CS 563 Advanced Topics in Computer Graphics

by Emmanuel Agu

Professor Background

- Dr. Emmanuel Agu (professor, “Emmanuel”)
- Research areas
  - Computer Graphics (appearance modeling, etc)
  - Mobile Computing (mobile graphics, SVG, imode, etc)
- Research opportunities
  - Independent Study Project
  - MQP
  - MS theses
  - PhD theses

Student Background

- Name
- Class (undergrad (seniors), masters, PhD ...)
- Full and Part-time student
- Programming experience (C, C++, java)
- Systems experience (Unix, windows,...)
- Helpful background
  - At least one graphics class taken
  - Solid math skills....
  - Other (Physics, computer vision, image science, ???)
- Students intro themselves!
- Important: fill in above info, say what you want from this class

Course Prerequisites

- No official prerequisite
- However, will assume you
  - Have probably taken at least 1 graphics course (OpenGL?)
  - Can quickly pick up graphics, vision and image processing representations and techniques, (will briefly cover them in class as needed)
  - have background in calculus, linear algebra
  - Can read book(s), research articles, fill in gaps
  - Can learn rendering package, shading language, use it
- Still have questions? See me

Syllabus

- http://www.cs.wpi.edu/~emmanuel/courses/cs563/
- Office hours:
  - Monday: 1:00-2:00 Thursday: 2:00-3:00
  - Tuesday: 1:00-2:00 Friday: 2:00-3:00
  - Note: Please use office hours or book appointments first
- Questions of general interest, post on myWPI
- Email me if you have specific questions
- Text Book: Real-Time Rendering ***plus*** selected papers
- Note: can select your own papers, discuss with me at least 2 weeks before your talk
- Names?

Course Structure

- Grading
  - No exams
  - 2 presentations each (30%)
  - Write critiques for any 4 weeks (not 4 papers) (20%)
  - Class participation (6%)
  - Two projects defined by me (24%)
  - Final project, chosen by you (20%)

Why This Class?

- WPI graduate course requirements
  - Masters, PhD, grad course requirements
- WPI research requirements
  - Want to do research in graphics (MS, PhD theses)
- Work in graphics
  - Rendering
  - Animation, etc.
- Hobbyist
  - Want to build cooler stuff
  - Understand more how visual effects, etc happen

Course Objectives

- Understand state-of-the-art techniques for real time rendering
- Become conversant with cutting edge graphics literature
- Hands-on exploration of one (or more) of the techniques encountered.
- Use cutting edge shading language(s), rendering package(s), graphics card(s)
- Possibly extend one of the techniques

Class Time

- Two halves with 15 minutes break
- Each half
  - 45 minute presentation *followed by*
  - 30 minute discussion of topic(s) and questions

Presentations

- Goal is to guide you how to present effectively
- I will be strict with time (no breaks when presenting at conferences!!)
- Get right to the point (core), offer motivation & insights
- Communicate basic ideas to fellow students
- Offer a 'roadmap' for studying the paper
- Look over reading list & let me know which paper you want to present
- **Note:** can use any resources to build your talk. Must give credit. If not.. **Cheating**!!!
- Don't just summarize! Find authors websites, videos, images, supplimentary cool stuff

Presentations

- Common mistakes:
  - Avoid: putting too much on a slide (talk!!)
  - Too many slides for alloted time (2-3 mins/slide)
- First two student presentations next week

Final Project

- Implement one of the RT rendering techniques discussed in class, use shading language?
- May also use high end package to create models
  - Maya
  - Renderman
  - Blender
  - PovRay, etc
- Must submit your final project proposal by March 31st, 2005
- Can get ambitious: convert a photorealistic technique to real time
- Ideas?? See Stanford rendering competition
- *http://graphics.stanford.edu/courses/cs348b-competition/*

Where to do Projects?

