Описание CMake инструкций

Автор:
Касьян Александр Иваночвич

Дата:
October 19, 2020

Оглавление

1 Вводное слово

CMake является одним из главных, если не главным инструментом в сборке даного проекта. CMake - это система автоматизации сборки из исходников. Сам CMake не собирает проект, он генерирует файлы для управления сборкой. В данном файле я разбиру инструкции, которые я написал для сборки **Emscripten_OpenGL**. По ходу проекта данный файл будет изменться все логи можно будет увидеть здесь

2 Изменения

3 Основной CMakeList.txt

```
# set the minimal version of the cmake
cmake_minimum_required(VERSION 3.9)

#============================ Main application information ========================#
# set the name of the project
project(Emscripten_Graphics)

# Showing the system type
message(STATUS "We are on a ${CMAKE_SYSTEM_NAME} system")

# setting the version of the project
set(${PROJECT_NAME}_MAJOR_VERSION 0)
set(${PROJECT_NAME}_MINOR_VERSION 1)
set(${PROJECT_NAME}_VERSION "${${PROJECT_NAME}_MAJOR_VERSION}.${${PROJECT_NAME}_MINOR_VERSION}")

# Output the version of the project
message("${PROJECT_NAME} version: ${${PROJECT_NAME}_VERSION}")
#==================================================================================#

#============================ Set Main Variables ==================================#
# Set cmake's modules path
set(CMAKE_MODULE_PATH
  ${CMAKE_SOURCE_DIR}/cmake/Modules
)

# set output dirictories dedicated for libs and binaries
set(CMAKE_ARCHIVE_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/lib)
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/lib)
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/bin)

# Adding options (flags)
# 1) The flag, which has to be ON if we compile via emcc
# 2) The flag, responsible for tests
```

```
option(FOR_EMSDK
        "The libraries compilation occurs for the emsdk compilers"
        OFF
    )
option(ENABLE_TESTS
        "Turn tests on/off"
        OFF
    )

# So far, tests available only for WINDOWS
if(FOR_EMSDK)
    set(ENABLE_TESTS OFF)
endif()

# Set the variables responsible for specific files.
set(${PROJECT_NAME}_SOURCES
                "application/sources/shader.cpp"
                "application/sources/camera.cpp"
                "application/sources/texture.cpp"
                "application/sources/application.cpp"
            )
set(${PROJECT_NAME}_INCLUDES
                "application/includes/shader.h"
                "application/includes/camera.h"
                "application/includes/texture.h"
                "application/includes/vertex.h"
                "application/includes/application.h"
            )
set(${PROJECT_NAME}_MAIN "application/main.cxx")

file(GLOB_RECURSE CFG_FILES ${CMAKE_SOURCE_DIR}/config/*.yaml)
file(GLOB_RECURSE ASSETS ${CMAKE_SOURCE_DIR}/assets/*)

source_group("Config Files" FILES ${CFG_FILES})
```

```
function(assign_source_group)
    foreach(_source IN ITEMS ${ARGN})
        if (IS_ABSOLUTE "${_source}")
            file(RELATIVE_PATH _source_rel
                "${CMAKE_CURRENT_SOURCE_DIR}" "${_source}")
        else()
            set(_source_rel "${_source}")
        endif()
        get_filename_component(_source_path "${_source_rel}" PATH)
        string(REPLACE "/" "\\" _source_path_msvc "${_source_path}")
        source_group("${_source_path_msvc}" FILES "${_source}")
    endforeach()
endfunction(assign_source_group)
assign_source_group(${ASSETS})
#============================================================================#

#============================ Include additional cmake-files ================#
# 1) Set compilation's flags
# 2) Safe guards against in-source builds and bad build types.
# 3) Including file looking for libraries for the graphics.
# 4) Include the file which compiles and looks for the configuration's library.
include(setflags)
include(safeguard)
include(graphics)
include(configloader)
#============================================================================#

#============================ Set libraries =================================#
# Not in-build emscripten libraries,
set(DEP_EMSDK_LIBS
    soil2
)

# The libraries, required by application, however they are already built in emscripten
# But they are not built in the ordinar one.
```

```
if (NOT FOR_EMSDK)
    set(DEP_LIBS
        glfw
        libglew_shared
    )
endif()

# Other Libs, actually created by my-self.
set(LIBS
    ConfigLoader
)

set(INTERFACE_LIB
    glm
)
#================================================================================#

#============================== Generate executable ==================================#
if (NOT ENABLE_TESTS)
    message(STATUS "Executable mode is on!")

    # adding executable
    add_executable(${PROJECT_NAME}
                    ${${PROJECT_NAME}_MAIN}
                    ${${PROJECT_NAME}_SOURCES}
                    ${${PROJECT_NAME}_INCLUDES}
                    ${CFG_FILES}
                    ${ASSETS}
                )
    target_link_libraries(${PROJECT_NAME}
                            ${LIBS}
                            ${DEP_LIBS}
                            ${DEP_EMSDK_LIBS}
                            ${INTERFACE_LIB}
                        )
```

```
target_include_directories(
    ${PROJECT_NAME} PUBLIC
    $<BUILD_INTERFACE:${PROJECT_SOURCE_DIR}/application/includes>
    $<INSTALL_INTERFACE:${PROJECT_SOURCE_DIR}/application/includes>
)

if (NOT FOR_EMSDK)
    find_package(OpenGL REQUIRED)
