CSCI 4239/5239

Advanced Computer Graphics

Spring 2014

Instructor

- Willem A (Vlakkies) Schreüder
- Email: willem@prinmath.com
  - Begin subject with 4239 or 5239
  - Resend email not answered promptly
- Office Hours:
  - CSEL Thursday 4-5pm
  - Other times by appointment
- Weekday Contact Hours: 6:30am - 9:00pm

Course Objectives

- Explore advanced topics in Computer Graphics
  - Pipeline Programming (Shaders)
  - Embedded System (OpenGL ES)
  - GPU Programming (CUDA&OpenCL)
  - Ray Tracing
  - Special topics
    - Particle systems
- Assignments: Practical OpenGL
  - Building useful applications

Course Organization and Grading

- Class participation (50% grade)
  - First hour: Discussion/Show and tell
    - Weekly homework assignments
    - Volunteers and/or round robin
  - Second hour: Introduction of next topic
- Semester project (50% grade)
  - Build a significant application in OpenGL
  - 10 minute presentation last class periods
- No formal tests or final

Assumptions

- You need to be fluent in C/C++
  - Examples are in C or simple C++
  - You can do assignments in any language
    - I may need help getting it to work on my system
- You need to be comfortable with OpenGL
  - CSCI 4229/5229 or equivalent
  - You need a working OpenGL environment

Grading

- Satisfactory complete all assignments => A
  - The goal is to impress your friends
- Assignments **must** be submitted on time unless prior arrangements are made
  - Due by Thursday morning
  - Grace period until Thursday noon
- Assignments must be completed individually
  - Stealing ideas are encouraged
  - Code reuse with attribution is permitted
- Grade <100 means not satisfactory (not A)

Class Attendance

- Attendance is highly encouraged
- More of a seminar than a lecture
  - Participation is important
- I don't take attendance
- Lectures are available if you miss class
  - If you are sick stay home
- Lecture video access
  - http://cuengineeringonline.colorado.edu/
  - Log in with Identikey credentials
  - Email me if you don't see this class

Code Reuse

- Code from the internet or class may be used
  - You take responsibility for any bugs in the code
    - That includes bugs in my code
  - Make the code your own
    - Understand it
    - Format it consistently
  - **Improve upon what you found**
    - **I may ask what improvements you made**
  - **Submitting code without crediting the source is violation of the CU honor code**
- The assignment is a minimum requirement

Code Expectations

- I expect professional standards in coding
  - Informative comments
  - Consistent formatting
    - Expand tabs
  - Clean code
- Good code organization
- Appropriate to the problem at hand

Text

- OpenGL Programming Guide (8ed)
  - Schreinder, Sellers, Kessenich & Licea-Kane
  - “OpenGL Red/Orange Book”
  - Implementing Shaders using GLSL
  - Don't get an older edition
- Ray Tracing from the Ground Up
  - Kevin Suffern
  - Theory and practice of ray tracing
- Recommended by not required

Other Texts

- OpenGL SuperBible: Comprehensive Tutorial and Reference (6ed)
  - Sellers, Wright & Haemel
  - Good all-round theory and applications
- Graphics Shaders: Theory and Practice (2ed)
  - Bailey & Cunningham
  - Great shader examples

Other Texts

- OpenGL ES 3.0 Programming Guide
  - Ginsburg & Purnomo
  - “OpenGL Purple Book”
  - Has a chapter specific to the iPhone
- iPhone 3D Programming
  - Rideout
  - Great introduction to portable programs
- WebGL Programming Guide
  - Matsuda & Lea

Other Texts

- Programming Massively Parallel Processors
  - Kirk & Hwu
  - Explains GPU programming using CUDA
  - Shows how to adopt OpenCL
- CUDA by Example
  - Sanders and Kandrot
  - Great introduction using examples

Other Texts

- Advanced Graphics Programming Using OpenGL
  - Tom McReynolds and David Blythe
  - Great reference for miscellaneous advanced topics

OpenGL Resources

- www.google.com
  - Need I say more?
- www.opengl.org
  - Code and tutorials
- nehe.gamedev.net
  www.lighthouse3d.com
  - Excellent tutorials
- www.mesa3d.org
  - Code of “internals”
- www.prinmath.com/csci5229
  - Example programs from CSCI 4229/5229