- Most self-respecting home PC’s have a graphics card
- Some even have sweet Nvidia or ATI cards
- You can use your home computer
- Only snag: demo project at the end
- Some students can also use movie lab
  - Gordon Library, Room 208, next to circulation desk
  - HP XW4100 Workstations (3.06GHz with HyperThreading, 1GB RAM)
  - 18" HP LCD monitors
  - PDF scanner, HP scanner
  - NVIDIA GeForce 4 TI 4800, 128MB on board RAM

Movie lab, Books

- I have requested installation of Cg toolkit (shader language)
- Cg installation complete within a week
- Other graphics software in movie lab:
  - 3D Studio Max, Maya personal learning edition, Adobe photoshop, illustrator
- Supplementary books:
  - On reserve in the library under CS 563 folder
  - Cg Tutorial by Randima Fernando and Mark Kilgard
  - Real-Time Shading by Olano et al
  - More to come..

About This Course

- Previous versions of class
  - Students chose any topics they liked
  - Students tend to pick what's easy
  - Sometimes big picture lost
- This version..
  - Suggested structure/papers based on hot trends
  - In fact, using an advanced text for most of the literatre for the first time
  - Creates better flow, students understand better
  - Still do your own literature survey, etc
  - Will get nice points for finding sweet links, videos, supplementary material

About This Course

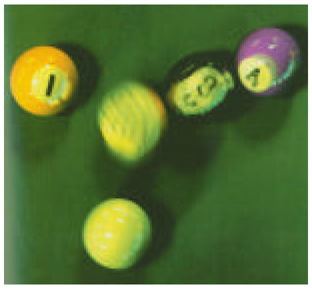
- Focus this semester on Real time rendering
- Last time, focussed on *Photo-realistic* Rendering
  - Rendering techniques
    - Advances in ray tracing
    - Photon mapping
  - Image-based rendering
  - Appearance Modeling
    - BRDFs (representations, viewers, acquisition)
    - Rendering humans (face, skin)
    - Rendering nature (water, trees, seashells)
    - Rendering animals (feathers, butterflies)

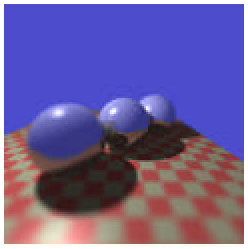
Why Ray Tracing Looks Fake

- Jagged edges
- Hard shadows
- Everything in focus
- Objects completely still
- Surfaces perfectly shiny
- Glass perfectly clear

Why Ray Tracing Looks Fake

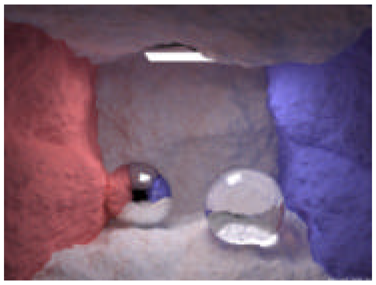
- Motion blur
- Depth of Field
  - Better simulation of camera model
    - f-stop
    - focus
- Others (soft shadow, glossy, etc)





Photon Mapping

- Jensen EGRW 95, 96
- Simulates the transport of individual photons
- Two parts.
  - Photons emitted from source
  - Photons deposited on surfaces
- Secondly:
  - Photons reflected from surfaces to other surfaces
  - Photons collected by rendering
- Good for:
  - Light through water
  - Cloud illumination
  - Marble



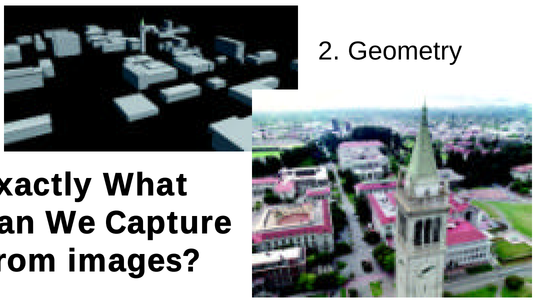
Rendering Techniques

- Photon mapping examples

Images: courtesy of Stanford rendering contest

Image-Based Rendering

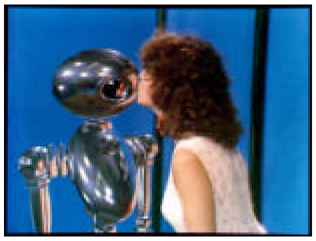
1. Appearance



2. Geometry

Exactly What Can We Capture From images?

