Dual State Framework

Project Report

Yu Chen - C00151352

2015/04/15 13:20:42

Abstract

As the quantity of today's most modern Multi-core processors [16] increase, the parallel computing becomes a very important computer technology.

For this project, a library is created to implements parallel programming mode. The main purpose of the library is to make up a solution for the disadvantage of locker in multiple threads programming. With this library, users can easily code programs implementing parallelism without knowledges of parallel programming mode, so that they have more time to focus on they main work.

A benchmark program is designed for profiling the library. It allows users using costumed settings to run the program.

Contents

1 Introduction

The purpose of this document is to provide an overview of the entire completed project. The focus of the document will be on how the development process was carried out during the project implementation. The document covers:

- description of completed project
- the major problems encountered and the solutions
- description of technical and personal learnings
- review of project
- acknowledgement

2 | Project Description

2.1 Installations

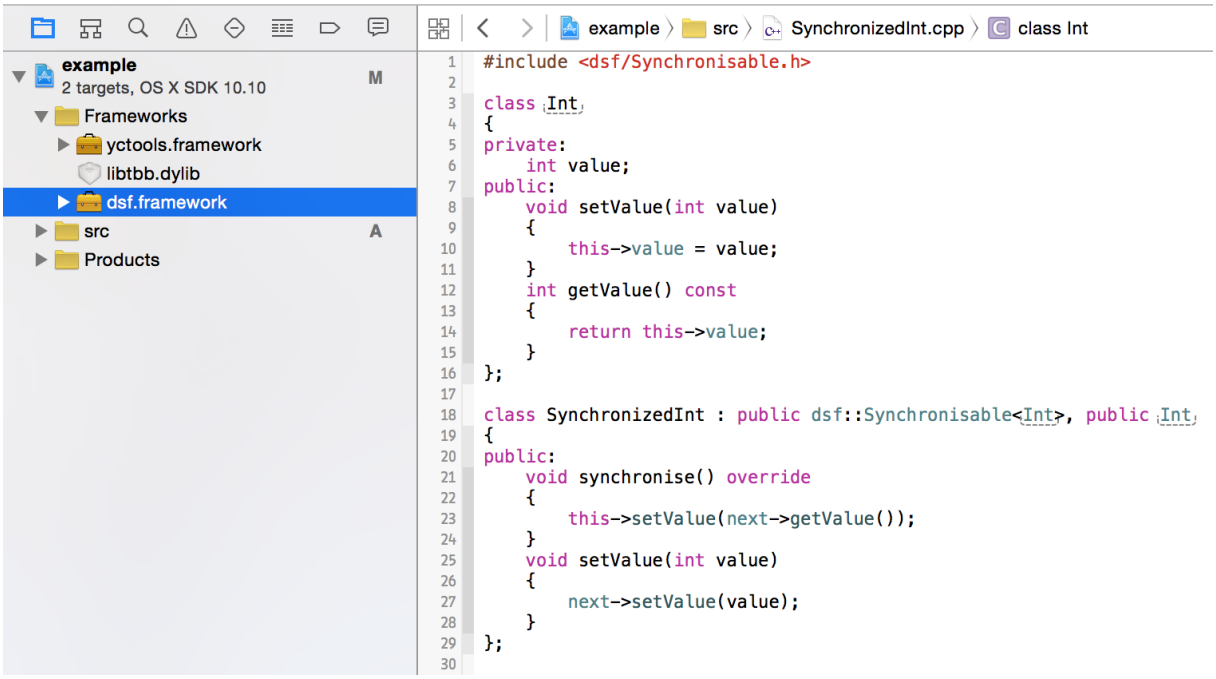
The installers were uploaded on SourceForge [10]. The download page is available at: https://sourceforge.net/projects/dualstateframework/. Click the "files" tab, you will see all versions for different platform.

