Lecture 1:

Course Intro:
Welcome to Computer Graphics!

Computer Graphics
CMU 15-462/15-662, Fall 2015

Hi!

Kayvon Fatahalian

Sky Gao

Bryce Summers

Keenan Crane

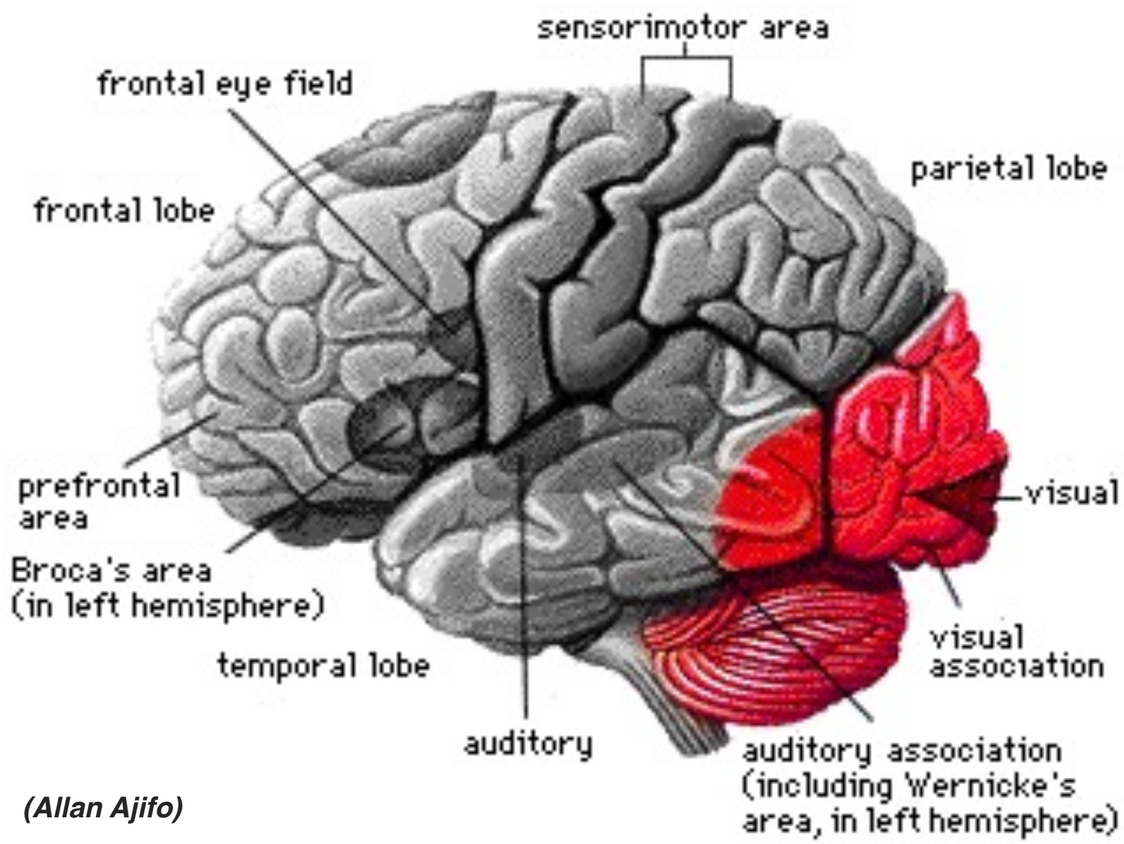
Michael Choquette

What is computer graphics?

com•put•er graph•ics /kəmˈpyo͞odər ˈgrafiks/ *n.*
The use of computers to synthesize and manipulate visual information.

Why *visual* information?

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



(Allan Ajifo)

(Petar Milošević)

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

Humans are visual creatures!

