Lecture 1:

Course Introduction:
Welcome to Computer Graphics!

Interactive Computer Graphics
Stanford CS248, Winter 2022
Hi!

Josephine the (Graphics) Cat

Kayvon Fatahalian

Doug James

Yanjun Chen

Haotian Zhang

James Hong
Discussion:
Why study computer graphics?
Why generate *visual* information?

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

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

Jurassic Park (1993)
Computer games

Uncharted 4 (Naughty Dog, 2016)

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

Assassin's Creed Odyssey (Ubisoft 2018)
Computer games

Screenshot: Red Dead Redemption
Supercomputing for games

NVIDIA Titan V GPU
(~ 15 TFLOPs fp32)

Specialized processors for performing graphics computations.
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)
Ubiquitous imaging

Cameras everywhere
Computational cameras

Panoramic stitching

High dynamic range (HDR) photography

Portait mode
(simulate effects of large aperture lens)
Imaging for mapping

Maps, satellite imagery, street-level imaging,...
Computer aided design

SolidWorks

SketchUp

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

Ikea - 75% of catalog is rendered imagery (several years ago... likely a lot more now)
Architectural design
Data visualization

Science, engineering, medicine, journalism, …
Simulation

Driving simulator
Toyota Higashifuji Technical Center

da Vinci surgical robot
Intuitive Surgical

Flight simulator, driving simulator, surgical simulator, ...
Graphics used for training ML models

**AI Habitat:**
simulator for training AI agents

**Carla:**
autonomous driving simulator

AI Habitat enables training of embodied AI agents (virtual robots) in a virtual environment before transferring the learned skills to reality. This empowers a parallel pipeline of real-world datasets (e.g., ImageNet, COCO, VQA) to embodied AI where agents actively perceive, act, and learn from interactions in this environment.

Why the name Habitat? Because that’s where AI agents live.

Habitat is a platform for embodied AI research that consists of Habitat, Habitat-Sim, Habitat-Render, and Habitat-EZ.

Habitat-Sim
A flexible, high-performance 3D simulator with configurable agents and environment handling (with built-in support for MatterPort3D, Gibson, Replica, and more). The Matterport3D dataset Habitat-Sim enables research on embodied AI.

Stanford CS248, Winter 2022
3D fabrication
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
  - Graphics-specific programming systems
  - Input/output devices

- 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 geometry 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 )

**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?
Perspective projection

- Objects look smaller as they get further away ("perspective")
- Why does this happen?
- Consider simple ("pinhole") model of a camera:
For those that didn’t do this in grade school

Pin hole

Place photosensitive paper here

http://janneinosaka.blogspot.com/2010/03/pinhole-time.html
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 \( v/1 = y/z \ldots v = y/z \)
- Likewise, horizontal offset \( u = x/z \)
ACTIVITY: now draw image made by pinhole camera

- Need 6 volunteers
  - I’ll do A and B, each volunteer will draw one vertex
  - Assume camera is at point c=(2,3,5)
  - Convert (X,Y,Z) of both endpoints of edge 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
  - Then I will draw a line between (u1,v1) and (u2,v2) for all edges

### 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)

### EDGES
- AB, CD, EF, GH, AC, BD, EG, FH, AE, CG, BF, DH
Render a cube!

- Assume camera is at point c=(2,3,5)
- Convert (X,Y,Z) of both endpoints of edge 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)
- Draw line between (u1,v1) and (u2,v2)

**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 )

**EDGES**

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

**Projected coordinates:**

A: (1/4, 1/2)
B: (3/4, 1/2)
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 about the origin \((0,0)\) and you get…
How did we do?
But wait…

How do we draw lines on a computer?
CNC sharpie drawing machine ;-)  

Oscilloscope
Cathode ray tube

[Credit: http://propagation.ece.gatech.edu/ECE3025/tutorials/CathodeRayTube/CRToverview.htm]
Oscilloscope art :-)  

https://www.youtube.com/watch?v=rtR63-ecUNo
Frame buffer: memory for a raster display

image = “2D array of colors”
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. ;-)}
Flat panel displays

Low-Res LCD Display

High resolution color LCD, OLED, ...
Close up photo of pixels on a modern display

Lorem
LCD screen pixels (closeup)

iPhone 6S

Galaxy S5
LCD screen

https://www.flexenable.com/blog/how-lcds-work/
LCD (liquid crystal display) pixel

- **Principle:** block or transmit light by twisting polarization

- **Illumination from backlight** (e.g. fluorescent or LED)

- **Intermediate intensity levels by partial twist**

[Image credit: H&B fig. 2-16]
DMD projection display

Array of micro-mirror pixels

DMD = Digital micro-mirror device
DMD projection display

Array of micro-mirror pixels

DMD = Digital micro-mirror device
What pixels should we color in to depict a line?

“Rasterization”: process of converting a continuous object (a line, a polygon, etc.) 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 painted when drawing 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)

\[
v = v_1;
\text{for( } u = u_1; u <= u_2; u++ \text{ } \{ \\
\quad v += s; \\
\quad \text{draw( } u, \text{ round}(v) \text{ )}
\}
\]

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

2D coordinates:

A: \((\frac{1}{4}, \frac{1}{2})\)
B: \((\frac{3}{4}, \frac{1}{2})\)
C: \((\frac{1}{4}, 1)\)
D: \((\frac{3}{4}, 1)\)
E: \((\frac{1}{6}, \frac{1}{3})\)
F: \((\frac{1}{2}, \frac{1}{3})\)
G: \((\frac{1}{6}, \frac{2}{3})\)
H: \((\frac{1}{2}, \frac{2}{3})\)

* 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

[Utah Teapot]

[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=6G3O60oSU7w

Luxo Jr. (Pixar 1986)
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
  - We expect that you can understand/write/debug C/C++ code
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 (~3 weeks)
Assignments / grading

- **(50%) Three programming assignments**
  - In teams of up to two students (yes, you can work alone if you wish)

- **(8%) Participation**
  - Weekly written exercises (think of these as possible exam problems)
    - Graded on effort/participation only

- **(22%) Self-selected final project**
  - Extend an earlier assignment, or do your own thing!

- **(20%) Exam**
  - Given in week 8 or week 9 of the course
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:

3D object

2D image

camera

Slide comments and discussion

Question: During class, Kayvon asked a question about why objects look smaller when they are viewed at a distance. I liked one of the arguments made because it appealed to the angle subtended by an object. Could someone elaborate on that here?
Thought question for next time:
What does it mean for a pixel to be covered by a triangle?

Question: which of these four 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