Programming into Slicer3

Sonia Pujol, Ph.D.

Surgical Planning Laboratory
Harvard University

Ferdinand Bol (1616-1680), The Officers of the Amsterdam Guild of Wine Merchants Alte Pinakothek, München
The NA-MIC Kit

Programming into Slicer3. Sonia Pujol, Ph.D., Harvard Medical School
National Alliance for Medical Image Computing
©2010,ARR
Slicer3

• An end-user application for image analysis

• An open-source environment for software development

• A software platform that is both easy to use for clinical researchers and easy to extend for programmers
Before Slicer3
#-------------------------------------------
# 1. Step
#-------------------------------------------
set f $fSeg.fStep1
DevAddLabel $f.lTitle "1. Select Input Channels: " WTA
pack $f.lTitle -side top -padx $Gui(pad) -pady 1 -anchor w
frame $f.fInput -bg $Gui(activeWorkspace)
pack $f.fInput -side top -padx 0 -pady 0 -anchor w
foreach frame "Left Right" {
    frame $f.fInput.f$frame -bg $Gui(activeWorkspace)
pack $f.fInput.f$frame -side left -padx 0 -pady $Gui(pad) }
foreach LABEL "T1 T2" Input "SPGR T2W" {
    DevAddLabel $f.fInput.fLeft.l$Input "  ${LABEL}:"
pack $f.fInput.fLeft.l$Input -side top -padx $Gui(pad) -pady 1 -anchor w
    set menubutton   $f.fInput.fRight.m${Input}Select
    set menu        $f.fInput.fRight.m${Input}Select.m
    eval {menubutton $menubutton -text [Volume($EMAtlasBrainClassifier(Volume,${Input}),node)GetName] -relief raised -bd 2 -width 9 -menu $menu} $Gui(WMBA)
    eval {menu $menu} $Gui(WMA)
    TooltipAdd $menubutton "Select Volume defining ${Input}"
    set EMAtlasBrainClassifier(mbSeg-${Input}Select) $menubutton
    set EMAtlasBrainClassifier(mSeg-${Input}Select) $menu
    # Have to update at UpdateMRML too
    DevUpdateNodeSelectButton Volume EMAtlasBrainClassifier Seg-${Input}Select Volume,$Input
    pack $menubutton -side top -padx $Gui(pad) -pady 1 -anchor w }
frame $f.fAlign -bg $Gui(activeWorkspace)
    TooltipAdd  $f.fAlign "If the input T1 and T2 are not aligned with each other set flag here"
pack $f.fAlign -side top -pady 0 -padx $Gui(pad) -anchor w
DevAddLabel $f.fAlign.lAlign "Align T2 to T1? "
    pack $f.fAlign.lAlign -side left -padx $Gui(pad) -pady 1 -anchor w
    foreach value "1 0" text "On Off" width "4 4" {
        eval {radiobutton $f.fAlign.r$value -width $width -indicatoron 0 -text "$text" -value "$value" -variable EMAtlasBrainClassifier(AlignInput) } $Gui(WCA)
pack $f.fAlign.r$value -side left -padx $Gui(pad) -pady 1 -anchor w }
frame $f.fAlign -bg $Gui(activeWorkspace)
    TooltipAdd  $f.fAlign "$f.fAlign if the input T1 and T2 are not aligned with each other set flag here"
pack $f.fAlign -side top -pady 0 -padx $Gui(pad) -anchor w
DevAddLabel $f.fAlign.lAlign "Align T2 to T1? "
    pack $f.fAlign.lAlign -side left -padx $Gui(pad) -pady 1 -anchor w
    foreach value "1 0" text "On Off" width "4 4" {
        eval {radiobutton $f.fAlign.r$value -width $width -indicatoron 0 -text "$text" -value "$value" -variable EMAtlasBrainClassifier(AlignInput) } $Gui(WCA)
pack $f.fAlign.r$value -side left -padx 0 -pady 0 }
From Slicer2 to Slicer3
The New Execution Model