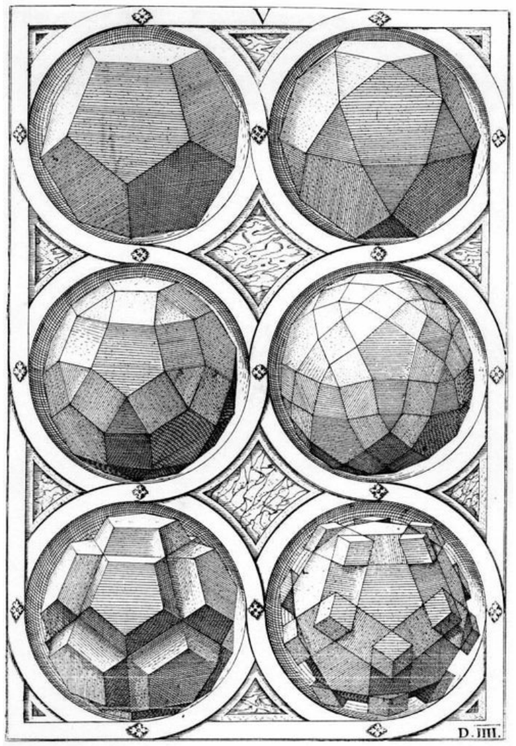
History of visual depiction

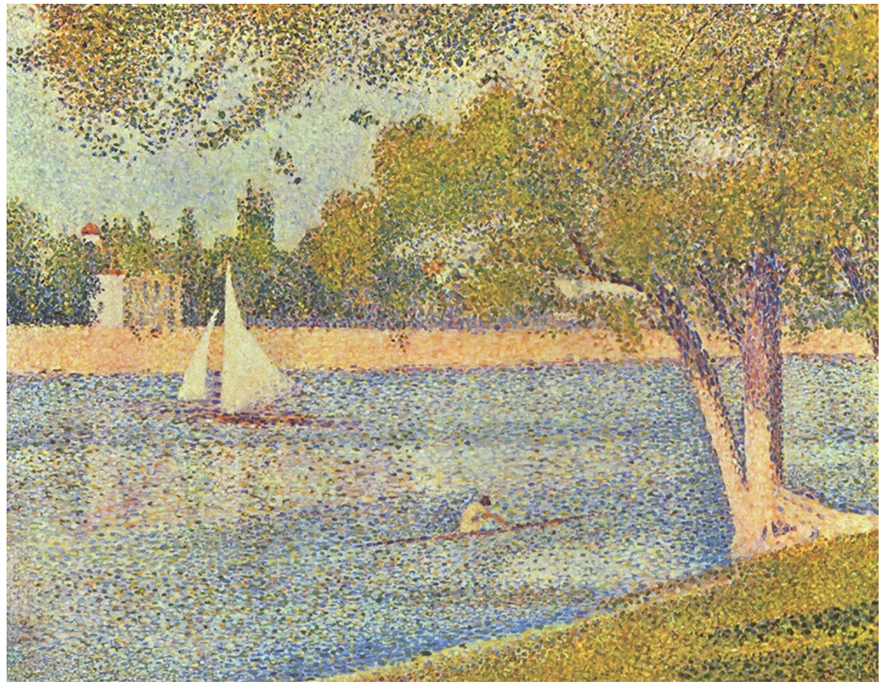
- **Humans have *always* been visual creatures!**

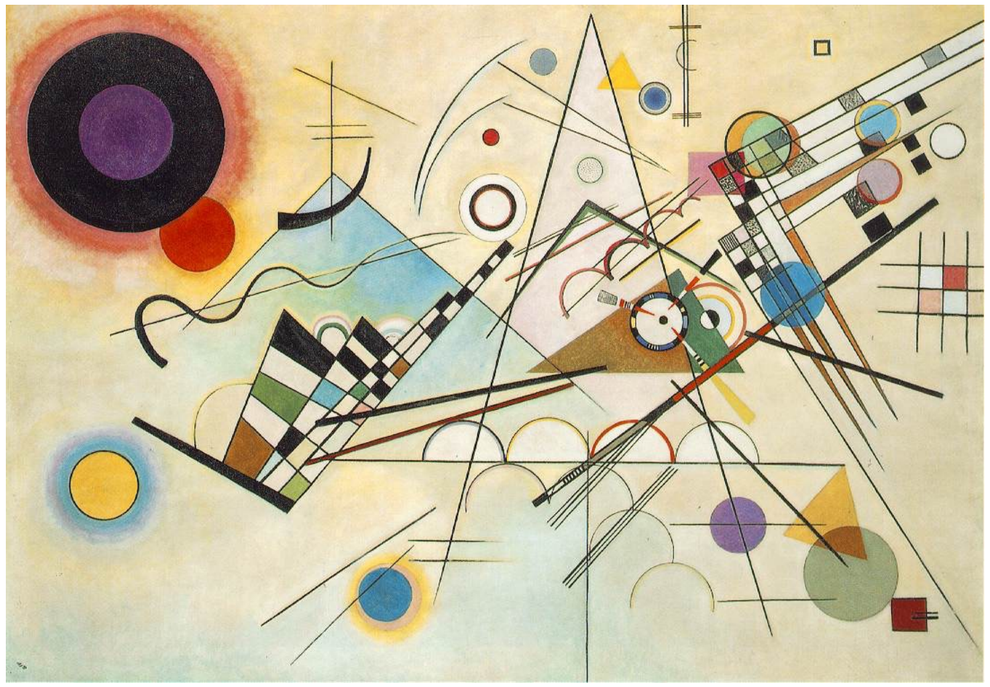
Indonesian cave painting (~38,000 BCE)

