

ENGN2912B Scientific Programming in C++

Instructor: Prof. Gabriel Taubin

taubin@brown.edu

TA: Ammar Hattab

Ammar_Hattab@brown.edu



Course Description

- Introduction to the C++ language with examples based on scientific applications
- As a prerequisite, some programming knowledge, e.g., MATLAB projects
- The course will cover the main C++ elements: data types; pointers; references; conditional expressions; streams; templates; Standard Template Library(STL); design and debugging techniques; and much more.



Course Web Site

- <https://canvas.brown.edu/courses/1074198>
- Slides and/or Lecture notes will be provided for all the lectures. Students are supposed to read the lecture notes before each class. Occasionally, additional materials will be handed out and will be available for download from the Canvas web site to complement some lectures.
- Expect one programming assignment per lecture to be submitted through this web site.



Prerequisites / Who should take this course

- This course is aimed at graduate students from Engineering, Computer Science, Applied Mathematics, Physics, Cognitive Science and Neuroscience, with some knowledge of Vector Calculus, Linear Algebra, and Data structures, and some exposure to computer programming



Office Hours

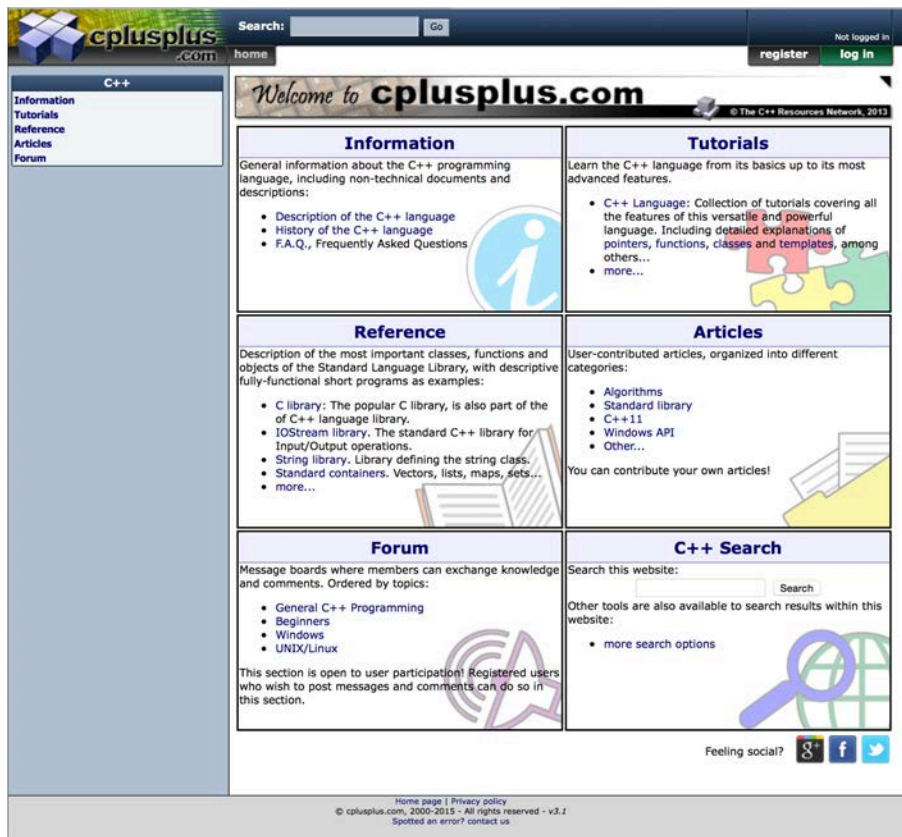
- Taubin / BH328
 - Mondays & Wednesdays 10 am to noon
 - Other times by appointment
- Hattab / BH317
 - Wednesdays 1-2 pm
 - Fridays 2-3 pm



Textbook

- For the first half of the course, we will use as a reference:
- <http://www.cplusplus.com>
- **There are plenty of tutorials and resources on-line !**
- There is no required book, although many good books are available as ebooks from the Brown Library.
- Slides and/or lecture notes will be provided for all the lectures.
- Students are supposed to download and read the lecture notes, which will be available before each class.
- Occasionally, additional materials will be handed out and will be available for download from the course web site to complement some lectures.





