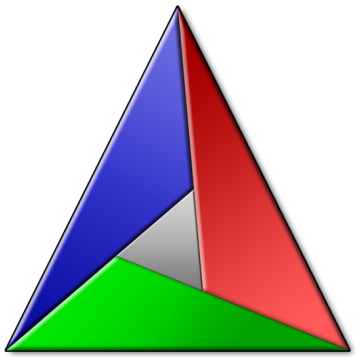
The Design and Evolution of a CMake Daemon

Stephen Kelly
steveire@gmail.com
@steveire


Qt



CMake Contributions

- GenerateExportHeader

CMake Contributions

- GenerateExportHeader
- Transitive usage requirements
- Generator expressions

CMake Contributions

- GenerateExportHeader
- Transitive usage requirements
- Generator expressions
- file(GENERATE)
- CMAKE_SYSROOT/ cross-compiling

CMake Contributions

- GenerateExportHeader
- Transitive usage requirements
- Generator expressions
- file(GENERATE)
- CMAKE_SYSROOT/ cross-compiling
- QNX QCC
- Qt AUTOUIC/AUTORCC
- export(EXPORT)

CMake Contributions

- GenerateExportHeader
- Transitive usage requirements
- Generator expressions
- file(GENERATE)
- CMAKE_SYSROOT/ cross-compiling
- QNX QCC
- Qt AUTOUIC/AUTORCC
- export(EXPORT)
- COMPILE_OPTIONS
- INTERFACE libraries
- Help manuals

CMake Contributions

- GenerateExportHeader
- Transitive usage requirements
- Generator expressions
- file(GENERATE)
- CMAKE_SYSROOT/ cross-compiling
- QNX QCC
- Qt AUTOUIC/AUTORCC
- export(EXPORT)
- COMPILE_OPTIONS
- INTERFACE libraries
- Help manuals
- Compile features ('C++11/14 support')
- WriteCompilerDetectionHeader
- target_sources()

CMake Contributions

- GenerateExportHeader
- Transitive usage requirements
- Generator expressions
- file(GENERATE)
- CMAKE_SYSROOT/ cross-compiling
- QNX QCC
- Qt AUTOUIC/AUTORCC
- export(EXPORT)
- COMPILE_OPTIONS
- INTERFACE libraries
- Help manuals
- Compile features ('C++11/14 support')
- WriteCompilerDetectionHeader
- target_sources()
- Porting from C++95 to C++98
- 'Minor fixes'
- Refactoring

IDE Metadata

- Buildsystem information
  - What targets are available?
  - Where are they?
  - What are the source files?

IDE Metadata

- Buildsystem information
  - What targets are available?
  - Where are they?
  - What are the source files?
- Target/'Object source' information
  - Include directories, compile definitions, flags

IDE Metadata

- Buildsystem information
  - What targets are available?
  - Where are they?
  - What are the source files?
- Target/'Object source' information
  - Include directories, compile definitions, flags
- CMake information
  - ???

IDE first-class features

- Code completion
- Semantic highlighting
- Validation
- Contextual help

IDE first-class features

- Code completion
- Semantic highlighting
- Validation
- Contextual help
- Debugging
- Understand the code
- 'Add class'

```
  app/projecttree.cpp
  app/projectcode.cpp
  app/projectdetail.cpp
  app/highlighter.cpp
  app/helpviewer.cpp

  app/icons.qrc
)
target_include_directories(cmake-browser
  PRIVATE
  app
)

target_
```

```
Command target_compile_definitions
Command target_compile_features
Command target_compile_options
Command target_include_directories
Command target_link_libraries
Command target_sources
Command get_target_property
Command set_target_properties
```

```
ake-browser
TH_TO_CMAKE}")

make-browser PRIVATE DARK_SCHEME


Y_DIR}/somefile)

target_link_libraries(cmake-browser cmake-client-lib)

target_compile_features(cmake-browser PRIVATE cxx_override)

if (TARGET KF5::ItemModels)
   target_link_libraries(cmake-browser KF5::ItemModels)
   target_compile_definitions(cmake-browser
     PRIVATE KF5_ITEMMODELS_LIB)
endif()
```

