Building JUCE projects made easy using FRUT and CMake

https://github.com/McMartin/FRUT

Alain Martin

Introduction

Projucer

JUCE is a C++ framework for developing cross-platform applications. Since version 4.2.0, JUCE provides Projucer, a development environment that contains a GUI Editor, hosts the Instant Compilation Environment engine, and generates files for native C++ build systems.

CMake

CMake is a cross-platform build system generator: like Projucer it can create Makefiles, Visual Studio solutions and Xcode projects. CMake has many more features for build system generation than Projucer and using it will allow you to simplify and automate building, testing (using ctest) and packaging (using cpack) your C++ applications.

However, due to the way JUCE projects are structured, it is not easy to use CMake for building them, especially when you are not an advanced CMake user. This is why FRUT was created.

FRUT

FRUT is a collection of tools dedicated to building JUCE projects using CMake instead of Projucer.

It currently contains:

- *Reprojucer.cmake*, a CMake module that provides high-level functions to reproduce how a JUCE project is defined in Projucer,

- *Jucer2Reprojucer*, a console application that converts .jucer project files into ready-to-use CMakeLists.txt files that include and use Reprojucer.cmake,

- several CMakeLists.txt files generated from existing .jucer files, including the examples and extras projects from JUCE.

Convert your JUCE project

Let's consider that you have a JUCE project called *Banana*. You can use Jucer2Reprojucer to convert the Projucer file, `Banana.jucer`, into a ready-to-use CMakeLists.txt file:

```
$ cd Banana/

$ <FRUT>/bin/Jucer2Reprojucer Banana.jucer
<FRUT>/cmake/Reprojucer.cmake

Banana/CMakeLists.txt has been successfully generated.
```

From now on, you only need CMake to build your JUCE project:

```
$ mkdir build && cd build/

$ cmake .. -G<generator> -DBanana_jucer_FILE=../Banana.jucer
-- Configuring done
-- Generating done
-- Build files have been written to: Banana/build

$ cmake --build .
```

<generator> can be one of many CMake Generators supported by your platform, including Ninja, NMake Makefiles (on Windows), Unix Makefiles (on Linux and macOS), Visual Studio 2013, 2015 and 2017 (on Windows), and Xcode (on macOS).

License

Advantages

Build all your JUCE projects

You can combine several projects in a parent CMakeLists.txt file with the `add_subdirectory()` command, in order to configure and build them together.

Integrate C++ libraries

Many C++ libraries are using CMake, and using FRUT will make it easier for you to integrate them into your project.

Write CMake scripts

Thanks to CMake's scripting language, you can add cross-platform pre-build and post-build actions, such as retrieving the current Git commit SHA-1, downloading third-party files, or running tests.

Create packages and installers with cpack

In your CMakeLists.txt file, you can use the `install()` command to define what files you want to distribute to your users, and then cpack will take care of creating packages and installers for you.

Keep using JUCE 4

FRUT supports JUCE 4.2.0 minimum, so no need to update to the latest JUCE to get the new features of Projucer.

Contact

You can write a direct message to @McMartin (Alain Martin) on the JUCE forum (https://forum.juce.com) or on the JUCE Discord server.

Limitations

Desktop only

There is currently no support for any mobile platform in FRUT. We need developers who have JUCE projects for these platforms to help implement the support.

Command-line only

Without any UI, FRUT cannot reproduce the GUI Editor of Projucer. The Instant Compilation Environment engine is also not supported.

CMake limitations

Projucer writes certain attributes in IDE files that CMake doesn't allow cutomizing, so FRUT cannot support these settings yet (for instance the "Development Team ID" setting). However, CMake is also open-source and if many of you are asking for these features, we could contribute the necessary changes to CMake.

Future plans

Finalize the support for JUCE 5

The support for JUCE 5 in FRUT is still experimental since some new behaviors and features of Projucer 5 haven't been implemented yet.

Support more exporters

The next Projucer exporter that FRUT will support is "Code::Blocks (Windows)" to allow building with MinGW on Windows. Then it will support the "Code::Blocks (Linux)" exporter.

Enable building JUCE projects in a pure CMake way

This might require contributing **Config.cmake* files to JUCE itself, but it will allow the following Modern CMake syntax:

```
project(Jucer2Reprojucer)

find_package(juce_data_structures REQUIRED)

add_executable(Jucer2Reprojucer "main.cpp")

target_link_libraries(Jucer2Reprojucer
  PRIVATE JUCE::juce_data_structures
)
```

How to contribute?

Contributions to FRUT are very welcomed and you can contribute even if you don't know anything about CMake.

Just do it!