Visual technology: painting / illustration

- Not purely representational: ideas, feelings, data, ...







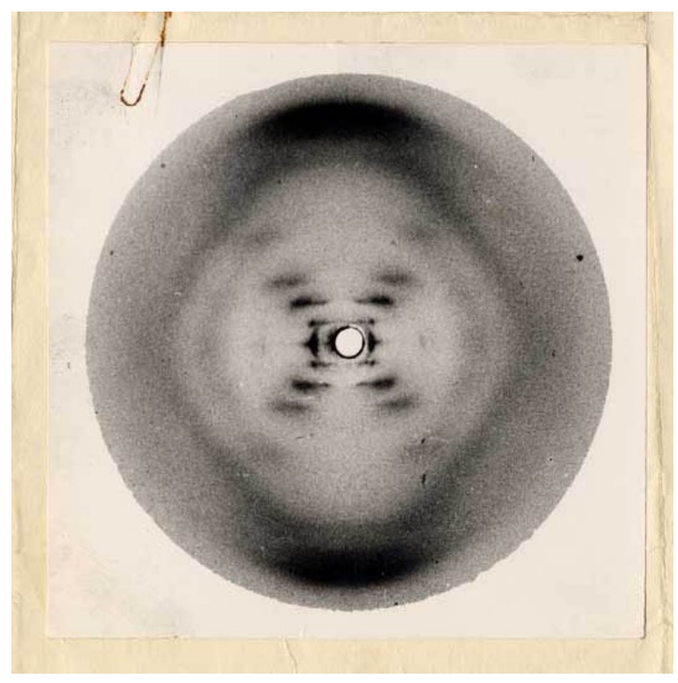
Visual technology: carving / sculpture

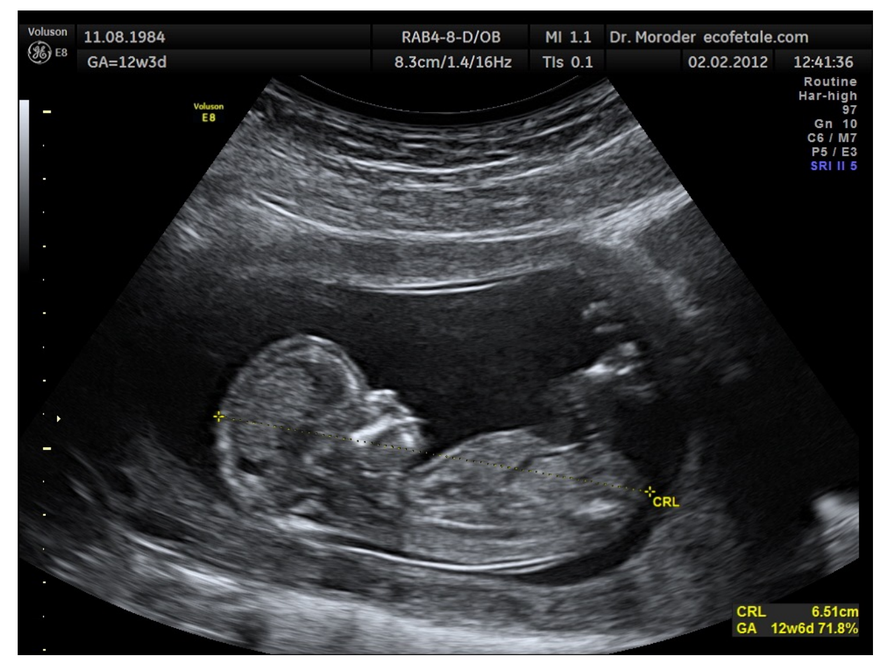
Visual technology: photography / imaging

- Processing of visual data no longer happening in the head!

