CS 5721 Computer Graphics

P. Willemsen

University of Minnesota Duluth

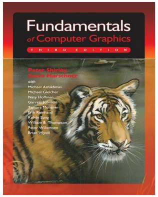
January 31, 2013

Important Info

- **Lecture** - T/Th 2:00pm - 3:15pm, HH 306
- **Lab** - T 4:00pm - 4:50pm, MWAH 187
- My Office Hours: T 10:00am - 11:00am, Th 9:30am - 11:00am
  - Email: willemsn@d.umn.edu
  - Will *try* to be available via Google chat during office hours
- TA: Alex Geng
  - Email: gengx064@d.umn.edu
  - TA Office Hours:
    - T 1:00pm - 2:00pm, HH 314
    - Th 10:00am - 12:00pm, MWAH 187
    - F 10:00am - 11:00am, MWAH 177

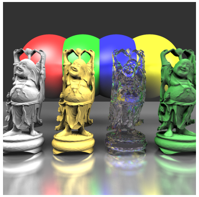
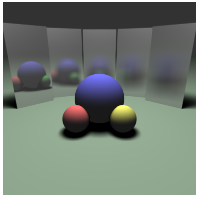
Reading Materials

- Fundamentals of Computer Graphics, 3rd Edition. Pete Shirley.
  - Required Reading!



Information

- Course Requirements
  - Programming Assignments (30%) - A major portion of the course will be through the implementation of computer graphics algorithms and applications! There will be 4 major programming assignment. The assignments will focus on building various components related to the generation of 3D computer graphics. Programming assignments will be coded using C++.

Information

- Course Requirements, cont'd.
  - Lab (30%) - The scheduled lab time will be used as a place to learn skills that will improve abilities and increase practical knowledge for creating computer graphics programs. Special emphasis will be placed on building an interactive, real-time ray tracer to complement your software ray tracer from the assignments.

Information

- Course Requirements, cont'd.
  - Readings, Discussions, Quizzes (10%) - About every other week, a reading, discussion, or small quiz will be given to gauge your reading of the course material.

Information

- Course Requirements, cont'd.
  - Readings, Discussions, Quizzes (10%) - About every other week, a reading, discussion, or small quiz will be given to gauge your reading of the course material.
  - Exams (30%) - There will be three exams over the course of the semester. The final exam is one of these exams. According to the Final Exam Schedule, the date of the final exam is Friday, May 11, 2011 from 2:00pm - 3:55pm.

Moodle

- http://moodle2.umn.edu/
  - Used for course management, forums, posting of assignments, etc...
  - Login via your X.500 login information.
- Simplified interface with approximately 4 sections.
- Programming assignments, lab, quizzes, and potentially exams will use moodle.
- Forums will be used extensively for feedback and assessment on programming assignments and labs.

First Lab

- Go to main CS Office (HH 320) and get key card access to MWAH 187. Also, pick up your hard drive key.

First Lab

- Go to main CS Office (HH 320) and get key card access to MWAH 187. Also, pick up your hard drive key.
- Sometime in the next week, install Ubuntu Linux 12.04 on your hard drive. Have this completed by the time the next lab starts.
- Alex will have some USB drives with Ubuntu on them, ready for you to install. Check them out from Alex and please return them!

First Lab

- Go to main CS Office (HH 320) and get key card access to MWAH 187. Also, pick up your hard drive key.
- Sometime in the next week, install Ubuntu Linux 12.04 on your hard drive. Have this completed by the time the next lab starts.
- Alex will have some USB drives with Ubuntu on them, ready for you to install. Check them out from Alex and please return them!
- If you have a preferred Linux distro, you are free to use that... however, my instructions will be based on Ubuntu 12.04, so be prepared.

The Appearance of Things

Much of what we will be doing involves writing computer programs to mimic, approximate the appearance of things in our world. For our first in-class assignment, you are to act as "scientists" making observations about the things you see.

- Pair up with a partner, grab some of the items I've brought along and start observing what these thing look like and why they look that way.

The Appearance of Things

Much of what we will be doing involves writing computer programs to mimic, approximate the appearance of things in our world. For our first in-class assignment, you are to act as "scientists" making observations about the things you see.

