
Prof. Dr. Stelian Coros
What is Computer Graphics?

**computer graphics** /kəmˈpyʊdər ˈɡræfɪks/ n.
The use of computers to synthesize and manipulate visual information.
Why visual information?

Humans are visual creatures!

About 30% of brain dedicated to visual processing...

...eyes are highest-bandwidth port into the head!
History of visual depiction

Humans have always been visual creatures!

Indonesian cave painting (~38,000 BCE)
History of visual depiction

Useful to represent MANY different types of information…
History of visual depiction

Of course, not restricted to “flat” representations
History of visual depiction: photography / imaging

Processing of visual data no longer happening just in our heads!

Joseph Niépce, “View from the Window at Le Gras” (1826)
History of visual depiction: photography / imaging
Digital imagery: the humble beginnings of Computer Graphics

Intersection of visual depiction and computation

Ivan Sutherland, “Sketchpad” (1963)
Digital imagery

Nowadays, Computer Graphics is everywhere!
Applications

Entertainment
Secondary Motion in Rig-Space
Applications

Entertainment – not just cartoons!
Applications

Entertainment – not just cartoons!
Applications

Medical Simulations
Applications

Scientific Visualization
Applications

Virtual/Augmented Reality
Applications

Computer-aided engineering
Applications

Art and design
Visual Technology: digital fabrication
Applications

Communication and social interactions
Performance Capture

Result - Leila

Camera 5 of 7

Camera 6 of 7

ETH Zürich
Recent research highlights…
What is Computer Graphics?

**computer graphics** /kəmˈpyʊdər ˈɡræfɪks/ n.
The use of computers to synthesize and manipulate visual information.
Foundations of computer graphics

• All these applications demand sophisticated theory & systems

• Theory
  – geometric representations
  – sampling theory
  – integration and optimization
  – Perception
  – etc

• Systems
  – parallel, heterogeneous processing
  – graphics-specific programming languages
Foundations of computer graphics

• We will begin exploring some of these concepts

• But first…
Class activity: modeling and drawing a cube

Let’s generate a drawing of a cube

Key questions:
- Modeling: how do we describe the cube?
- Rendering: how do we then visualize this model?
Class activity: modeling and drawing a cube

Goal: generate a realistic drawing of a cube

Suppose our cube:
- Centered at the origin (0,0,0)
- Has dimensions 2x2x2
- Edges are aligned with x,y,z axes

What are the coordinates of the cube vertices?

A: (1, 1, 1)  E: (1, 1, -1)
B: (-1, 1, 1)  F: (-1, 1, -1)
C: (1, -1, 1)  G: (1, -1, -1)
D: (-1, -1, 1)  H: (-1, -1, -1)
Class activity: modeling and drawing a cube

Goal: generate a realistic drawing of a cube

What are the coordinates of the cube vertices?

A: (1, 1, 1)  E: (1, 1, -1)
B: (-1, 1, 1)  F: (-1, 1, -1)
C: (1, -1, 1)  G: (1, -1, -1)
D: (-1, -1, 1)  H: (-1, -1, -1)

What about the edges?

AB, CD, EF, GH,
AC, BD, EG, FH,
AE, CG, BF, DH
Class activity: modeling and drawing a cube

We now have a digital description of the cube:

<table>
<thead>
<tr>
<th>Vertices</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: (1, 1, 1)</td>
<td>E: (1, 1, -1)</td>
</tr>
<tr>
<td>B: (-1, 1, 1)</td>
<td>F: (-1, 1, -1)</td>
</tr>
<tr>
<td>C: (1, -1, 1)</td>
<td>G: (1, -1, -1)</td>
</tr>
<tr>
<td>D: (-1, -1, 1)</td>
<td>H: (-1, -1, -1)</td>
</tr>
<tr>
<td></td>
<td>AB, CD, EF, GH,</td>
</tr>
<tr>
<td></td>
<td>AC, BD, EG, FH,</td>
</tr>
<tr>
<td></td>
<td>AE, CG, BF, DH</td>
</tr>
</tbody>
</table>

How do we draw this 3D cube as a 2D (flat) image (e.g. screen)?

Basic strategy:
1) map 3d vertices to 2D points in the image
2) draw straight lines between 2D points corresponding to edges

Ok, but how?
Perspective projection

- Objects look smaller as they get further away ("perspective")
- Why does this happen?
- Consider simple ("pinhole") model of a camera:
Perspective projection: side view

- Where does a point \( p=(x,y,z) \) from the world end up in the image (\( q=(u,v) \))?
Perspective projection: side view

- Where does a point $p = (x, y, z)$ from the world end up in the image $q = (u, v)$?
- Notice the two similar triangles!
- Assume camera has unit size, origin is at pinhole $c$
- Then $v/1 = y/z$ (vertical coordinate is just the slope $y/z$)
- Likewise, horizontal coordinate is $u = x/z$
Class activity: Let’s draw that cube!

