# DECS CL for OpenCL C++ Bindings 1.1

0.1

Generated by Doxygen 1.7.6.1

Sun Apr 15 2012 01:03:44

# Contents

# Chapter 1

# Main Page

## 1.1 Introduction

OpenCL enable to give effectivness to various kinds of software. It is good acpect of OpenCL. However, It has complex prosedure to use it for hobby programings and to go through its advantages. Many lines of code must be to written before the caluculations which you want are begun. The C++ classes including in this file help to use OpenCL easily.

The interface for wrapping OpenCL is contained with a single header file DECS/OpenCL/OpenCL.hpp and all definitions are within the namespace DECS::CL. However, some definitions included DECS/OpenCL/OpenCL.hpp are implemented in a binary file, for example decsopencl.lib on Windows. The binary library should be lined when the software is built.

For detail or other information see:

WASABI Tokyo (Author's Blog in English) http://www.wasabi-tokyo.net/

INCHOKI Journal (Author's Blog in Japanese) http://blog.inchoki.com/

## 1.2 Example

The following example shows a simple use case for the DECS CL for OpenCL C++ Bindings.

```
#include "OpenCL.hpp"
#include <iostream>
#include <cstdlib>

const int nElements = 9000000;
float input1[nElements];
float input2[nElements];
float output[nElements];

int main(int argc, char* argv[])
```

```
{
    for(int i = 0; i < nElements; i++){
        input1[i] = (float)i * 10.0f;
        input2[i] = (float)i / 20.0f;
        output[i] = 0.0f;
    }

    try{
        DECS::OpenCL::OpenCL ocl;
        ocl.setSourceFromFile("addVector.cl");
        ocl.buildProgram();
        ocl.setKernel(std::string("addVector"));
        ocl.setKernelInputArgument(0, input1, nElements);
        ocl.setKernelInputArgument(1, input2, nElements);
        ocl.setKernelOutputArgument(2, nElements);
        ocl.enqueueNDRange(nElements);
        ocl.readBuffer(2, output, nElements);

        for(int i = 0; i < 20; i++){
            std::cout << "input1[" << i << "], input2[" << i << "], output["
    << i << "] : ";
            std::cout << input1[i] << ", " << input2[i] << ", " << output[i]
    << std::endl;
        }

    }catch(cl::Error err){
         std::cerr << "ERROR: " << err.what() << "(" << err.err() << ")" <<
    std::endl;

    }

    return EXIT_SUCCESS;
}
```

# Chapter 2

# Todo List

**Class DECS::OpenCL::OpenCL**

Preparing the function to load binary program (Priority: High)

Preparing the function to make kernel with binary program (Priority: High)

**Member DECS::OpenCL::OpenCL::setKernel (std::string functionName)**

Change due to use binary program (Priority: High)

**Member DECS::OpenCL::OpenCL::setKernelInputArgument (cl_int index, float *argument, const int n)**

Change due to deal with other type argument (Priority: High)

Change due to switch multi kernels (Priority: Mid)

**Member DECS::OpenCL::OpenCL::setKernelOutputArgument (cl_int index, const int n)**

Change due to deal with other type argument (Priority: High)

Change due to switch multi kernels (Priority: Mid)

# Chapter 3

# Namespace Index

## 3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

# Chapter 4

# Class Index

## 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 5

# File Index

## 5.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 6

# Namespace Documentation

## 6.1 DECS Namespace Reference

The top level namespace for Discrete Element Calculation Suite(DECS)

### 6.1.1 Detailed Description

The top level namespace for Discrete Element Calculation Suite(DECS) DECS(*D* escrete *E* lement *C* alculation *S* uite) is the set of utilities and libraries for Granular and Powder pysical simulation. Their definitions are contained within this namespace or sub-namespace under this namespace.

## 6.2 OpenCL Namespace Reference

The DECS CL for OpenCL C++ Bindings is defined within this namespace.

### 6.2.1 Detailed Description

The DECS CL for OpenCL C++ Bindings is defined within this namespace. All Classes, structures, constants and macros are defined within this namespace. This namespace is one of sub-namespace of DECS.

# Chapter 7

# Class Documentation

## 7.1 DECS::OpenCL::OpenCL Class Reference

This class is the main class of DECS CL for OpenCL C++ Bindings.