Home / rel-1.0.0

Name	Modified	Size	Downloads / Week
Parent folder			
dsf.exe	2015-04-05	477.4 kB	1
libdsf_1.0.0_amd64.deb	2015-04-04	28.1 kB	1
dsf.pkg	2015-04-04	16.3 kB	1
Totals: 3 Items		521.9 kB	3

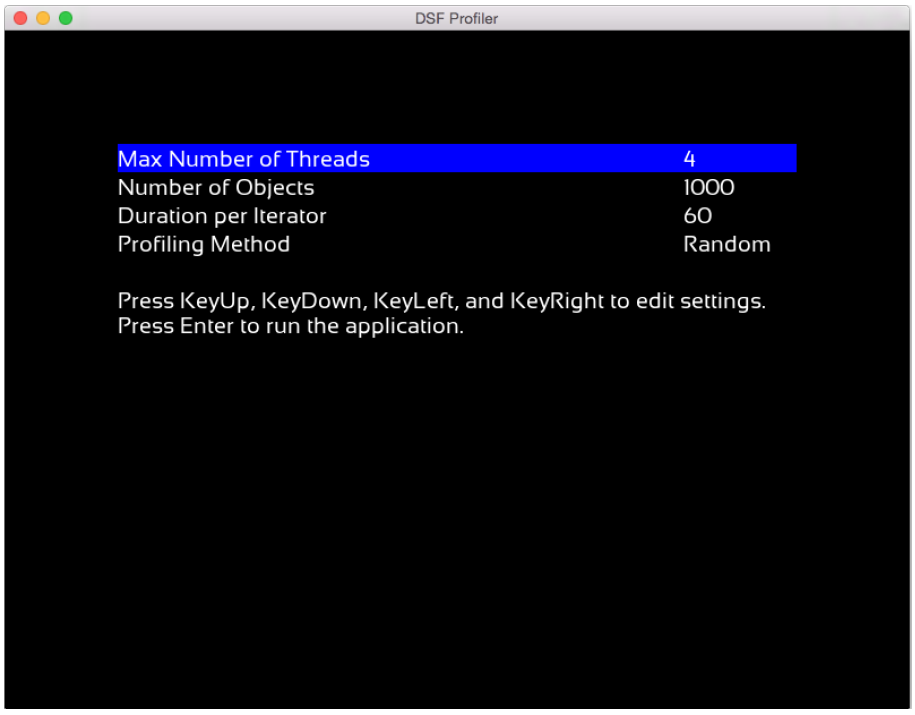
2.2 Framework with Xcode

The library is designed as a bundle [1] for Mac OS X. To use it in Xcode you just need to drag it into your project explorer.