Grading

- The course evaluation will be based on an average of one programming assignment per class, and a final project.
- There will not be homework assignments other than the programming assignments and the final project, and there will be neither midterms nor final exam.
- All the programming will in C++.
- As new concepts and techniques are introduced in the lectures, the programming assignments will expose the students to concepts of increasing complexity.
- The final project will be graded incrementally as several assignments. It will span several weeks and will integrate several advanced topics.

Programming Assignments

- Programming assignments are to be submitted electronically through the course web site by midnight Sunday each week.
- Students must be logged in using their user id for the course website.
- Submissions are only accepted as zip files containing your source code, configuration files, documentation, and answers to questions.
- Further details on how to submit homework assignments will be distributed through the course web site.



Programming Assignments Not Graded

- If they are not submitted according to the instructions
- If they do not compile
- If they do not run after compilation
- Submitted assignments may contain bugs
- Late submission of assignments will be penalized
- Not graded assignments can be resubmitted



Plagiarism

- Discussing strategies with fellow students is OK
- But each student has to write its own code
- Cheating is strictly forbidden !!!
- We use advanced software to detect cheating
- If cheating is detected, the case will be reported to the University, and it will be processed as described in The Academic Code
- Don't even think about cheating
- It may lead to dismissal



Class Attendance and Participation

- Class attendance and participation in the class discussions is mandatory, and will contribute to the final grade.
- The programming assignments will contribute 60% to the final grade
- The final project will contribute 30% to the final grade
- Class participation will contribute the remaining 10%



Tentative Lecture Plan (1)

- Introduction
- Multi-Platform development / Cmake / Qt
- Data Types and Arithmetic
- Control Structures
- C++ Memory Allocation
- Pointers and References
- Functions
- Object Oriented Programming / Classes
- More About Classes
- Strings and Streams
- Preprocessor and Templates



Tentative Lecture Plan (2)

- The Standard Template Library (STL)
- Using Libraries / Qt / Boost
- Numerical Methods / Solving Linear Systems / Singular Value Decomposition
- Non-Linear Optimization / Levenberg-Marquardt
- Developing Interactive Applications with Graphical User Interfaces
- Developing Distributed Applications
- Multi-threading and GPU Programming / OpenCL
- Time allowing, other advanced topics



First Task: Set Up Development Environment

- Windows
 - Visual Studio
 - Cygwin
- OSX
 - Xcode
 - Homebrew package manager <http://brew.sh>
- Multi-Platform IDEs
 - Cmake + (Xcode, VisualStudio, or Make)
 - Qt Creator (we will not use)



First Task: Set Up Development Environment

- Command Line Tools
 - Windows: Cygwin
 - OSX / Linux: built-in
 - Linux Tutorial for Beginners
 - <http://www.ee.surrey.ac.uk/Teaching/Unix/>



UNIX Tutorial for Beginners

A beginners guide to the **Unix** and **Linux** operating system. Eight simple tutorials which cover the basics of UNIX / Linux commands.

Introduction to the UNIX Operating System

- What is UNIX?
- Files and processes
- The Directory Structure
- Starting an UNIX terminal

Tutorial One

- Listing files and directories
- Making Directories
- Changing to a different Directory
- The directories . and ..
- Pathnames
- More about home directories and pathnames

Tutorial Two

- Copying Files
- Moving Files
- Removing Files and directories
- Displaying the contents of a file on the screen
- Searching the contents of a file

Tutorial Three

- Redirection
- Redirecting the Output
- Redirecting the Input
- Pipes

Tutorial Four

- Wildcards
- Filename Conventions
- Getting Help

UNIX and Linux books


► If you wish to continue learning Unix, here is a [list of good Unix and Linux books](#), ranging from beginners to advanced.



Package Managers

- OSX
 - Homebrew
- Windows 10
 - PackageManagement (a.k.a. OneGet)
 - Chocolatey
- Linux (Raspberry PI)
 - apt-get





Homebrew
The missing package manager for OS X

English

Install Homebrew

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Paste that at a Terminal prompt.

The script explains what it will do and then pauses before it does it. There are more installation options [here](#) (needed on 10.5).