`#include <OpenCL.hpp>`

### Public Member Functions

- OpenCL ()
  *The default constructor.*
- OpenCL (cl_device_type deviceType)
  *The constructor with the device type.*
- OpenCL (int platformIndex, int deviceIndex)
  *The constructor of DECS::OpenCL::OpenCL class to select a platform and a device.*
- ~OpenCL ()
  *The default destructor.*
- void setSource (std::string &source)
  *The function to set source strings for the program.*
- int setSourceFromFile (PATH srcFilePath)
  *The function to load and set a source file.*
- cl_int buildProgram ()
  *The function to build the program.*
- cl_int setKernel (std::string functionName)
  *The function to set kernel.*
- cl_int setKernelInputArgument (cl_int index, float *argument, const int n)
  *The function to set the input argument into the private variable kernel_.*
- cl_int setKernelOutputArgument (cl_int index, const int n)
  *The function to set the output argument into the private variable kernel_.*
- cl_int enqueueNDRange (const int n)

*The function to enqueue the kernel.*

- cl_int readBuffer (cl_int index, float *output, const int n)

  *The fucntion to read the buffer.*

### Protected Member Functions

- cl_int **setPlatform** ()
- cl_int **setContext** (int platformIndex)
- cl_int **setContext** (cl_device_type deviceType, int platformIndex)
- cl_int **setDevice** ()

## 7.1.1 Detailed Description

This class is the main class of DECS CL for OpenCL C++ Bindings.

The OpenCL class supply to the series of procedures to use OpenCL in the softwares from initialization to finalization. The initialization process, for example getting platforms, contexts, devices and creating command queues is run in constructor.

**Todo** Preparing the function to load binary program (Priority: High)

Preparing the function to make kernel with binary program (Priority: High)

## 7.1.2 Constructor & Destructor Documentation

### 7.1.2.1 DECS::OpenCL::OpenCL::OpenCL ( )

The default constructor.

This is the default constructor of DECS::OpenCL::OpenCL class. This constructor initialize the class as following steps.

1. All platforms set into the private variabe std::vector<cl::Platform> platforms_ using the setPlatform protected function.

2. The Context with DEFAULT_DEVICE_TYPE set into the private variable cl::Context context_ using the setContext protected function.

3. All devices related with context_ set into the private varible std::vector<cl::Device> device_ using the setDevice protected function.

4. The Command Queue related with the first device in device_ set into the private varible cl::CommandQueue queue_.

#### 7.1.2.2 DECS::OpenCL::OpenCL::OpenCL ( cl_device_type *deviceType* )

The constructor with the device type.

This is the constructor of DECS::OpenCL::OpenCL class with the device type. This constructor initialize the class as following steps.

1. All platforms set into the private variabe std::vector<cl::Platform> platforms_- using the setPlatform protected function.
2. The Context with the argument deviceType set into the private variable cl::Context context_ using the setContext protected function.
3. All devices related with context_ set into the private varible std::vector<cl::- Device> device_ using the setDevice protected function.
4. The Command Queue related with the first device in device_ set into the private varible cl::CommandQueue queue_.

**Parameters**

| | | |
|---|---|---|
| in | *deviceType* | The device type of context. |

All parameters are defined in OpenCL.

For detail see: The OpenCL Specification http://www.khronos.org/registry/cl/specs/opencl-1.-1.pdf

#### 7.1.2.3 DECS::OpenCL::OpenCL::OpenCL ( int *platformIndex*, int *deviceIndex* )

The constructor of DECS::OpenCL::OpenCL class to select a platform and a device.

This is the constructor to select a platform and a device. This constructor initialize the class as following steps.

1. All platforms set into the private variabe std::vector<cl::Platform> platforms_- using the setPlatform protected function.
2. The Context is created from the platform specified the argument platformIndex and its all type devices. The Context set into the private variable cl::Context context_ using the setContext protected function.
3. All devices related with context_ set into the private varible std::vector<cl::- Device> device_ using the setDevice protected function.
4. The Command Queue related with the device specified the argument deviceIndex in device_ set into the private varible cl::CommandQueue queue_.