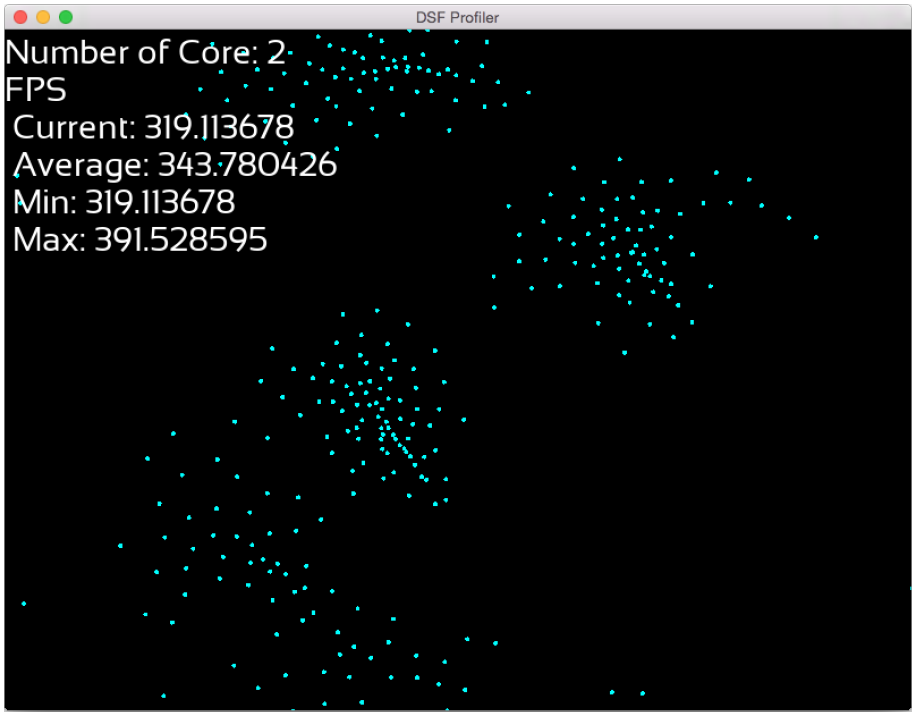


2.3 Benchmark Program

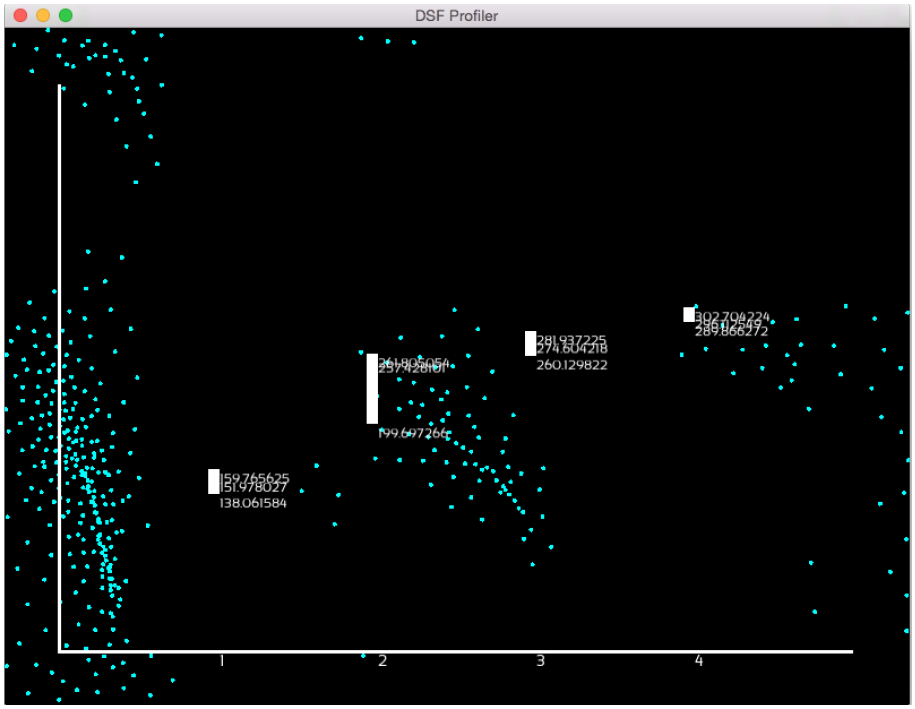
The benchmark program is designed for profiling the library. It allows user to configure settings.



While running it shows the maximum, minimum, and average FPS on the screen.



The output is a bar graph. X axis is the number of threads, and y axis in the FPS.


DSF Profiler
159.765625
151.978027
138.061584
261.937225
274.604218
260.129822
302.798224
289.866272
1
2
3
4

3 | Conformance to Specification and Design Manual

Fortunately, the submitted product matches all functionalities that my specification and design documents mentioned.

3.1 Setup Development Environment

The operating system of my laptop is Mac OS X Yosemite [8]. I have installed VMware Fusion [14] and have created two virtual machine for other operating systems(Microsoft Windows 7 [7] and Ubuntu 14.04 LTS [12]). The development will be mainly taken place under Mac, and tested in all these three operating systems.