Need 12 volunteers
- each person will draw one edge of the cube
- assume camera is at c=(2,3,5)
- convert (X,Y,Z) of both endpoints to (u,v)
  1. subtract camera c from vertex (X,Y,Z) to get (x,y,z)
     (cube vertices expressed relative to camera)
  2. divide (x,y) by z to get (u,v) – write as a fraction
- draw line between (u1, v1) and (u2, v2)

<table>
<thead>
<tr>
<th>Vertices</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: ( 1, 1, 1 )</td>
<td>E: ( 1, 1, -1 )</td>
</tr>
<tr>
<td>B: (-1, 1, 1 )</td>
<td>F: (-1, 1, -1 )</td>
</tr>
<tr>
<td>C: ( 1,-1, 1 )</td>
<td>G: ( 1,-1,-1 )</td>
</tr>
<tr>
<td>D: (-1,-1, 1 )</td>
<td>H: (-1,-1,-1 )</td>
</tr>
</tbody>
</table>

Edges: AB, CD, EF, GH, AC, BD, EG, FH, AE, CG, BF, DH
Class activity: Let’s draw that cube!
Class activity: How did we do?

2D coordinates:

A: 1/4, 1/2
B: 3/4, 1/2
C: 1/4, 1
D: 3/4, 1
E: 1/6, 1/3
F: 1/2, 1/3
G: 1/6, 2/3
H: 1/2, 2/3
Class activity: Previous year’s result
But wait... how do we draw lines (on a computer)?
Drawing on a raster display

• Common abstraction:
  – Image represented as a 2D grid of “pixels”
  – Each pixel can take on a unique color value
Close up photo of pixels on a modern display
Which pixels should we color in to depict the line?

- Rasterization: converting continuous object to a discrete representation on a pixel grid (raster grid)
Which pixels should we color in to depict the line?

- Light up all pixels intersected by the line?
Which pixels should we color in to depict the line?

- Light up all pixels intersected by the line?
Which pixels should we color in to depict the line?

- Diamond rule (used by modern graphics hardware)
Which pixels should we color in to depict the line?

- Is there a “right” answer?
How do we find the pixels satisfying a chosen rasterization rule?

• Could check every single pixel in the image to see if it meets the condition…
  – $O(n^2)$ pixels in image vs at most $O(n)$ “lit up” pixels
  – Must be able to do better (work proportional to the number of pixels in the drawing of the line)
Incremental line rasterization

- Let's say a line is represented with integer endpoints \((u_1, v_1)\) and \((u_2, v_2)\).
- Slope of line: \(s = (v_2-v_1)/(u_2-u_1)\).
- Consider a very easy special case:
  - \(u_1 < u_2, v_1 < v_2\) (line points toward upper-right)
  - \(0 < s < 1\) (more change in \(x\) than in \(y\)).

Common optimization: rewrite with only integer arithmetic (Bresenham algorithm):

\[
\begin{align*}
v &= v_1; \\
&\text{for}( u=u_1; u<=u_2; u++ ) \\
&\quad \{ \\
&\quad \quad v += s; \\
&\quad \quad \text{draw}( u, \text{round}(v) ) \\
&\quad \}\n\end{align*}
\]
Our beautiful cube, now on a computer…
But… we rendered only a simple line drawing…

- We want much richer models of the world…

  Shapes and surfaces
  
  Materials
  
  Lighting environments
  
  Physics-based simulation
## Schedule

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Topic</th>
<th>Exercise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nov. 7/9</td>
<td>Introduction, How to draw a triangle</td>
<td>Ex. 6: OpenGL Rendering</td>
</tr>
<tr>
<td>Nov. 14/16</td>
<td>Transformations, Rendering Pipeline</td>
<td>Ex. 7: Shaders in OpenGL</td>
</tr>
<tr>
<td>Nov. 21/23</td>
<td>Geometry and Textures</td>
<td>Ex. 8: Theory: Light and Colors</td>
</tr>
<tr>
<td>Nov. 28/30</td>
<td>Lighting and Shading, Visibility and Shadows</td>
<td>Ex. 9: Matrices and Quaternions</td>
</tr>
<tr>
<td>Dec. 5/7</td>
<td>Ray Tracing, Animation</td>
<td>Ex. 10: Lighting and Shading</td>
</tr>
<tr>
<td>Dec. 12/14</td>
<td>Physically-based simulation, PDEs</td>
<td>Ex. 11: Rigid body dynamics</td>
</tr>
<tr>
<td>Dec. 19/21</td>
<td>Geometry Processing, Digital Fabrication</td>
<td>(none)</td>
</tr>
</tbody>
</table>
Advanced Courses

• **Computer Graphics**
  – Modeling, rendering, raytracing

• **Shape Modeling and Geometry Processing**
  – Geometric modeling, splines, meshes, processing

• **Game Programming Laboratory**
  – Design & development of a game in small groups

• **Image Synthesis**
  – Photorealistic rendering, light transport

• **Physically-based Simulation**
  – Animation, deformation, fracture, collision detection

• **Seminar: Advanced Topics in CG & Vision**
  – State-of-the-art research papers
That’s all for today...