Lecture 1:
Course Introduction:
Welcome to Computer Graphics!

Interactive Computer Graphics
Stanford CS248, Winter 2019
Hi!

Kayvon Fatahalian

Colin Dolese

Nikki Nikolenko

Katherine Sun

Elbert Lin
Discussion: Why study computer graphics?
What is computer graphics?

**computer graphics** /kəmˈpyʊdər ˈɡrɑːfiks/ n.
The use of computers to synthesize and manipulate visual information.
Humans are visual creatures!
Why *visual* information?

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

...eyes are highest-bandwidth port into the head!
Movies

Jurassic Park (1993)
Movies

The Matrix (1999)
Games

Uncharted 4 (Naughty Dog, 2016)

This image is rendered in real-time on a modern GPU
Games

Assassin’s Creed Origins (Ubisoft 2017)
Supercomputing for games

NVIDIA Titan X GPU
(~ 7 TFLOPs fp32)

Tesla generation NV chip ~ ASCI Red Supercomputer
Virtual reality experiences
Augmented reality

Microsoft Hololens augmented reality headset concept
Illustration

Indonesian cave painting (~38,000 BCE)
Digital illustration

Meike Hakkart
http://maquenda.deviantart.com/art/Lion-done-in-illustrator-327715059
Graphical user interfaces

Ivan Sutherland, “Sketchpad” (1963)

Doug Engelbart
Mouse
Modern graphical user interfaces

2D drawing and animation are ubiquitous in computing. Typography, icons, images, transitions, transparency, . . . (all rendered at high frame rate for rich experience)
Digital photography

NASA | Walter Iooss | Steve McCurry
Harold Edgerton | NASA | National Geographic
Ubiquitous imaging

Cameras everywhere
Imaging in mapping

Maps, satellite imagery, street-level imaging, ...
Computational cameras

Panoramic stitching, HDR photos, light field cameras, ...
Computer aided design

SolidWorks

SketchUp

For mechanical, architectural, electronic, optical, …
Product design and visualization

Ikea - 75% of catalog is rendered imagery
Architectural design

Bilbao Guggenheim, Frank Gehry
Visualization

Science, engineering, medicine, journalism, ...
Simulation

Driving simulator
Toyota Higashifuji Technical Center

da Vinci surgical robot
Intuitive Surgical

Flight simulator, driving simulator, surgical simulator, ...
Visual technology: 3D fabrication
Computer graphics is *everywhere!*
Foundations of computer graphics

- All these applications demand *sophisticated* theory and systems

- **Science and mathematics**
  - Physics of light, color, optics
  - Math of curves, surfaces, geometry, perspective, …
  - Sampling

- **Systems**
  - parallel, heterogeneous processing
  - input/output devices
  - graphics-specific programming systems

- **Art and psychology**
  - Perception: color, stereo, motion, image quality, …
  - Art and design: composition, form, lighting, …
ACTIVITY: modeling and drawing a cube

- Goal: generate a realistic drawing of a cube
- Key questions:
  - **Modeling**: how do we describe the cube?
  - **Rendering**: how do we then visualize this model?
ACTIVITY: modeling the cube

- Suppose our cube is...
  - centered at the origin (0,0,0)
  - has dimensions 2 x 2 x 2

QUESTION: 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)

QUESTION: What about the edges?

AB, CD, EF, GH,
AC, BD, EG, FH,
AE, CG, BF, DH
ACTIVITY: drawing the cube

Now have a digital description of the cube:

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

EDGES

AB, CD, EF, GH,
AC, BD, EG, FH,
AE, CG, BF, DH

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

Basic strategy:

1. Project 3D vertices to 2D points in the image
2. Connect 2D points with straight lines

...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 exactly does a point $p = (x, y, z)$ end up on the image?
- Let’s call the image point $q = (u, v)$
Perspective projection: side view

- Where exactly does a point \( p = (x, y, z) \) end up on the image?
- Let’s call the image point \( q = (u, v) \)
- Notice two similar triangles:

Assume camera has unit size, coordinates relative to pinhole \( c \)

- Then \( v/1 = y/z \), i.e., vertical coordinate is just the slope \( y/z \)
- Likewise, horizontal coordinate is \( u = x/z \)
ACTIVITY: now draw image made by pinhole camera

- Need 12 volunteers
  - each person will draw one cube edge
  - assume camera is at point $c=(2,3,5)$
  - convert $(X,Y,Z)$ of both endpoints to $(u,v)$:
    1. subtract camera $c$ from vertex $(X,Y,Z)$ to get $(x,y,z)$
    2. divide $x$ and $y$ by $z$ to get $(u,v)$—write as a fraction
  - draw line between $(u_1,v_1)$ and $(u_2,v_2)$

