Graphics and OpenGL Programming – Comp 175 Notes

Table of Contents

About this Guide

Please view this guide as a supplement to the course resources (website, lectures, -in-class labs. recommended course book, etc.). There is no guarantee that this will be updated along with the course, however I will put here examples and other discussion about questions people ask. Please take notes in class as you normally would. Please do not redistribute this guide without our permission.

Get Inspired!

SIGGRAPH

Link: http://www.siggraph.org/

SIGGRAPH is the ACM Association for computer graphics. It is the top conference where the latest and greatest advances in computer graphics. The techniques presented in these papers are often the results found from research and animation studios around the world (What you see at these conferences is also a preview of what will be in gaming in 5-10 years).

The latest and greatest in graphics from SIGGRAPH 2013:
http://www.youtube.com/watch?v=JAFhkdGtHck

Questions asked during office hours

This section is for me to post questions during office hours, or e-mails from students. If there is something related to the course logistics, then please consult the course webpage (http://www.cs.tufts.edu/comp/175/), as this section is more for explaining conceptual questions. My rationale is I can better draw out figures in this document as opposed to an e-mail, as well as keep a record of questions that multiple students may have.

Piazza Forums

Please use the Piazza forums to ask questions and help your classmates. Please do not post solutions on the forums however. If you need to be added to the forum, please send Remco or the TAs an e-mail.

Question: My program runs, but really really slow!

Sol. (work in progress)

Setting up OpenGL on your platform

Setting up OpenGL to work on your Unix machine, PC, or Mac requires that you download the appropriate libraries for each platform. Go ahead and skip below to your selected platform to get started. Optionally, you may want to read the explanation below to understand what is going on, in case you receive some error messages.

Explanation of linking libraries:

You have previously had experience adding header files to your programs which give you additional functionality (For example, once you #include <string> you can use the string command set). We are going to be adding libraries to our code, which are essentially pre-compiled versions of libraries that have functionality ready for us to use. The functionality we have available is (add some more later).

Setting up OpenGL on the Mac through the Terminal

(Tested on Mac OSX 10.8.4 | December 16, 2013)

Setting up FreeGlut

0.) Download freeglut 2.8.1 (http://freeglut.sourceforge.net/)
1.) Extract the files to a directory
2.) Navigate to this directory in a terminal
3.) Run the command:
“env CPPFLAGS="-I/opt/X11/include" LDFLAGS="-L/opt/X11/lib" ./configure”

./configure CFLAGS="-I/usr/X11/include/X11/extensions -L/usr/X11/lib" LIBS="-lXrandr -lXxf86vm -lXi" --with-x --x-includes=/usr/X11/include

4.) Next type the command ‘sudo make install’ and then enter your password for your machine.
5.) You will see a make script being executed and when it finishes you will be ready t go.

Directions based on this External Resource:
http://lazyfoo.net/tutorials/OpenGL/01_hello_opengl/mac/index.php

Setting up FreeGlut through homebrew

Some students have had trouble using the above instructions. Alternatively they can try this:

1.) Go to: http://brew.sh/
2.) In a terminal enter:
ruby -e "$(curl -fsSL https://raw.github.com/Homebrew/homebrew/go/install)"
3.) When installation is finished, you should then run “brew doctor”
4.) It may recommend you install XCode or some other libraries, so go ahead and do so if you like.
5.) Next enter “brew install freeglut” in the terminal. This should generate the correct libraries.

Setting up GLUI

0.) Download the GLUI Package from here:
https://lukecyca.com/2008/glui-235-framework-for-mac-os-x.html
1.) Install the package (double click it)
2.) The package will be installed in /Library/Frameworks/
(The header files will be here if you search ‘cd /Library/Frameworks/GLUI.Framework/)
3.) Navigate to here: “cd /Library/Frameworks/GLUI.framework/Headers”
4.) Create a link to this directory “sudo ln –s ../Headers/ GL”

Note: If the files were not put into “/Library/Frameworks” please move them there to ensure the libraries can be found.

Your first Mac OpenGL Program

1.) Create a directory (“mkdir hello_opengl”)
2.) cd hello_opengl
3.) Download one of the example GLUI files(example1.cpp example2.cpp etc.) and run the following command line.

g++ -Wall -Wextra example.cpp -I/Library/Frameworks/GLUI.framework/Headers/ -framework OpenGL -framework GLUT -framework GLUI -o hello_opengl

4.) Run with “./hello_opengl”

If you’re having trouble compiling, alternatively try:

```
g++ example2.cpp -L/path_to_glui_lib -I/path_to_glui_include -framework OpenGL -framework GLUT -lglui -o hello_opengl
```

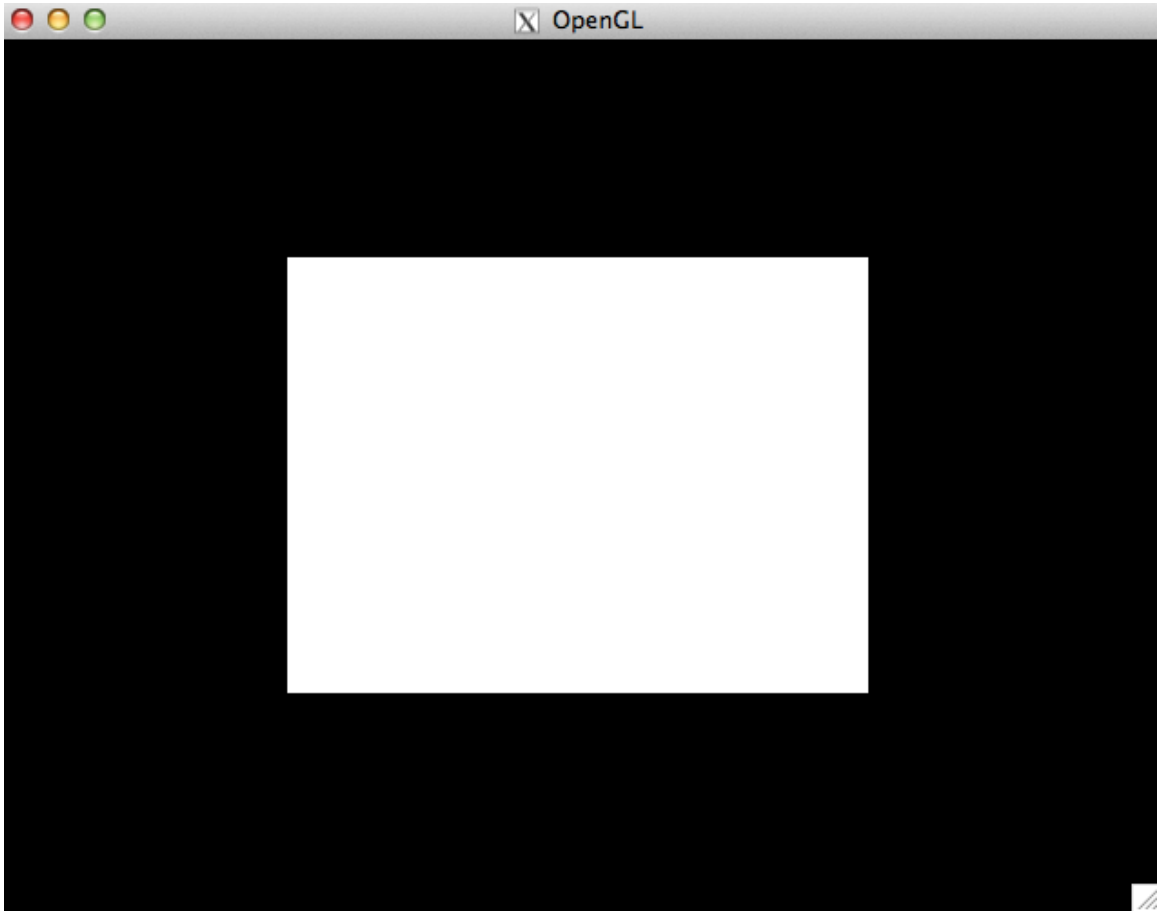
Your first OpenGL Program (Testing the installation)

