Luxo Jr.

- Pixar Animation Studios, 1986
- Director: John Lasseter
Plan for Today

- What this class is (and is not) about
- Administrivia
- Overview of the semester
- Overview of assignments
- Quick-and-dirty intro to OpenGL & Assignment 0
Why are YOU here?
What are the applications of graphics?
Movies / Visual Effects (VFX)
Video Games
Alan Wake Trailer, E3 2009
Simulation
CAD-CAM & Design, Advertising
Architecture
Global Illumination in Architecture

Henrik Wann Jensen
Global Illumination using Photon Mapping
Virtual Reality
Visualization
Medical Imaging
Differences between applications

- Games vs. movies?
- Architectural vs. medical visualization?
- Etc. etc. etc.
Differences between applications

- Games vs. movies?
  - Games need real-time performance (30 frames per second, FPS)
  - Movie frames usually take hours to render

- Architectural vs. medical visualization?
  - Building visualization needs accurate simulation of illumination (want to know what it’ll look like)
  - Medical visualization aims to highlight important features in real-world datasets

- Etc. etc. etc.
What you will learn in 6.837

• Fundamentals of computer graphics algorithms
  – Will give a pretty good idea of how to implement lots of the things just shown
• We’ll concentrate on 3D, not 2D illustration or image processing
• Basics of real-time rendering and graphics hardware
• Basic OpenGL
  – Not the focus, though: Means, not the end.
• You will get C++ programming experience
  – Most things are written in it IRL
What you will NOT learn in 6.837

• OpenGL and DirectX hacks
  – Most become obsolete every 18 months anyway!
  – Doesn’t really matter either: Graphics is becoming all software again (OpenCL, Larrabee, etc.)

• Software packages
  – CAD-CAM, 3D Studio MAX, Maya
  – Photoshop and other painting tools

• Artistic skills

• Game design
How much Math?

• Lots of simple linear algebra
  – Get it right, it will help you a lot!

• Some more advanced concepts
  – Homogeneous coordinates
  – Quaternions for interpolating rotations/orientations
  – Ordinary differential equations (ODEs) and their numerical solution
  – Sampling, antialiasing (some gentle Fourier analysis)
  – Monte-Carlo integration

• Always in a concrete and visual context

• Deeper mathematic exposition in 6.839
Questions?
Team

• Instructor
  – Jaakko Lehtinen

• TAs
  – Jordan Sorensen
  – Greg Pintilie

• Course secretary
  – Bryt Bradley
Administrivia: Prerequisites

- Not strictly enforced
- All assignments are in C++
  - *Optional review/introductory session*
    Monday Sept 14, 7:30-9pm in 32-155
- Calculus, Linear Algebra
  - Solving equations, derivatives, integrals
  - Vectors, matrices, basis, solving systems of equations
  - *Optional review/introductory session*
    Monday Sept 21, 7:30-9pm in 32-155
Administrivia: Website, Staff Email

• Course website at Stellar
  – Announcements
  – Slides (posted soon after each lecture)
  – Message board
  – Assignments, both instructions and turn-in
  – You need a MIT certificate!

• 6837-staff@csail.mit.edu
  – Reaches all of us, preferred method of communication
Administrivia: Grading Policy

• Assignments: 75%
  – Two-week programming assignments
  – Must be completed individually
  – *No final project*

• Quiz: 10%
  – Tuesday, Oct 20 (in class)

• Final Exam: 10%
  – TBA during finals week

• Participation: 5%
Administrivia: Assignments

• Turn in code and executable (Athena Linux)
• Always turn in a README file
  – Describe problems, explain partially-working code
    Say how long the assignment took
• Coding style important
  – Some assignments are cumulative
• Late policy
  – The deadline is absolute: 0 if not on time
  – Due Wednesday @ 8pm
  – Extensions only considered if requested 1 week before due date
  – Medical problems must be documented, get note from dean
Administrivia: Assignments

• The assignments are a lot of work. Really.
  – Start early!