Joseph Niépce, "View from the Window at Le Gras" (1826)

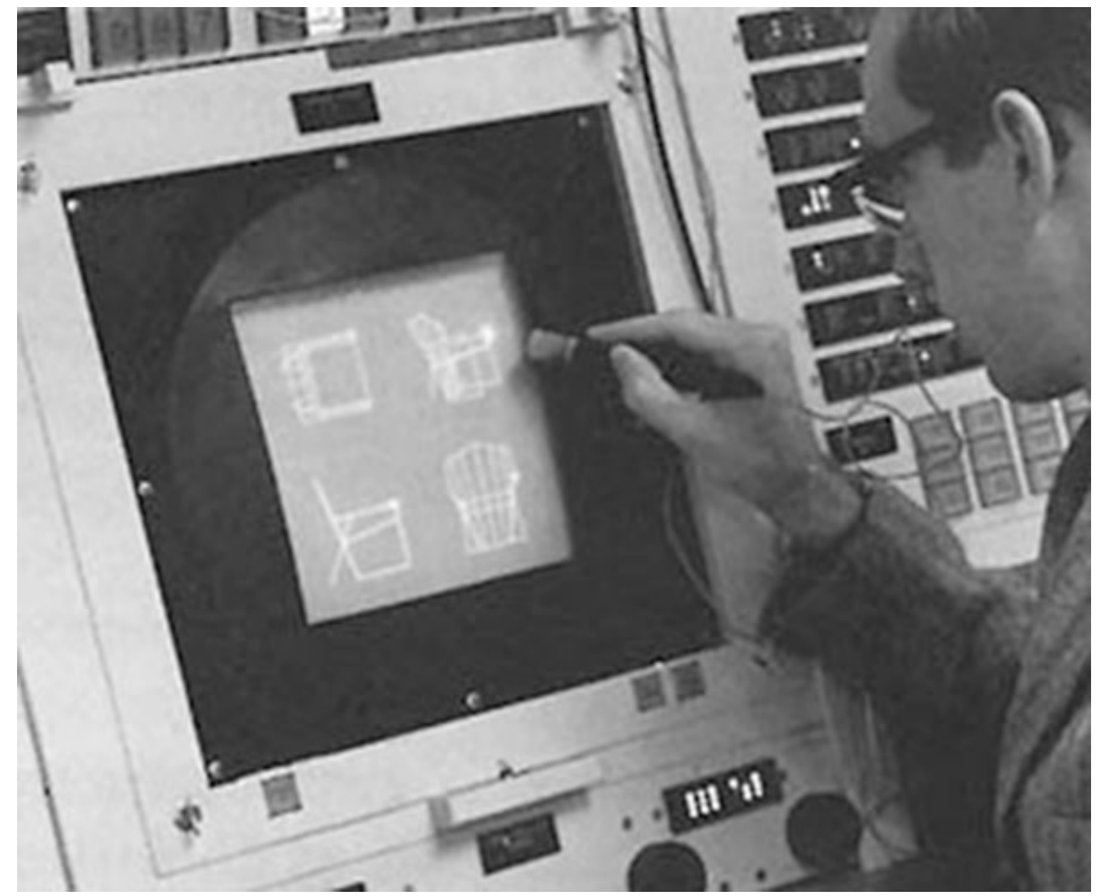
Visual technology: photography / imaging