0.) Create a new directory on your system (e.g. ‘mkdir inclass01’)

1.) Copy the from this directory We’ll have some directory here

2.) Compile using the command

3.) You should see the following: (Might need to change window depending on what we render)



Setting up GLEW (Do not worry about this for first half of course)

0.) Download glew version 1.10.0 .TGZ from http://glew.sourceforge.net/1.) Navigate to the directory that contains the glew folder you downloaded.

2.) Type in 'make install' (note: you might have to do 'sudo make install' and enter your password if you do not have permission to write to your usr/bin folders.

```
Michaels-MacBook-Air-3:glew-1.10.0 michaelshah$ sudo make install
Password:
install -d -m 0755 /usr/include/GL
install -m 0644 include/GL/wglew.h /usr/include/GL/
install -m 0644 include/GL/glew.h /usr/include/GL/
install -m 0644 include/GL/glxew.h /usr/include/GL/
cc -DGLEW_NO_GLU -O2 -Wall -W -Iinclude -dynamic -fno-common -fPIC  -o tmp/darwi
n/default/shared/glew.o -c src/glew.c
```

3.) Wait while the program builds, and if all goes correct you will see the following:

```
install -d -m 0755 /usr/lib
install -m 0644 lib/libGLEW.1.10.0.dylib /usr/lib/
ln -sf libGLEW.1.10.0.dylib /usr/lib/libGLEW.1.10.dylib
ln -sf libGLEW.1.10.0.dylib /usr/lib/libGLEW.dylib
install -m 0644 lib/libGLEW.a /usr/lib/
install -d -m 0755 /usr/lib
install -d -m 0755 /usr/lib/pkgconfig
install -m 0644 glew.pc /usr/lib/pkgconfig/
```

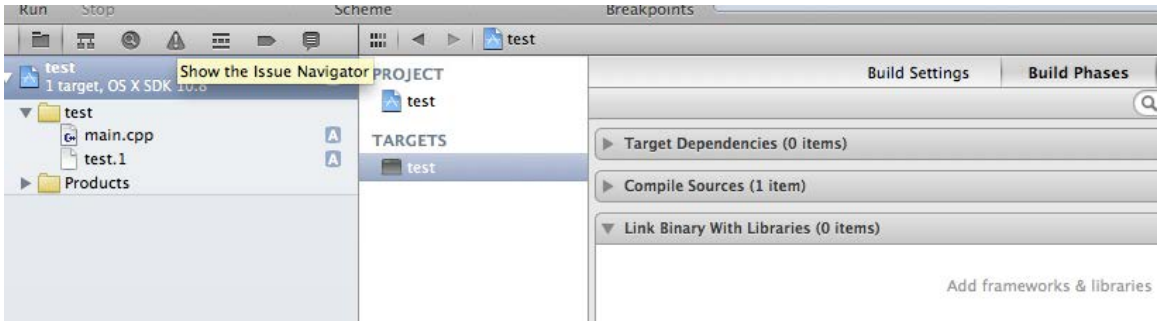
Setting up OpenGL on Mac with XCode

(Note that for this class using the terminal is recommended as I'm not an XCode guru. I'm happy to write more instructions though if it helps you)

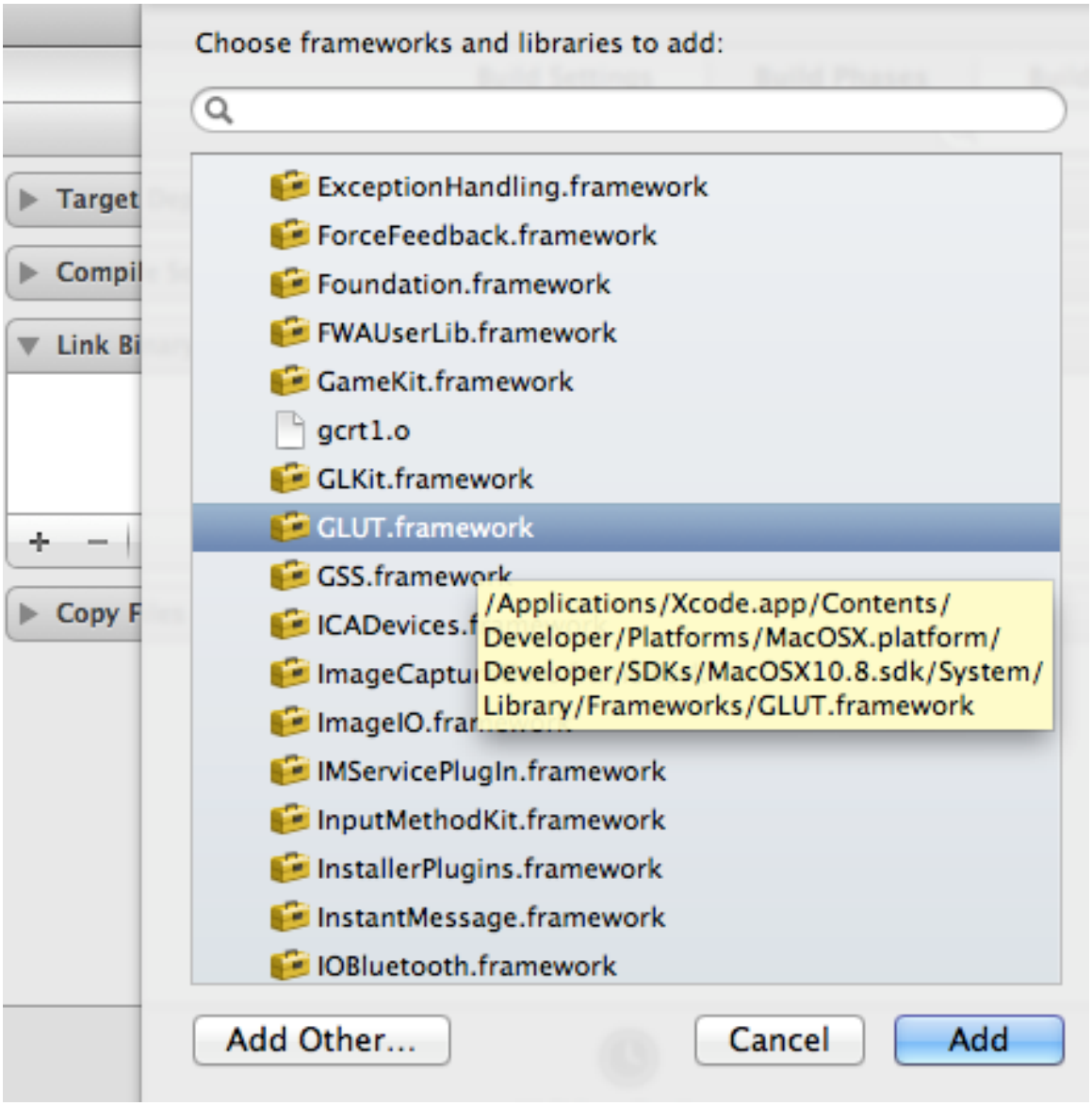
`.) Create a new Command Line Project

2.) Name your project, and ensure that it's a C++ project.