• Collaboration policy
  – You can chat, but code on your own
    (we use automated plagiarism detection software!)
  – Acknowledge your collaboration (in README)
  – Help others on Stellar message board
    (will help your grade!)

• Talk to each other, get a community going
  – Graphics is fun!
Administrivia: Lab & Office Hours

• Jaakko
  – Thursdays 4-5pm (after class) at 32-D412

• Jordan and Greg
  – at the Athena cluster on 5th floor of the Student Center
  – Mondays at 7.30-8.30pm starting Sep 28
  – Tue Sep 15 & Tue Sep 22 at 7.30-8.30pm
    • Review sessions on Mon 14 and Mon 21

• Send email to make
  an appointment for other times
Textbooks

• No textbook is required
• Recommendations
  – 3D Computer Graphics (Watt)
    • There is a free online version available from Books24x7 (http://libraries.mit.edu/get/books24x7, requires MIT Certificate)
  – Fundamentals of Computer Graphics, 3rd ed. (Shirley, Marschner)
Questions?
How do you make this picture?
How do you make this picture?

• Modeling
  – Geometry
  – Materials
  – Lights
How do you make this picture?

- Modeling
  - Geometry
  - Materials
  - Lights
- Animation
  - Make it move
How do you make this picture?

- **Modeling**
  - Geometry
  - Materials
  - Lights
- **Animation**
  - Make it move
- **Rendering**
  - I.e., draw the picture!
  - Lighting, shadows, textures...
How do you make this picture?

- **Modeling**
  - Geometry
  - Materials
  - Lights
- **Animation**
  - Make it move
- **Rendering**
  - I.e., draw the picture!
  - Lighting, shadows, textures...
Modeling/Viewing Pipeline

Meet the **Stanford Bunny**. He is one of the best-known characters in graphics.

See [http://www.cc.gatech.edu/~turk/bunny/bunny.html](http://www.cc.gatech.edu/~turk/bunny/bunny.html) for history.
1. Model the geometry
   - Here a triangle mesh
   - Also, specify materials
Modeling/Viewing Pipeline

1. Model the geometry
   - Here a triangle mesh
   - Also, specify materials
Modeling/Viewing Pipeline

1. Model the geometry
2. Place the objects in world space
   - Each object has its own object space
   - Only one world space
Modeling/Viewing Pipeline

1. Model the geometry
2. Place the objects in world space
3. Pick viewing position and direction
Modeling/Viewing Pipeline

1. Model the geometry
2. Place the objects in world space
3. Pick viewing position and direction
4. Transform objects to view space and project to image plane
   - Compute shading and draw picture!

Object coordinates
World coordinates
View coordinates
Image coordinates
Modeling/Viewing Summary

Object coordinates → World coordinates → View coordinates → Image coordinates

6.837 Fall 09 – Lehtinen

Emil Praun
Some algorithms go the same sequence in the opposite direction (e.g. ray tracing)
Animation:
Make these transformations vary with time

Object coordinates ➔ World coordinates ➔ View coordinates ➔ Image coordinates
Another View

Modeling, animation

Viewing/rendering

Object coordinates  World coordinates

View coordinates  Image coordinates
Questions?
Overview of the Semester

• Modeling, Transformations (All Assignments)
• Animation (Assignments 2, 3)
• Quiz (in class, Tue Oct 20)
• Ray Casting / Ray Tracing (Assignments 4, 5)
• Textures, Shadows (Assignment 5)
• Graphics Pipeline (Rasterization vs. Ray Tracing)
• Sampling, Global Illumination (Assignment 5)
Coordinate Transformations

- Mostly linear algebra
- Homogeneous coordinates
  - Neat way of treating affine and perspective transforms as linear
- Perspective (for viewing)
Modeling

• 2D curves, triangle meshes, smooth surfaces (Bézier, splines), subdivision surfaces
Assignment 1: Curves & Surfaces