**Parameters**

| | | |
|---|---|---|
| in | *platform-Index* | The index of Platform |
| in | *deviceIndex* | The index of devices |

#### 7.1.2.4 DECS::OpenCL::OpenCL::~OpenCL ( )

The default destructor.

The default destructor does not have the specific process at this time.

### 7.1.3 Member Function Documentation

#### 7.1.3.1 DECS::OpenCL::OpenCL::buildProgram ( )

The function to build the program.

This fucntion initialize the private variable program_ and build the programs from the sources stored in the private variable source_. If an error occur in initializing program, this function return the error code of cl::Program and throw the exception. If an error occur in building program, this function return the error code of cl::Program::Build and throw the exception. These exception is defined in OpenCL C++ Bindings.

For detail of the error code see:

The OpenCL Specification http://www.khronos.org/registry/cl/specs/opencl-1.-1.pdf

For Detail of the exception see:

The OpenCL C++ Wrapper API http://www.khronos.org/registry/cl/specs/opencl-cplusp 1.pdf

**Returns**

The error code of cl::Program or cl::Program::Build.

#### 7.1.3.2 DECS::OpenCL::OpenCL::enqueueNDRange ( const int *n* )

The function to enqueue the kernel.

This function enqueue the private variable kernel_ into the command queue which created as the private variable queue_. If a error occur, this function return the error code of cl::CommandQueue::enqueueNDRangeKernel, and throw the exception of its.

For detail of the error code see:

The OpenCL Specification http://www.khronos.org/registry/cl/specs/opencl-1.-1.pdf

For Detail of the exception see:

The OpenCL C++ Wrapper API http://www.khronos.org/registry/cl/specs/opencl-cplusp 1.pdf

**Parameters**

| | | |
|---|---|---|
| in | *n* | The number of index of the arguments. |

**Returns**

The error code of cl::CommandQueue::enqueueNDRangeKernel

#### 7.1.3.3 DECS::OpenCL::OpenCL::readBuffer ( cl_int *index*, float * *output*, const int *n* )

The fucntion to read the buffer.

This function read the output from the buffer into the argument "output". If a error occur duaring the reading buffer, this function return the error code of cl::CommandQueue-::enqueueReadBuffer and throw the exception of cl::CommandQueue::enqueueRead-Buffer and throw.

For detail of the error code see:

The OpenCL Specification http://www.khronos.org/registry/cl/specs/opencl-1.-1.pdf

For Detail of the exception see:

The OpenCL C++ Wrapper API http://www.khronos.org/registry/cl/specs/opencl-cplusplus-1.-1.pdf

**Warning**

At this time this function can deal with only float type.

**Parameters**

| | | |
|---|---|---|
| in | *index* | The index of argument order of the kernel. |
| out | *output* | The variable to store the output. |
| in | *The* | number of element of the argument. |

**Returns**

The error code of cl::CommandQueue::enqueueReadBuffer

#### 7.1.3.4 DECS::OpenCL::OpenCL::setKernel ( std::string *functionName* )

The function to set kernel.

This function set the kernel with the OpenCL function included in the private variable program_instance into the private variable cl::Kernel kernel_. If an error occur in making the kernel instance, this function return the error code of cl::Kernel and throw the Open-CL C++ Bindings exception.

For detail of the error code see:

The OpenCL Specification http://www.khronos.org/registry/cl/specs/opencl-1.-1.pdf

For Detail of the exception see:

The OpenCL C++ Wrapper API http://www.khronos.org/registry/cl/specs/opencl-cplusp 1.pdf

**Parameters**

| | | |
|---|---|---|
| in | *functionName* | The function name to set into kernel. |

**Returns**

The error code of cl::Kernel.

**Todo** Change due to use binary program (Priority: High)

#### 7.1.3.5 DECS::OpenCL::OpenCL::setKernelInputArgument ( cl_int *index*, float * *argument*, const int *n* )

The function to set the input argument into the private variable kernel_.

This Function set the input argument for kernel into the private variable kernel_. The memory object is create based on the pointer of argument and the number of element of argument. In actual, this memory object is set into the kernel. If a error occur in making the memory object, this function return the error code of cl::Buffer and throw the exception. And if a error occur in setting the argument into the kernel, this function return the error code of cl::kernel::setArg and throw the exception.

**Warning**

At this time this function can deal with only float type. And this function can only deal with the memory object with only CL_MEM_READ_ONLY.