3. Reflectance & Illumination

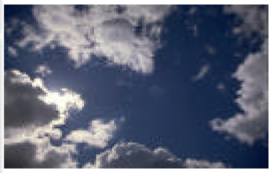
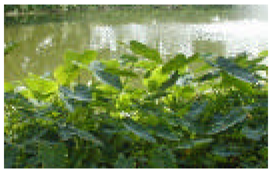
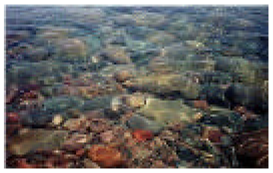


4. Motion

IBR Pros and Cons

- Pros
  - Modest computation compared to classical C.G.
  - Cost independent of scene complexity
  - Imagery from real or virtual scenes
- Cons
  - Static scene geometry
  - Fixed lighting
  - Fixed look-from or look-at point

Appearance Modeling

Appearance Models

- Why does the sky **appear** blue?
- Why does wet sand **appear** darker than dry sand?
- Why do iridescent surfaces (CD-ROM, butterflies, hummingbird) wings) **appear** to have different colors when viewed in different directions ?
- Why do old and weathered surfaces **appear** different from new ones?
- Why do rusted surfaces **appear** different from un-rusted ones?
- **Appearance models** in computer graphics and vision try to answer these questions

Real Time Rendering

- Photo-realistic rendering does NOT care how long it takes
- If we have a technique that renders realistic sea shells but it takes days, still use it
- Some applications require images to be displayed relatively quickly = Real-time rendering
- Examples:
  - Games
  - Flight simulators
  - Virtual reality
  - Augmented reality, etc

What is Real-Time Rendering?

- Purist may argue that real-time rendering should happen instantaneously (too strict)
- We can relax this a little
- Rendering speed measured in Frames Per Second (fps)
- Frank Cho, electronics arts, used algorithms must run in at most 30 fps (minimum)
- About 72 fps guarantees that user:
  - becomes immersed in graphics experience
  - interacts freely
  - No distraction of waiting for rendering to complete
- So, we can say 15 fps upwards is real-time
- All images produced must have feel of 3D graphics

Real-Time Rendering

- How can we achieve real-time speeds
  - Pre-process graphics models (simplify, replace polygons with textures, etc)
  - Graphics Hardware acceleration
- Graphics Hardware:
  - Previously, SGI was king
  - Today: 3D graphics cards from ATI, Nvidia on PCs
  - 3Dfx Voodoo 1 was first card in 1996
    - beginning of real-time graphics era?
  - Most of advances in real time graphics are due to innovation in graphics cards
    - Chip on card also called Graphics Processing Unit (GPU)
    - Speed: GPUs renders graphics faster CPUs
    - Programmability: Flexibility

Comparison: SGI InfiniteReality (1998) Vs Nvidia GeForce4 (2002)

Metric	SGI IR	Nvidia GF4
Triangles/sec	13 million	136 million
Pixels/sec		4.8 million
Texture memory	64MB	128MB
Bump mapping	No	Yes
Programmable Vertex engine?	No	Yes
Programmable Pixel engine?	No	Yes
Physical size	Mini-Fridge	Video Cassette
Cost	$100,000	$300

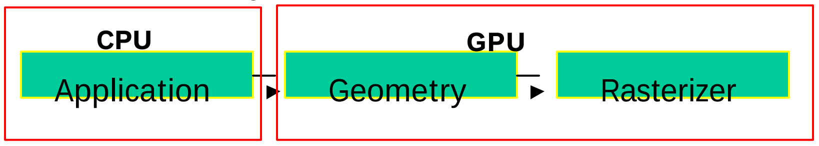
Major advance

How quickly are these developments happening?