- Pair up with a partner, grab some of the items I've brought along and start observing what these thing look like and why they look that way.
- Think about the light hitting the objects
- Think about how these objects may interact with each other

The Appearance of Things

Much of what we will be doing involves writing computer programs to mimic, approximate the appearance of things in our world. For our first in-class assignment, you are to act as "scientists" making observations about the things you see.

- Pair up with a partner, grab some of the items I've brought along and start observing what these thing look like and why they look that way.
- Think about the light hitting the objects
- Think about how these objects may interact with each other
- Why are you seeing what you are seeing when you look at things
- Think about the *computation* behind what you are seeing

The Appearance of Things

Much of what we will be doing involves writing computer programs to mimic, approximate the appearance of things in our world. For our first in-class assignment, you are to act as "scientists" making observations about the things you see.

- Pair up with a partner, grab some of the items I've brought along and start observing what these thing look like and why they look that way.
- Think about the light hitting the objects
- Think about how these objects may interact with each other
- Why are you seeing what you are seeing when you look at things
- Think about the *computation* behind what you are seeing
- Write down your observations and discuss them. After a while, switch partners and/or objects.

Instances of Computer Graphics

Course will focus on hitting the following topics in computer graphics through programming assignments and labs.

- Image-order algorithms - Ray Tracing

Instances of Computer Graphics

Course will focus on hitting the following topics in computer graphics through programming assignments and labs.

- Image-order algorithms - Ray Tracing
- Object-order algorithms - Rasterization

Instances of Computer Graphics

Course will focus on hitting the following topics in computer graphics through programming assignments and labs.

- Image-order algorithms - Ray Tracing
- Object-order algorithms - Rasterization
- Real-time graphics - OpenGL

Instances of Computer Graphics

Course will focus on hitting the following topics in computer graphics through programming assignments and labs.

- Image-order algorithms - Ray Tracing
- Object-order algorithms - Rasterization
- Real-time graphics - OpenGL

- These topics will provide solid overview and understanding of computer graphics.

Useful and Necessary Tools

You will need to be familiar with the following (ASAP):

- Linux editors and C++ compiler

Useful and Necessary Tools

You will need to be familiar with the following (ASAP):

- Linux editors and C++ compiler
- CMake - build system meta tool

Useful and Necessary Tools

You will need to be familiar with the following (ASAP):

- Linux editors and C++ compiler
- CMake - build system meta tool
- Subversion - code repository software
  - Requirement! You must demonstrate that you are using a subversion system.

Writing Code - Editor and Compiler

Editing your code with popular and not-so popular editors:

- gedit
- emacs
- vi
- *others???*

You will compile your code with the GNU C++ compiler for your assignments. Here's some helpful review:

- Use C++ classes well! Try C++-11 for extra credit (some compilers will work better than others).
- Declarations go in header file (.h); Definitions go in source file (.cpp)
- What's an inline function and when to use them?
- Use the *-g* and *-Wall* flags and pay attention to it
- Use the debugger! It's called *gdb* on Linux.

Building your Code

Use CMake!
CMake is a high level build system that will let you cross compile your code on numerous build environments.
Why you want to learn about it and use it:

- Good for bigger projects; no messing around typing command line stuff to compile your code!
- Good for building libraries of your code!
- Cross-platform development.

All CMake-based project will contain a file, called `CMakeLists.txt`:

- CMakeLists.txt contains the build instructions for your project.
- Specifies which files are part of the project and any dependencies.

Simple CMake Example

You have one file set that contains your class and a file that contains your main.

```
main.cpp  ClassTest.h  ClassTest.cpp
```

Normally, you could use a Makefile to *build* this into your executable. But Makefiles are not native to Visual Studio (Windows) and Xcode (OS X) and you want to develop there also.

CMake can help. Here is what the CMakeLists.txt file looks like to build your project:

```
CMAKE_MINIMUM_REQUIRED (VERSION 2.8)
PROJECT (ClassTest)

ADD_EXECUTABLE (classTest
                classTest.cpp classTest.h
                main.cpp)
```

Simple CMake Example Cont'd

Once you have your CMakeLists.txt file, you can build your project. A couple notes:

- Keep your sources separate from the build directories. I create a build directory to hold the "build" files. This is similar to what VS and other IDEs do.
- Only commit your sources, not your build directories to your software versioning repository (svn).