Assignment 0

- Due: **Friday** Jan 17 by 9pm
- Sign up with moodle.cs.colorado.edu
  - Enrollment key: 42395239
  - A picture will help me remember your names
- Submit
  - Your study area
  - Platform (Hardware, Graphics, OS, ...)
  - Any specific interests in computer graphics
  - Specific topics you want to see covered
  - Initial project idea(s)
  - CAETE students propose a schedule for work

My information

- Mathematical modeling and data analysis
  - PhD Computational Fluid Dynamics [1986]
  - PhD Parallel Systems (*CU Boulder*) [2005]
  - President of *Principia Mathematica*
- Use graphics for scientific visualization
- Open source bigot
- Program in C, C++, Fortran and Perl
- Outside interests
  - Aviation
  - Amateur radio (voice and digital)

How to get started

- Make sure you have a working OpenGL environment
  - Compile and run examples from CSCI [45]229
    - Ex 5 (hello world) for basics
    - Ex 35 (shaders) for advanced
- Make sure advanced examples work
  - Windows may need GLEW
  - Linux/nVidia may need driver and library from nVidia
  - OS/X may need Xcode

OpenGL Extension Wrangler (GLEW)

- Maps OpenGL extensions at run time
  - Provides headers for latest OpenGL
  - Finds vendor support at run time
- Check support for specific functions or OpenGL version at run time
  - Crashes if unsupported features are used
- Use only if you have to (Windows mostly)
  - Set -dUSEGLEW to selectively invoke it
  - Do NOT require GLEW (I don't need it)

Assignment 1

- Due: Thursday January 23
- Alternate OpenGL wrappers
  - GLUT is an easy to use, cross platform wrapper for running OpenGL
    - Created for the Red Book
    - Used by most OpenGL texts
  - Investigate alternative wrappers and compare it to GLUT, for example
    - Qt (Cross platform C++ application framework)
    - Simple DirectMedia Layer (SDL)
    - iPhone or Android
  - Write a simple visualizer of a 3D scene

Assignment 2

- Due: Thursday January 30
- NDC to RGB shader
  - For every point on the objects, the color should be determined by its position in normalized device coordinates
- The goal is to make this as short and elegant as possible
  - Shader Golf
  - Figure this out for yourself

Project

- Should be a program with a significant graphics component
  - Something useful in your research/work
  - Graphical front end to simulation
  - Graphical portion of a game
  - Expect more from graduate students
- Deadlines
  - Proposal: Thursday March 13
  - Progress: Thursday April 3
  - Review: Thursday April 17
  - Final: **Monday** April 28

Nuts and Bolts

- Complete assignments on any platform
  - Assignments reviewed under Ubuntu 12.04 LTS
  - Set #ifdef so I can compile and run it
- Submit using moodle.cs.colorado.edu
  - ZIP or TAR (no RAR)
  - Name executables hw1, hw2, ...
  - Create a makefile so I can do *make clean;make*
  - Set window title to *Assignment X: Your Name*
- Include number of hours spent on assignment
- ***Check my feedback and resubmit if no grade or less than 100%***

A few hints

- My machine runs Linux x86_64
  - gcc/g++ with nVidia & GLX
    - -Wall is a **really** good idea
  - case sensitive file names
  - int=32bit, long=64bit
  - little-endian
  - fairly good performance
- How to make my life easier
  - Try it in CSEL or a Linux box
  - Stick to C/C++ unless you have a good reason
- **Maintain thy backups...**

Class Discussions

- If have a special interest in the topic and have something special to contribute VOLUNTEER to lead the discussion
- If by Sunday there are no volunteers, I will appoint volunteers some on a round robin basis (in order by MD5 of names)
  - You can trade places, but **you** are responsible for arranging a substitute
- Everybody should do this at least once, but you can do more if you want
  - CAETE students Skype or screencast
- Popular topics may have more presenters

What to Present

- Should be (mostly) the assigned topic
  - Feel free to push the envelope
  - Keep it within reach of the class
- Show what you did for the assignment
  - Cover principles or theory I omitted
  - Show and describe code of interest
  - Demonstrate “gotchas” you encountered
  - Impress your friends
- Keep it interesting

How to Present

- 15 minutes can be forever or over in a wink
  - Plan your time (practice a bit)
  - If you use slides figure 2 minutes per slide
- Plan your presentation
  - What are the key points you want to convey?
  - How do you illustrate the key points?
- The presentation should TEACH
  - Teaching is learning twice
  - Adapt to the questions

How to Listen

- If you don't understand, ask
  - Helps the presenter understand what's new to you
- If you disagree, say so
  - Maybe the presenter misspoke or has an different opinion worth discussing
- Be nice – you may be next!