Spline curves

Surfaces of revolution

Sweep surfaces
Animation: Keyframing

ACM © 1987 “Principles of traditional animation applied to 3D computer animation”

FIGURE 3. Squash & stretch in Luxo Jr.'s hop.
Character Animation: Skinning

- Animate simple “skeleton”
- Attach “skin” to skeleton
  - Skin deforms smoothly with skeleton
- Used everywhere (games, movies)
Assignment 2: Hierarchical Modeling

• Animate character skeleton as tree of transformations

• Skinning: smooth surface deformation
Particle systems (ODEs)
“Physics” (ODEs)

- Fire, smoke
- Cloth

- Quotes because we do “visual simulation”
Assignment 3: Physics

- Simulate cloth as a mass-spring network
  - ODE numerics
Color

- Saturation
- Hue

- Color diagram
- Relative response vs. wavelength

Diagram showing the relationship between saturation, hue, and color space, along with spectral sensitivity curves for different wavelengths.
Rendering: Ray Casting

• For every pixel
  construct a ray from the eye
    – For every object in the scene
      • Find intersection with the ray
      • Keep if closest

Visibility or “hidden surface” problem
Assignment 4: Ray Casting

- Cast rays from the viewpoint
- Intersect with scene primitives
- Compute simple shading
Rendering: Ray Tracing

• **Shading**: Interaction of light and material
• Secondary rays (shadows, reflection, refraction, indirect lighting)
Traditional Ray Tracing
Ray Tracing + Soft Shadows
Ray Tracing + Caustics
Global (Indirect) Illumination
Textures and Shading

At what point do things start looking real?

For more info on the computer artwork of Jeremy Birn see [http://www.3drenderer.com/jbirm/productions.html](http://www.3drenderer.com/jbirm/productions.html)
Sampling & Antialiasing

There are two diagrams illustrating the concept of sampling and antialiasing. The left diagram shows a pattern where the sampling is incorrect, leading to aliasing artefacts, indicated by the red 'X'. The right diagram shows a corrected sampling pattern, where the sampling is uniform and does not cause aliasing, indicated by the green checkmark.

The text below the diagrams reads:

- "not a box!"
- "not a circle!"

These texts highlight the visual differences between the correct and incorrect sampling scenarios.

Additionally, there is a graph showing a comparison between point sampling within and beyond the Nyquist limit, indicating the importance of accurate sampling to prevent aliasing.

- (a) Point sampling within the Nyquist limit
- (b) Point sampling beyond the Nyquist limit

This graph helps demonstrate the effects of aliasing and the importance of adhering to the Nyquist criterion in digital signal processing.
Shadows

Figure 12. Frame from *Luxo Jr.*

Figure 13. Shadow maps from *Luxo Jr.*
Assignment 5: Ray Tracing

- Shadows, reflection, refraction
- ..and extensions!
- Antialiasing
- Jittered Sampling
The Graphics Pipeline

Ray Casting

For each pixel
  For each object
    Does object hit pixel?

GPU

For each triangle
  For each pixel
    Does pixel hit triangle?
The Graphics Pipeline

Ray Casting

For each pixel
For each object
Send pixels to scene

GPU

For each triangle
For each projected pixel
Project scene to pixels

Both are ways of determining what is visible in each pixel – just in different order.
Phew! That’s a lot of stuff!

• BUT: Mastering all this takes you a long way towards cool applications!

Little Big Planet
Questions?
Simple 3D with OpenGL

• Scene represented as triangles
  – A triangle is a set of 3 vertices
  – A vertex is a set of 3 floating point numbers (x, y, z)

• We’ll use OpenGL to send this to the graphics card (GPU)
  – The GPU will do its magic to display the scene from the current viewpoint (Later, we’ll get to see how this happens)
How to Draw?