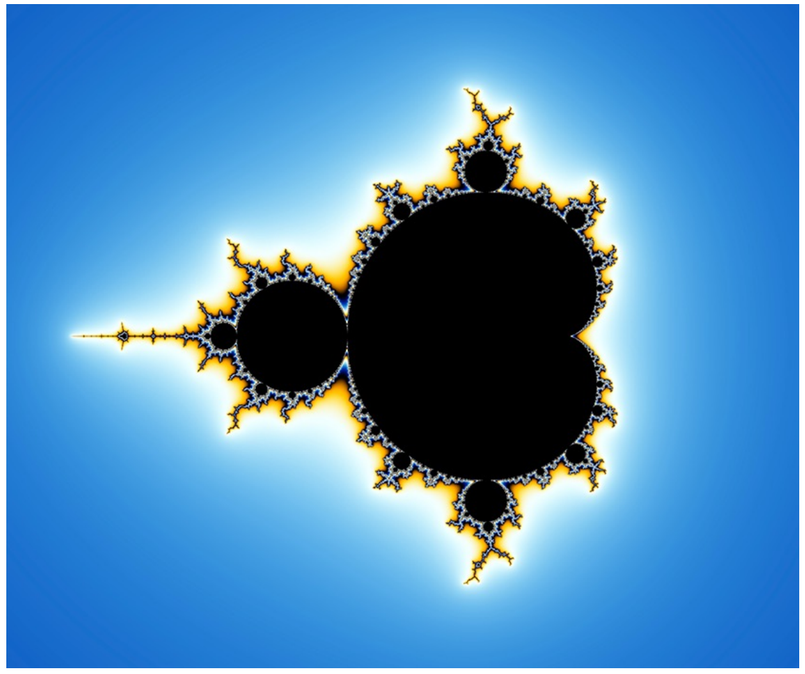
Visual technology: digital imagery

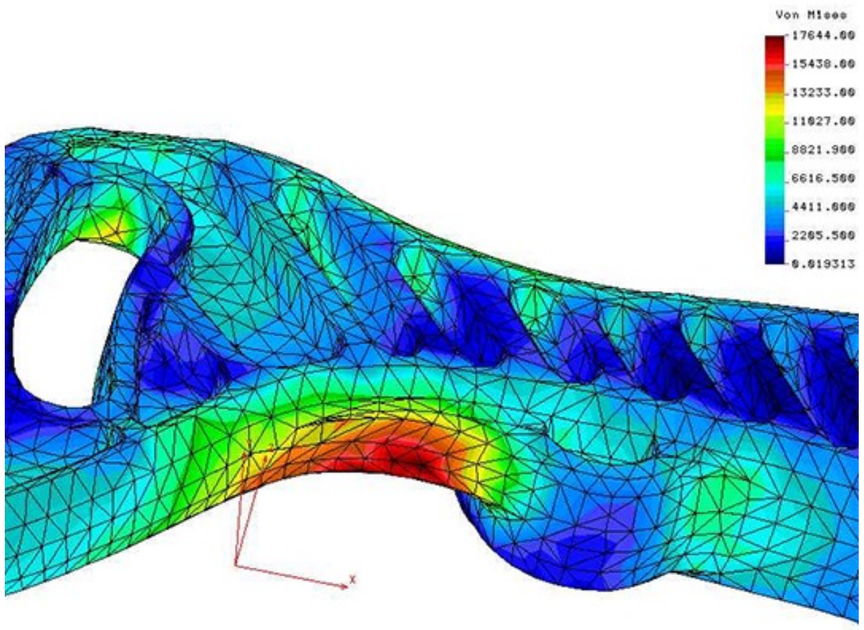
- **Intersection of visual depiction & computation**



Ivan Sutherland, "Sketchpad" (1963)