<?xml version="1.0" encoding="utf-8"?>
<executable>
  <category> Demonstration </category>
  <title> Hello World </title>
  <description> Slicer Developer Course </description>
  <version> 1.0 </version>
  <documentation-url> </documentation-url>
  <license></license>
  <contributor>
    Sonia Pujol, Ph.D., Surgical Planning Laboratory, Harvard Medical School
  </contributor>
  <acknowledgements> National Alliance for Medical Image Computing (NAMIC), Grant U54 EB005149. 
  <acknowledgements>
  </parameters>
  <label>Input/Output</label>
  <description>Input/output parameters</description>
  <image>
    <name>helloWorldInputVolume</name>
    <label>Input Volume</label>
    <channel>input</channel>
    <index>0</index>
    <default>None</default>
    <description>Input volume</description>
  </image>
  <image>
    <name>helloWorldOutputVolume</name>
    <label>Output Volume</label>
    <channel>output</channel>
    <index>1</index>
    <default>None</default>
    <description>Output filtered</description>
  </image>
</parameters>
</executable>
Slicer3 Execution Model

• This course is based on the Execution Model which provides a mechanism for incorporating command line programs as Slicer modules.

• Slicer Communication with external executables

• Jim Miller, Dan Blezek, Bill Lorensen (GE)

Learning objective

Following this course, you’ll be able

1) to plug-in an external program into Slicer3
2) to implement an image filter and to run the analysis from Slicer3
3) to write and run a test using the CTest tool
Material

This course requires the following material

- Slicer 3.6 Software
- HelloWorld_Plugin.zip

Disclaimer
It is the responsibility of the user of 3DSlicer to comply with both the terms of the license and with the applicable laws, regulations and rules.
HelloWorld_Plugin.zip archive

Unzip HelloWorld_Plugin.zip

HelloWorld.xml (Execution Model)

HelloWorld.cxx (application)

CMakeLists.txt (CMake)

spgr.nhhdr
spgr.raw.gz
(124 SPGR images)
Overview

• **Part A**: integration of the HelloWorld program into Slicer3

• **Part B**: implementation of a Discrete Gaussian filter within the HelloWorld module

• **Part C**: implementation of a test for the HelloWorld module
Slicer Programming Course

Programming into Slicer3. Sonia Pujol, Ph.D., Harvard Medical School
National Alliance for Medical Image Computing
©2010,ARR
Part A: Integrating an executable into Slicer3
Slicer3 Execution Model

• The **Execution Model** which provides a mechanism for incorporating command line programs as Slicer modules.

• The Slicer modules are described using **XML files** which are used to generate
  – the C++ command line code
  – the Graphical User Interface (GUI).
Modifying CMakeLists.txt

Open the file CMakeLists.txt located in the directory /HelloWorld_Plugin/HelloWorld/

```cmake
project(HelloWorld)
cmake_minimum_required(VERSION 2.6)

# Slicer3
find_package(Slicer3 REQUIRED)
include(Slicer3 USE FILE)

# Default install prefix
slicer3_set_default_install_prefix_for_external_projects()

# Hello World plugin

# Tests for the plugin
enable_testing()
```
Add the following lines to CMakeLists.txt:

```cmake
set (CLP HelloWorld)
set (${CLP}_SOURCE ${CLP}.cxx)
generateclp(${CLP}_SOURCE ${CLP}.xml)
```

GENERATECLP generates the file HelloWorldCLP.h for parsing the command line arguments.

‘CLP’ means Command Line Processing.
ADD_EXECUTABLE creates the stand-alone executable HelloWorld.exe that can be run from a command line.

