Real-time graphics with OpenGL

Gaëtan Blaise-Cazalet - Head of programming section

Pre-requisites

Modern C++

The evolution of C++

C++ is a standard, implemented by multiple compilers. From 2011, the standard evolved a lot, while keeping backward compatibility. The goal is to keep the performance and flexibility of C++, while proposing safer and simpler interfaces to code.

This trend started in 2011 with C++11, and still continues today with C++14, C++17 and C++20. In studios, those norms are not necessarilly adopted, but they may. Thus it is wise two know the fundamentals of Modern C++.

This lecture, though very small in term of code, use modern C++. It is a prerequisites to understand its main concepts.

An overview of modern C++

Here is a good paper on microsoft C++ website :

https://docs.microsoft.com/en-us/cpp/cpp/welcome-back-to-cpp-modern-cpp?view=msvc-160

Please consult its links to go deeper into each concept.

A small framework

An evolution of the OpenGL intro

This lectures uses an evolution of the Introduction to OpenGL and Shaders (year 2) lecture.

Before you start to code, it is wise to understand its simple structure. You can find the code at :

https://github.com/Gaetz/opengl-training/tree/master/AdvancedOpenGL

Main concepts

In addition to the Timer, the Log system and the Texture class, please take a look at :

- The Window interface and the WindowSDL implementation, used this time as a pointer. It shows how to use a std::unique_ptr in a Strategy pattern.
- The Game class that, this time, implement a Scene (GameState) pattern and handle scene transitions and stacking
- From Scene_006_..., the code use a system of macros to auto-change scenes and shaders. Macros are visible in the MacroUtils.h files

CMake compilation

CMake introduction

CMake is a well known and performant compilation system. It uses specific files, called CMakeLists.txt, to configure compilation.

Each source folder usually have it's CMakeLists.txt file. It allows to use each folder as a library, whose dependencies are inner folders.

This lectures uses Visual Code and CMake compilation. After installing CMake, you should install the CMake and CMake Tools extensions. You should then configue the project (Ctrl + Shift + P then CMake Configure), then build with F7.

CMake project configuration

Here is the main CMake file of the project :

```
cmake_minimum_required(VERSION 3.15)
project(OpenGLTraining VERSION 0.1.0)

set(OpenGL_GL_PREFERENCE "GLVND")
set(CMAKE_CXX_STANDARD 17)

include(CTest)
enable_testing()

# Includes and libraries
if (WIN32)
  set(SDL2_DIR ${CMAKE_SOURCE_DIR}/external/SDL2-2.0.12)     ←  Set the directories to find libs
  set(GLEW_DIR ${CMAKE_SOURCE_DIR}/external/glew-2.1.0)
endif (WIN32)


find_package(OpenGL REQUIRED COMPONENTS OpenGL)

find_package(SDL2 REQUIRED)
include_directories(${SDL2_INCLUDE_DIRS})                     ←  Includes the libs, so that the
                                                                 find_package function can find them
find_package(GLEW REQUIRED)
include_directories(${GLEW_INCLUDE_DIRS})

find_package(OpenGL)

# subdirectories
add_subdirectory( src/engine )                                ←  Add code subfolders. They will also
add_subdirectory( src/game )                                     possess a CMakeLists file

# Executable and link
if (NOT WIN32)
  string(STRIP ${SDL2_LIBRARIES} SDL2_LIBRARIES)
endif (NOT WIN32)
add_executable(OpenGLTraining src/main.cpp)
target_link_libraries(OpenGLTraining game engine maths input ${GLEW_LIBRARIES} ${SDL2_LIBRARIES} OpenGL::GL OpenGL::GLU)


if (WIN32)                                                     ←  We can copy files
  file(COPY external/SDL2-2.0.12/lib/x64/SDL2.dll DESTINATION ${CMAKE_BINARY_DIR}/Debug)
  file(COPY external/SDL2_mixer-2.0.4/lib/x64/SDL2_mixer.dll DESTINATION ${CMAKE_BINARY_DIR}/Debug)
  file(COPY external/SDL2_ttf-2.0.15/lib/x64/SDL2_ttf.dll DESTINATION ${CMAKE_BINARY_DIR}/Debug)
  file(COPY external/SDL2_ttf-2.0.15/lib/x64/zlib1.dll DESTINATION ${CMAKE_BINARY_DIR}/Debug)
  file(COPY external/glew-2.1.0/bin/Release/x64/glew32.dll DESTINATION ${CMAKE_BINARY_DIR}/Debug)
endif (WIN32)

if (WIN32)
  file(COPY assets DESTINATION ${CMAKE_BINARY_DIR}/Debug)
else (WIN32)
  file(COPY assets DESTINATION ${CMAKE_BINARY_DIR})
endif (WIN32)

set(CPACK_PROJECT_NAME ${PROJECT_NAME})
set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
include(CPack)
```

An example of a subdirectory CMakeLists file, for the engine folder :

```
file( GLOB engine_SOURCES *.cpp )                ←  Get all cpp files
add_subdirectory( maths )                        ←  Other subfolders
add_subdirectory( input )
add_library( engine ${engine_SOURCES} )          ←  Declare this folder as a library
target_include_directories(engine PUBLIC ${CMAKE_CURRENT_SOURCE_DIR})
```

Finding libraires

Under Windows at least, you need a specific file in the folder of your dependancies to help CMake find them.

For instance the ./external/SDL2-2.0.12 contains a SDL2Config.cmake file. This file details where to find the includes and compiled libs.

Overview of the complete graphics pipeline (WIP)