CS 536
Computer Graphics

Introduction
Week 1, Lecture 1

David Breen, William Regli and Maxim Peysakhov
Department of Computer Science
Drexel University
Overview

• Course Policies/Issues
• Brief History of Computer Graphics
• The Field of Computer Graphics: A view from 66,000ft
• Structure of this course
• Homework overview
• Introduction and discussion of homework #1
Computer Graphics: Course Goals

• Provide introduction to *fundamentals* of 2D and 3D computer graphics
  – Representation (curves/surfaces/solids/hierarchical models)
  – Drawing, clipping, transformations and viewing
  – Evaluation of geometric models
    • Convert to Open Inventor for display
Interactive Computer Graphics
CS 537

• Should be offered again next year
• Learn and program webGL
• Computer Graphics was a pre-requisite
  – Not anymore
Advanced Rendering Techniques (Advanced Computer Graphics)

- Not sure when it will be offered again
- 3D Computer Graphics
- CS 536 or CS 537 is a pre-requisite
- Implement Ray Tracing algorithm
- Lighting, rendering, photorealism
- Study Radiosity and Photon Mapping
ART Student Images
Computer Graphics: Technical Material

• Course coverage
  – Mathematical preliminaries
  – 2D lines and curves
  – Geometric transformations
  – Line and polygon drawing
  – 3D viewing, 3D curves and surfaces
  – Bezier & Hermite curves, Splines, B-Splines and NURBS
  – Solid Modeling
  – Hierarchical models
  – Color, hidden surface removal, Z-buffering
Computer Graphics:
Course Management Issues

• All course policies are in the syllabus
• Extensive use of PDF handouts
• Must read email every day
• There will be 7 programming assignments (*plan on 8-to-15 hrs*)
• Suggestion: print out handouts before class, use them to take notes
• Read and summarize a research paper
• Final exam on material not covered by the programming assignments
• *READ THE SYLLABUS!!*
Computer Graphics: Collaboration Policies

• Thou Shall
  – write your own code
  – do your own math
  – attribute any work that is not your own
  – talk amongst yourselves, share ideas

• Thou Shall Not
  – Share/copy code
  – Work in groups
  – Use generative tools
  – Utilize geometry/graphics libraries

• All code will be auto checked for plagiarism
• Violations will result in an automatic $F$
Go to class web page
CG Technical Areas

• Geometric Modeling
  – Mathematics and algorithms that define 2D and 3D geometric objects
CG Technical Areas

- Human/Computer Interaction
  - Methods for creating graphics data via user input

Surface Drawing, Steven Schkolne
CG Technical Areas

• Lighting and Shading
  – Math, physics and algorithms that specify how light interacts with matter
CG Technical Areas

• Rendering
  – Algorithms that take geometry, lighting, shading and viewing information and generate an image
CG Technical Areas

• Visualization
  – Techniques for visually communicating and exploring scientific, medical or abstract data
CG Technical Areas

- Perception
  - Study of how humans perceive light and information
CG Technical Areas

• Animation
  – Algorithms for making models change over time
CG Technical Areas

- Simulation
  - Using physics to make models move
CG Technical Areas

• Software and Hardware
  – Designing software and hardware systems to implement graphics algorithms
Computer Graphics: In The Beginning

- MIT - 1963
  Ivan Sutherland’s *Sketchpad*
- Modified oscilloscope for drawing
- The original CAD system

Courtesy Marc Levoy @ Stanford U
Computer Graphics from 66,000ft

- Display types
- Display/Rendering algorithms
- Application areas
  - Entertainment
  - CAD/CAM
  - Scientific & medical visualization
  - Training & education
  - Synthetic realities
  - Art and design
  - Games
2D Graphics

- **Raster:**
  
  Pixels
  - X11 bitmap, XBM
  - X11 pixmap, XPM
  - GIF
  - TIFF
  - PNG
  - JPG

  Lossy, jaggies when transforming, good for photos.

- **Vector:**
  
  Drawing instructions
  - Postscript
  - CGM
  - Fig
  - DWG

  Non-lossy, smooth when scaling, good for line art and diagrams.
2D Graphics

• Raster:

• Vector:
Adobe Photoshop: 2D Raster Graphics
2D Raster Graphics
Adobe Illustrator: 2D Vector Graphics
2D Vector Graphics
3D Rendering

• 1960s - the visibility problem
  – Roberts (1963), Appel (1967) - hidden-line algorithms
  – Sutherland (1974) - visibility = sorting

• 1970s - raster graphics
  – Gouraud (1971) - diffuse lighting
  – Phong (1974) - specular lighting
  – Blinn (1974) - curved surfaces, texture
  – Crow (1977) - anti-aliasing

Courtesy Marc Levoy @ Stanford U
3D Rendering

Toward Reality in the 1980s

• global illumination
  – Whitted (1980) - ray tracing
  – Goral, Torrance et al. (1984), Cohen (1985) - radiosity
  – Kajiya (1986) - the rendering equation