Add the following lines to CMakeLists.txt after the ‘generateclp’ line you just added:

```cmake
add_executable(${CLP} ${${CLP}_SOURCE})
slicer3_set_plugins_output_path(${CLP})
target_link_libraries (${CLP} ${ITK_LIBRARIES})
```

Save the file after editing.
• Launch the CMake executable located in the directory Slicer3-lib/CMake-build/bin

/cygdrive/c/slicer3dev/Slicer3-lib/CMake-build/bin

Sonia PUJOL@SONIA-DELL-M70 ~
$ cd c:slicer3dev/Slicer3-lib/CMake-build/bin/

Sonia PUJOL@SONIA-DELL-M70 /cygdrive/c/slicer3dev/Slicer3-lib/CMake-build/bin
$ ./CMakeSetup.exe
Configuring HelloWorld - WINDOWS (2/5)

Enter the path to the \HelloWorld directory that contains the source code

Enter the path to the \HelloWorld-build directory where the binaries will be built

Click on Configure
Select the compiler which is installed on your machine, and click on OK.
A CMake message appears as CMake cannot find the path to Slicer3. Click on Cancel.
Enter the path to the \Slicer3-build directory, and click on Configure.
Click on OK to generate the build files in the HelloWorld-build directory, and exit.
• From the HelloWorld-build/ directory, launch the `ccmake` executable located in the Slicer3-lib/CMake-build/bin/ directory

```
• cd HelloWorld_Plugin/HelloWorld-build/
• /path/to/Slicer/build/Slicer3-lib/CMake-build/bin/ccmake ../HelloWorld
```
You need to enter the path to Slicer3 manually:
Press e to get to the configuration options
Enter the path to the directory Slicer3-build/:

- Arrow down to the Slicer3_DIR and Hit Enter to edit the path
- Arrow up once you have finished editing the path
Configuring HelloWorld (Linux & Mac) 4/4

Press C to configure
Press C to configure again
Press G to generate the Makefile
Open the file HelloWorld.xml located in the directory HelloWorld_Plugin/HelloWorld

Module Description

Module Parameters

<?xml version="1.0" encoding="utf-8"?>
<executable>
  <category>
    Demonstration </category>
  <title>
    Hello World </title>
  <description>
    Slicer Developer Course </description>
  <version>
    1.0 </version>
  <documentation-url>
  <license>
  </license>
  <contributor>
    Sonia Pujol, Ph.D., Surgical Planning Laboratory, Harvard Medical School </contributor>
  <acknowledgements>
    This work is part of the National Alliance for Medical Image Computing (NAMIC),
    funded by the National Institutes of Health through the NIH Roadmap for Medical Research,
    Grant U54 EB005149. </acknowledgements>

  <parameters>
    <label>Input/Output</label>
    <description>Input/output parameters</description>
    <image>
      <name>helloWorldInputVolume</name>
      <label>Input Volume</label>
      <channel>input</channel>
      <index>0</index>
      <default>None</default>
      <description>Input volume</description>
    </image>
    <image>
      <name>helloWorldOutputVolume</name>
      <label>Output Volume</label>
      <channel>output</channel>
      <index>1</index>
      <default>None</default>
      <description>Output filtered</description>
    </image>
  </parameters>
</executable>
<?xml version="1.0" encoding="utf-8"?>
<executable>
  <category>
    Demonstration</category>
  <title>
    Hello World</title>
  <description>
    Slicer Developer Course
  </description>
  <version>
    1.0
  </version>
  <documentation-url></documentation-url>
  <license></license>
  <contributor>Sonia Pujol, Ph.D., Surgical Planning Laboratory, Harvard Medical School</contributor>
  <acknowledgements>
    This work is part of the National Alliance for Medical Image Computing (NAMIC), funded by the National Institutes of Health through the NIH Roadmap for Medical Research, Grant U54 EB005149.
  </acknowledgements>
</executable>
Module Parameters

<parameters>

<label>Input/Output</label>
<description>Input/output parameters</description>

<image>
  <name>helloWorldInputVolume</name>
  <label>Input Volume</label>
  <channel>input</channel>
  <index>0</index>
  <default>None</default>
  <description>Input volume</description>
</image>