You can convert your existing JUCE projects using Jucer2Reprojucer and build them using CMake (as presented on this poster in the section "Convert your JUCE project"). Then create issues on GitHub (https://github.com/McMartin/FRUT/issues) to report any unwanted differences with how your projects are built when using Projucer.

Show your interest

Several issues on GitHub are labeled `missing feature` and are only waiting for you to comment on them to express your interest for these features. If FRUT is missing another feature that you need and there is no GitHub issue for that feature, feel free to create one!

Help needed

Some of the GitHub issues are also labeled `help wanted` because they concern features that require acquiring third-party SDKs and tools (for instance the AAX SDK and ProTools). We need developers who are familiar with these SDKs and tools to build and test their JUCE projects while we are adding these features to FRUT.

Example CMakeLists.txt

```
# This file was generated by Jucer2Reprojucer from "Plugin Host.jucer"

cmake_minimum_required(VERSION 3.4)


list(APPEND CMAKE_MODULE_PATH
  "${CMAKE_CURRENT_LIST_DIR}/../../../../cmake"
)
include(Reprojucer)


if(NOT DEFINED Plugin_Host_jucer_FILE)
  message(FATAL_ERROR "Plugin_Host_jucer_FILE must be defined")
endif()

get_filename_component(Plugin_Host_jucer_FILE
  "${Plugin_Host_jucer_FILE}" ABSOLUTE
  BASE_DIR "${CMAKE_BINARY_DIR}"
)


jucer_project_begin(
  JUCER_VERSION "5.2.0"
  PROJECT_FILE "${Plugin_Host_jucer_FILE}"
)

jucer_project_settings(
  PROJECT_NAME                "Plugin Host"
  PROJECT_VERSION             "1.0.0"
  COMPANY_NAME                "ROLI Ltd."
  PROJECT_TYPE                "GUI Application"
  BUNDLE_IDENTIFIER           "com.roli.pluginhost"
  CXX_LANGUAGE_STANDARD       "C++11"
)

jucer_project_files("Plugin Host"
# Compile   Xcode     Binary
#           Resource  Resource
  x         .         .         "Source/FilterGraph.cpp"
  .         .         .         "Source/FilterGraph.h"
  x         .         .         "Source/FilterIOConfiguration.cpp"
  .         .         .         "Source/FilterIOConfiguration.h"
  x         .         .         "Source/GraphEditorPanel.cpp"
  .         .         .         "Source/GraphEditorPanel.h"
  x         .         .         "Source/HostStartup.cpp"
  x         .         .         "Source/InternalFilters.cpp"
  .         .         .         "Source/InternalFilters.h"
  x         .         .         "Source/MainHostWindow.cpp"
  .         .         .         "Source/MainHostWindow.h"
)

jucer_project_module(juce_audio_basics      PATH "../../modules")
jucer_project_module(juce_audio_devices     PATH "../../modules"
  JUCE_WASAPI ON
  JUCE_DIRECTSOUND ON
  JUCE_ALSA ON
)
jucer_project_module(juce_audio_formats     PATH "../../modules"
  JUCE_USE_FLAC OFF
  JUCE_USE_OGGVORBIS OFF
)
jucer_project_module(juce_audio_processors  PATH "../../modules"
  JUCE_PLUGINHOST_VST ON
  JUCE_PLUGINHOST_VST3 ON
  JUCE_PLUGINHOST_AU ON
)
jucer_project_module(juce_audio_utils       PATH "../../modules")
jucer_project_module(juce_core              PATH "../../modules")
jucer_project_module(juce_cryptography      PATH "../../modules")
jucer_project_module(juce_data_structures   PATH "../../modules")
jucer_project_module(juce_events            PATH "../../modules")
jucer_project_module(juce_graphics          PATH "../../modules")
jucer_project_module(juce_gui_basics        PATH "../../modules")
jucer_project_module(juce_gui_extra         PATH "../../modules"
  JUCE_WEB_BROWSER OFF
)
jucer_project_module(juce_opengl            PATH "../../modules")
jucer_project_module(juce_video             PATH "../../modules"
  JUCE_USE_CAMERA OFF
)

jucer_export_target("Xcode (MacOSX)"
  EXTRA_COMPILER_FLAGS
    "-Wall"
    "-Wshadow"
    "-Wstrict-aliasing"
    "-Wconversion"
    "-Wsign-compare"
    "-Woverloaded-virtual"
    "-Wextra-semi"
)
jucer_export_target_configuration("Xcode (MacOSX)"
  NAME         "Debug"
  DEBUG_MODE   ON
  BINARY_NAME "Plugin Host"
)
jucer_export_target_configuration("Xcode (MacOSX)"
  NAME         "Release"
  DEBUG_MODE   OFF
  BINARY_NAME "Plugin Host"
)

jucer_project_end()
```