Why this class?
Who's in charge here?

• Instructor: Christian Miller
  • PhD student in graphics lab, under Don Fussell
  • Office hours: W 3-5 PM

• TA: Randall Smith
  • Also a PhD student in the graphics lab
  • Office hours: ????
When / where?

• Hopefully you’ve figured this one out.
• T/Th 9:30 - 11:00 AM :
• Burdine Hall (BUR) 220
Where’s the website?


- Contact info, syllabus, lecture notes, projects, and so forth will be posted there
What will you learn?

• Fundamentals:
  • Physics of light and the human visual system
  • Vector algebra and coordinate systems
  • Transformations: affine and perspective
  • Shading: color, lighting, texturing
  • Rendering: rasterization and ray tracing
What will you learn?

• Topics:
  • How to program in OpenGL
  • Cool graphics techniques and effects
  • What GPUs actually are, and how to use them
  • Animation and simulation
  • Whatever else I can cram in the schedule
What is this class not?

• An OpenGL tutorial
  • Aside from covering the basics, this part is mostly self-taught

• A Maya / 3DS Max / Softimage tutorial
  • Content creation is hard, go get an art degree
How will grades work?

- Projects (60%)
  - Four of ‘em, each worth 15%
- Exams (40%)
  - Two midterms, each 10%
  - One final, worth 20%
What about the book?

• Interactive Computer Graphics: A Top-Down Approach with OpenGL (6e)
  - Decent background, math reference, and OpenGL reference
  - They made a terrible change in this edition of the book...
Some Background

- The OpenGL API was originally released in 1991
- A review board (ARB) was formed to keep it up to date over the years
- Graphics and graphics hardware have changed a lot since then
- Stuff that used to be fine now doesn’t map cleanly to hardware, and leads to inefficiencies (BAD)
OpenGL housekeeping

- By the late 2000’s, programmable hardware had stabilized, and clear design trends had emerged
- The ARB decided to recenter the OpenGL API around current hardware and how it is used
- They developed a shader-driven version of the spec, cleaned out a bunch of old crud (GOOD)
- Released in 2008 as OpenGL 3.0, however...
Trouble brews...

- The ARB got a little carried away, and deprecated / removed a whole ton of useful stuff
  - Like all immediate mode rendering
  - And any rendering without shaders
  - And all image processing functions
  - And named vertex properties, etc, etc...
The OpenGL schism

- Everybody agrees that the new shader stuff is good
- Most everybody hates that so much was removed
- Nvidia and ATI (at the time) implemented drivers in a few months, but refused to deprecate things
- Others (Apple) refuse to fully update their shader implementations, even three years later
- (All of these companies are on the ARB...)

What this means to you

• The new OpenGL stuff is harder to use and requires significantly more setup to get running

• It’s hard to understand the API if you don’t already know graphics

• We’re on OpenGL 4.2 now, and support for 3.0 still isn’t universal
And the book?

• This version of the textbook moved to an OpenGL standards compliant, shader-based approach

• This isn’t as good as it sounds, and just serves to make the material more confusing

• The included source code is no longer portable, too

• This is the only book to have done this, and we’re stuck with it
My solution

• I’ll teach you OpenGL as it’s always been done

• The book is still a good reference for the unchanged parts of OpenGL, as well as graphics in general
Other resources

- Two books really worth having:
  - Mathematics for 3D Game Programming and Computer Graphics (3e), by Eric Lengyel
  - Real-Time Rendering (3e), by Tomas Akenine-Moller, Eric Haines, and Naty Hoffman
- The standard OpenGL reference:
  - The OpenGL Programming Guide (Red Book)
A BRIEF HISTORY: 50’s

- It starts with vacuum tubes & output devices
  - Large computers needed outputs
  - Strip charts and pen plotters for output
  - Slow-phosphor CRTs
- Computers are too slow & unreliable to do realtime
Cathode Ray Tube (CRT)
Vector graphics
History: 60’s

• Computers got faster, refresh now possible
• Able to do interactive vector drawing on CRTs
• Wireframes become (relatively) common
• One man put this to particularly good use...
Sketchpad

- Ivan Sutherland’s PhD thesis at MIT (1963)
- Vector CRT system, with a light pen
- User could draw and interact on the screen
- Along the way, he invented many of the fundamental concepts of CG
  - And all other computer interaction that we now take for granted...
• http://www.youtube.com/watch?v=t3ZsiBMnGSg
Other stuff from the 60’s

- Display Processor Unit (DPU): Special purpose computer just for drawing
  - Host computer sends commands to DPU
  - DPU handles drawing, host does other things
- The first GPUs
**History: 70’s**

- Enough people use graphics that standards emerge
- Mainframes dominate graphics, but not everything is custom-built anymore
- Ray tracing invented
- Video games show up, explode in popularity
Raster graphics

- Image stored as an array of pixels
- Memory intensive, finite resolution, arbitrary data
The University of Utah

- In the 70’s their graphics lab was unrivaled
- Visibility determination, splines, subdivision
- Flat, Gouraud, Phong shading
- First digitization of a real object: a VW beetle
- Notable graduates: Ed Catmull, Jim Blinn, Alan Kay, John Warnock, Jim Clark
History: 80’s

• PCs come out, all hell breaks loose
• Silicon Graphics releases high-power workstations
• Exponential improvements in realism
• Texturing becomes common
• Early CG shorts, TRON, The Abyss
• Video games implode, explode again
History: 90’s

- Game consoles in every living room
- Commercially successful CG movies, starting with Toy Story
- OpenGL, DirectX released
- Graphics cards start becoming standard in PCs
- 3D becomes a household term
- SIGGRAPH becomes largest ACM conference
History: 2000’s

- Computer graphics ubiquitous
- Near-perfect realism, even in realtime
- GPUs go from fixed functions to almost total programmability, overflow into other fields
- Over 40,000 people attend SIGGRAPH
- Industry reaches tens of billions of dollars per year