Status quo

```
44    app/projectcode.cpp
45    app/projectdetail.cpp
46    app/highlighter.cpp
47    app/helpviewer.cpp
48
49    app/icons.qrc
50  )
51  target_include_directories(cmake-browser
52    PRIVATE
53    app
54  )
55  target_
56    TARGET_ARCHIVES_MAY_BE_SHARED_LIBS
57    TARGET_FILE                                    er
58    TARGET_OBJECTS                                 KE}")
59    TARGET_SUPPORTS_SHARED_LIBS
60  #target_compile_definitions(cmake-browser PRIVATE DARK_SCHEME
61
62  execute_process(COMMAND
63    touch ${CMAKE_CURRENT_BINARY_DIR}/somefile)
64
65  target_link_libraries(cmake-browser cmake-client-lib)
66
67  target_compile_features(cmake-browser PRIVATE cxx_override)
68
69  if (TARGET KF5::ItemModels)
70    target_link_libraries(cmake-browser KF5::ItemModels)
```

```
  app/projecttree.cpp
  app/projectcode.cpp
  app/projectdetail.cpp
  app/highlighter.cpp
  app/helpviewer.cpp

  app/icons.qrc
)
target_include_directories(cmake-browser
  PRIVATE
  app
)
target_comp
  PRIVATE C

#target_com

execute_pro
  touch ${C

target_link

target_comp

if (TARGET KF5::ItemModels)
   target_link_libraries(cmake-browser KF5::ItemModels)
   target_compile_definitions(cmake-browser
     PRIVATE KF5_ITEMMODELS_LIB)
endif()
if (TARGET Qt5::Help)
  target_link_libraries(cmake-browser Qt5::Help)
endif()
```

target_include_directories

Add include directories to a target.

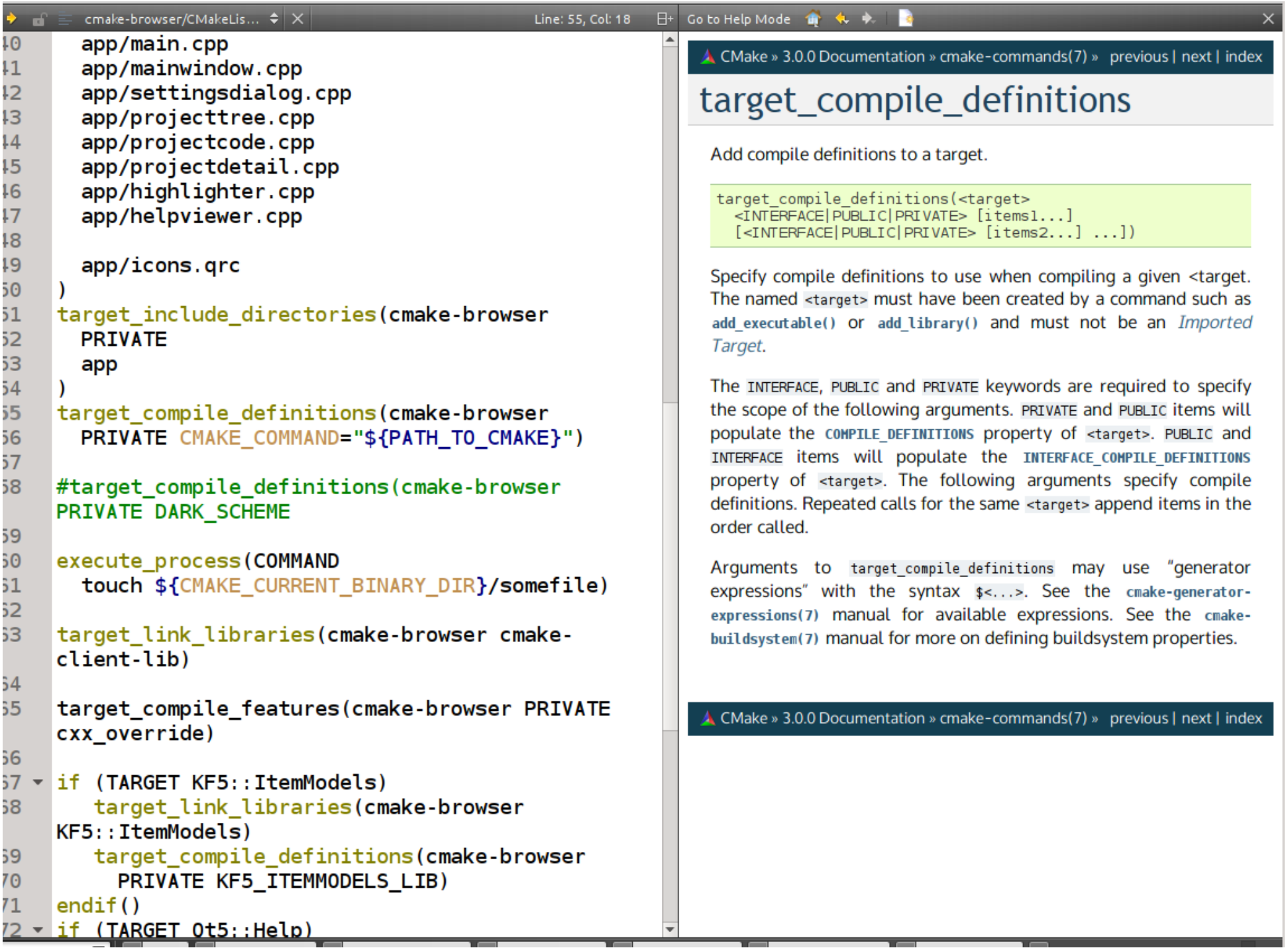
::

```
 target_include_directories(<target> [SYSTEM] [BEFORE]
   <INTERFACE|PUBLIC|PRIVATE> [items1...]
   [<INTERFACE|PUBLIC|PRIVATE> [items2...] ...])
```