CAETE Students

- Suggest ways you can present remotely
- Provide screen cast or similar demonstration
- Skype or other desktop sharing
  - Performance may be an issue
- Stick to the class schedule if possible

Parallel Flight Simulator Project

- Consider joining a project with many members
  - Each member has a specific subtask
    - World visualization
    - Special effects
    - Flight dynamics
    - Multi-function displays (instruments)
    - Networking
    - Flight controls
    - Sound
  - Rotating project manager
    - Responsible for managing the project for a week
    - Provide concise report of what was done the last week
    - Lay out a plan for what should be done the next week
- Somewhat like a real software project
  - I will be the client

Review of OpenGL

- OpenGL is an API or library
  - GLU for some utilities and convenience functions
  - GLUT or other library for OS interface
  - Just gets a picture inside an OS supplied window
- OpenGL uses a state machine
  - Function calls set attributes that apply to subsequent objects
  - Objects are built from primitive elements
    - Vertexes define points in 3D space
    - Build lines and polygons from vertexes
    - Build 3D objects from polygons (skin only)

Vertex Attributes

- Position (x,y,z,w)
  - glVertex
- Color (R,G,B,α)
  - glColor
- Normal (x_n,y_n,z_n,w_n)
  - glNormal
- Texture Coordinates (s,t,r,q)
  - glTexture
  - glMultiTex
- User defined attributes

Points

- One at each vertex
  - glBegin(GL_POINTS)
- Point size
  - glPointSize()