    target_link_libraries(${PROJECT_NAME} OpenGL::GL)
endif()

# Set some properties related only with emsdk
# Specifically, we creates some output and supporting dirictories:
#+-------------------------------------------------------------------------------------------+
#| To begin with, I tie volumes in the docker-compose and for these purposes                 |
#| I pass executable (.html, .js, .wasm, .data) to the tied directory. So that we            |
#| may acquire those files on our host machine.                                              |
#+-------------------------------------------------------------------------------------------+
#| 2&3) Due to, whereas building occurs in the docker container, I created so-called warapper|
#| in python, which lunches a server and handles requests if those appears. Thus I have to   |
#| shove the entry.py and its config to the binary dir, in purpose to launch it properly.    |
#+-------------------------------------------------------------------------------------------+
#| I'd like to emphasize the problem appears when the compilation occurs via emcc.           |
#| Emscripten creates its own filesystem and assign the start-dir as the root one.           |
#| In c++ code I have to step back to the previous directory to derive the config.yaml       |
#| (../config/config.yaml) if project was built via msvc-compiler, however for the emcc it   |
#| looks like (/config/config.yaml). I do not really want to do a mess from the code,        |
#| therefore I copy the config file to bin dir to make paths equal.                          |
#+-------------------------------------------------------------------------------------------+
if (FOR_EMSDK)
    set(CMAKE_EXECUTABLE_SUFFIX ".html")
    set_target_properties(${PROJECT_NAME}
                PROPERTIES RUNTIME_OUTPUT_DIRECTORY "${CMAKE_BINARY_DIR}/application/out")
    configure_file(${CMAKE_SOURCE_DIR}/application/entry.py
                   ${CMAKE_BINARY_DIR}/application/entry.py
```

```
                    COPYONLY
                )
    configure_file(${CMAKE_SOURCE_DIR}/config/servconfig.yaml
                    ${CMAKE_BINARY_DIR}/config/servconfig.yaml
                    COPYONLY
                )
else()
    configure_file(${CMAKE_SOURCE_DIR}/config/appconfig.yaml
                    ${CMAKE_BINARY_DIR}/config/appconfig.yaml
                    COPYONLY
                )
    macro(copy_files srcDir destDir)
        message(STATUS "Configuring directory ${destDir}")
        make_directory(${destDir})

        file(GLOB templateFiles RELATIVE ${srcDir} ${srcDir}/*)
        foreach(templateFile ${templateFiles})
            set(srcTemplatePath ${srcDir}/${templateFile})
            if(NOT IS_DIRECTORY ${srcTemplatePath})
                message(STATUS "Configuring file ${templateFile}")
                configure_file(
                        ${srcTemplatePath}
                        ${destDir}/${templateFile}
                        COPYONLY)
            endif(NOT IS_DIRECTORY ${srcTemplatePath})
        endforeach(templateFile)
    endmacro(configure_files)
    copy_files(${CMAKE_SOURCE_DIR}/assets/* ${CMAKE_BINARY_DIR}/assets/*)
endif()

# If msvs then set the Emscripten_Graphics as startup project
if(MSVC)
    if(${CMAKE_VERSION} VERSION_LESS "3.6.0")
        message("\n\t[ WARNING ]\n\n\tCMake version lower than 3.6.\n\n\t - Please update CMake and rerun; OR\n\t - Manually
    else()
```

```
            set_property(DIRECTORY ${CMAKE_CURRENT_SOURCE_DIR} PROPERTY VS_STARTUP_PROJECT ${PROJECT_NAME})
        endif()
    endif()
else()
    message(STATUS "Test mode is on!")
    if(MSVC)
        enable_testing()

        # adding test's dir
        add_subdirectory(${CMAKE_SOURCE_DIR}/application/tests)
        configure_file(${CMAKE_SOURCE_DIR}/application/tests/config/appconfig.yaml
                       ${CMAKE_BINARY_DIR}/application/tests/config/appconfig.yaml
                       COPYONLY
                    )
        file(COPY ${CMAKE_SOURCE_DIR}/application/tests/badconfig
                    DESTINATION ${CMAKE_BINARY_DIR}/application/tests/badconfig
                )
    endif()
endif()# endif (NOT ENABLE_TEST)

# Set the only directory for the outsource libs.
set_property(GLOBAL PROPERTY USE_FOLDERS ON)
    if (FOR_EMSDK)
        set_target_properties(${DEP_EMSDK_LIBS} PROPERTIES FOLDER "dependencies")
    elseif(DEP_EMSDK_LIBS)
        set_target_properties(${DEP_LIBS} PROPERTIES FOLDER "dependencies")
        set_target_properties(${DEP_EMSDK_LIBS} PROPERTIES FOLDER "dependencies")
    else()
        set_target_properties(${DEP_LIBS} PROPERTIES FOLDER "dependencies")
endif()

# If the application is builded via emcc sets appropriate flags
if (FOR_EMSDK)
    message(STATUS "Setting EMCC flags\n")
    set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -s FORCE_FILESYSTEM=1 -s USE_WEBGL2=1 -s USE_GLFW=3  -s FULL_ES3=1 -s ALLOW_MEMORY_G
```

```
endif()
#=============================================================================#
```

Для начала output-дириктории:

```
set(CMAKE_ARCHIVE_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/lib)
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/lib)
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/bin)
```

Все рантайм объекты будут находится по пути `bin/<Configuration>/`, где Configuration - это конфигурация, такая как Debug/Release. Все статические библиотеки будут храниться в `lib/<Configuration>/`.