ingowald / **OpenCLHelper**

A set of cmake scripts to more easily build opencl based programs

32 commits | **1** branch | **0** releases | **1** contributor

Branch: **master** ▾ | **New pull request** | **Find file** | **Clone or download** ▾

ingowald Merge branch 'master' of github.com:ingowald/OpenCLHelper — Latest commit `0fece36` on 14 Feb

clHelper	Merge branch 'master' of github.com:ingowald/OpenCLHelper	7 months ago
examples	aobench example now computes time	8 months ago
.gitignore	initial import	8 months ago
CMakeLists.txt	aobench example now computes time	8 months ago
LICENSE.txt	first import - not working	8 months ago
README.md	fixes to readme.md	8 months ago
helperTest.cpp	fix opencl.cmake to create custom target	8 months ago

README.md

OpenCL-Helpers - Some Helper Infrastructure for building OpenCL based Projects w/ CMake

Introduction

Whehter on is a fan of OpenCL or not, and no matter whatever limitations OpenCL may or may not have as a *language*, the key issues I stumbled over in my working with OpenCL were actually mostly related to the build system - cl kernels only being compiled during runtime, the app not finding the right .cl files during runtime, hash-include's and hash-define's in cl kernels causing problems when running the executable from a different directory than intended, compile errors in the cl-file only appearing when the cl code is jit'ed during program execution, etc.

This project aims at fixing this through a set of cmake and c++ helper functions. In particular, this library

- allows for build-time (pre-)compilation of all opencl-kernels
- proper preprocessor-expansion (#include, #ifdef, ...) during build time
- cmake commands to specify OPENCL_INCLUDE_DIRS(...) and OPENCL_ADD_DEFINITIONS(...) for .cl files
- 'embedding' of properly preprocessor-expanded .cl-files as global char[] arrays in .c files that can be linked (and thus, embedded) into the application. Ie, cl file "myFile.cl" will always be accesible through global symbols

```
    char myFile_cl[];
```

  and

```
    int myFile_cl_len;
```

- some (optional) c/c++ helper functions to access these embedded kernerls.

Usage

To use this library, do roughly the following in your CMakeLists.txt

```
SET(clHelper_DIR <path to this clHelper directory>)

# this defines all the helper macros
INCLUDE(${clHelper_DIR}/clHelper.cmake)

# this builds the clHelper library (optional, if you want
# to manually access the globally generated symbols)
ADD_SUBDIRECTORY(${clHelper_DIR})

# specify include paths for #include's in opencl files:
OPENCL_INCLUDE_DIRECTORIES(mySearchPath/subdir1 mySearchPath/subdir2)

# specify global defines for opencl files
OPENCL_ADD_DEFINITIONS(-DMY_DEFINE=32)

# compile some opencl kernels. This properly preprocessor-exapands
# and test-compiles the given .cl files (once to asm, once to llvm),
# and puts the preprocessor-expanded code (as a char[] array)
# into dedicated .c files (that can be accessed through the
# implicit EMBEDDED_OPENCL_KERNELS variable
COMPILE_OPENCL(myFile1.cl myFile2.cl)

# build a library/executable that embeds these kernels
ADD_LIBRARY(myLib
myFile1.c myFile2.c
${EMBEDDED_OPENCL_KERNELS})
```

From within the application the embedded kernels can be accessed through global variables

```
extern char myFile1_cl[];
extern int  myFile1_cl_len;
```

or through the clHelper-library (in clHelper/ subdir)

```
size_t source_size;
const char *source_str = clhGetEmbeddedProgram("myFile1.cl",&source_size);
```

respectively

```
std::string source = clHelper::getEmbeddedProgram("myFile1.cl")
```

Note in particular in these examples the "myFile1.cl" is *not* an actual file that is opened during runtime, but that is fully expanded and embedded in the generated executable. Ie, the executable can be called from any directory, and can be shipped without the .cl files.

Dependencies

To use this project you need

- a reasonably new version of CMake
- a c++-11 capable compiler (for the clHlper lib only, *not* for the cmake tools)
- a opencl-capable from of clang, in your path
- for he c++ clHelper library: some sort of OpenCL runtime and devel files (ie, libOpenCL.so and CL/OpenCL.h) in a way that CMake's FindOpenCL can find it.

License

This project comes under MIT license, use as you see fit, without any warranties, expressed or implied, whatsoeever. See accompanying LICENSE.txt for details.

History

This project started Jan 2017, to help some experimentatoin with OpenCL. The cmake build functionality was heavily inspired by - but significantly expanded from - the cmake macros the OSPRay project introduced to build ISPC files (see https://github.com/ospray/OSPRay, and more generally http://www.ospray.org).