- SGI: new product every 3 years
- Nvidia/ATI:
  - new product every 6/18 months
  - Cards performance double every 10 months
  - Moore’s law cubed??@!#@#!!
  - More and more algorithms/features being moved to graphics card
    - Programmable pipelines
    - Floating point support
    - Hardware occlusion

Graphics Pipeline Revisited

- Conceptual graphics pipeline fits into 3 parts
  - Application stage
  - Geometry stage
  - Rasterizer stage



- OpenGL pipeline fits into geometry and rasterizer stages
- Application stage is “stuff” that main program may do before sending vertices down the pipeline

Graphics pipeline

- What about modelview, clipping, projection, etc?
- Functional stages fit into conceptual stages
- Application stage:
  - Stuff that you thought about as your OpenGL program
  - Camera movement: slide, pitch, yaw, roll
  - Collecting user interaction with models
  - Animation calculations
- Geometry stage (also called vertex pipeline):
  - Model and view transforms
  - Lighting
  - Projection
  - Clipping
  - Screen mapping

Graphics Pipeline

- Rasterizer stage (also called pixel pipeline):
  - Fill algorithm (line drawing, polygon fill, etc)
  - Z-buffer
  - Texturing: Look up/fill textures
- Application stage => usually in software
- Geometry stage => may be in hardware of software
- For high performance graphics rasterizer stage has to be in hardware!!!

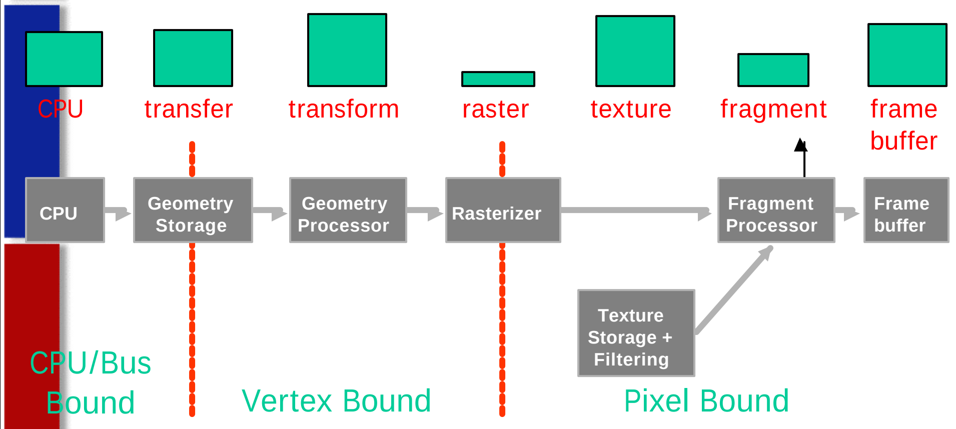
Programmable Pipeline

- With OpenGL, programs focussed on generating and pumping primitives (actually vertices) down pipeline
- Little control of vertices once in pipeline, fixed functions
- Recent hardware offers option of replacing portions of pipeline with user-programmed stages
  - ***Vertex shader:*** replaces fixed-function transform and lighting
  - ***Pixel shader:*** replaces texture stages
- GPU typically programmed using shading languages
  - Examples: Cg, HLSL, OpenGL shading language, RTSL, etc

Performance Bottlenecks

- A pipeline's fastest speed is defined by it's *slowest* stage
- Depending on specific configuration, application, geometric or rasterizer stages could be the bottleneck
- We say:
  - Application stage bottleneck => ***application/CPU/bus-bound***
  - Geometry stage bottleneck => ***vertex/geometry-bound***
  - Rasterizer stage bottleneck => ***pixel-bound***

Possible Bottlenecks

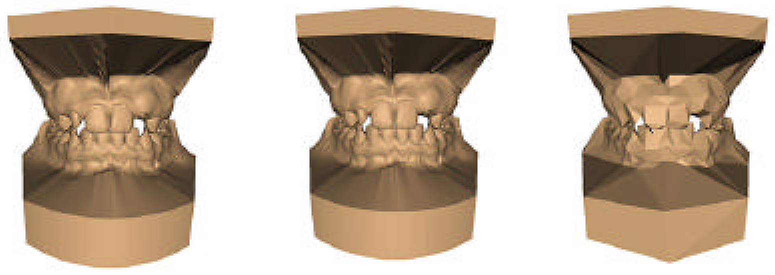


List of Topics

- Real time applications: games, augmented reality, etc
- Texturing to improve RT performance
- BRDF factorization, SH lighting
- Pixel/vertex shading
- Shader languages/programming
- Image-based rendering
- Polygonal techniques/geometric simplification
- Point-based rendering
- Mobile graphics