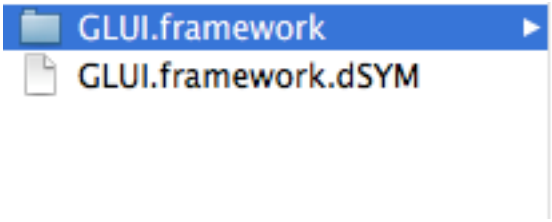
3.) Click on your project (in this example, called 'test') and then navigate to the Build Phases tab.:



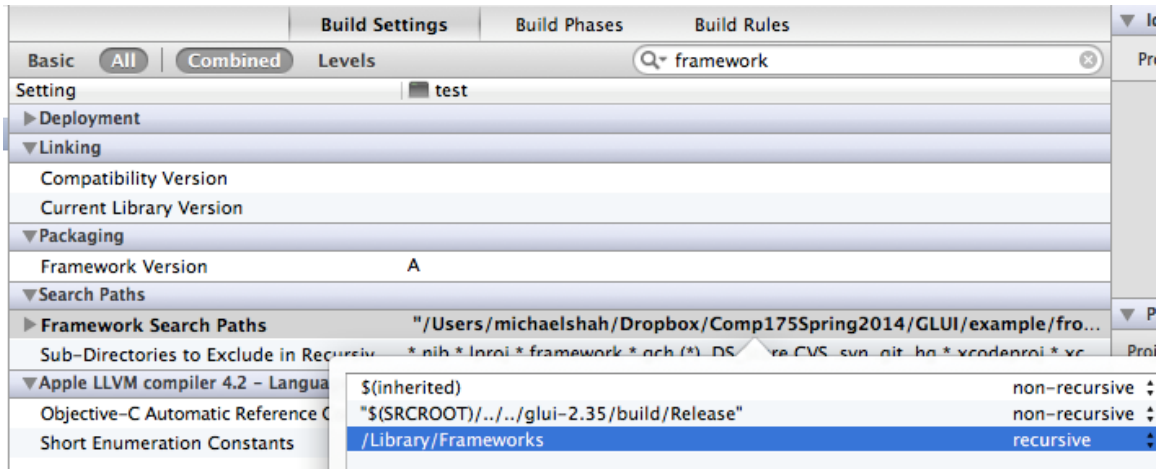
4.) Click the (+) sign and Add GLUT, and OpenGL



5.) Select 'Add Other…' and then add GLUI.framework



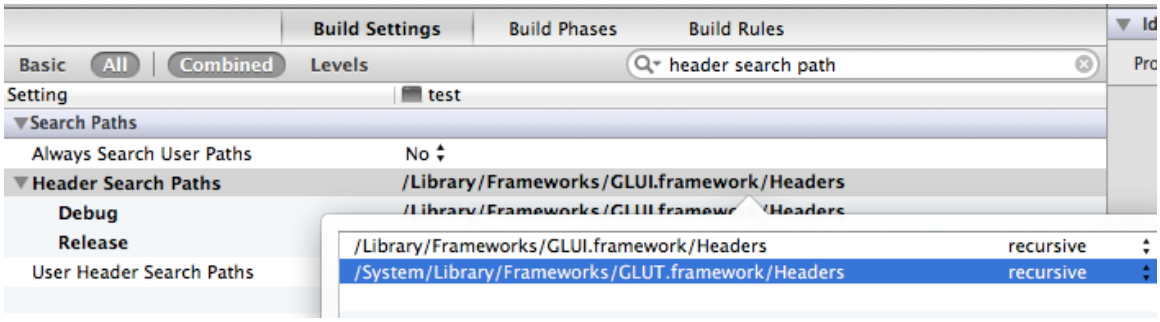
5.) Under the 'Build Settings' tab, add /Library/Frameworks and set to recursive



6.) Under the 'Build Settings' tab, add:

/Library/Frameworks/GLUI.framework/Headers

/System/Library/Framework/GLUT.framework/Headers



External Resources notes based on: http://www.cs.unh.edu/~cs770/docs/gluiMacInstall-10.8.pdf

Setting up FreeGlut on your Unix Machine for your Account

(Note freeglut may be installed on your Unix system already)

1.) Download freeglut 2.8.1

Follow these instructions (Also listed posted below in steps 2-7):
http://asymptote.sourceforge.net/doc/Compiling-from-UNIX-source.html

2.) If you are root:

2a) Type in a terminal in the root directory ./configure –prefix=/usr

3.) If you are not root:

4) Create a directory called 'freeglut'
5.) ./configure –prefix=$HOME/Desktop/comp175/freeglut
6.) make all
7.) make install

Setting up GLUI on your Unix Machine

1.) Download GLUI 2.3.6
2.) Run gunzip on the file, then untar it using (tar -xvf filename.tar)
3.) Open makefile
4.) uncomment out sections for glut, and comment in sections of code for glut.
5.) Set the file paths for libglut to:

-L/h/mshah08/Desktop/comp175/freeglut/lib –freeglut
-I/h/ mshah08/Desktop/comp175/mshah08/freeglut/include --DGLUI_FREEGLUT

6.) type 'make all'

Setting up GLEW on your Unix Machine

1.) Go download tgz source of GLEW from: http://glew.sourceforge.net/
2.) untar the file
3.) setenv GLEW_DEST $HOME/
4.) 'setenv GLEW_DEST $HOME/Desktop/comp175/'
5.) 'echo GLEW_DEST' to make sure it worked correctly
6.) 'make install'

Compiling and running an example

Testing

Sample makefile will be given

1.) setup an environment variable for your $LD_LIBRARY_PATH

```
Example: setenv $LD_LIBRARY_PATH
/user/local/lib:/usr/sup/lib:/h/mshah08/Desktop/comp175/lib
```

2.) If you don't like linking dynamically, you may also link statically with the .a file.

3.) Run 'make'

(Note you may have to set the environment variable for every session).

Setting up OpenGL, FreeGLUT, GLUI, and GLEW on your own Linux machine via apt-get

This was done on Lubuntu 12.04, so there may be new version numbers available. Use 'apt-cache search' to find them.