For detail of the error code see:

The OpenCL Specification http://www.khronos.org/registry/cl/specs/opencl-1.- 1.pdf

For Detail of the exception see:

The OpenCL C++ Wrapper API http://www.khronos.org/registry/cl/specs/opencl-cplusp 1.pdf

**Parameters**

| | | |
|---|---|---|
| in | *index* | The index of argument order of the kernel. |
| in | *argument* | The pointer of a argument for the kernel. |
| in | *n* | The number of element of the argument. |

**Returns**

The error code of cl::Buffer or cl::Kernel::setArg.

**Todo** Change due to deal with other type argument (Priority: High)

Change due to switch multi kernels (Priority: Mid)

#### 7.1.3.6 DECS::OpenCL::OpenCL::setKernelOutputArgument ( cl_int *index*, const int *n* )

The function to set the output argument into the private variable kernel_.

This Function set the output argument for kernel into the private variable kernel_. The memory object is create based on the pointer of argument and the number of element of argument. In actual, this memory object is set into the kernel. If a error occur in making the memory object, this function return the error code of cl::Buffer and throw the exception. And if a error occur in setting the argument into the kernel, this function return the error code of cl::kernel::setArg and throw the exception.

**Warning**

At this time this function can deal with only float type. And this function can only deal with the memory object with only CL_MEM_WRITE_ONLY.

For detail of the error code see:

The OpenCL Specification http://www.khronos.org/registry/cl/specs/opencl-1.-1.pdf

For Detail of the exception see:

The OpenCL C++ Wrapper API http://www.khronos.org/registry/cl/specs/opencl-cplusplus-1.-1.pdf

**Parameters**

| | | |
|---|---|---|
| in | *index* | The index of argument order of the kernel. |
| in | *n* | The number of element of the argument. |

**Returns**

The error code of cl::Buffer or cl::Kernel::setArg.

**Todo** Change due to deal with other type argument (Priority: High)

Change due to switch multi kernels (Priority: Mid)

#### 7.1.3.7 DECS::OpenCL::OpenCL::setSource ( std::string & *source* )

The function to set source strings for the program.

This function set the source strings into the private variable cl::Program::Sources source_. cl::Program::Sources is std::vector for source string and its size. When this

function call second time and more, the source strings push back the std::vector. The size of std::vector is managed in this function, so the user don't need to conserned with the size.

**Parameters**

| | | |
|---|---|---|
| in | *source* | The source string refference. |

#### 7.1.3.8 DECS::OpenCL::OpenCL::setSourceFromFile ( PATH *srcFilePath* )

The function to load and set a source file.

This function load a souce from source file and set its contents into the private variable source_. This function call the function setSource in its internal process. If you define the macro "__ENABLE_BOOST__", the type "PATH" is "boost::filesystem::path". And "std::string" is used in the default settings.

**Parameters**

| | | |
|---|---|---|
| in | *srcFilePath* | The path to source file. |

**Returns**

0: Success, 1: "srcFilePath cannot be opened

The documentation for this class was generated from the following files:

- OpenCL.hpp
- OpenCL.cpp

# Chapter 8

# File Documentation

## 8.1 OpenCL.hpp File Reference

DECS CL for OpenCL C++ Bindings 1.1.

`#include <CL/cl.hpp> #include <iostream> #include <fstream> ×`

### Classes

- class DECS::OpenCL::OpenCL

  *This class is the main class of DECS CL for OpenCL C++ Bindings.*

### Namespaces

- namespace DECS

  *The top level namespace for Discrete Element Calculation Suite(DECS)*
- namespace OpenCL

  *The DECS CL for OpenCL C++ Bindings is defined within this namespace.*

### Typedefs

- typedef std::string **DECS::OpenCL::PATH**
- typedef std::ifstream **DECS::OpenCL::IFSTREAM**

### Variables

- const cl_device_type **DECS::OpenCL::DEFAULT_DEVICE_TYPE** = CL_DEVICE_TYPE_GPU

  *This variable is the constant default device type.*

### 8.1.1 Detailed Description

DECS CL for OpenCL C++ Bindings 1.1.

**Author**

INCHOKI Studio

**Version**

0.1

**Date**

April 2012