3.1.1 C++ Compiler and IDE

Mac OS X

To setup Xcode [15] and default C++ compiler LLVM [6] in Mac OS X is very easy.

- Open App Store → search "Xcode" → click "get" button

After the installation, the IDE Xcode and the compiler LLVM is installed. To see the information about clang by command "clang --version".

```
MacBook-Pro:~ yuchen$ clang --version
Apple LLVM version 6.1.0 (clang-602.0.49) (based on LLVM 3.6.0svn)
Target: x86_64-apple-darwin14.3.0
```

Microsoft Windows

Download Visual Studio Community [13] from https://www.visualstudio.com and install it. Visual Studio has its own embedded C++ compiler.

Ubuntu

Open a terminal and type following command to install the GNU Compiler Collection(GCC) [3]

```
sudo apt-get install build-essential
```

Type "gcc -v" to print the descriptions.

```
yu@ubuntu:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/4.8/lto-wrapper
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 4.8.2-19ubuntu1' --with-bugurl=file:
Thread model: posix
gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1)
```

3.1.2 Third party libraries

Intel Threading Building Blocks

Intel TBB [5] is a C++ library for parallel computing. Use Homebrew [4] to Install the library for Mac OS X:

```
$ brew install libtbb
```

For Ubuntu use following command

```
$ sudo apt-get install libtbb2
```

For Window need go to its download page https://www.threadingbuildingblocks.org/download. Download the Windows version binaries package and unzip it.

- First, create a a new folder. The full path of it is "C:\Program Files\tbb". Inside the folder create three folders "include", "lib", and "bin".
- Secondly, go to the unzipped folder. Copy everything inside the "include" to the "include" has just been created.
- Inside the "lib", there are two folders "ia32" and "intel64" that means the architecture of operating system. My Window 7 is 32 bits, so I chose the "ia32". Again, inside "ia32", there are many folders those are the versions of Visual Studio. The version of Visual Studio that I installed is 2013. The core of Visual Studio 2013 is vs2012, so copy all files inside "vc12" to the folder "lib" which has just been created.
- Inside the "bin", do the same thing as "lib" did.
- Create a environment variable "tbb" refers to the directory "C:\Program Files\tbb".

SFML

Simple and Fast Multimedia Library(SFML [9]) is a graphics library. For Mac, just go to the download page http://www.sfml-dev.org/download/sfml/2.1/ to get the installer and run it. For Ubuntu, just need one line command:

```
sudo apt-get install libsfml-dev
```

However, there is a problem for Windows. When I went to its download page, I could not find the version for Visual Studio 2013. The solution is to compile the source code which can be cloned from https://github.com/LaurentGomila/SFML.git.

3.1.3 Other Tools

- SourceTree [11], a GUI git front-end, which is available in both Mac and Windows.
- cmake-gui [2], the official GUI front-end for CMake, available in Mac, Windows, and Linux.

3.2 Start Development

3.2.1 Source Control

I used git for source control. The repository is available at https://github.com/kuyoonjo/DualStateFramework.git.

3.2.2 Code Configurations

The project is designed for Mac OS X, Microsoft Window, and other Unix-like operating systems. Therefore, the code should be portable in all of them. Write different copy for different operating system is not a good idea. The best solution is prepare some configuration file, then different operating system will generate different copies.