Start-Date: 2014-01-07 17:13:48
Commandline: apt-get install mesa-common-dev:i386
Upgrade: libgl1-mesa-dev:i386 (8.0.4-0ubuntu0.6, 8.0.4-0ubuntu0.7), libgl1-mesa-glx:amd64 (8.0.4-0ubuntu0.6, 8.0.4-0ubuntu0.7), libgl1-mesa-glx:i386 (8.0.4-0ubuntu0.6, 8.0.4-0ubuntu0.7), libglapi-mesa:amd64 (8.0.4-0ubuntu0.6, 8.0.4-0ubuntu0.7), libglapi-mesa:i386 (8.0.4-0ubuntu0.6, 8.0.4-0ubuntu0.7), mesa-common-dev:i386 (8.0.4-0ubuntu0.6, 8.0.4-0ubuntu0.7)
End-Date: 2014-01-07 17:14:07

Start-Date: 2014-01-07 17:17:07
Commandline: apt-get install freeglut3
Install: freeglut3:amd64 (2.6.0-1ubuntu3)
End-Date: 2014-01-07 17:17:12

Start-Date: 2014-01-07 17:17:44
Commandline: apt-get install freeglut3-dev
Install: libdrm-nouveau2:amd64 (2.4.46-1ubuntu0.0.0.1, automatic), libgl1-mesa-dev:amd64 (8.0.4-0ubuntu0.7, automatic), mesa-common-dev:amd64 (8.0.4-0ubuntu0.7, automatic), libglu1-mesa-dev:amd64 (8.0.4-0ubuntu0.7, automatic), libdrm-dev:amd64 (2.4.46-1ubuntu0.0.0.1, automatic), libxext-dev:amd64 (1.3.0-3ubuntu0.1, automatic), libkms1:amd64 (2.4.46-1ubuntu0.0.0.1, automatic), freeglut3-dev:amd64 (2.6.0-1ubuntu3)
Upgrade: libdrm-nouveau2:i386 (2.4.43-0ubuntu0.0.2, 2.4.46-1ubuntu0.0.0.1), libdrm-radeon1:amd64 (2.4.43-0ubuntu0.0.2, 2.4.46-1ubuntu0.0.0.1), libdrm-radeon1:i386 (2.4.43-0ubuntu0.0.2, 2.4.46-1ubuntu0.0.0.1), libdrm2:amd64 (2.4.43-0ubuntu0.0.2, 2.4.46-1ubuntu0.0.0.1), libdrm2:i386 (2.4.43-0ubuntu0.0.2, 2.4.46-1ubuntu0.0.0.1), libdrm-dev:i386 (2.4.43-0ubuntu0.0.2, 2.4.46-1ubuntu0.0.0.1), libdrm-nouveau1a:amd64 (2.4.43-0ubuntu0.0.2, 2.4.46-1ubuntu0.0.0.1), libdrm-nouveau1a:i386 (2.4.43-0ubuntu0.0.2, 2.4.46-1ubuntu0.0.0.1), libdrm-intel1:amd64 (2.4.43-0ubuntu0.0.2, 2.4.46-1ubuntu0.0.0.1), libdrm-intel1:i386 (2.4.43-0ubuntu0.0.2, 2.4.46-1ubuntu0.0.0.1), libkms1:i386 (2.4.43-0ubuntu0.0.2, 2.4.46-1ubuntu0.0.0.1), libglu1-mesa:amd64 (8.0.4-0ubuntu0.6, 8.0.4-0ubuntu0.7), libglu1-mesa:i386 (8.0.4-0ubuntu0.6, 8.0.4-0ubuntu0.7)
Remove: libqt4-opengl-dev:i386 (4.8.1-0ubuntu4.4), libgl1-mesa-dev:i386 (8.0.4-0ubuntu0.7), mesa-common-dev:i386 (8.0.4-0ubuntu0.7), libglu1-mesa-dev:i386 (8.0.4-0ubuntu0.6), libxext-dev:i386 (1.3.0-3ubuntu0.1)
End-Date: 2014-01-07 17:19:19

Start-Date: 2014-01-14 14:55:49
Commandline: apt-get install libglui-dev libglui2c2
Install: libglui2c2:amd64 (2.36-4ubuntu1), libglui-dev:amd64 (2.36-4ubuntu1)
End-Date: 2014-01-14 14:56:17

Start-Date: 2014-01-17 15:40:39
Commandline: apt-get install libglewmx1.6 libglew1.6-dev
Install: libglew1.6-dev:amd64 (1.6.0-4), libglewmx1.6:amd64 (1.6.0-4)
End-Date: 2014-01-17 15:40:54

Setting up FreeGLUT on Windows

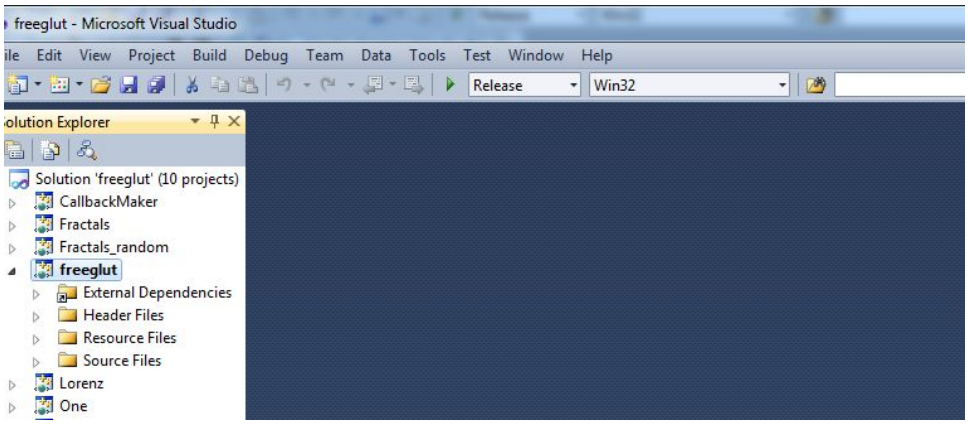
1) download freeglut 2.8.1 from http://freeglut.sourceforge.net/index.php#download

2) unzip, untar the directory

3) go into the folder "VisualStudio", and select the appropriate sub-folder based on the version of your Visual Studio

4) Double click on the freeglut.sln file to launch Visual Studio

5) Make sure that the compilation setting says "Release" and "Win32"



6) Right click on the project "freeglut" and click "Build"

7) When the compilation completes, go back to the main freeglut-2.8.1 folder (two levels up), go into the folders lib\x86

7a) copy freeglut.lib into C:\Program Files (x86)\Microsoft Visual Studio 10.0*\VC\lib

7b) copy freeglut.dll into C:\Windows\system

8) go back to freeglut-2.8.1 folder, this time go into the folder include

8a) copy the GL folder into C:\Program Files (x86)\Microsoft Visual Studio 10.0*\VC\include

*or whatever version you are currently using

Setting up GLUI on Windows

1) download glui-2.35 from http://glui.sourceforge.net/

2) unzip, untar the directory

3) go into the folder glui-2.35\src\msvc

4) double click on glui.sln

4a) It is possible that your Visual Studio will ask you to convert the files, say yes to that (and it doesn't really matter if you choose to backup the fold files)

5) Make sure that the compilation says "Release" and "Win32"

6) Right click on the project _glui library and check "Build"

7) When the compilation completes, go back to glui-2.35\src\msvc\lib, copy glui32.lib into C:\Program Files (x86)\Microsoft Visual Studio 10.0*\VC\lib

8) Go back to Visual Studio, this time make sure that the compilation says "Debug" and "Win32"

9) Right click on the project _glui library and check "Build"