<image>
  <name>helloWorldOutputVolume</name>
  <label>Output Volume</label>
  <channel>output</channel>
  <index>1</index>
  <default>None</default>
  <description>Output filtered</description>
</image>

</parameters>

A file that specifies the image

Input Volume

Output Volume
Modifying the source code

Open the file HelloWorld.cxx

```cpp
#include <iostream>

int main(int argc, char * argv [])
{
    std::cout << "Hello World!
" << std::endl;

    return 0;
}
```
Modifying the source code

Add the following lines to the file HelloWorld.cxx

```cpp
#include <iostream>
#include "HelloWorldCLP.h"

int main(int argc, char * argv [])
{
    PARSE_ARGS;
    std::cout<< "Hello World !"<<std::endl;
    return EXIT_SUCCESS ;
}
```
Building HelloWorld.exe

Mac/Linux

Run ‘make’ in the directory HelloWorld-build/

Windows

In Visual Studio, select Build→Build Solution to build the solution HelloWorld.sln located in HelloWorld-build/
Building HelloWorld.exe

Mac/Linux

HelloWorld.exe is located in
/HelloWorld-build/lib/slicer3/plugins

Windows

HelloWorld.exe is located in
/HelloWorld-build/lib/slicer3/plugins/debug
Running Slicer3

Mac/Linux

Run `./Slicer3` in Slicer3-build/

Windows

Run `./Slicer3` in Slicer3-build/
Running Slicer3

Click on the View → Application Settings in the main menu
Setting the HelloWorld plugin path

Select Module Settings from the Application Settings GUI
Setting the HelloWorld plugin path

Click on the ‘Add a preset’ button, and enter the path to HelloWorld.exe.
Setting the HelloWorld plugin path

The path to the HelloWorld executable is set-up in Slicer3.

Click on the Close to exit the Application Settings window.
Setting the HelloWorld plugin path

Select File → Exit to exit Slicer3.
Running Slicer3

Mac/Linux

Run `./Slicer3' in Slicer3-build/

Windows

Run `./Slicer3.exe' in Slicer3-build/
Select the category ‘Demonstration’, and the module ‘HelloWorld’ in the Modules menu.
The program ‘HelloWorld’ is now integrated into Slicer3
Part B: Implementing an image filter
Goal

• In this section, we’ll implement a **Gaussian smoothing operator** to ‘blur’ the images and remove detail and noise.

• This implementation will allow us to run the filter on volumes loaded in Slicer, and to integrate the resulting filtered volumes as MRML nodes.
Discrete Gaussian Filter

Variance

Input volume

Output volume

Discrete Gaussian Filter

Programming into Slicer3. Sonia Pujol, Ph.D., Harvard Medical School
National Alliance for Medical Image Computing
http://na-mic.org
©2010,ARR
Editing the file HelloWorld.xml

Add a new parameter group to HelloWorld.xml

```xml
<executable>
  <category>
    Demonstration
  </category>
  <title>
    Hello World
  </title>
  <description>
    Slicer Developer Example
  </description>
  <version>
    1.0
  </version>
  <documentation-url>
  </documentation-url>
  <license>
  </license>
  <contributor>
    Sonia Pujol, Ph.D, Surgical Planning Laboratory, Harvard Medical School
  </contributor>
  <acknowledgements>
    This work is part of the National Alliance for Medical Image Computing (NAMIC), funded by the National Institutes of Health through the NIH Roadmap for Medical Research, Grant U54 EB005149.
  </acknowledgements>

<parameters>
  <label>Input/Output</label>
  <description>Input/output parameters</description>
  ....
</parameters>

<parameters>
  <label>Discrete Gaussian Parameters</label>
  <description>Parameters of the Discrete Gaussian Filter</description>
</parameters>
```
Add the parameter ‘variance’ which corresponds to the variance of the Discrete Gaussian Filter to HelloWorld.xml
Implementing I/O functionalities