Specify include directories to use when compiling a given target.
The named ``<target>`` must have been created by a command such
as ``add_executable()`` or ``add_library()`` and must not be an
``IMPORTED`` target.

If ``BEFORE`` is specified, the content will be prepended to the property
instead of being appended.

The ``INTERFACE``, ``PUBLIC`` and ``PRIVATE`` keywords are required to specify
the scope of the following arguments. ``PRIVATE`` and ``PUBLIC`` items will
populate the ``INCLUDE_DIRECTORIES`` property of ``<target>``.
``PUBLIC`` and ``INTERFACE`` items will populate the


cmake-browser/CMakeLis...
Line: 55, Col: 18
Go to Help Mode
app/main.cpp
app/mainwindow.cpp
app/settingsdialog.cpp
app/projecttree.cpp
app/projectcode.cpp
app/projectdetail.cpp
app/highlighter.cpp
app/helpviewer.cpp
app/icons.qrc
)
target_include_directories(cmake-browser
PRIVATE
app
)
target_compile_definitions(cmake-browser
PRIVATE CMAKE_COMMAND="${PATH_TO_CMAKE}")
#target_compile_definitions(cmake-browser
PRIVATE DARK_SCHEME
execute_process(COMMAND
touch ${CMAKE_CURRENT_BINARY_DIR}/somefile)
target_link_libraries(cmake-browser cmake-
client-lib)
target_compile_features(cmake-browser PRIVATE
cxx_override)
if (TARGET KF5::ItemModels)
target_link_libraries(cmake-browser
KF5::ItemModels)
target_compile_definitions(cmake-browser
PRIVATE KF5_ITEMMODELS_LIB)
endif()
if (TARGET Qt5::Help)
CMake » 3.0.0 Documentation » cmake-commands(7) » previous | next | index
target_compile_definitions
Add compile definitions to a target.
target_compile_definitions(<target>
<INTERFACE|PUBLIC|PRIVATE> [items1...]
[<INTERFACE|PUBLIC|PRIVATE> [items2...] ...])
Specify compile definitions to use when compiling a given <target. The named <target> must have been created by a command such as add_executable() or add_library() and must not be an Imported Target.
The INTERFACE, PUBLIC and PRIVATE keywords are required to specify the scope of the following arguments. PRIVATE and PUBLIC items will populate the COMPILE_DEFINITIONS property of <target>. PUBLIC and INTERFACE items will populate the INTERFACE_COMPILE_DEFINITIONS property of <target>. The following arguments specify compile definitions. Repeated calls for the same <target> append items in the order called.
Arguments to target_compile_definitions may use "generator expressions" with the syntax $<...>. See the cmake-generator-expressions(7) manual for available expressions. See the cmake-buildsystem(7) manual for more on defining buildsystem properties.
CMake » 3.0.0 Documentation » cmake-commands(7) » previous | next | index