10) When the compilation completes,

7a) go back to glui-2.35\src\msvc\lib, rename glui32.lib as glui32d.lib

7b) copy glui32d.lib into C:\Program Files (x86)\Microsoft Visual Studio 10.0*\VC\lib

11) Go back to glui-2.35\src\include\GL, copy the glui.h file into C:\Program Files (x86)\Microsoft Visual Studio 10.0*\VC\include\GL

*or whatever version you are currently using

Setting up GLEW on Windows

Compiling and running an example

1) Go to File -> `New -> Project

2) Enter the following information

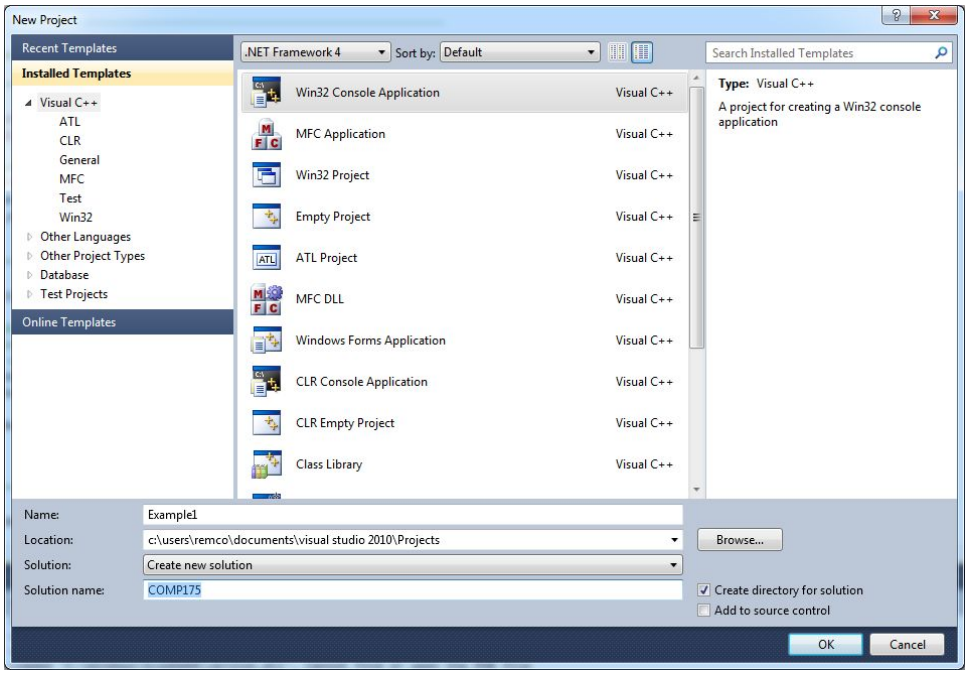
2a) Select Win32 Console Application

2b) Name: Example1

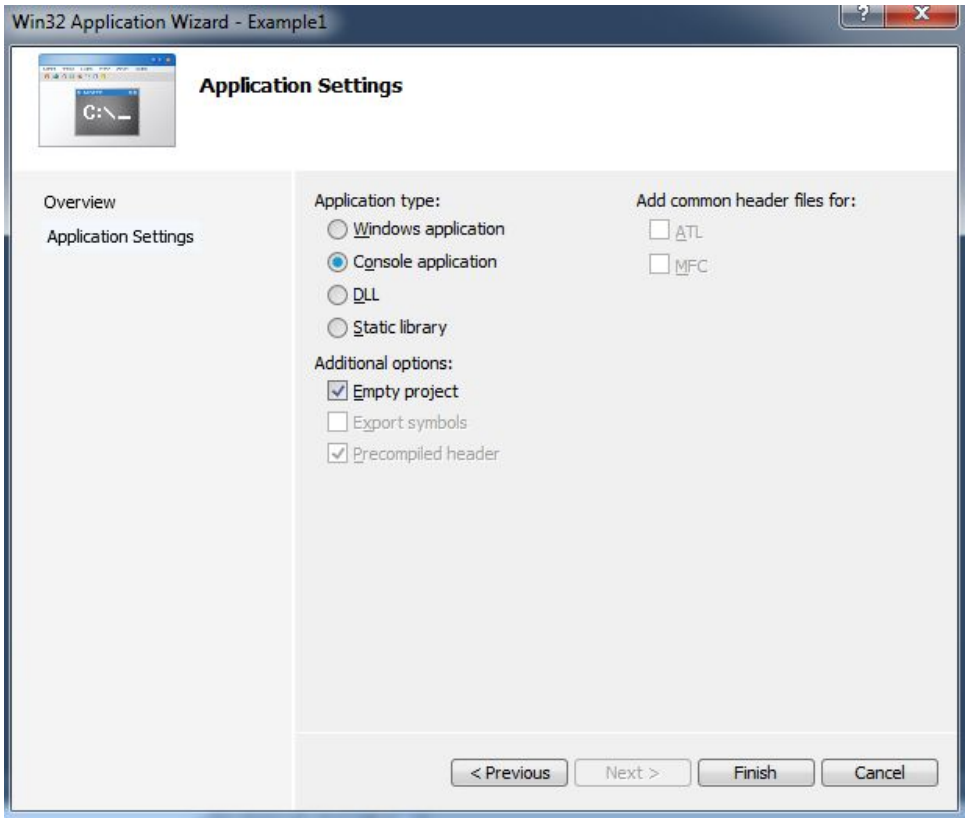
2c) Location: (default is fine)

2d) Solution: Create New Solution

2e) Solution Name: COMP175



3) Click Next on the following screen, and then Click on "Empty project"

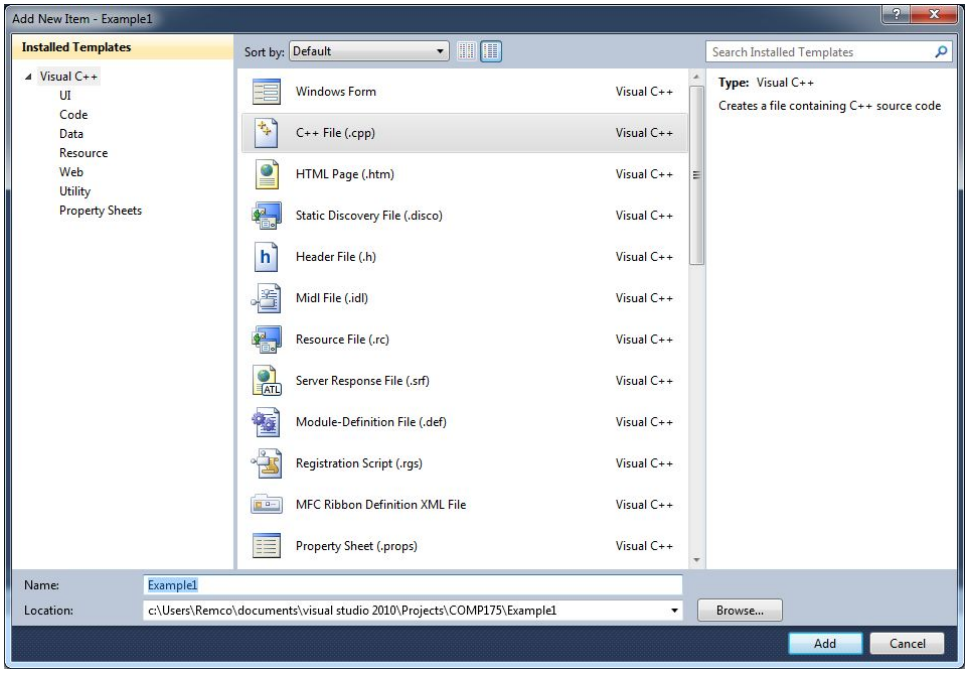


4) Right click on "Source Files", select "Add" and select "New Item..."