Add the following lines to HelloWorld.cxx

```cpp
#include <iostream>
#include "HelloWorldCLP.h"
#include "itkImage.h"
#include "itkImageFileReader.h"
#include "itkImageFileWriter.h"

int main(int argc, char * argv [])
{
    PARSE_ARGS;
    std::cout << "Hello World!" << std::endl;
    return EXIT_SUCCESS ;
}
```
Implementing I/O functionalities

Add the following command lines to set-up the reading and writing functionalities in the ‘main’ procedure in HelloWorld.cxx

```cpp
int main ( int argc, char * argv[]) {
    PARSE_ARGS;
    std::cout << "Hello World!" << std::endl;
    
    typedef itk::Image<short,3>   ImageType;
    typedef itk::ImageFileReader<ImageType>  ReaderType;
    typedef itk::ImageFileWriter<ImageType>  WriterType;
    ReaderType::Pointer reader = ReaderType::New();
    WriterType::Pointer writer = WriterType::New();

    return EXIT_SUCCESS;
}
```
Implementing I/O functionalities

Set the input and output volumes parameters defined in HelloWorld.xml

```cpp
int main ( int argc, char * argv[])
{
    PARSE_ARGS;
    std::cout << "Hello World!" << std::endl;
    typedef itk::Image< short, 3 >   ImageType;
    typedef itk::ImageFileReader< ImageType  >  ReaderType;
    typedef itk::ImageFileWriter< ImageType >  WriterType;
    ReaderType::Pointer reader = ReaderType::New();
    WriterType::Pointer writer = WriterType::New();
    reader->SetFileName(helloWorldInputVolume.c_str());
    writer->SetFileName (helloWorldOutputVolume.c_str());
    return EXIT_SUCCESS;
}
```
Implement the filter `itk::DiscreteGaussianImageFilter`

```cpp
#include "itkDiscreteGaussianImageFilter.h"
int main ( int argc, char * argv[])
{
    PARSE_ARGS;
    std::cout << "Hello World!" << std::endl;
    typedef itk::Image< short, 3 >  ImageType;
    typedef itk::ImageFileReader< ImageType >  ReaderType;
    typedef itk::ImageFileWriter< ImageType >  WriterType;
    ReaderType::Pointer reader = ReaderType::New();
    WriterType::Pointer writer = WriterType::New();
    reader->SetFileName( helloWorldInputVolume.c_str() );
    writer->SetFileName(helloWorldOutputVolume.c_str());

typedef itk::DiscreteGaussianImageFilter <ImageType, ImageType> FilterType;
    FilterType::Pointer filter = FilterType::New();

    return EXIT_SUCCESS;
}
```
int main ( int argc, char * argv[])
{
    PARSE_ARGS;
    std::cout << "Hello World!" << std::endl;
    typedef itk::Image< short, 3 >   ImageType;
    typedef itk::ImageFileReader< ImageType >  ReaderType;
    typedef itk::ImageFileWriter< ImageType >  WriterType;
    ReaderType::Pointer reader = ReaderType::New();
    WriterType::Pointer writer = WriterType::New();
    reader->SetFileName( helloWorldInputVolume.c_str() );
    writer->SetFileName (helloWorldOutputVolume.c_str());
    typedef itk::DiscreteGaussianImageFilter <ImageType, ImageType> FilterType;
    FilterType::Pointer filter = FilterType::New();
    try {
        filter->SetInput(reader->GetOutput());
        filter->SetVariance(variance);
        writer->SetInput(filter->GetOutput());
        writer->Update();
    }
    catch (itk::ExceptionObject &excep){
        std::cerr << argv[0] << ": exception caught !" << std::endl;
        return EXIT_FAILURE;
    }
    return EXIT_SUCCESS;}