What Does Homebrew Do?

Homebrew installs **the stuff you need** that Apple didn't.

```
brew install wget
```

Homebrew installs packages to their own directory and then symlinks their files into `/usr/local`.

```
cd /usr/local
find Cellar
Cellar/wget/1.16.1
Cellar/wget/1.16.1/bin/wget
Cellar/wget/1.16.1/share/man/man1/wget.1
ls -l bin
bin/wget -> ../Cellar/wget/1.16.1/bin/wget
```

Fork me on GitHub




Chocolatey

Install About Kickstarter Blog Pricing Packages FAQ Docs Login Signup

The package manager for Windows

Chocolatey - Software Management Automation

The sane way to manage software on Windows

- ✓ Chocolatey builds on technologies you know - unattended installation and PowerShell. Chocolatey works with all existing software installation technologies like MSI, NSIS, InnoSetup, etc, but also works with runtime binaries and zip archives. Go [Pro/Business](#) to dial that experience up to 11!
- ✓ Easily manage all aspects of Windows software (installation, configuration, upgrade, and uninstallation). Chocolatey is the most reliable when software is included in the package, but can also easily download resources.
- ✓ Take advantage of PowerShell to provide automated software management instructions and Chocolatey's [built-in module](#) to turn complex tasks into one line function calls!

```
C:\> choco upgrade nodejs
```

[Install Chocolatey Now](#)

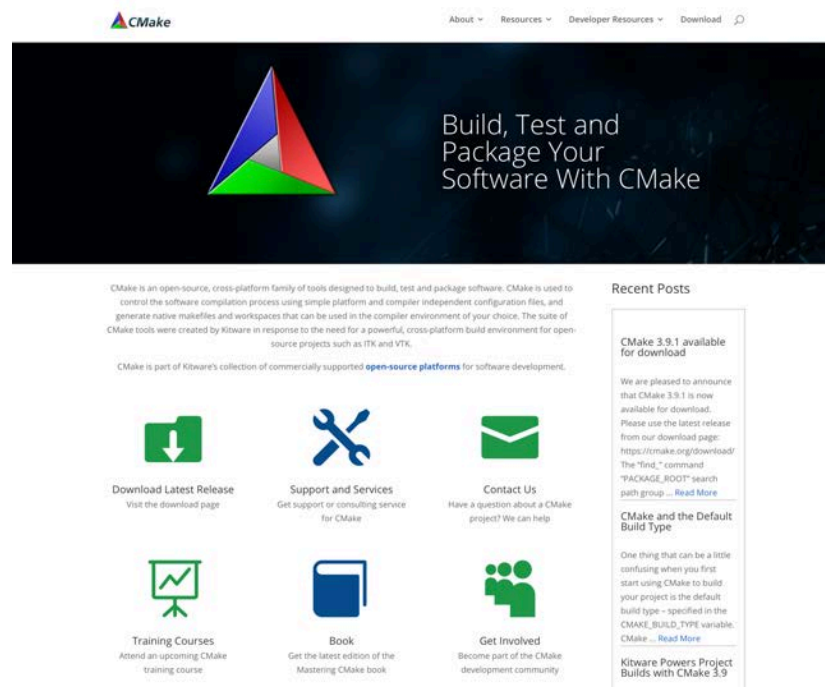


CMAKE

- An extensible, open-source system that manages the build process in an operating system and in a compiler-independent manner.
- <http://www.cmake.org>
- <http://www.cmake.org/Wiki/CMake>



<http://www.cmake.org>



Qt

- **Qt** is a cross-platform application framework that is widely used for developing application software that can be run on various software and hardware platforms with little or no change in the underlying codebase, while having the power and speed of native applications.



One Qt Code

Create Powerful Applications & Devices

We believe modern software development must include a cross-platform user experience and that your tech strategy should be based on easy creation of connected devices, UIs and applications that run anywhere on any device, on any operating system at any time – making your end users' life easier. With Qt, you can do this and more.

Before you begin, make the right license choice.



1

Framework

Write your source code once.
Run it anywhere on any device.

1

Million downloads

Latest version of Qt had over 1M downloads.