Geometric Simplification

- Produce lower resolution approximation with fewer polygons
- Jeff Somers' viewer demo

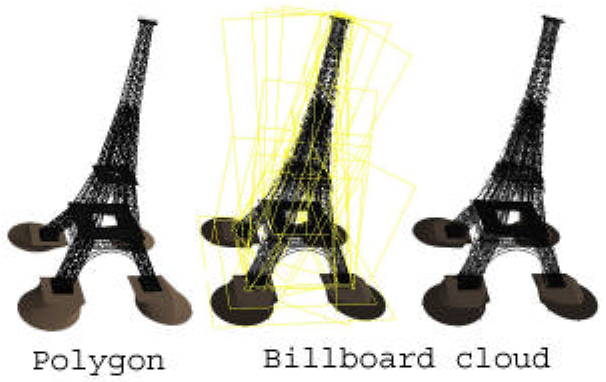


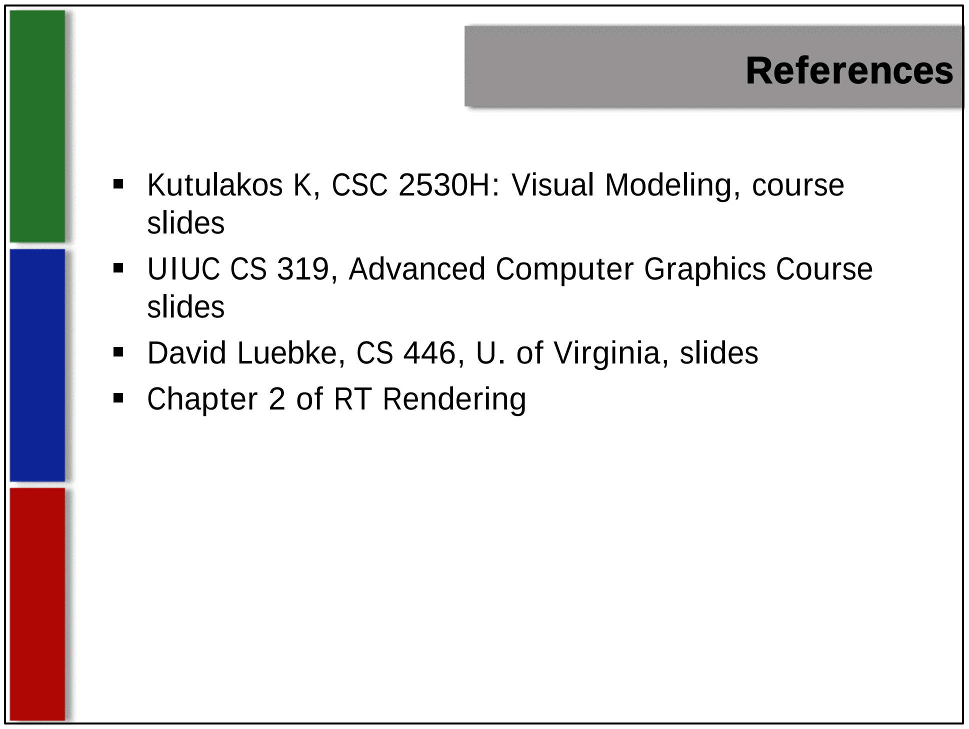
Original: 424,000 triangles (laptop) **60,000 triangles (14%) (PDA)** **1000 triangles (0.2%) (cellphone)**

(courtesy of Michael Garland and Data courtesy of Iris Development.)

Image-based Simplification

- **Billboard Clouds**, Decoret, Durand *et al* [SIGGRAPH'03]
- Represent mesh as sequence of images
- **Pros:** images take less memory, realism doesn't suffer
- **Cons:** interactivity suffers





References

- Kutulakos K, CSC 2530H: Visual Modeling, course slides
- UIUC CS 319, Advanced Computer Graphics Course slides
- David Luebke, CS 446, U. of Virginia, slides
- Chapter 2 of RT Rendering