Config.h

```
#ifndef dsf_Config_h
#define dsf_Config_h

#if defined(_WIN32)

// Windows compilers need specific (and different) keywords for export and import
#define DSF_API_EXPORT __declspec(dllexport)
#define DSF_API_IMPORT __declspec(dllimport)

// For Visual C++ compilers, we also need to turn off this annoying C4251 warning
#ifdef _MSC_VER

#pragma warning(disable : 4251)

#endif

#else // Linux, FreeBSD, Mac OS X

#if __GNUC__ >= 4

// GCC 4 has special keywords for showing/hidding symbols,
// the same keyword is used for both importing and exporting
#define DSF_API_EXPORT __attribute__ ((__visibility__ ("default")))
#define DSF_API_IMPORT __attribute__ ((__visibility__ ("default")))

#else

// GCC < 4 has no mechanism to explicitely hide symbols, everything's exported
#define DSF_API_EXPORT
#define DSF_API_IMPORT

#endif

#endif


#endif
```

Export.h

```
#ifndef dsf_Export_h
#define dsf_Export_h

// Headers
#include "Config.h"


// Define portable import / export macros
#if defined(dsf_EXPORTS)

#define DSF_API DSF_API_EXPORT

#else

#define DSF_API DSF_API_IMPORT

#endif

#endif
```

3.2.3 Build Process Management

Different IDEs use different build process managements.

- A bundle with extension name "xcodeproj" is an Xcode project.
- A file with extension name "sln" is a Visual Studio Solution.
- GCC uses Makefile.
- Many others.

CMake can generate different build process systems for those. What I need is a little bit configurations for different IDEs.

```
if (MSVC)
```

```
    # Windows VC
    # Activate C++ exception handling
    if (NOT CMAKE_CXX_FLAGS MATCHES "/EHsc")
    set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} /EHsc")
    endif ()

    # Set Warning level always to 4
    if (CMAKE_CXX_FLAGS MATCHES "/W[0-4]")
        string(REGEX REPLACE "/W[0-4]" "/W4" CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS}")
    else ()
        set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} /W4")
    endif ()
elseif(APPLE)
    # Mac OS X Xcode
    set(CMAKE_MACOSX_RPATH 1)
    ADD_DEFINITIONS(-std=c++11)
else()
    # Unix
    ADD_DEFINITIONS(-std=c++11)
endif()
```

3.3 Implementation

3.3.1 Dual State

The idea to make an object two states is to create an abstract class named "Synchronisable" that any other class inherits it will have two copies. The class is designed as this:

```
template<class T> class Synchronisable
{
protected:
    T* next; //A copy of current object.
public:
    virtual ~Synchronisable() {
        delete this->next;
    }
    virtual void synchronise() = 0; //Signs current value to next value.
};
```

Any class implements the abstract class will automatically generate a copy "next" when an object is created. What we need to do is just override the method "synchronise". For example:

```
#include <dsf/Synchronisable.h>

class Int
{
private:
    int value;
public:
    void setValue(int value)
    {
        this->value = value;
    }
    int getValue() const
    {
        return this->value;
    }
};

class SynchronizedInt : public dsf::Synchronisable<Int>, public Int
{
public:
    void synchronise() override
    {
        this->setValue(next->getValue());
    }
};
```

3.3.2 Read operation and Write operation

Now, we need to make the current value for all read operations, and the next value for all write operations. To implement this, we just need to override the method "setValue".

```
void setValue(int value)
{
    next->setValue(value);
}
```

Every time we call the method "setValue" will effect the next, but not the current. The example code will now like this:

```
#include <dsf/Synchronisable.h>

class Int
{
private:
    int value;
public:
    void setValue(int value)
    {
        this->value = value;
    }
    int getValue() const
    {
        return this->value;
    }
};

class SynchronizedInt : public dsf::Synchronisable<Int>, public Int
{
public:
    void synchronise() override
    {
        this->setValue(next->getValue());
    }
    void setValue(int value)
    {
        next->setValue(value);
    }
};
```

3.3.3 Benchmark Program

For benchmarks, I used SFML to create graphics. Frames per Second is the measurement for the benchmark program. Three methods of different algorithms are used:

- Random
- Elastic collision
- Flocking boids

Because of SFML using opengl, and opengl not supporting multiple-thread rendering, the step of draw elements should be in serial programming phase. A class was designed for SFML:

```
//
//  DSFSFML.h
//  profiler
//
//  Created by Yu Chen on 2/16/15.
//
//

#ifndef profiler_DSFSFML_h
#define profiler_DSFSFML_h

#include <SFML/Graphics.hpp>
#include <vector>

namespace dsf
{
    namespace sfml
    {
        class RenderWindow
        {
        public:
            explicit RenderWindow();
            virtual ~RenderWindow();
            sf::RenderWindow* window;
```

```
            std::vector<sf::Drawable*>* drawables;
        protected:
            virtual void draw() = 0;
        };
    }
}
#endif
```

In parallel programming phase, all elements need to be drawn will be pushed in to the list drawables. In serial phase, all elements in the list will be drawn out.

4 | Description of Learning

4.1 Technical

By developing this framework I learned a lot of information:

- Dynamic-link library.
  - how dynamic-link works on different operating systems
  - export symbols and import symbols
- Intel TBB
  - how Intel TBB implements parallel programming
  - how to execute a program by a specified number of threads
- C++11 features
  - when and how to use smart pointers
  - function pointers and Lambdas
  - override and final keywords
- Mac os X related
  - Mac library bundle
  - Mac application bundle
  - @rpath, @excutable_path, and @load_path
  - how to use otool command to display specified parts of object files or libraries
  - how to use install_name_tool command to edit specified parts of object files or libraries
- Algorithms
  - Elastic collision
  - Flocking Boids
- CMake
  - how to create a Mac bundle, and copy relative files to the bundle
  - how to detect a specified library is installed or not
  - how to add extra headings and libraries
- Others
  - how to use Doxygen to generate LaTex format documents
  - how to use Graphviz and DOT to create inheritance diagrams
  - how to use LaTex to create pdf documents.
  - how to use BibTeX to manage bibliography.

4.2 Personal

I did not practice good Time management and Project management skills during the most of project development. As a result of this project work, I have once more realized that the Time management and Project management are important personal skills, which should be always improved. Also, I have achieved presentation skills through project presentations and demos.

5 | Review of Project

Overall, the project went very smoothly. One of the things I was very happy with was the fact that the benchmark program gave me a surprise. Before the benchmarks, I thought that if the number of threads increases, then the FPS will grow proportionally. However, the result depends on the CPU. For example, my CPU is Intel i5 with 2 cores running 4 threads. The best performance is using 4 threads to run the framework.

5.1 Problems Encountered & Fixes

5.1.1 Memory Leaks

Background

I tries to implement a class type that can be signed as any type of value like this:

```
Any one = int(1);
Any pi = float(3.14);
Any rect = Rectangle(2, 3);
```

Originally, I used void pointer to save the value.

```
class Any
{
private:
    void* value;
}
```

Reason

When I ran the code, there was a memory leak issue. The reason is that a void pointer cannot be released.

```
void* i = new int() // no problem
delete i; // it will not deallocate the memory
```

Solution

I could cast the pointer to a specified type and delete it.

```
void* i = new int() // no problem
delete (int)i; // works fine
```

However, if I did that way, the class Any should be designed as a template class. That would be meaningless. The solution was using smart pointer indeed of traditional pointer.

5.1.2 Wrong path with dynamic-link library

Background

When I tried to package resources to an App bundle, I got an error message:

```
dyld: Library not loaded: libtbb.dylib
  Referenced from: /Users/yuchen/Documents/XCode/profiler/build/Release/profiler.app/Contents/MacOS/
      profiler
  Reason: image not found
```

Reason

Mac App Bundle uses @rpath, @excutable_path, and @load_path to handle the resource path. By using otool command, I got the message as below:

```
MacBook-Pro:libs-osx yuchen$ otool -L libtbb.dylib
libtbb.dylib:
    libtbb.dylib (compatibility version 0.0.0, current version 0.0.0)
    /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1197.1.1)
    mac64/libcilkrts.5.dylib (compatibility version 0.0.0, current version 0.0.0)
    /usr/lib/libstdc++.6.dylib (compatibility version 7.0.0, current version 60.0.0)
    /usr/lib/libgcc_s.1.dylib (compatibility version 1.0.0, current version 2577.0.0)
```

The third line is the reference path for the dynamic-link library.

