CS 430
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
    • draw lines using Postscript
    • simple frame buffer with PBM & PPM 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
- Looks at graphics “one level up” from CS 430
- Useful for Games classes
- Part of the HCI and Game Development & Design tracks?
Advanced Rendering Techniques (Advanced Computer Graphics)

• Might be offered in the Spring term
• 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 5 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 $F$
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
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
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
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

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

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

![Telepresence](image1)

![UCLA](image2)

![Augmented Reality](image3)

![FakeSpace Cave](image4)
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…
• Just line and filled polygon rendering
  – Color in last assignment
• Output in the form of
  – Lines (Postscript)
  – 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, …)
For programming assignments

• Compute line segments. Export as Postscript.
• 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 out lines or images
• HW1 - Clip 2D lines and export 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>HW1</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>Write PBM file</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read &quot;Postscript&quot; File</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2D Transformations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line Drawing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2D Line Clipping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW 2</td>
<td>Polygon Clipping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW 3</td>
<td>Window to Viewport Transformation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Polygon Filling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW 4</td>
<td>3D Geometry</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3D Viewing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW 5</td>
<td>Color</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Z-Buffering</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Depth-Cueing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC</td>
<td>Bezier Curve Drawing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data Structures

• 2D Lines
• Initially B/W, then RGB Frame Buffer
• 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 Lines as Postscript
- 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 clipping
- Due October 3rd
- Get started now!