Status quo

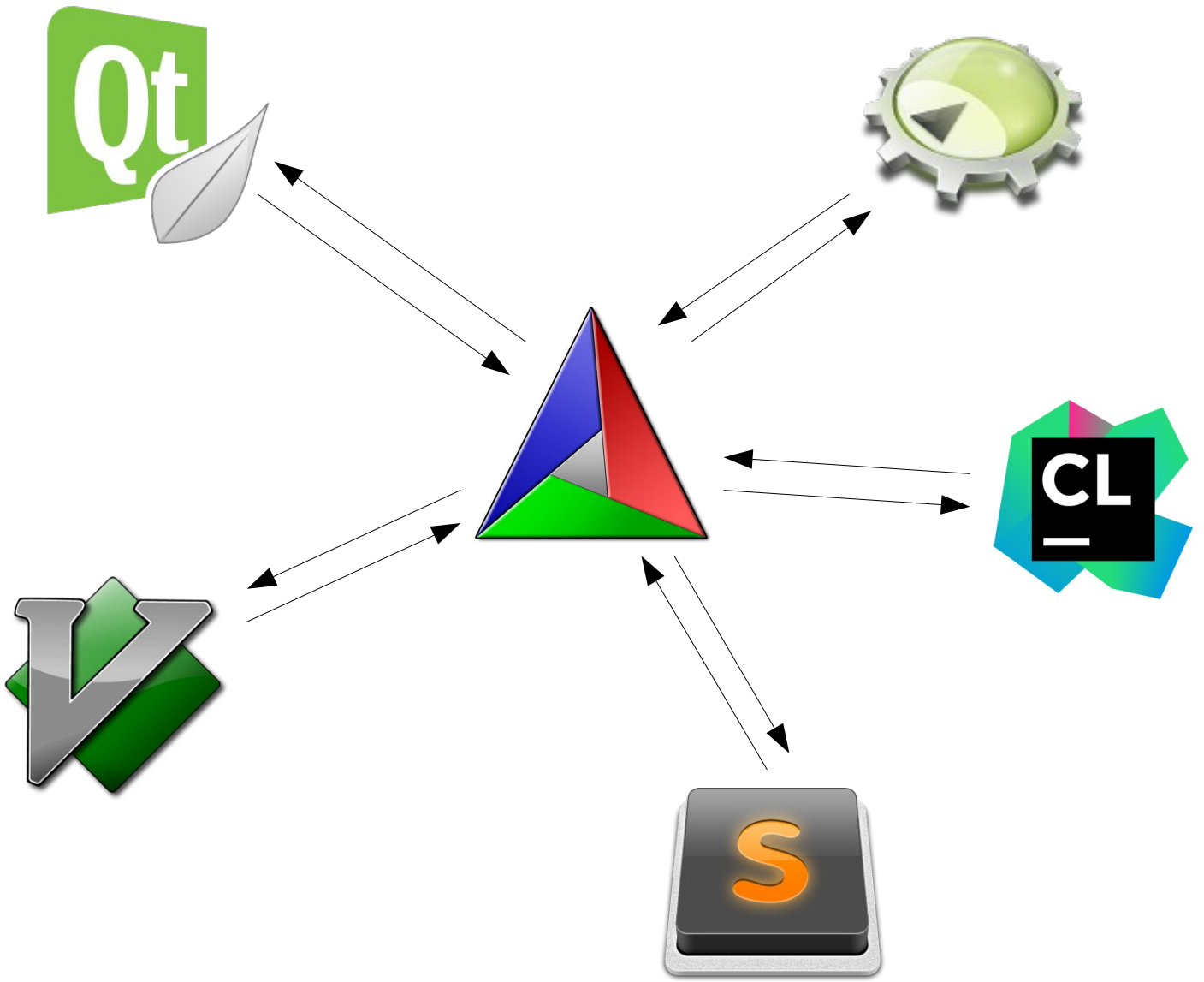
- Hardcoded lists of CMake built-ins
- Syntax Highlighting – various quality
- Contextual help – various quality