Solution

The solution was using install_name_tool command to change the reference path.

```
MacBook-Pro:libs-osx yuchen$ install_name_tool -id @executable_path/../Frameworks/libtbb.dylib libtbb.dylib

MacBook-Pro:libs-osx yuchen$ otool -L libtbb.dylib
libtbb.dylib:
    @executable_path/../Frameworks/libtbb.dylib (compatibility version 0.0.0, current version 0.0.0)
    /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1197.1.1)
    mac64/libcilkrts.5.dylib (compatibility version 0.0.0, current version 0.0.0)
    /usr/lib/libstdc++.6.dylib (compatibility version 7.0.0, current version 60.0.0)
    /usr/lib/libgcc_s.1.dylib (compatibility version 1.0.0, current version 2577.0.0)
```

5.1.3 Some functions and variables not defined

Background

I used C++ std library for this project. It worked fine in Mac OS X. However, when I debugged it in Windows and Linux, some error message prompted out.

```
Error   3   error C2065: 'M_PI' : undeclared identifier C:
      \Users\Administrator\Documents\DualStateFramework\profiler\src\BouncingCircleManager.cpp  104 1   profiler
```

Reason

The reason is that M_PI is not "pure" standard. Most compilers regard it as a standard, but Visual C++ not. The similar problem I met is GCC does not know what std::powf is, but both LLVM and Visual C++ use it as a standard.

Solution

Once I knew the reason, the solution was easily made up. For M_PI, I added a preprocessor definition _US↩
E_MATH_DEFINES for Visual Studio by CMakeList:

```
# Add Math definitions
add_definitions(-D_USE_MATH_DEFINES)
```

For std::powf, I changed it to std::pow, which is the standard.

5.2 Attempting the Project Again

If I were to attempt the project again I would avoid using pointers if they are not necessary. I think this is because my first Object Oriented Programming language is Java. In Java word, all objects should be created by keyword "new". However, that is not guaranteed in C++. The reason is Java virtual machine has garbage collection but C++ not, which means in C++, objects created by "new" need user to deallocate them manually.

Pointers sometimes make users confused in a framework. For example, in this framework, some pointers passed by functions will be deleted automatically.

```
void send(SynchronizedObject* to,
          TaskFunction* taskFunction,
          TaskArgument* args);
```

The method "send" takes three pointers as arguments. The third one "args" will be deleted automatically, but "task↩ Function" or "to" not. Users may be confused when should they manually free the pointers. If I did not use pointers as arguments, the code may look like this:

```
void send(SynchronizedObject& to,
          const TaskFunction& taskFunction,
          const TaskArgument& args);
```

Users do not need to worry about memory allocations.

In addition, I would use C++ 14 indeed of C++ 11, because it is the new standard and some features are really useful.

5.3 Advice for Someone Else Attempting a Similar Project

I would advise them to spend more time researching C++ Object-Oriented Concepts. This is because I found that a lot people use OOP languages not in OOP concept. The understanding of OOP concept can make the code usable. Also, I would them to spend a little time figuring out Agile Project Management, which is helpful during the hole project. Some smart tools are recommended:

- Git, for source control
- Doxygen with Graphviz, very powerful for framework API document
- Umlet, for UML diagrams

5.4 Technology Choices

I believe I made the right choices of technologies that I used during this project. The finished project is proof in itself. Intel TBB is powerful and compatible with standard C++ threading library.

6 | Acknowledgement

First of all, I would like to thank my classmates who have been there to help in whatever way they could. Next, I would like to thank my supervisor Joseph Kehoe [20] for the planning of my project. The weekly project meeting was really helpful.