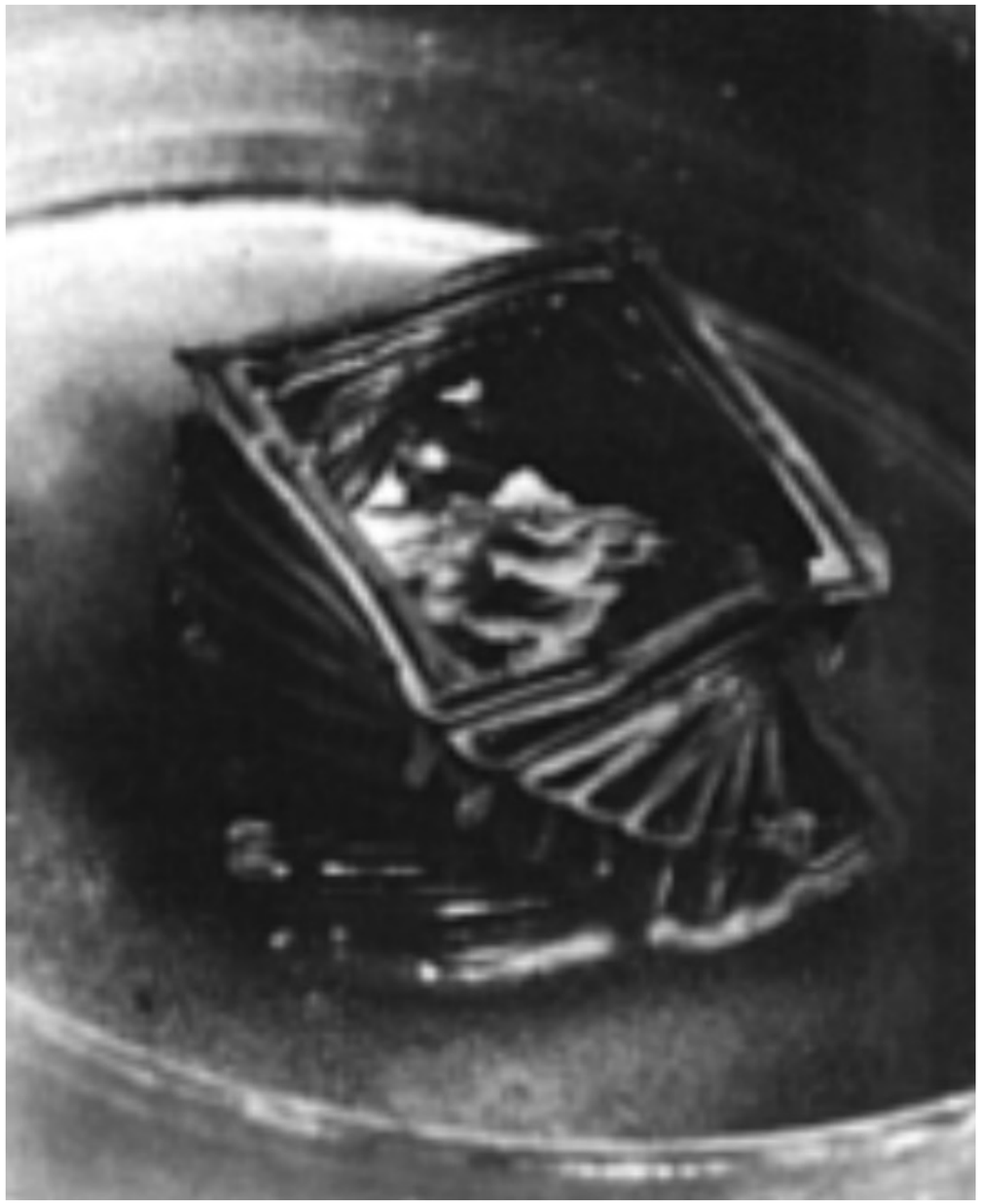
Visual technology: digital imagery

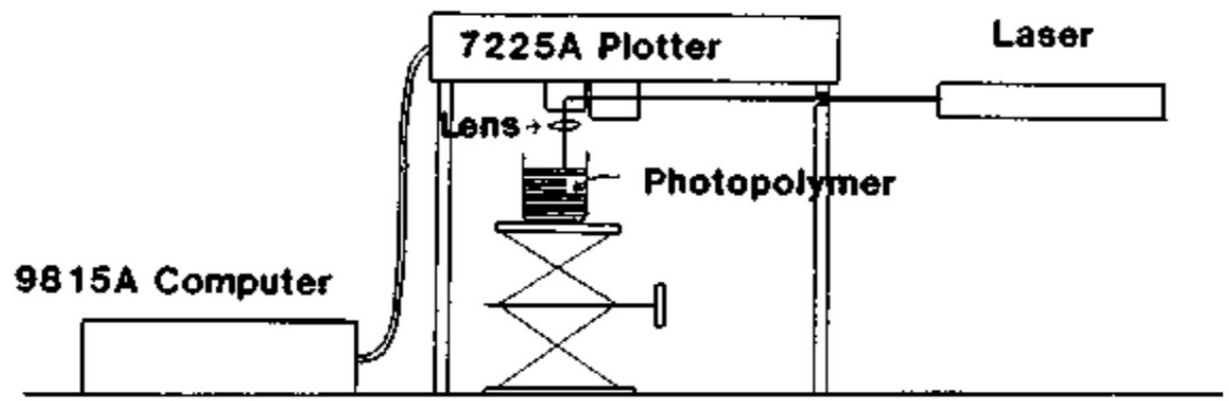




Visual technology: 3D fabrication

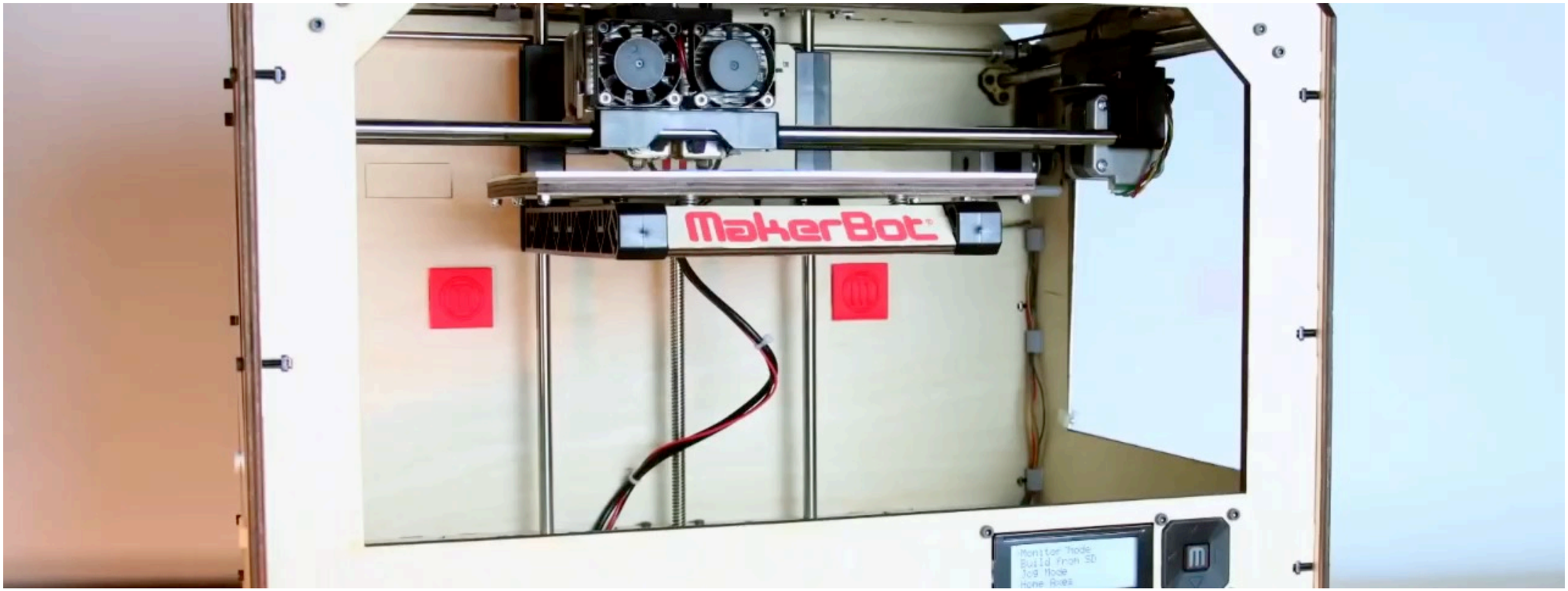
