cmake\_minimum\_required(VERSION 3.5.1)

project(grid\_map\_rviz\_plugin)

set(CMAKE\_CXX\_FLAGS "-std=c++11 ${CMAKE\_CXX\_FLAGS}")

add\_compile\_options(-Wall -Wextra -Wpedantic)

set(CMAKE\_EXPORT\_COMPILE\_COMMANDS ON)

find\_package(catkin REQUIRED

COMPONENTS

rviz

grid\_map\_ros

grid\_map\_msgs

)

catkin\_package(

INCLUDE\_DIRS

include

LIBRARIES

${PROJECT\_NAME}

CATKIN\_DEPENDS

rviz

grid\_map\_ros

grid\_map\_msgs

)

include\_directories(

include

SYSTEM

${catkin\_INCLUDE\_DIRS}

)

link\_directories(${catkin\_LIBRARY\_DIRS})

## This setting causes Qt's "MOC" generation to happen automatically.

set(CMAKE\_AUTOMOC ON)

set(INCLUDE\_FILES\_QT

include/grid\_map\_rviz\_plugin/GridMapDisplay.hpp

include/grid\_map\_rviz\_plugin/modified/message\_filter\_display.h

include/grid\_map\_rviz\_plugin/modified/frame\_manager.h

)

## This plugin includes Qt widgets, so we must include Qt.

## We'll use the version that rviz used so they are compatible.

if(rviz\_QT\_VERSION VERSION\_LESS "5")

message(STATUS "Using Qt4 based on the rviz\_QT\_VERSION: ${rviz\_QT\_VERSION}")

find\_package(Qt4 ${rviz\_QT\_VERSION} EXACT REQUIRED QtCore QtGui)

## pull in all required include dirs, define QT\_LIBRARIES, etc.

include(${QT\_USE\_FILE})

qt4\_wrap\_cpp(MOC\_FILES

${INCLUDE\_FILES\_QT}

)

else()

message(STATUS "Using Qt5 based on the rviz\_QT\_VERSION: ${rviz\_QT\_VERSION}")

find\_package(Qt5 ${rviz\_QT\_VERSION} EXACT REQUIRED Core Widgets)

## make target\_link\_libraries(${QT\_LIBRARIES}) pull in all required dependencies

set(QT\_LIBRARIES Qt5::Widgets)

QT5\_WRAP\_CPP(MOC\_FILES

${INCLUDE\_FILES\_QT}

)

endif()

## Avoid Qt signals and slots defining "emit", "slots", etc.

add\_definitions(-DQT\_NO\_KEYWORDS)

## The list of source files.

## The generated MOC files are included automatically as headers.

set(SOURCE\_FILES

src/GridMapDisplay.cpp

src/GridMapVisual.cpp

)

## An rviz plugin is just a shared library, so here we declare the

## library to be called ${PROJECT\_NAME} and specify the list of

## source files we collected above in ${SOURCE\_FILES}.

add\_library(${PROJECT\_NAME}

${SOURCE\_FILES}

${MOC\_FILES}

)

## Link the executable with whatever Qt libraries have been defined by

## the find\_package(Qt4 ...) line above, or by the

## set(QT\_LIBRARIES Qt5::Widgets), and with whatever libraries

## catkin has included.

target\_link\_libraries(${PROJECT\_NAME}

${QT\_LIBRARIES}

${catkin\_LIBRARIES}

)

## Install rules

install(

DIRECTORY include/${PROJECT\_NAME}

DESTINATION ${CATKIN\_PACKAGE\_INCLUDE\_DESTINATION}

)

install(

TARGETS ${PROJECT\_NAME}

ARCHIVE DESTINATION ${CATKIN\_PACKAGE\_LIB\_DESTINATION}

LIBRARY DESTINATION ${CATKIN\_PACKAGE\_LIB\_DESTINATION}

RUNTIME DESTINATION ${CATKIN\_PACKAGE\_BIN\_DESTINATION}

)

install(

FILES plugin\_description.xml

DESTINATION ${CATKIN\_PACKAGE\_SHARE\_DESTINATION}

)

install(

DIRECTORY icons

DESTINATION ${CATKIN\_PACKAGE\_SHARE\_DESTINATION}/icons

)

install(

DIRECTORY doc

DESTINATION ${CATKIN\_PACKAGE\_SHARE\_DESTINATION}

)

#############

## Testing ##

#############

if (CATKIN\_ENABLE\_TESTING)

catkin\_add\_gtest(${PROJECT\_NAME}-test

test/test\_grid\_map\_rviz\_plugin.cpp

test/empty\_test.cpp

)

target\_include\_directories(${PROJECT\_NAME}-test PRIVATE

include

)

target\_include\_directories(${PROJECT\_NAME}-test SYSTEM PUBLIC

${catkin\_INCLUDE\_DIRS}

${EIGEN3\_INCLUDE\_DIRS}

)

target\_link\_libraries(${PROJECT\_NAME}-test

${PROJECT\_NAME}

${catkin\_LIBRARIES}

)

###################

## Code\_coverage ##

###################

find\_package(cmake\_code\_coverage QUIET)

if(cmake\_code\_coverage\_FOUND)

add\_gtest\_coverage(

TEST\_BUILD\_TARGETS

${PROJECT\_NAME}-test

)

endif()

endif()

#################

## Clang\_tools ##

#################

find\_package(cmake\_clang\_tools QUIET)

if(cmake\_clang\_tools\_FOUND)

add\_default\_clang\_tooling(

DISABLE\_CLANG\_FORMAT

)

endif(cmake\_clang\_tools\_FOUND)