1

Ranked #1

Qt is ranked #1 of all cross-platform tools*

1

Unified ecosystem

One ecosystem - Stronger together - One Qt

Leading companies in over 70 industries use Qt to power millions of devices and applications.

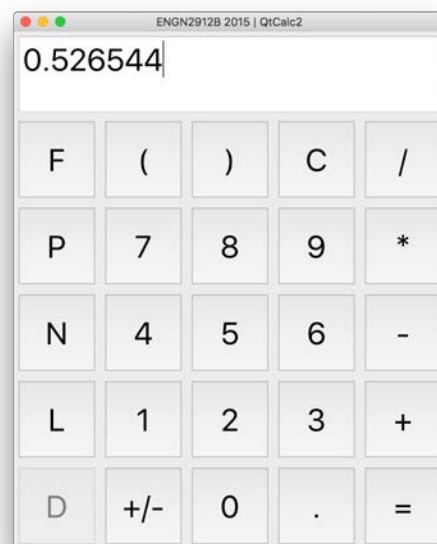
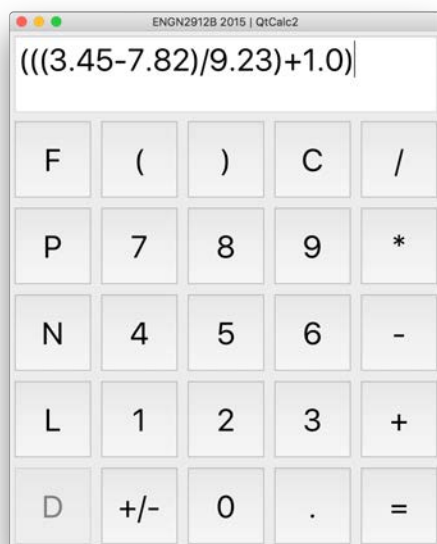


Cmake + Qt

- We will use Cmake to create project file for the native compiler
- Then we will use the native IDE as our development environment
 - Xcode in OSX
 - VisualStudio in Windows
 - Makefiles (command line) in Linux
 - We will not cover Android or iOS development
 - But you can explore what Qt offers for Android and iOS development



Interactive Applications



Final Project = Several Assignments

- IfsViewer

Brown University ENGN2912B Fall 2012 | Scientific Computing in C++ | Lecture 13

The IfsViewer Application

In this lecture we will start to build an interactive application of moderate complexity application for reading, visualizing, operating on, and saving polygon meshes and point clouds. This is a much more complex project than the ones we have worked on so far, which requires careful class design, partitioning the project files into libraries, and also linking your program using external libraries. These libraries are part of even more complex packages, which you will have to install in your machine, and configure for use in conjunction with CMake.

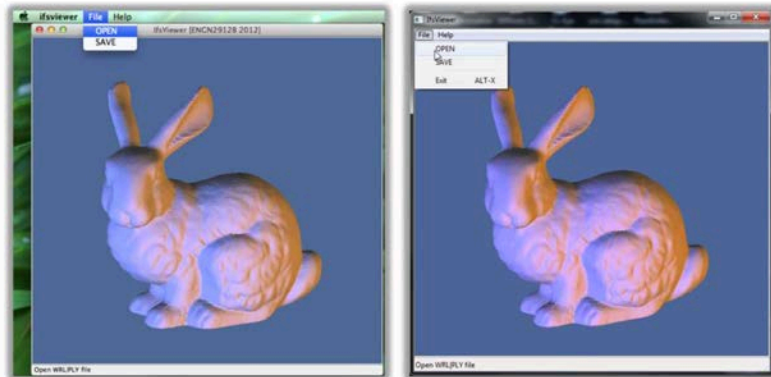


Figure 1 The IfsViewer application shown running in OSX and Windows.