5) At the pop up window:

5a) select C++ File (.cpp)

5b) Name: Example1



6) Copy in a sample OpenGL program

7) Right click on the Project "Example1", select "Properties"

8) At the pop up window, first make sure that the Configuration says "Debug", and the Platform says "Win32"

9) Click "Configure Properties" -> "Linker" -> Input

10) At the line "Additional Dependencies", add the following to the start of the list:

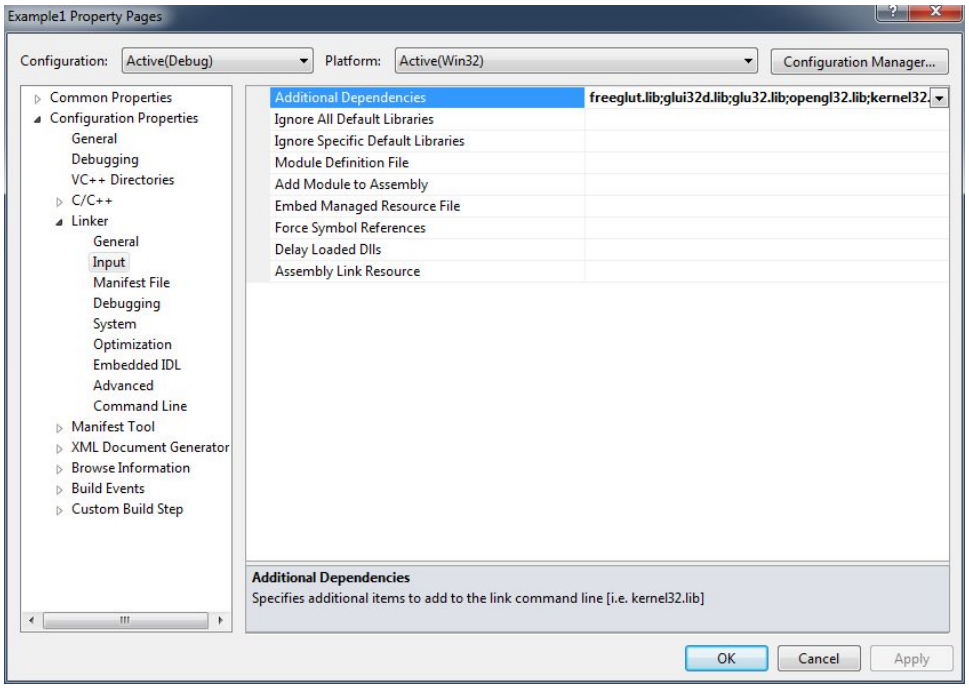
freeglut.lib;glui32d.lib;glu32.lib;opengl32.lib;

11) Click “Configuration Properties” → “C/C++” → Preprocessor

12) At the line “Preprocessor Definitions”, add the following to the start of the list:

_CRT_SECURE_NO_WARNINGS;

NOTE: At the start of every project in this class, steps 7-12 will have to be repeated for each project.



11) Note that if you are compiling for the "Release" configuration, you will use glui32.lib (instead of glui32d.lib)

12) Click OK, and then F5 to run the program (you will be prompted to compile the program)

Graphics and Mathematics

Vector Math

Scalar
Point
Vector

Vector Magnitude:

The vectors magnitude is the length of the vector (or the norm). The two bars around the 'A' represent 'length of A'
$|A| = sqrt(A_x^2 + A_y^2 + A_z^2)$

Vector Normalization:

We often want to normalize a vector, which means we are reducing its size to a length of 1.
$n = N / |N|$

Vector Addition:

Vector-Scalar Multiplication:

Dot-Product (also called inner-product)

Calculates the angle between two vectors.

Cross Product

Returns a normal vector from two vectors.
Value returned tells us 'how perpendicular(or orthogonal)' two vectors are.

Basis vector

Right-hand rule

Matrix Math

Identity Matrix:

Zero Matrix:

Addition and Subtraction:

Scalar Matrix Multiplication:

Matrix-Matrix Multiplication: Order matters, matrix multiplication is not commutative.

Transformations

Translation:

$Matrix_T$ = |1 0 0 dx |
|0 1 0 dy |
|0 0 1 dz |
|0 0 0 1 |

Rotation:

$Matrix_{rx}$ = |1 0 0 0 |
|0 cos -sin 0 |
|0 sin cos 0 |
|0 0 0 1 |

$Matrix_{ry}$ = |cos 0 sin 0 |
|0 1 0 0 |
|-sin 0 cos 0 |
|0 0 0 1 |

$Matrix_{rz}$ = |cos -sin 0 0 |
|sin cos 0 0 |
|0 0 1 0 |
|0 0 0 1 |

Scaling:

$Matrix_S$ = |sx 0 0 0 |
|0 sy 0 0 |
|0 0 sz 0 |
|0 0 0 1 |

Local versus Global transformations:

Projections

Parallel Projection(Orthogonal): All objects appear the same size on the projection plane(the projection plane being the 2D screen, i.e. our camera lens looking into the world). This view is often used in CAD applications for designers

Read further: glOrtho

Perspective Projection: Determine objects size based on distance from the projection plane. This is how humans view the world.

Perspective transformation Matrix:

$Matrix_P$ = |1 0 0 0 |

|0 1 0 0 |

|0 0 1 1/focus |

|0 0 -1 0 |

Clipping

Viewing Volume: Do not render objects that appear within this 3D space, and thus do not need to go through the process of being rendered or undergo perspective transformations.

Viewing Transformation

Specifies the location of the camera. This is the first transformation that is applied to your scene.

Eye Coordinates

Cartesian coordinate system applied to camera. By default OpenGL looks down negative z-axis.

Modeling Transformation

Moves objects around the scene. During this set of transformations we can move, rotate, and scale our objects. Remember though, that this set of transformations is a series of matrix multiplications, so order does matter!

Projection Transformation

Defines viewing volume and Clipping Plane. This series of transformations determines which objects appear in or final scene.

Read further: glFrustum

Field of View (fov)

Modelview transformation

Combination of viewing and modeling transformation.

The ModelView Matrix

The modelview matrix is a 4x4 matrix that represents where objects and vertices will be placed after transformations.

We use glMatrixMode(GL_MODELVIEW) when we want to modify the object with any transformations(rotate, scale, translate).

glMatrixMode(GL_PROJECTION) is used to specify the projection matrix.

glMatrixMode(GL_TEXTURE) is used to modify the textures we wrap around our objects.

Whichever matrix mode you are in, remember that these are each stacks(modelview, projection, and texture). As you perform translations, rotations, and scaling, then you create more sophisticated transformations with glPushmatrix() and glPopMatrix().