Implementing the filter in HelloWorld.cxx

Add the following lines for the filter execution:
Building HelloWorld

Mac/Linux

Run ‘make’ in the directory HelloWorld-build/

Windows

Select Build→Build Solution to build the solution HelloWorld.sln located in HelloWorld-build/
Running Slicer3

Mac/Linux

Run `./Slicer3' in Slicer3-build/

Windows

Run `./Slicer3.exe' in Slicer3-build/
Running the Filter

Go in File→Add Volume and load the dataset spgr.nrrd located in the directory HelloWorld_Plugin/data/
Running the Filter

Browse to the Category ‘Demonstration’ in the Modules menu, and select the module ‘HelloWorld’
Running the Filter

Select the input volume ‘spgr.nhdr’, the output volume ‘Create New’. Enter the value ‘0.9’ for the variance and click on Apply.
Running the Filter

Slicer displays the filtered volume ‘Output Volume’.
Running the Filter

Select the Foreground volume ‘spgr.nhdr’ and fade between the Background and Foreground images to visualize the effect of the filter.
Part C: Testing
Goal

• This section describes a simple example for testing that the ‘help’ functionality of our newly implemented module ‘HelloWorld’ works correctly.

• CTest is a core element of Slicer3’s quality control system for software development.
  
  http://www.cmake.org/Wiki/CMake_Testing_With_CTest

• The goal of ‘HelloWorldTest1’ is to test the following command:
  
  ./HelloWorld --help
To implement the test HelloWorldTest1, add the following lines to the CMakeLists.txt file located in the HelloWorld directory:

```cmake
set (SLICER_EXE ${Slicer3_HOME}/Slicer3)
set (BUILD_SUBDIR "")
if (WIN32)
    set (BUILD_SUBDIR Debug)
endif (WIN32)
add_test (HelloWorldTest1 ${SLICER_EXE} --launch ${Slicer3_INSTALL_PLUGINS_BIN_DIR}/${BUILD_SUBDIR}/$ {CLP} --help)
```
Building HelloWorld

Mac/Linux

Run ‘make’ in the directory HelloWorld-build/

Windows

Select Build→Build Solution to build the solution HelloWorld.sln located in HelloWorld-build/
Testing HelloWorld

Mac/Linux

- In the directory `/HelloWorld-build/` run the following command:

```
/path/to/Slicer/build/Slicer3-lib/CMake-build/bin/ctest -R HelloWorldTest1
```

Windows

- In the directory `/HelloWorld-build/` run the following command:

```
/path/to/Slicer/build/Slicer3-lib/CMake-build/bin/ctest.exe -R HelloWorldTest1
```
Running **HelloWorldTest1**

When the module successfully passes the test, the output below is generated:

```
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
bash-3.2$
pwd
/Users/stuartwallace/Desktop/HelloWorld_Plugin/HelloWorld-build
bash-3.2$ /Users/stuartwallace/Desktop/Slicer3.6/Slicer3-lib/CMake-build/bin/ctest -R HelloWorldTest1
1
Test project /Users/stuartwallace/Desktop/HelloWorld_Plugin/HelloWorld-build
  Start 1: HelloWorldTest1
1/1 Test #1: HelloWorldTest1 ..................  Passed  2.00 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =  2.00 sec
bash-3.2$ ```
Conclusion

• This course described functionalities for integrating, developing and testing an external program within Slicer3.

• The **Execution Model** of Slicer3 provides a simple mechanism for incorporating command line programs as Slicer modules.

• The pipeline guided you through 6 components of the NA-MIC kit.
Slicer Programming Course

Programming into Slicer3. Sonia Pujol, Ph.D., Harvard Medical School
National Alliance for Medical Image Computing
©2010,ARR

http://na-mic.org
Acknowledgments

National Alliance for Medical Image Computing
NIH U54EB005149

Neuroimage Analysis Center
NIH P41RR013218