BROWN

ENGN 2912B / Lecture 1

WED Sep 06 2017

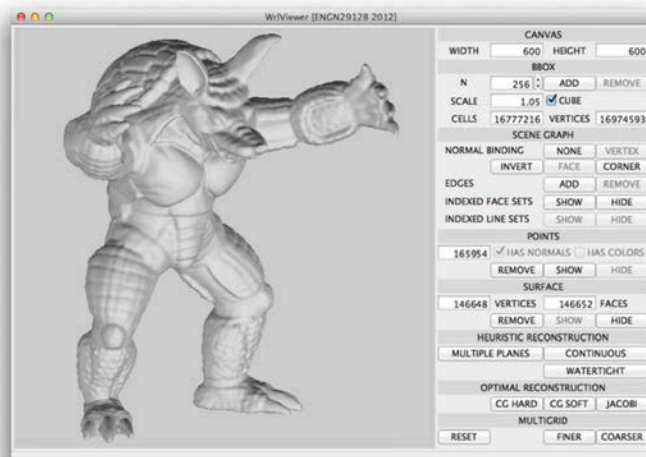
27

Final Project

- WrlViewer

Brown University ENGN2912B Fall 2012
Scientific Computing in C++ | Lectures 20-24

Final Project : WrlViewer4



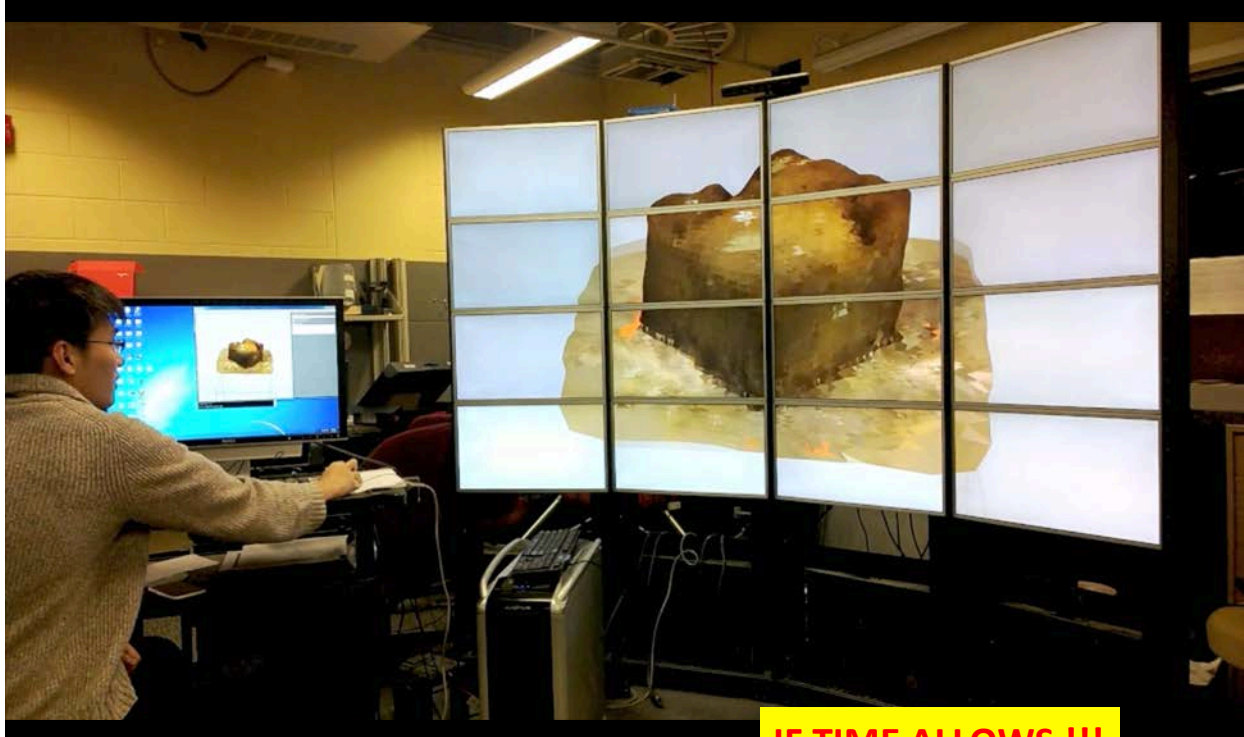
BROWN

ENGN 2912B / Lecture 1

WED Sep 06 2017

28

C++ / MPI / Raspberry PI 2 ?



IF TIME ALLOWS !!!

Distributed Video Wall 3D Viewer

