FYP Project musings:

milestones

* Compiled several examples for the Altera/Intel SDK on the snowball departmental server. It takes a long time to compile.
* Examined the CrowdSim program for Intel OpenCL 2.0

Crowdsim analysis:

Interestingly the biggest part of it is an external library created by a team at the University of North Carolina- RVO2. It’s for the “Reciprocal Collision Avoidance for Real-Time Multi-Agent Simulation”. In other words, this is the thing that does the crowd simulation. This is located in the RVO2 folder- it can be replaced with a (newer?) version from the library homepage which contains a makefile/CMakeList that can be used to compile it for linux- not much work to be done here as it is in C++.

Oddly enough compiling with my machine (OpenCL 2.1 is supported) works, however when running the program, it can’t compile the OpenCL code. Errors such as this appear:

1:31:9: error: unknown type name 'float2'; did you mean 'float'?

1:56:5: error: unknown type name 'uint'; did you mean 'int'?

1:78:5: error: unknown type name 'uint'; did you mean 'int'?

1:79:5: error: unknown type name 'uint'; did you mean 'int'?

1:84:5: error: unknown type name 'uint'; did you mean 'int'?

1:85:5: error: unknown type name 'uint'; did you mean 'int'?

Which is strange as these are standard OpenCL types defined since OpenCL 1.1 (I think?). Perhaps it can’t find the correct OpenCL run-time?

It seems like these errors were caused by the OpenCL compiler having an include directive that included the working directory. For some reason this prevented the compiler from being able to use the proper CL header files/libs.

The source files (w/o RVO2 library):

**basic.cpp**

So this appears to implement basic reusable functions in the code, things like is\_number, etc.

But more importantly it also implements functions which work with aligned pointers, such as malloc and free. Need to look into this.

**cmdoptions.cpp**

This file contains the display for any command line flags, for help (display all) and if any invalid parameters are specified. Not very interesting.

**cmdparser.cpp**

Similar to cmdoptions, this one works the handling of cmd flags.

**main.cpp**

Point of entry.

**oclobject.cpp**

It describes a class called OpenCLBasic, and presumably handles the initialization of the OpenCL context.

**RenderGL.cpp**

This is as the name suggests, the file that handles the rendering of the simulation. Unfortunately it is mostly rigged up for windows so we will exclude this file for now.

**SimulateCL.cpp**

This handles the CL execution of the RVO library.