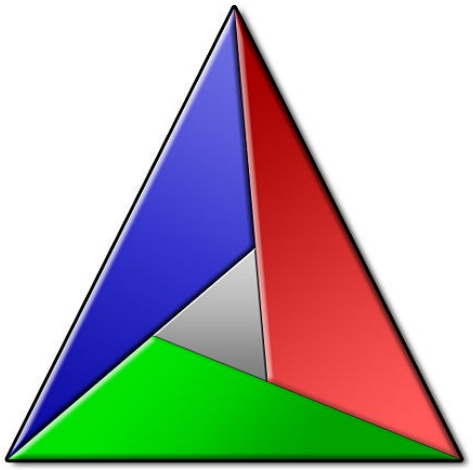
Status quo

- Hardcoded lists of CMake built-ins
- Syntax Highlighting – various quality
- Contextual help – various quality

This needs to be easier

Desire

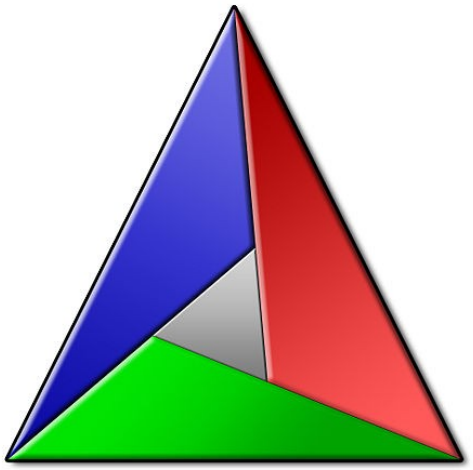




Desire

- Project structure
- Semantic highlight
- Help urls
- Code completion
- State inspection
- Go to definition

Demo



Demo

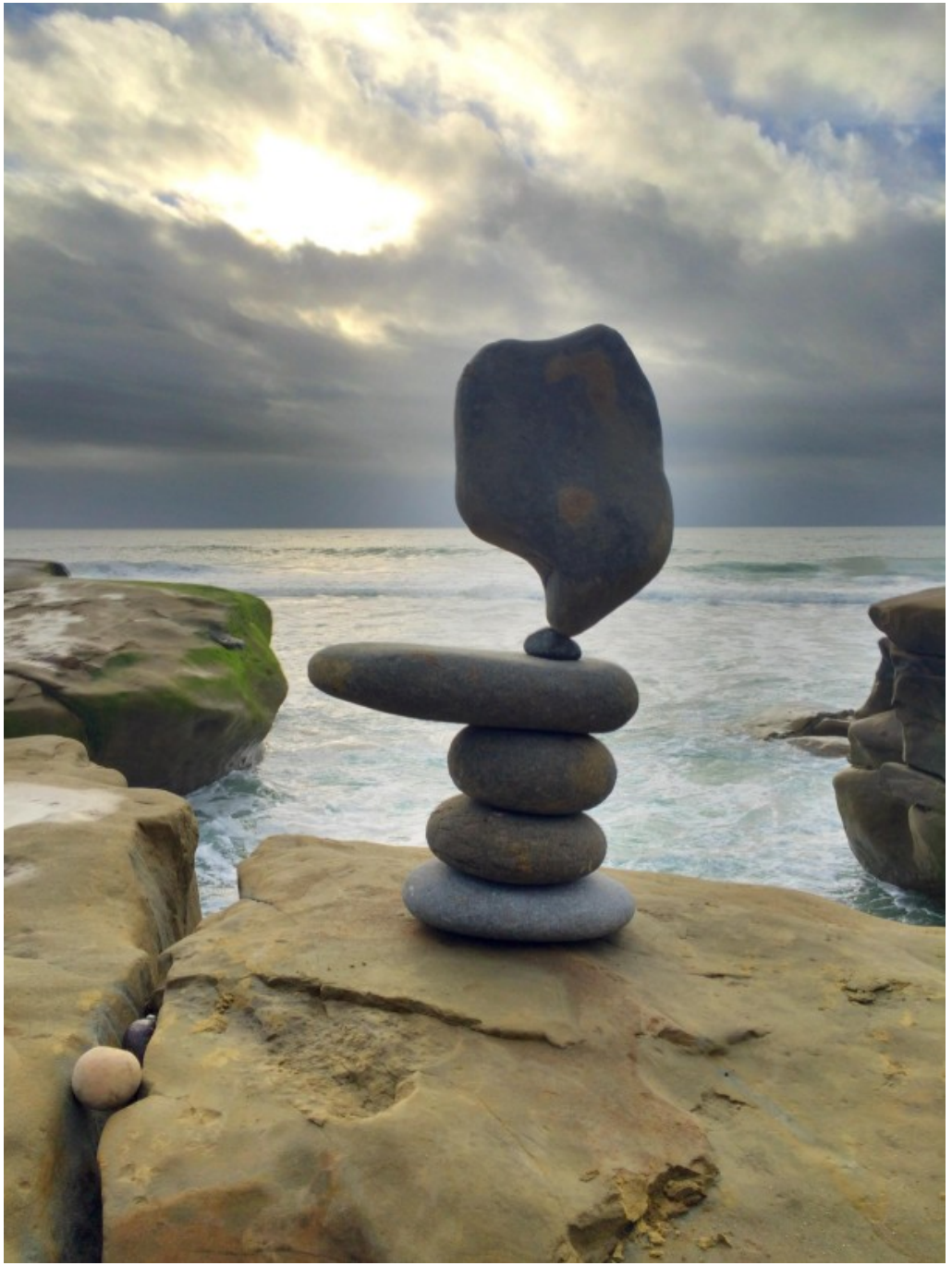
- Project structure
- Semantic highlight
- Help urls
- Code completion
- State inspection
- Go to definition