How to build:

```
mkdir build
cd build
cmake ..
make
```

- If your build directory is already there, you don't need to re-create it, nor run the cmake command again.
- You only need run the make command
- You can clean and build all with the *make clean* command provided in your auto-generated Makefile (CMake did this for you)

Keeping your Code

Use a software repository and versioning system!
Subversion is good, but there are others that are useful if you know how to use them:

- git
- mercurial
- *numerous others...*

Keeps a backup of your code!
Let's your code be accessed from many places!
For multi-person projects, this is a MUST!

subversion

Documentation - http://svnbook.red-bean.com/

- Revision control system
- Store your code at various states in time
- Tag your code with easy to remember names
- Branch your code for explorative dev

- Possible to setup with an Apache server
- But even simpler to setup without a dedicated server!
  - Requires that your machine has *ssh* and *subversion* installed
  - Machine should also be "online"

subversion Setup

First thing to do

- Create directory in your account to house your projects

Next,

- Create subversion repository for your project
  - *cd your_directory*
  - *svnadmin create project1*
- This creates the "database" that subversion uses to keep track of your project.

Checking out your Project

Use the SSH protocol do to this securely

- svn co svn+ssh://*userid*@*machine*/path_to_repo_here *your local name*

Example

- svn co svn+ssh://willemsn@marengo/home/vol/svnrepo/project1 project1

Creating the Initial Project Structure

- I tend to like having a *trunk* (mainline of development), a *tags*, and a *branches* directory structure to organize my subversion projects. After creating project with svnadmin, follow the example below to get setup this way:
  - svn co svn+ssh://willemsn@marengo/home/vol/svnrepo/project1 project1
  - cd project1
  - mkdir -p trunk tags branches
  - svn add trunk tags branches
  - svn commit -m "Added initial dir structure." trunk tags branches
  - cd ..
  - /bin/rm -fr project1
  - svn co svn+ssh://willemsn@marengo/home/vol/svnrepo/project1/trunk project1
- You'll then be ready to go with your main trunk of development!

Important SVN Commands

- Add new files or directories (recursively) to the repository
  - svn add [files or dirs]
- Update your copy from the repository
  - svn update
- Commit your updates
  - svn commit -m "My comment about updates" [dirs or specific files]
- Differences between your copy and repository
  - svn diff -rHEAD [my file or directory]
- Local status of your repository
  - svn status

git

Documentation - http://git-scm.com/documentation

- Revision control system
- Store your code at various states in time
- Manages changes to a tree of files
- Optimized for distributed development and large file set
- Also good at merges and branching
- Originally developed by Linus Torvalds

Stanford has a nice basic overview of git commands:
http://www-cs-students.stanford.edu/˜blynn/gitmagic/ch02.html

git Setup

We'll use a directory on a CS machine to hold your git repository (marengo would be good to use):

- Create directory in your account to house your projects

Next,

- Create git repository for your project
  - *cd your_git_directory*
  - *GIT_DIR=project1.git git init*
  - *cd project1.git*
  - *cp hooks/post-update.sample hooks/post-update*
- This creates the main git master repository to keep track of your project.

Cloning the Project with git

On a remote machine

- git clone username@marengo.d.umn.edu: /your_git_directory/project1.git
- cd project1
- emacs README
- git commit -m "Added the readme file"
- git push origin master

I suggest you read up on git.

Setting up SVN with an External Repository

Say you want to write your own code and place it in your own repository **but**, you need to use an external subversion repository with your code. Here's one way to handle that with an external linkage:

- First, change into the directory to where you want the external working repository to eventually exist.
- Next, issue the `svn propset` command in the following manner to link a directory in your repository to an external svn repository:

s

vn propset svn:externals 'cs5721Lib https://wind.d.umn.edu/cssvn/cs5721/trunk' .

- After you do this, you'll need to commit those changes.
- And, then, you can update to pull the recent changes from the linked repository.