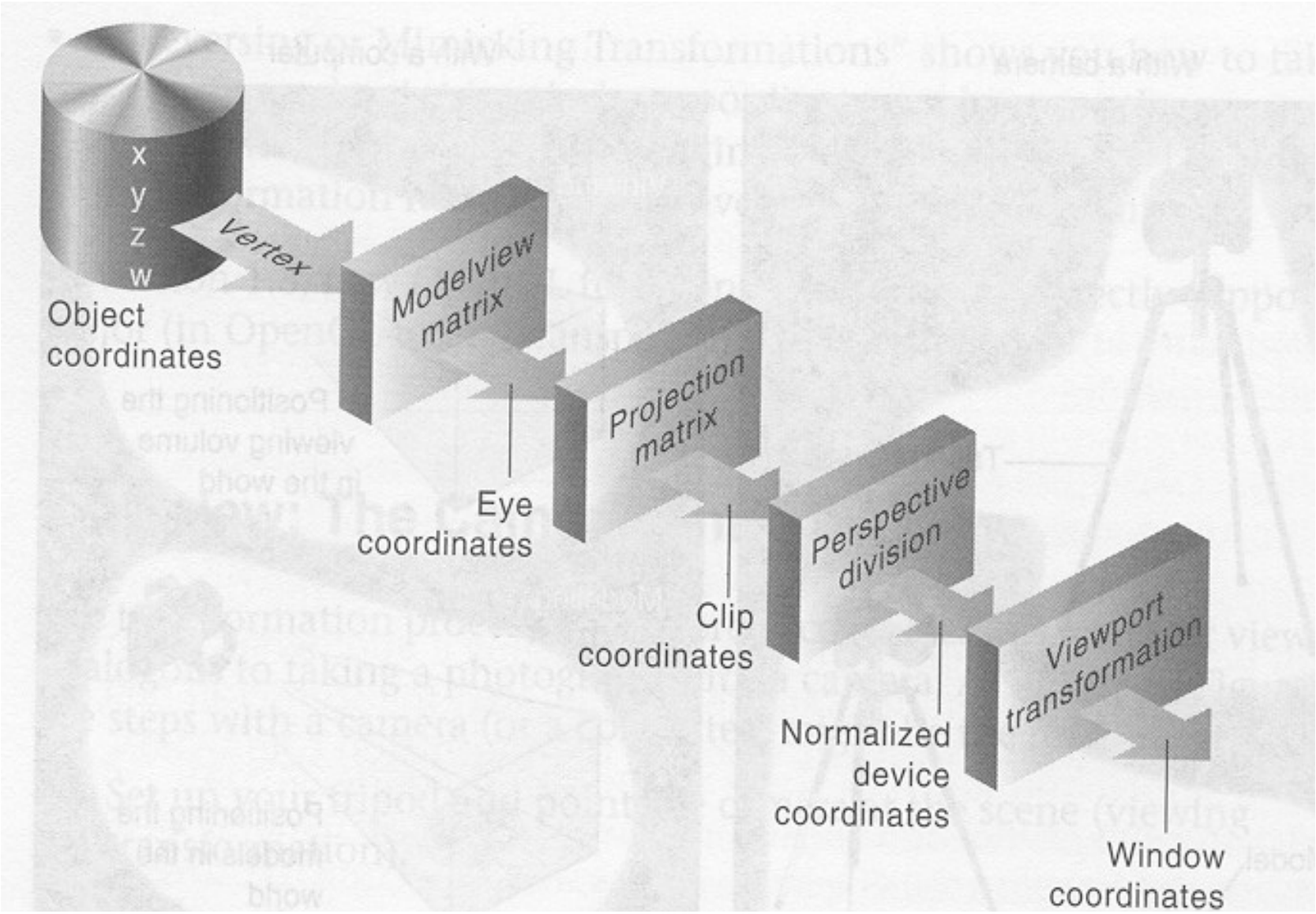
Lines

- Built from vertexes
  - glBegin(GL_LINES)
  - glBegin(GL_LINE_STRIP)
  - glBegin(GL_LINE_LOOP)
- Line pattern
  - glLineStipple()
- Line width
  - glLineWidth()

Polygons

- Build from vertexes
  - glBegin(GL_POLYGON)
  - glBegin(GL_TRIANGLES)
  - glBegin(GL_QUADS)
  - many more
- Must be planar
- Fill pattern
  - glPolygonStipple()
  - glShadeModel()

OpenGL Transformation Pipeline



Transformations

- glMatrixMode()
- Primitive transforms
  - glTranslate
  - glScaled
  - glRotated
- Utilities
  - gluPerspective()
  - gluLookat()
- Direct manipulation
  - glMultMatrix
- Deprecated in OpenGL4 Core Profile

Modes

- glEnable(), glDisable()
  - GL_DEPTH_TEST
  - GL_CULL_FACE
  - GL_LIGHTING
  - GL_TEXTURE_2D
  - GL_SHADOWS

Blinn-Phong Lighting

- Enable GL_LIGHTING & GL_LIGHTx
- glLightXXX
  - Ambient, Diffuse, Specular
  - Position, Direction, Spot Angle
  - Local/Global, One/Two sided
- glMaterialXXX
  - Emission, Ambient, Diffuse, Specular, Shininess
- glNormal

Textures

- glEnable(GL_TEXTURE_xD)
- glGenTexture()
- glBindTexture()
- glTexImage2D()
  - you need to load the image yourself
- glTexParameter()
- glTexEnv()

Blending and Transparency

- glEnable(GL_BLEND)
- glBendFunc(source,destination)
  - GL_ZERO
  - GL_ONE
  - GL_DST_COLOR
  - GL_ONE_MINUS_DST_COLOR
  - GL_SRC_ALPHA
  - GL_ONE_MINUS_SRC_COLOR
  - GL_DST_ALPHA
  - GL_ONE_MINUS_DST_ALPHA
  - GL_SRC_ALPHA_SATURATE
- Anti-aliasing

Other Features

- Display Lists
- Polygon Offset
- Fog
- Bezier curves and surfaces
- Masking
  - Color, Z-buffer, etc can be made read-only
  - Stencil buffer
- Shadows (Shadow Volume & Shadow Map)
- Many more..

OpenGL 4

- Core Profile
  - High performance rendering
  - Deprecates many OpenGL functions
    - Transformations
    - Display lists
  - Steep learning curve
- Compatibility profile
  - Provides deprecated functionality
  - Flatter learning curve
- I will mostly use OpenGL 2.3

Interfacing with the OS

- Displaying the image
  - Request window features
  - Redraw window
  - Window resized
- User interaction
  - Keyboard input
  - Mouse input
- Other OS functions
  - Elapsed time
  - Sound

GLUT (OpenGL Utility Toolkit)

- Used by many textbooks
  - Implementations for Unix, Windows, MacOS, ...
- Hides OS specific complexity
  - Create window and request properties
  - Callback functions to process events
    - Display
    - Keyboard
    - Mouse
  - glutGet to access time, dimensions, ...
- Limited support for menus

Alternatives to GLUT

- Simple Directmedia Layer
  - Similar to GLUT
  - Add sound, menus, etc..
- Trolltech/Nokia/Digia Qt
  - Cross Platform C++ Application Framework
  - Comprehensive sets of widgets
  - OpenGL window is a widget
- iPhone SDK
- Android SDK or NDK
- Many others