- You need to tell OpenGL
  - The geometry of the object
    - Vertex positions
    - Vertex normals
    - 3 x vertex makes a triangle!
  - How the object sits in world space
  - How the camera sits in world space
    - Above two combine into the “modelview matrix”
  - Camera parameters
    - Field of view, aspect ratio, (depth range)
    - The “projection matrix”
OpenGL Example: Viewing

```c
// Current matrix affects objects positions
glMatrixMode(GL_MODELVIEW);
// Initialize to the identity
glLoadIdentity();
// Position the camera at [0,0,5], looking at [0,0,0], with [0,1,0] as the up direction.
gluLookAt(0.0, 0.0, 5.0,
    0.0, 0.0, 0.0,
    0.0, 1.0, 0.0);
// Rotate by -20 degrees about [0,1,0]
glRotated(-20.0, 0.0, 1.0, 0.0);

// Draw a teapot.
glutSolidTeapot(1.0);
```
OpenGL Example: Viewing

```c
// Set up a perspective view,
// with square aspect ratio
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
// perspective camera:
// field of view, aspect ratio, ...
gluPerspective(50.0, 1.0, 1.0, 100.0);
```
Vertex Data

• What information do we need at each vertex?
  – Coordinates (3 floats)
  – Color (optional, 3 floats)
  – Normal vector (optional, 3 floats)
  – More to come (texture coordinates, shininess, ...)

Why normals?

- To compute color as a function of light direction
- Simplest: Diffuse or Lambert model
  - Intensity = dot product (normal, light direction)
OpenGL Code

```c
glBegin(GL_TRIANGLES);  //what follows describes triangles
glColor3d (1,1,0);     //red, green and blue components=>(yellow)
glNormal3d (0, 0, 1);  //normal pointing up
glVertex3d (2,3,3);    //3D position x, y, z
glColor3d (1,0,0);
grNormal3d (0, 0, 1);
glVertex3d (5,3,3);
glColor3d (1,0,1);
glNormal3d (0, 0, 1);
glVertex3d (3,6,3);
glEnd();
```
OpenGL Pseudocode

• Initialize
  – Setup window, load mesh, ...

• For each frame
  – Manage UI
  – Set appropriate viewpoint
  – Set light source directions
  – For each triangle T
    For i=0 to 2
    Send data for vertex Ti
OpenGL is a state machine

- Commands change the state
  - `glVertex` also “pushes” data

- For example, `glColor3f` changes current color
  - Color remains valid until `glColorXX` called again
  - Use it before each vertex to get per-vertex color

- Modelview and projection are part of state, too
- Other state to manage lighting, texturing, etc. etc.
- Can make it hard to debug
- (Note: This is conceptually simple, but not quite how you write efficient code these days.)
Assignment 0

• Read a file with triangle mesh data
  – Including mesh normals
• Display it using OpenGL
  – Colors, simple movement

• Due next Wednesday!
What’s Missing

- Shadows
- Shininess
- Texture
- Etc.

- Be patient, you’ll get there!
Linear Algebra is Everywhere

- Vertices are 3-vectors
- Normals are 3-vectors
  - Orthogonal to surface tangent plane
  - Cross product
- Colors are 3-vectors
- Diffuse shading is a dot product
- A non-bending object moving in a scene undergoes a rigid transformation
- Changing the viewpoint is a linear transformation of the scene coordinate
- Brush up in the review session!
Questions?
What Makes Graphics Fun?

- You can look at it from several angles
  - Anything that looks good will do in many applications...
    - Means you can really be creative once you know the basics.
    - ...but doing stuff “right” can be really involved.
  - Feels pretty nice when all that math and CS gives you a pretty picture or animation!
  - There is a continuum in between
That’s all for Today

• Further reading
  – OpenGL Red Book
    • http://www.glprogramming.com/red/

• Remember, Assignment 0 due already next Wednesday!

• (Keywords for OpenGL speed freaks: vertex buffer object, index buffer)