_t' to `long' if <sys/types.h> does not define. */
#define HAVE_NO_TYPEDEF_SSIZE_T 1
#ifdef HAVE_NO_TYPEDEF_SSIZE_T
typedef long ssize_t;
#endif
```

with this code:

```
/* Define `size_t' to `unsigned' if <sys/types.h> does not define.
*/
/* #undef HAVE_NO_TYPEDEF_SIZE_T */
#ifdef HAVE_NO_TYPEDEF_SIZE_T
#ifndef TYPEDEF_SSIZE_T_DEFINED
#define TYPEDEF_SSIZE_T_DEFINED
typedef unsigned size_t;
#endif
#endif

/* Define `ssize_t' to `long' if <sys/types.h> does not define. */
#define HAVE_NO_TYPEDEF_SSIZE_T 1
#ifdef HAVE_NO_TYPEDEF_SSIZE_T
#ifndef TYPEDEF_SSIZE_T_DEFINED
#define TYPEDEF_SSIZE_T_DEFINED
typedef long ssize_t;
#endif
#endif
```

10 – Compile the solution

Compiling wxWidgets

1 – Download **wxWidgets 2.8.12** and unpack the sources into a folder of your choice (i.e.: `C:\Libs\wxWidgets-2.8.12`)

2 – Open the project file which can be found inside the “build” subfolder (i.e.: `C:\Libs\wxWidgets-2.8.12\build\msw`)

3 – Open the file
`C:\Libs\wxWidgets-2.8.12\include\wx\defs.h`
and replace the code around line 1015:

```
#ifndef HAVE_SSIZE_T
    #if SIZEOF_SIZE_T == 4
        typedef wxInt32 ssize_t;
    #elif SIZEOF_SIZE_T == 8
        typedef wxInt64 ssize_t;
    #else
        #error "error defining ssize_t, size_t is not 4 or 8 bytes"
    #endif

    /* prevent ssize_t redefinitions in other libraries */
    #define HAVE_SSIZE_T
#endif
```

with

```
#ifndef HAVE_SSIZE_T
    #if SIZEOF_SIZE_T == 4
           #ifndef TYPEDEF_SSIZE_T_DEFINED
           #define TYPEDEF_SSIZE_T_DEFINED
              typedef wxInt32 ssize_t;
           #endif
    #elif SIZEOF_SIZE_T == 8
           #ifndef TYPEDEF_SSIZE_T_DEFINED
           #define TYPEDEF_SSIZE_T_DEFINED
                typedef wxInt64 ssize_t;
           #endif
    #else
        #error "error defining ssize_t, size_t is not 4 or 8 bytes"
    #endif
#endif
```

4 – Compile wxWidgets

Download GLUT

1 - Download GLUT headers and pre-compiled libraries for Intel platforms from http://www.opengl.org/resources/libraries/glut/glut_downloads.php#windows

Download JPEG

1 - Download the Jpeg “Developer files” from http://gnuwin32.sourceforge.net/packages/jpeg.htm

Download Windows Driver Kit (WDK)

1 - Download the WDK for your platform from
http://msdn.microsoft.com/en-us/windows/hardware/gg487463

Download Microsoft DirectX SDK

1 - Download and install the Microsoft DirectX SDK from
http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=6812

Compiling MITO v2.0

1 – Create a new folder to hold MITO source files (i.e:. `C:\Mito`)

2 – Open the `C:\Mito\CMakeList.txt` file with a text editor and modify the following variables to point to the correct path on your file system: `DCMTK_DIR`, `GLUT_DIR`, `JPEGLIB_INCLUDE_DIR`, `JPEGLIB_LIB_DIR`, `CXIMAGE_INCLUDE_DIR`, `CXIMAGE_LIB_DIR`, `WDK_LIB_DIR`, `WDK_INC_DIR`, `WBEMUUID_LIB_DIR` and `DIRECTX_LIBRARY_DIR`.

3 – Run CMake and fill the “Where is the source code” field with the path of the library (i.e.: `C:\MITO`)

4 – Fill the “Where to build the binaries” field with the path where the binaries should be built (i.e.: `C:\MITO\bin`)

5 – Click on “Configure”

6 – Choose Yes when CMake asks for the permission to create the binaries folder.

7 - Choose the compiler for which CMake has to generate the binaries from the dropdown list (i.e.: Microsoft Visual Studio 9 - 64 Bit)

8 – Set the `ITK_DIR` and `VTK_DIR` to the folders containing the related binaries generated by CMake (i.e.: `C:\Libs\ITK-3.20.0\bin` and `C:\Libs\VTK-5.8.0\bin`)

9 – Set the `DCMTK_DIR` variable to the folder containing DCMTK binaries generated by CMake (i.e.: `C:\libs\dcmtk-3.5.4\bin`)

10 – Set the `wxWidgets_ROOT_DIR` variable to the folder containing wxWidgets (i.e.: `C:\Libs\wxWidgets-2.8.12`)

11 – Click on “Configure” and then on “Generate”

11 - If there were no problems, you should now be able to open the generated binaries

12 – In case Microsoft Visual Studio 2008 (VC9) is being used, the modification of an include folder may be needed. Right click on the MITO folder inside Visual Studio and choose “Properties”. Then click on the “…” button near “Configuration Properties->C/C++->General->Additional Include Directories”. In the new window, scroll down the path list until a path terminating with `SOME_PATH/$(VCInstallDir)include` is found (it should be at the very bottom of the list). Just remove the part preceding the dollar sign, leaving just `$(VCInstallDir)include`

13 – Compile MITO