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 I:
Course Goals
• Provide introduction to fundamentals of 2D and 3D computer graphics
  – Representation (lines/curves/surfaces)
  – Drawing, clipping, transformations and viewing
  – Implementation of a basic graphics system
    • simple frame buffer with PBM format
    • ties together 3D projection and 2D drawing

Interactive Computer Graphics
CS 432
• Learn and program WebGL
• Computer Graphics was a pre-requisite
  – Not anymore
• Useful for Games classes
• Part of the HCI and Intelligent Systems tracks

Advanced Rendering Techniques
(Advanced Computer Graphics)
• Not sure when it will be offered again
• 3D Computer Graphics
• CS 430/536 is a pre-requisite
• Implement Ray Tracing algorithm
• Lighting, rendering, photorealism
• Study Radiosity and Photon Mapping

ART Student Images
Computer Graphics I: Technical Material

- Course coverage
  - Mathematical preliminaries
  - 2D lines and curves
  - Geometric transformations
  - Line and polygon drawing
  - 3D viewing, 3D curves and surfaces
  - Splines, B-Splines and NURBS
  - Solid Modeling
  - Color, hidden surface removal, Z-buffering

Computer Graphics I: Course Highlights

- Bresenham’s scan conversion algorithm
- Cohen-Sutherland clipping algorithm
- Sutherland-Hodgman polygon clipping
- The De Casteljau Algorithm
- Polygon filling
- B-Splines, NURBS, De Boor’s Algorithm
- Z-buffer algorithm; backface culling

Computer Graphics I: Course Management Issues

- All course policies are in the syllabus
- Extensive use of PDF handouts
- Must read email every day
- There will be 6 programming assignments *(plan on 8-to-15 hrs)*
- Suggestion: print out handouts before class, use them to take notes
- Final exam on material not covered by the programming assignments
- **READ THE SYLLABUS!!**

Computer Graphics I: 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
  - Use ideas without attribution
  - Utilize geometry/graphics libraries
- All code will be auto checked for plagiarism
- Violations will result in an automatic **F**

CG Technical Areas

- Geometric Modeling
  - Mathematics and algorithms that define 2D and 3D geometric objects

Go to class web page
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

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

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.
Adobe Photoshop: 2D Raster Graphics

Adobe Illustrator: 2D Vector Graphics

3D Rendering

- 1960s - the visibility problem
  - Roberts (1963), Appel (1967) - hidden-line algorithms
  - Warnock (1968), Watkins (1970) - hidden-surface algorithms
  - Sutherland (1974) - visibility = sorting

- 1970s - raster graphics
  - Courant (1973) - diffuse lighting
  - Phong (1974) - specular lighting
  - Binn (1974) - curved surfaces, texture
  - Crow (1977) - anti-aliasing

2D Raster Graphics

2D Vector Graphics

3D Rendering

Toward Reality in the 1980s

- global illumination
  - White (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
  - Perlman (1985) - shading languages
  - Hanrahan and Lawson (1980) - 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

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

Lord Of the Rings Troll

Application Areas

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

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

FakeSpace Cave

Augmented Reality

Programming Assignments

- No APIs: OpenGL, GLUT, Mesa, DirectX...
- Just line and filled polygon rendering
  - Color in last assignment
- Output in the form of 2D ASCII bitmaps
  - PBM and PPM
- Program source (and makefile) turned in via Bb Learn
- Executable MUST RUN on Linux (tux)
- Whatever language you want, so long as you can deliver a program that TA can run (c, c++, java, python, ...)

Kimmel Center
For programming assignments
• Use PBM as B/W “software” frame buffers
• We will be implementing parts of the 2D Engine, 3D Engine and Pixel Cache of a graphics accelerator
• Issue: How to translate the mathematics of 2D/3D shapes to the 2D screen?
• Tip: Renew your friendship with your linear algebra textbook
• Read homeworks ahead. It will help you to structure your code for future requirements.

Assignment Dependencies
• Every HW - Read in geometry and write images
• HW1 - Clip 2D lines and draw them
• HW2 - Clip 2D polygons and draw edges with HW1
• HW3 - Use HW2 to clip 2D polygons and fill in interiors
• HW4 - Project 3D polygon edges (3D lines) into 2D. Draw them with HW2
• HW5 - Project 3D triangles into 2D. Combine HW3 and HW4, and add depth buffer
• EC - Generate lines and draw with HW1

Programming Assignments Dependencies

<table>
<thead>
<tr>
<th>HW</th>
<th>HW 1</th>
<th>HW 2</th>
<th>HW 3</th>
<th>HW 4</th>
<th>HW 5</th>
<th>EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW1</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW2</td>
<td>2D Lines cleaning</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW3</td>
<td>Polygon filling</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW4</td>
<td>Window to Viewport Transformation</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW5</td>
<td>3D Geometry</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>EC</td>
<td>Color</td>
<td>Depth-Cueing</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

Data Structures
• Initially B/W, then RGB Frame Buffer
• 2D Lines
• 2D/3D polygons (vertices and edges)
• Triangle mesh
• 3D camera/scene

When it’s all over!

Don’t forget ICG & ART!
Programming assignment 1

• Input PostScript-like file
• Output B/W PBM
• Primary I/O formats for the course
• Create data structure to hold points and lines in memory (the world model)
• Implement 2D translation, rotation and scaling of the world model
• Implement line drawing and clipping
• Due October 6th
• Get started now!