I would like to thank Ken Power [23] for Graphics design and Mathematics related technologies. I would like to thank Philip Bourke [18] for teaching me SFML and recommending me to use Doxygen for API documentation. I would like to thank Christophe Meudec [21], Paul Barry [17], Greg Doyle [19], Marian Murphy and [22] for the Project Management and documentations.

Bibliography

[1] Bundle. webpage, Last checked: Apr. 2015. Available at: https://developer.apple.com/library/mac/documentation/CoreFoundation/Conceptual/CFBundles/BundleTypes/BundleTypes.html. 3

[2] cmake-gui. webpage, Last checked: Apr. 2015. Available at: http://www.cmake.org/cmake/help/v3.0/manual/cmake-gui.1.html. 8

[3] Gcc. webpage, Last checked: Apr. 2015. Available at: https://gcc.gnu.org. 7

[4] Homebrew. webpage, Last checked: Apr. 2015. Available at: http://brew.sh. 8

[5] Intel tbb. webpage, Last checked: Apr. 2015. Available at: https://www.threadingbuildingblocks.org. 8

[6] Llvm. webpage, Last checked: Apr. 2015. Available at: http://llvm.org. 7

[7] Microsoft windows 7. webpage, Last checked: Apr. 2015. Available at: https://support.microsoft.com/en-us/product/windows/windows-7. 7

[8] Os x yosemite. webpage, Last checked: Apr. 2015. Available at: https://www.apple.com/osx/. 7

[9] Sfml. webpage, Last checked: Apr. 2015. Available at: http://www.sfml-dev.org/index.php. 8

[10] Sourceforge. webpage, Last checked: Apr. 2015. Available at: https://sourceforge.net. 3

[11] Sourcetree. webpage, Last checked: Apr. 2015. Available at: https://www.atlassian.com/software/sourcetree/overview. 8

[12] Ubuntu 14.04 lts. webpage, Last checked: Apr. 2015. Available at: http://releases.ubuntu.com/14.04/. 7

[13] Visual studio community. webpage, Last checked: Apr. 2015. Available at: https://www.visualstudio.com. 7

[14] Vmware fusion. webpage, Last checked: Apr. 2015. Available at: www.vmware.com/products/fusion. 7

[15] Xcode. webpage, Last checked: Apr. 2015. Available at: https://developer.apple.com/xcode/. 7

[16] Jernej Barbic. Multiple-core architectures. May 3, 2007, pdf, Last checked: Apr. 2015. Available at: http://www.cs.cmu.edu/~fp/courses/15213-s07/lectures/27-multicore.pdf. iii

[17] Paul Barry. Email: paul.barry@itcarlow.ie. Homepage, Last checked: Apr. 2015. Available at: http://glasnost.itcarlow.ie/~barryp/. 19

[18] Philip Bourke. Email: philip.bourke@itcarlow.ie. Linkedin, Last checked: Apr. 2015. Available at: https://www.linkedin.com/in/bourkephilip. 19

[19] Greg Doyle. Email: Greg.doyle@itcarlow.ie. Linkedin, Last checked: Apr. 2015. Available at: https://www.linkedin.com/in/gregdoyleitcarlow. 19

[20] Joseph Kehoe. Email: joseph.kehoe@itcarlow.ie. Linkedin, Last checked: Apr. 2015. Available at: https://ie.linkedin.com/in/josephkehoe. 19

[21] Christophe Meudec. Email: echancrure@gmail.com. Linkedin, Last checked: Apr. 2015. Available at: https://ie.linkedin.com/pub/christophe-meudec/46/178/811. 19

[22] Marian Murphy. Email: Marian.murphy@itcarlow.ie. 19

[23] Ken Power. Email: ken.power@itcarlow.ie. Linkedin, Last checked: Apr. 2015. Available at: https://ie.linkedin.com/pub/ken-power/46/935/496. 19