6.837 Introduction to Computer Graphics
Luxo Jr.

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

- Overview of computer graphics
- Administrivia
- Intro to OpenGL & assignment 0
- Overview of the semester
- Overview of assignments
What do you expect to learn?

• And why?
Movies
Games
Simulation
CAD-CAM & Design
Architecture
Virtual Reality
Visualization
Medical Imaging
What you will learn in 6.837

- Fundamentals of computer graphics algorithms
- How to implement most of the applications just shown
- How graphics APIs and the graphics hardware work

- And you will gain programming experience (C++)
What you will NOT learn in 6.837

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

• Artistic skills

• Game design
  – But some of the algorithms are used in games

• Graphics API in depth
  – Although you will use OpenGL
How much math?

• Not very deep
• Lots of simple linear algebra, polynomial roots
• Superficial exposure to cool math
  – Homogeneous coordinates
  – Quaternions
  – Monte-Carlo integration
  – Sampling, antialiasing
  – Ordinary differential equations
• Always in a concrete and visual context
• Deeper mathematic exposition in 6.839
Questions?
Plan

• Introduction

• Overview of the Semester

• Administrivia

• Iterated Function Systems (Fractals)
Team

• Instructor
  – Frédo Durand

• TA
  – Paul Green

• Course secretary
  – Bryt Bradley
Administrivia: Prerequisites

• Not strictly enforced
• All assignments are in C++
  – *Optional review/introductory session*
    *Monday Sept 8, 7:30-9pm,*
• Calculus, Linear Algebra
  – Solving equations, derivatives, integral
  – vectors, matrices, basis, solving systems of equations
  – *Optional review/introductory session*
Administrivia: Grading Policy

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

• Quiz: 10%
  – Tuesday, Oct 28 (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
• Collaboration policy:
  – You can chat, but code on your own
  – Acknowledge your collaboration! (in readme file)
• 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
Assignments

0: Warm up (mesh display with OpenGL)
1: Curves & surfaces
2: Hierarchical modeling, skinning
3: Physically-based simulation
4: Ray casting
5: Ray tracing
Web, email

- Slides will be online a few hours after lecture
  - Before if I happen to be ready early
- **6837-staff@csail.mit.edu**
  - To reach all 3 of us
  - Recommended to increase response chances
- **6837-discuss@csail.mit.edu**
  - To ask a question to your classmates
- Please subscribe to **6837-students@csail.mit.edu**
  - Class announcements
Textbooks

• No textbook is required

• Recommendations:
  
  – *3D Computer Graphics* by *Alan H. Watt*

Questions?
Basic 3D with OpenGL

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

• We will use the OpenGL API to send this to the graphics card
  – The card will do its magic to display the scene from the current viewpoint
  – Later this term, we’ll see how it works.
OpenGL high-level pseudocode

- **Initialize**
  - (get graphics context, etc.)

- **For each frame**
  - Manage UI
  - Set appropriate viewpoint
  - Set light source directions
  - For each triangle
    - For i=0 to 2
      - Send vertex data
Vertex data

- What information do we need at each vertex?
  - Coordinates (3 floats)
  - Color (optional, 3 floats)
  - Normal information (optional, 3 floats)
  - Transparency (optional, 1 float)
  - More to come (texture information, shininess)
Why normals?

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

• Shadows
• Shininess
• Texture
• Etc.

• Be patient, you’ll have plenty enough
OpenGL high-level pseudocode

• Initialize
  - (get graphics context, etc.)

• For each frame
  - Manage UI
  - Set appropriate viewpoint
  - Set light source directions
  - For each triangle
    For i=0 to 2
    Send vertex data
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);
glNormal3d (0, 0, 1);
glVertex3d (5,3,3);
glColor3d (1,0,1);
glNormal3d (0, 0, 1);
glVertex3d (3,6,3);
glEnd();
```
OpenGL is a state machine

- Each command changes the state
  - But `glVertex` also “pushes” data
- For example, `glColor3f` changes the current color.
  - The color remains valid until we call `glColorxx` again
  - Use it before each vertex to get per-vertex color.
- Other state to manage lighting and other rendering aspects
- Can make it hard to debug
Assignment 0

• Read a file with triangle mesh data
  – Including mesh normals
• Display it using OpenGL
Linear algebra everywhere

- Vertices are 3-vectors
- Normals are 3 vectors
  - Notion of orthogonality
  - Cross product becomes useful
- Colors are 3-vectors
- Diffuse shading is a dot product
- A solid object moving in a scene undergoes a rigid transformation
- Changing the viewpoint is a linear transform of the scene coordinate
Questions?
Overview of the Semester

- Modeling, Transformations
- Animation, Color
- Ray Casting / Ray Tracing
- The Graphics Pipeline
- Textures, Shadows
- Sampling, Global Illumination
Transformations

- Yep, good old linear algebra
- Homogeneous coordinates
  - (Adding dimensions to make life harder)
- Perspective

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ax+by+cz+d \\ ex+fy+gz+h \\ ix+jy+kz+l \\ 1 \end{bmatrix}$$
Modeling

- Curved surfaces
- Subdivision surfaces
Animation: Keyframing

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

FIGURE 3. Squash & stretch in Luxo Jr.'s hop.
Particle system (PDE)
Rigid Body Dynamics

- Simulate all external forces and torques
Color
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
Ray Tracing

- Shade (interaction of light and material)
- Secondary rays (shadows, reflection, refraction)
Ray Tracing

- Original Ray-traced image by Whitted
- Image computed using the Dali ray tracer by Henrik Wann Jensen
- Environment map by Paul Debevec
Textures and Shading

At what point do things start looking real?

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

not a box!

not a circle!

(a) Point sampling within the Nyquist limit
(b) Point sampling beyond the Nyquist limit
Traditional Ray Tracing
Ray Tracing + Soft Shadows
Ray Tracing + Caustics
Global Illumination
The Graphics Pipeline

Ray Casting

For each pixel
  For each object
  Send pixels to scene

Rendering Pipeline

For each triangle
  For each projected pixel
  Project scene to pixels
The Graphics Pipeline

- Transformations
- Clipping
- Rasterization
- Visibility
Shadows

Figure 12. Frame from *Luxo Jr.*

Figure 13. Shadow maps from *Luxo Jr.*
Assignment 1: curves & surfaces

Bezier curves

Surfaces of revolution

Sweep surfaces
Assignment 2: hierarchical modeling

- Animate character skeleton as tree of transformations
- Skinning: smooth surface deformation
Assignment 3: physics

- Simulate cloth as a mass-spring network
  - ODE integration
Assignment 4: ray casting

- Cast rays from the viewpoint
- Intersect with scene primitives
Assignment 5 ray tracing

- Shadows, reflection, refraction
- + flexible extension
Questions?