Viewport Transformation

Maps 2D projection of scene into rendering window.

Read further: glViewport

Lighting

Ambient: Light rays have no direction, thus objects illuminated equally on all sides with equal intensity.

Diffuse: Light has a particular direction and reflected evenly across a surface.

Specular: Light coming from a direction that is heavily reflected.

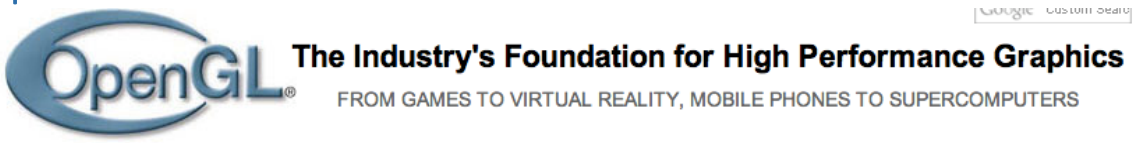
Texture Mapping

Textures: These are patterns that can be 'mapped' or wrapped around 3D objects to give 3D objects more distinct surfaces. Think of a brown box that you wrap with wrapping paper to make it look more exciting.

Texture Coordinate: A (s, t) coordinate on a 2D image

Parametric Cooridantes: (u,v) coordinates for a texture which are used to map a 2D image to a 3D surface (uv mapping).

OpenGL Basics



[IMAGE SOURCE: HTTP://WWW.OPENGL.ORG/]

What is it?

OpenGL is a free API that allows programmers to interface with the graphics hardware on your machines. It is itself a very low level framework giving you lots of flexibility to create visuals on all of the major computing platforms. Many video game consoles use OpenGL for rendering their graphics (the alternative is Microsoft's Direct3D, which is the graphics part of the DirectX API).

History

OpenGL was originally developed by Silicon Graphics.

The Fixed Rendering Pipeline

What is GLUT? (and Freeglut)

GLUT is the OpenGL Utility Toolkit, which is a free library that helps us with basic functionality for creating a window, menus, and some forms of input. Freeglut is a more modern and open-source version of the GLUT library that we will be using for all of our examples.

OpenGL Architecture

OpenGL is a state machine.

What is Computer Graphics?

Think of it like a real world camera, and how we capture a 3D environment and plot it on a 2D screen.

Think about perspective? How do painters draw a picture of the scene to make it look like the real world.

Ways to do 3D Graphics

Geometric transform – Map one coordinate system to another. (An x,y,z to a u,v, w for example)
Per pixel level – Ray tracing (Shoot a ray from the canvas to the geometry and collect light. Remember integrals? We are summing the total amount of color that appears, think about how light is illuminated from different sources).

How to think about color?

1.) What do we need to see color—light!
2.) We also need the color of the object (because remember the color of an object is the light that is reflected back).

How to think about light? (and perspective)

(Insert picture of view volume)

|------| object
|------|

_____ screen (2d canvas)

(o) eye

What if we have multiple light sources?

What does your graphics card give you?

- Dedicated memory and processor to do math computations
- Can do parallel computations (multiple cores)
    - What computations can we parallelize?
    - Which must be serial?
-

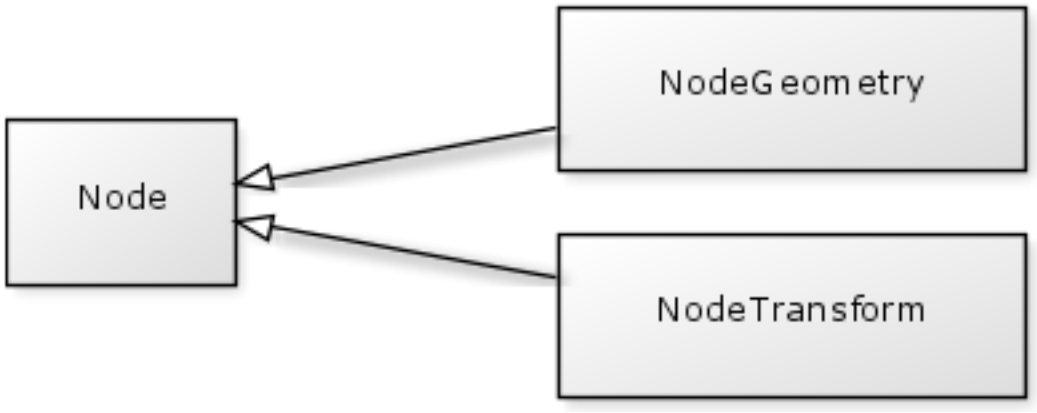
C++ (Refresher) Topics to be familiar with

Polymorphism

Inheritence

Event-Based Programming

Scene Graph

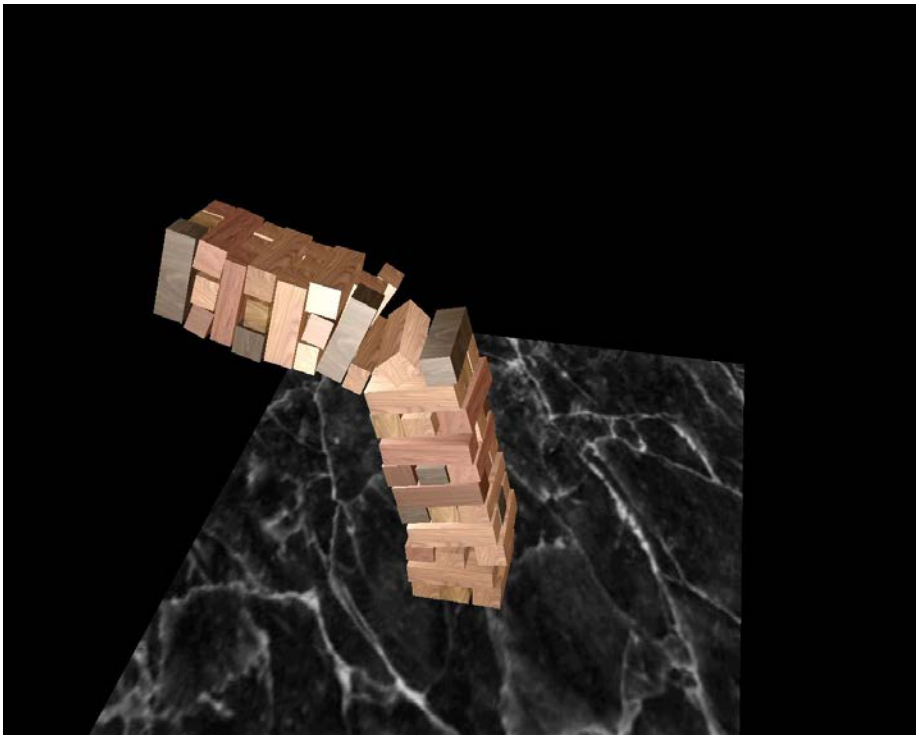


[IMAGE SOURCE: HTTP://YUML.ME/DIAGRAM/CLASS/%5BNODE%5D%5E%5BNODETRANSFORM%5D,%20%5BNODE%5D%5E%5BNODEGEOMETRY%5D]

Animation

[IMAGE SOURCE:]