<table>
<thead>
<tr>
<th>VERTICES</th>
<th></th>
<th>EDGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: ( 1, 1, 1 )</td>
<td>E: ( 1, 1, -1 )</td>
<td>AB, CD, EF, GH,</td>
</tr>
<tr>
<td>B: (-1, 1, 1 )</td>
<td>F: (-1, 1, -1 )</td>
<td>AC, BD, EG, FH,</td>
</tr>
<tr>
<td>C: ( 1,-1, 1 )</td>
<td>G: ( 1,-1,-1 )</td>
<td>AE, CG, BF, DH</td>
</tr>
<tr>
<td>D: (-1,-1, 1 )</td>
<td>H: (-1,-1,-1 )</td>
<td></td>
</tr>
</tbody>
</table>
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: \((1/2, 2/3)\)

* keep in mind, this image is mirrored since it is a pinhole projection. Mirror the result and you get…
But wait…

How do we draw lines on a computer?
Output for a raster display

- Common abstraction of a raster display:
  - Image represented as a 2D grid of “pixels” (picture elements) **
  - Each pixel can take on a unique color value

** Kayvon will strongly challenge this notion of a pixel “as a little square” next class. But let’s go with it for now. ;-)
Close up photo of pixels on a modern display
What pixels should we color in to depict a line?

“Rasterization”: process of converting a continuous object to a discrete representation on a raster grid (pixel grid)
What pixels should we color in to depict a line?

Light up all pixels intersected by the line?
What pixels should we color in to depict a line?

Diamond rule (used by modern GPUs):
light up pixel if line passes through associated diamond
What pixels should we color in to depict a line?

Is there a right answer?
(consider a drawing a “line” with thickness)
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! (e.g., seek algorithm that does work proportional to number of pixels in the drawing of the line)
Incremental line rasterization

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

```plaintext
v = v1;
for( u=u1; u<=u2; u++ )
{
    v += s;
    draw( u, round(v) )
}
```

Common optimization: rewrite algorithm to use only integer arithmetic (Bresenham algorithm)
Line drawing of cube

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: \((1/2, 2/3)\)

* keep in mind, this image is mirrored since we simulated the result of pinhole projection.
We just rendered a simple line drawing of a cube.

But to render more realistic pictures (or animations) we need a much richer model of the world.

surfaces
motion
materials
lights
cameras
2D shapes

[Source: Batra 2015]
Complex 3D surfaces

(Stanislav Orekhov)

Platonic noid

[Kaldor 2008]
Modeling material properties

[Wann Jensen 2001]

[Zhao 2013]

[Jakob 2014]
Realistic lighting environments

Wall-E, (Pixar 2008)
Animation: modeling motion

https://www.youtube.com/watch?v=6G3O60o5U7w
Physically-based simulation of motion

https://www.youtube.com/watch?v=tT81VPk_ukU

[James 2004]
Course Logistics
About this course

- A broad overview of major topics and techniques in interactive computer graphics: geometry, rendering, animation, imaging

- Learn by implementing:
  - Focus on implementing fundamental data structures and algorithms that are reused across all areas of graphics
Getting started

- Sign up for an account on the course web site

- Sign up for the course on Piazza

- There is no textbook for this course, but please see the course website for references (there are some excellent graphics textbooks)
Course programming assignments

1. 2D drawing (2 weeks)

2. Geometry editing (2 weeks)

3. Materials and lighting in a 3D renderer (2 weeks)

4. Self-selected project: extend existing project, take on optional animation project, choose your own (4 weeks)
Assignments / Grading

- (45%) Three programming assignments
  - Each worth 15% of grade
  - In teams of up to two

- (25%) Self-selected final project
  - Extend an earlier assignment, or do your own thing!

- (25%) Midterm / final
  - Both cover cumulative material seen so far

- (5%) Class participation
  - In-class/website comments, other contributions to class
The course web site

We have no textbook for this class and so the lecture slides and instructor/TA/student discussions on the web are the primary course reference.

Perspective projection

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

Slide comments and discussion

"Add private note" button: You can add notes to yourself about this slide here.
Thought question for next time:
What does it mean for a pixel to be covered by a triangle?

Question: which triangles “cover” this pixel?
See you next time!

Next time, we’ll talk about drawing a triangle
- And it’s a lot more interesting than it might seem . . .
- Also, what’s up with these “jagged” lines?

Slide acknowledgements:
Thanks to Keenan Crane and Ren Ng