- QTreeView
- QSyntaxHighlighter
- QtHelp/QtWebKit
- QCompleter
- QTextDocument
- QTextCursor

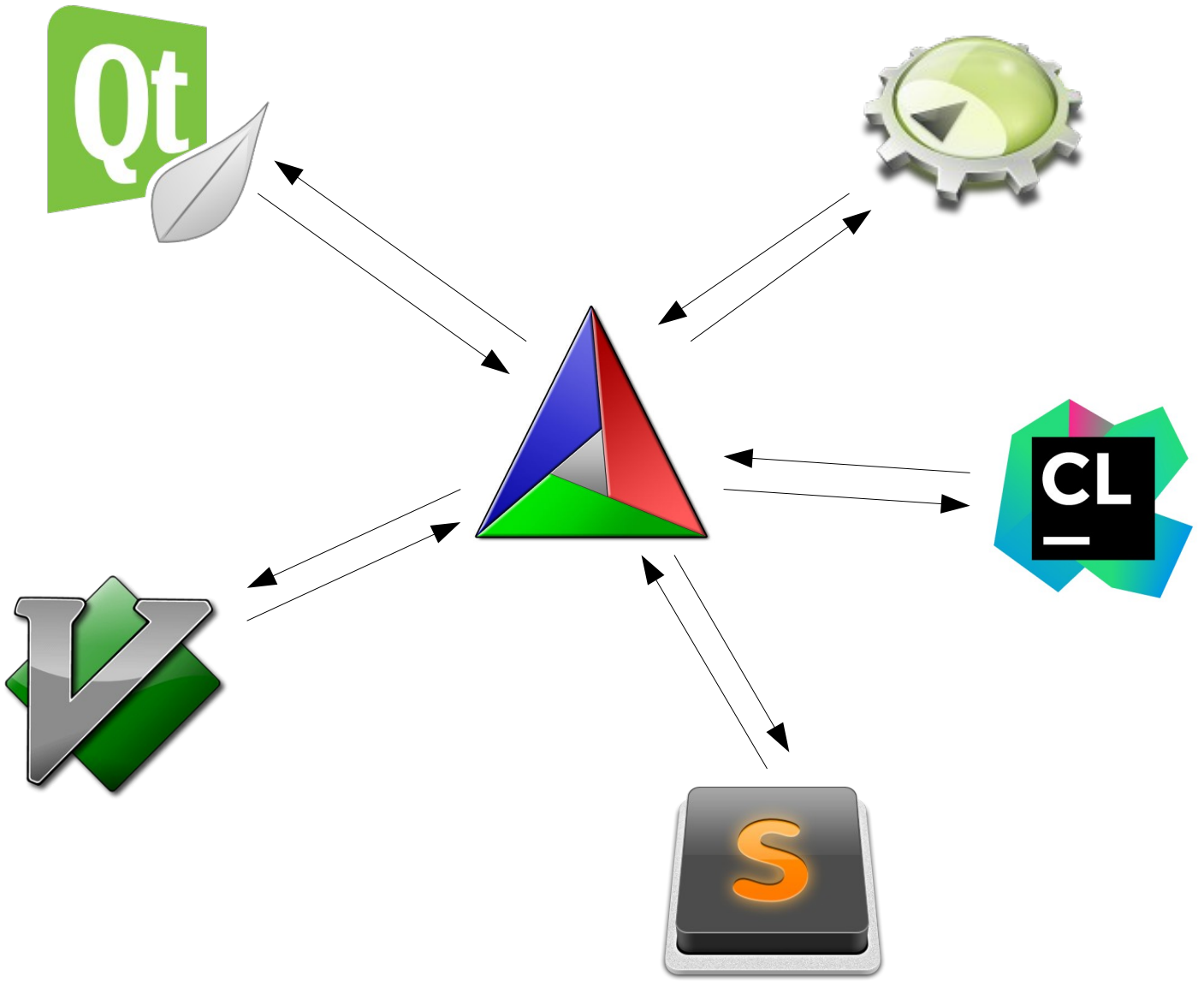
Demo

- ~2500 LOC
- 50% CMake-specific
- 50% GUI composition

- 200 LOC



Daemon



Daemon Design

- Configure, creating snapshots
- Listen to incoming requests
- Respond with requested data

Incremental Snapshots

```
27
28  find_package(Qt5Help QUIET)
29  # Snapshot
30
31  find_package(Qt5TestLib QUIET)
32  # Snapshot
33
34  find_package(Qt5WebKitWidgets QUIET)
35  # Snapshot
36
37  if (POLICY CMP0053)
38    cmake_policy(SET CMP0053 NEW)
39  endif()
40
41  if (POLICY CMP0063)
42    cmake_policy(SET CMP0063 NEW)
43  endif()
44
45  add_subdirectory(lib)
46  # Snapshot
47
48  add_executable(cmake-browser
49    app/main.cpp
50    app/mainwindow.cpp
51    app/settingsdialog.cpp
52    app/projecttree.cpp
53    app/projectcode.cpp
```

Protocol

```
{
  "type": "code_completion_at",

  "line": 31,
  "path": "/path/to/CMakeLists.txt",
  "column": 1
}
```

Protocol

```
{
  "type": "code_completion_at",
  "buffer": "add_library(foo)...",
  "line": 31,
  "path": "/path/to/CMakeLists.txt",
  "column": 1
}
```

Protocol

```
{
  "type": "content_at",
  "buffer": "add_library(foo)...",
  "line": 31,
  "path": "/path/to/CMakeLists.txt"
}
```

Protocol

```
{
  "type": "content_diff",
  "buffer": "add_library(foo)...",
  "line1": 31,
  "line2": 36,
  "path": "/path/to/CMakeLists.txt"
}
```

Protocol API

- `buildsystem`
- `target_info`
- `content_at`
- `content_diff`

Protocol API

- `buildsystem`
- `target_info`
- `content_at`
- `content_diff`
- `completion_at`
- `semantic_parse`
- `contextual_help`
- `definition_location`

Doing

- Collaborators needed
- Implement, test and maintain daemon
- Add more stuff to snapshots
  - Target properties
  - Directory properties
- Extend protocol with new features

Questions

- Will it scale?
- Will IDE makers adopt it?
- I think I can use this to implement <…>!

Stephen Kelly

steveire@gmail.com

@steveire

steveire.wordpress.com