Physics



[IMAGE SOURCE: HTTP://WWW.CENTRALQUESTION.COM/TOPALOV/TOPALOV2.JPG]

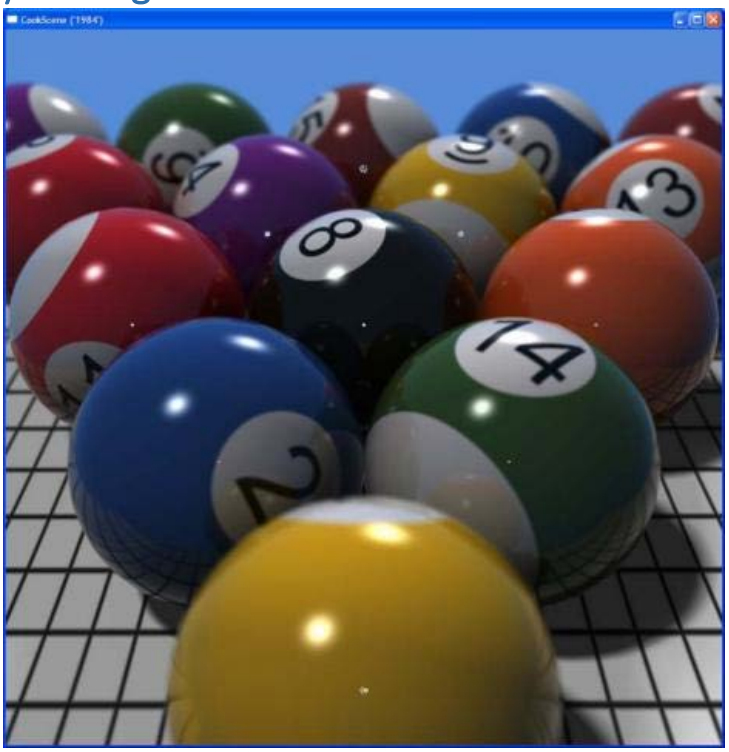
Some cool demoes

1.) Music, scripting, and Physics - http://www.youtube.com/watch?v=Xu-A0jqMPd8
   a. This demo is probably ten years old, but still quite impressive.
   b. As an exercise, think about how could it have been implemented? Different materials have different sounds associated with them. The objects themselves have forces and directions, and are controlled as a particle system for each instrument.

Ray Casting

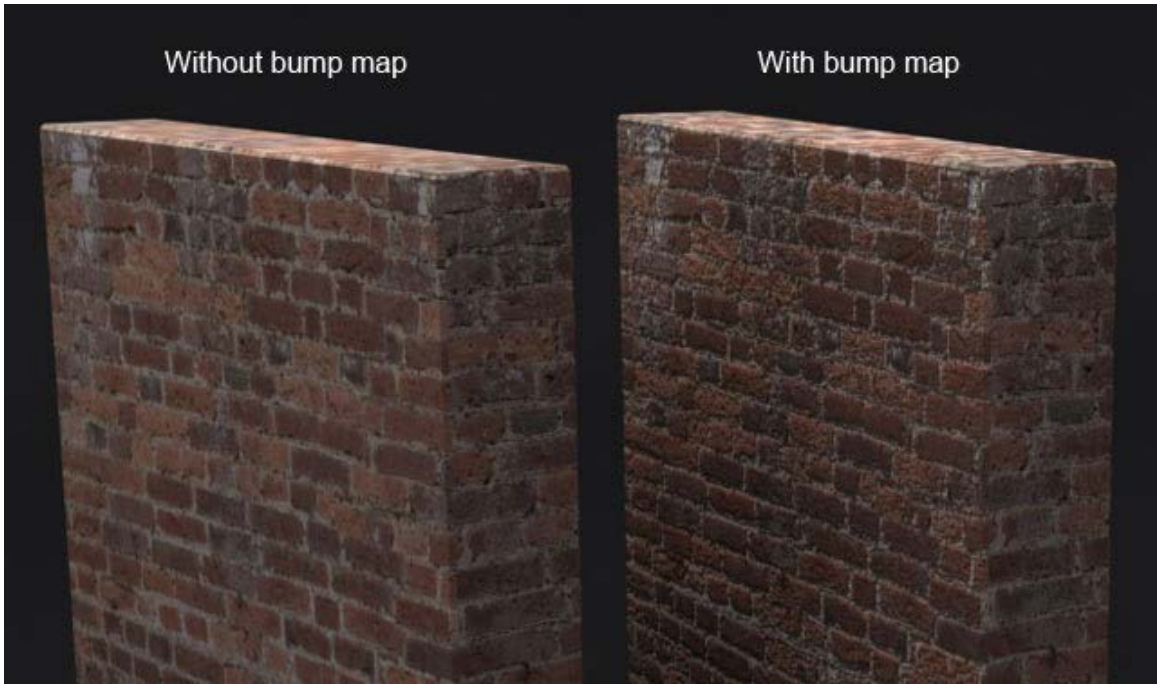
Illumination and Intersection Normals

Recursive Ray Tracing



[IMAGE SOURCE: HTTP://WWW.PCGAMESHARDWARE.COM/SCREENSHOTS/MEDIUM/2009/05/NVIDIA-CUDA-RAYTRACING-02.JPG]

Shaders



[IMAGE SOURCE: HTTP://3DMODELING4BUSINESS.COM/WP-CONTENT/UPLOADS/2012/10/BUMP-MAP-3.JPG]

Final Projects

There are certain projects I feel that I can help supervise reasonably well. Below I am listing some ideas, and I will also link some sample code to a basic version of the problem for you to build on. As the semester progresses I will flesh out more details, but for now, you can keep these ideas in mind.

Extend any of the in-class assignments

Add interesting shaders to the planets in the solar system demo

Maker Bot Simulation

Simulate the making of a 3D object. You will need to learn about volume rendering for this.

Explore WebGL

Build some application on the web

Deploy your project to the iPhone/Android

Do something interesting with shaders on the iPhone and Android

Appendix

Sharpening up on C++

C++ Reference: http://www.cplusplus.com/

Advanced C++ Tips: http://www.gotw.ca/gotw/

My notes: http://www.eecs.tufts.edu/~mshah08/comp15.html

Getting a Job

Getting a job as a graphics programmer: http://www.p1xelcoder.com/2013/10/29/what-i-expect-from-a-graphics-programmer-candidate/

Getting a job as a games programmer: http://www.altdevblogaday.com/2013/11/08/how-to-become-a-graphics-programmer-in-the-games-industry/

Useful (but unsorted) links I'm collecting.

http://www.opengl.org/discussion_boards/showthread.php/133880-Diference-between-GL_MODELVIEW-and-GL_PROJECTION

http://stackoverflow.com/questions/628796/what-does-glloadidentity-do-in-opengl

http://stackoverflow.com/questions/5367361/opengl-what-is-matrixmode

http://stackoverflow.com/questions/21558/in-c-what-is-a-virtual-base-class

http://en.wikipedia.org/wiki/Callback_(computer_programming)

http://en.wikipedia.org/wiki/Delete_(C%2B%2B)

http://www.opengl.org/archives/resources/faq/technical/viewing.htm

http://www.songho.ca/opengl/gl_projectionmatrix.html

http://sjbaker.org/steve/omniv/projection_abuse.html