MITK Master News

Sascha Zelzer

Build System (modernize and simplify)

- Minimum CMake version is **3.1** (probably going to be 3.2)

Build System (modernize and simplify)

- Minimum CMake version is **3.1** (probably going to be 3.2)
- No more *module conf* files

Build System (modernize and simplify)

- Minimum CMake version is **3.1** (probably going to be 3.2)
- No more *module conf* files
- New export macro style `<MODULENAME>_EXPORT` (CMake default)
  (old: custom or `<ModuleName>_EXPORT`)

```
class MITKCORE_EXPORT Foo {};
```

Build System (modernize and simplify)

- Minimum CMake version is **3.1** (probably going to be 3.2)
- No more *module conf* files
- New export macro style `<MODULENAME>_EXPORT` (CMake default)
  (old: custom or `<ModuleName>_EXPORT`)

```
class MITKCORE_EXPORT Foo {};
```

- New super-build directory layout

```
MITK-superbuild
  ├ ep              <-- external project install prefix
  │ ├ bin
  │ ├ include
  │ └ ...
  ├ MITK-build      <-- actual MITK build tree
  └ MITK-Data       <-- MITK test data
```

Build System (modernize and simplify)

- Minimum CMake version is **3.1** (probably going to be 3.2)
- No more *module conf* files
- New export macro style `<MODULENAME>_EXPORT` (CMake default)
  (old: custom or `<ModuleName>_EXPORT`)

```
class MITKCORE_EXPORT Foo {};
```

- New super-build directory layout

```
MITK-superbuild
  ├ ep              <-- external project install prefix
  │ ├ bin
  │ ├ include
  │ └ ...
  ├ MITK-build      <-- actual MITK build tree
  └ MITK-Data       <-- MITK test data
```

- External projects use a common install prefix

Build System - continued

Support for CMake usage requirements

```
mitk_create_module([moduleName]
  [DEPENDS
    [PUBLIC|PRIVATE|INTERFACE] module1 ...]
  [PACKAGE_DEPENDS
    [PUBLIC|PRIVATE|INTERFACE] package1[|comp1[+comp2]...] ...]
)
```

Build System - continued

Support for CMake usage requirements

```
mitk_create_module([moduleName]
  [DEPENDS
    [PUBLIC|PRIVATE|INTERFACE] module1 ...]
  [PACKAGE_DEPENDS
    [PUBLIC|PRIVATE|INTERFACE] package1[|comp1[+comp2]...] ...]
)
```

Example (from `MitkCore`):

```
mitk_create_module(
  DEPENDS
    PUBLIC mbilog CppMicroServices
  PACKAGE_DEPENDS
    PUBLIC  ITK|ITKTransform+ITKImageGrid+ITKImageFeature
            VTK|vtkFiltersTexture+vtkFiltersParallel
    PRIVATE tinyxml OpenGL ITK
)
```

C++11 (baby steps...)

MITK and all its dependencies build with C++11 enabled.

Minimum compiler versions:

- gcc 4.6 (Ubuntu 12.04)
- Clang 3.2
- AppleClang 5.0
- Visual Studio 2010 (probably bumped to 2012)

C++11 (baby steps...)

MITK and all its dependencies build with C++11 enabled.

Minimum compiler versions:

- gcc 4.6 (Ubuntu 12.04)
- Clang 3.2
- AppleClang 5.0
- Visual Studio 2010 (probably bumped to 2012)

Supported C++11 language features:

- `auto`
- `nullptr`
- `override`

C++11 (baby steps...)

MITK and all its dependencies build with C++11 enabled.

Minimum compiler versions:

- gcc 4.6 (Ubuntu 12.04)
- Clang 3.2
- AppleClang 5.0
- Visual Studio 2010 (probably bumped to 2012)

Supported C++11 language features:

- `auto`
- `nullptr`
- `override`

Supported C++11 library features:

- `std::unique_ptr`
- ... to be determined

C++11 examples

Supported C++11 language features:

- auto

```
for(auto iter = weird_template_container.begin(); ...)
```

C++11 examples

Supported C++11 language features:

- auto

```
for(auto iter = weird_template_container.begin(); ...)
```

- nullptr

```
void func(double*) {}
void func(int) {}

func(0); // or func(NULL);
func(nullptr);
```

C++11 examples - continued

- override

```
struct A
{
  virtual void foo() const;
  void bar();
};

struct B : A
{
  void foo() override; // Error: B:foo does not override A::foo
                       // (signature mismatch)
  void bar() override; // Error: A:bar is not virtual
};
```

MITK source code organization

- No *Core* top-level directory any more
- *Core* and *CppMicroServices* libraries moved into the Modules directory

MITK source code organization

- No *Core* top-level directory any more
- *Core* and *CppMicroServices* libraries moved into the Modules directory
- New recommended directory layout for modules (de-facto standard for C/C++ libraries):

```
MyModule
  ├ doc           <-- documentation (dox, png, md, ...)
  ├ include       <-- public header files (h, txx)
  ├ resource      <-- Qt and CppMicroServices resources
  ├ src           <-- sources and private headers (cpp, h, qrc, ui, ...)
  └ test          <-- unit tests
```

Many modules use this already

Poll - Lecture series topics

Proposed topics for lectures á 45 min:

- CMake and the MITK build system
  integrating external projects, using MITK's module and plug-in CMake macros, do's and dont's, ...

Poll - Lecture series topics

Proposed topics for lectures á 45 min:

- CMake and the MITK build system
  integrating external projects, using MITK's module and plug-in CMake macros, do's and dont's, ...
- C++ Micro Services (maybe 2 sessions)
  core features, module management, handling dynamic services, service properties, hooks, settings, ...

Poll - Lecture series topics

Proposed topics for lectures á 45 min:

- CMake and the MITK build system
  integrating external projects, using MITK's module and plug-in CMake macros, do's and dont's, ...
- C++ Micro Services (maybe 2 sessions)
  core features, module management, handling dynamic services, service properties, hooks, settings, ...
- BlueBerry
  main concepts and features, best practices, use cases, customization points, ...

Questions?