• photorealism
  – Cook & Torrance (1982) – rough surface reflectance
  – Cook (1984) - shade trees
  – Perlin (1985) - shading languages
  – Hanrahan and Lawson (1990) - RenderMan

Courtesy Marc Levoy @ Stanford U
Away from Reality

- early 1990s - non-photorealistic rendering
  - Drebin et al. (1988), Levoy (1988) - volume rendering
  - Haeberli (1990) - impressionistic paint programs
  - Salesin et al. (1994-) - automatic pen-and-ink illustration
  - Meier (1996) - painterly rendering

Courtesy Marc Levoy @ Stanford U
And Back Again

• late 1990s & 2000s - photon mapping, subsurface scattering and participating medium
  – H. Wann Jensen
Application Areas

- *Entertainment*
- CAD/CAM
- Scientific & Medical visualization
- Training & Education
- Synthetic Realities
  - VR, AR, etc.
- Art and design
- Games

Pixar
Lord Of the Rings Troll
Application Areas

- Entertainment
- **CAD/CAM**
- Scientific & Medical visualization
- Training & Education
- Synthetic Realities
  - VR, AR, etc.
- Art and design
- Games

\[
\tau = M(\ddot{e}) \cdot \dot{e} + V(\dot{e}, \dot{e}) + G(\dot{e}) + F(\dot{e}, \dot{e})
\]

Regli et al @ Drexel
Application Areas

• Entertainment
• CAD/CAM
• **Scientific & Medical Visualization**
• Training & Education
• Synthetic Realities
  – VR, AR, etc.
• Art and design
• Games

Lombeyda & Breen @ CalTech
Application Areas

- Entertainment
- CAD/CAM
- Scientific visualization
- *Training & Education*
- Synthetic Realities
  - VR, AR, etc.
- Art and design
- Games
Application Areas

- Entertainment
- CAD/CAM
- Scientific visualization
- Training & Education
- **Synthetic Realities**
  - VR, AR, etc.
- Art and design
- Games
Application Areas

- Entertainment
- CAD/CAM
- Scientific visualization
- Training & Education
- Synthetic Realities
  - VR, AR, etc.
- **Art and design**
- Games
Application Areas

- Entertainment
- CAD/CAM
- Scientific visualization
- Training & Education
- Synthetic Realities – VR, AR, etc.
- Art and design
- Games
Programming Assignments

• **No APIs**: OpenGL, GLUT, Mesa, DirectX…

• May use math (vector/matrix), argument processing and Open Inventor writing libraries

• Just compute lines and polygons from higher-level descriptions
  – Sorry, no color until ICG or ART

• Output in the form of Open Inventor
  – A standard (now a bit out-of-fashion) for 3D graphics

• Program source (and makefile/script) turned in via Bb Learn

• **Executable/Programs MUST RUN on Linux (tux)**

• **Whatever language you want**, so long as the TA can run code on tux (c, c++, java, python, …)
For programming assignments

• Use Open Inventor as your graphics representation
• We will be evaluating a variety of 3D graphics primitives
• GPUs can (only) display lines and polygons. So higher-level geometry descriptions need to be converted into these lower-level primitives
• Tip: Renew your friendship with your linear algebra & multi-variate calculus textbook
• Read homeworks ahead. It will help you to structure your code for future requirements.
Assignment Dependencies

- Every HW – Read in geometry parameters and write out (to standard out) Open Inventor
- HW1 – Evaluate arbitrary-degree Bezier curves
  - Output polylines & spheres
- HW2 – Link together cubic Bezier curves from HW1
  - Output polylines & spheres
- HW3 – Evaluate bicubic Bezier patch
  - Output triangles, normals & spheres
- HW4 – Evaluate surface of revolution
  - Output triangles & normal. Similar to HW3
- HW5 – Union of two polygons
  - Input and output Postscript
- HW6 – Evaluate a (biparametric) superellipsoid. Similar to HW4
  - Output triangles & normals
- HW7 – Evaluate a hierarchical model consisting of cuboids
- EC – Evaluate points on the surface of a CSG model
  - Output spheres
Data Structures

• Control points and tangents
• 3D Polylines
• 3D Polygons (vertices, edges, faces and normals)
• Triangle mesh (topology of mesh)
• Model hierarchy
When it’s all over!
Don’t forget ICG & ART!
Programming Assignment 1

• Input list of 3D control points
• Output polyline & spheres as Open Inventor
• Primary output format for the course
• Create data structures to hold control points and polyline
• Implement code to evaluate (i.e., compute points on) an arbitrary-degree 3D Bezier curve
• Draw spheres at control point locations
• Due April 19th
• Get started now!
Programming Assignment 1

• Process command-line arguments
• Read in 3D control points
• Iterate through parameter space by du
  – for loop should use integers!
• At each u value evaluate Bezier curve
  formula to produce a sequence of 3D
  points
• Output points by printing them to
  standard out as a polyline and control
  points as spheres in Open Inventor