- Create *physical* realization of digital shape

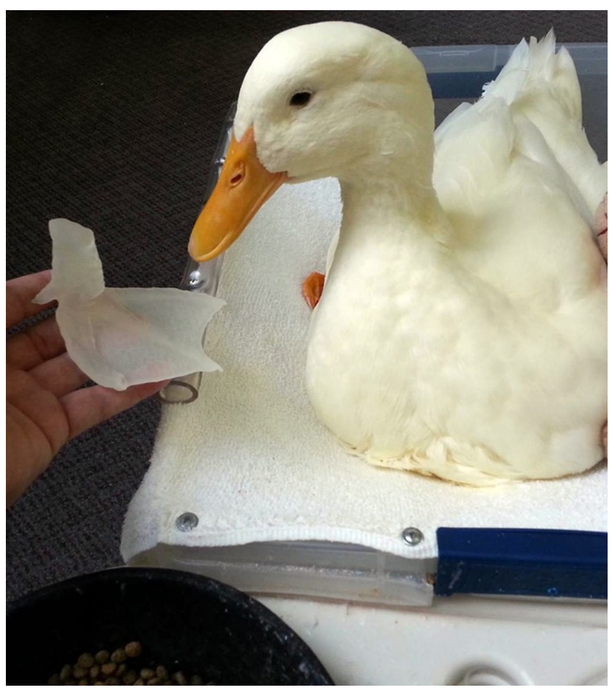




A.J. Herbert / 3M (1979)

Visual technology: 3D fabrication





Technologies for visual depiction

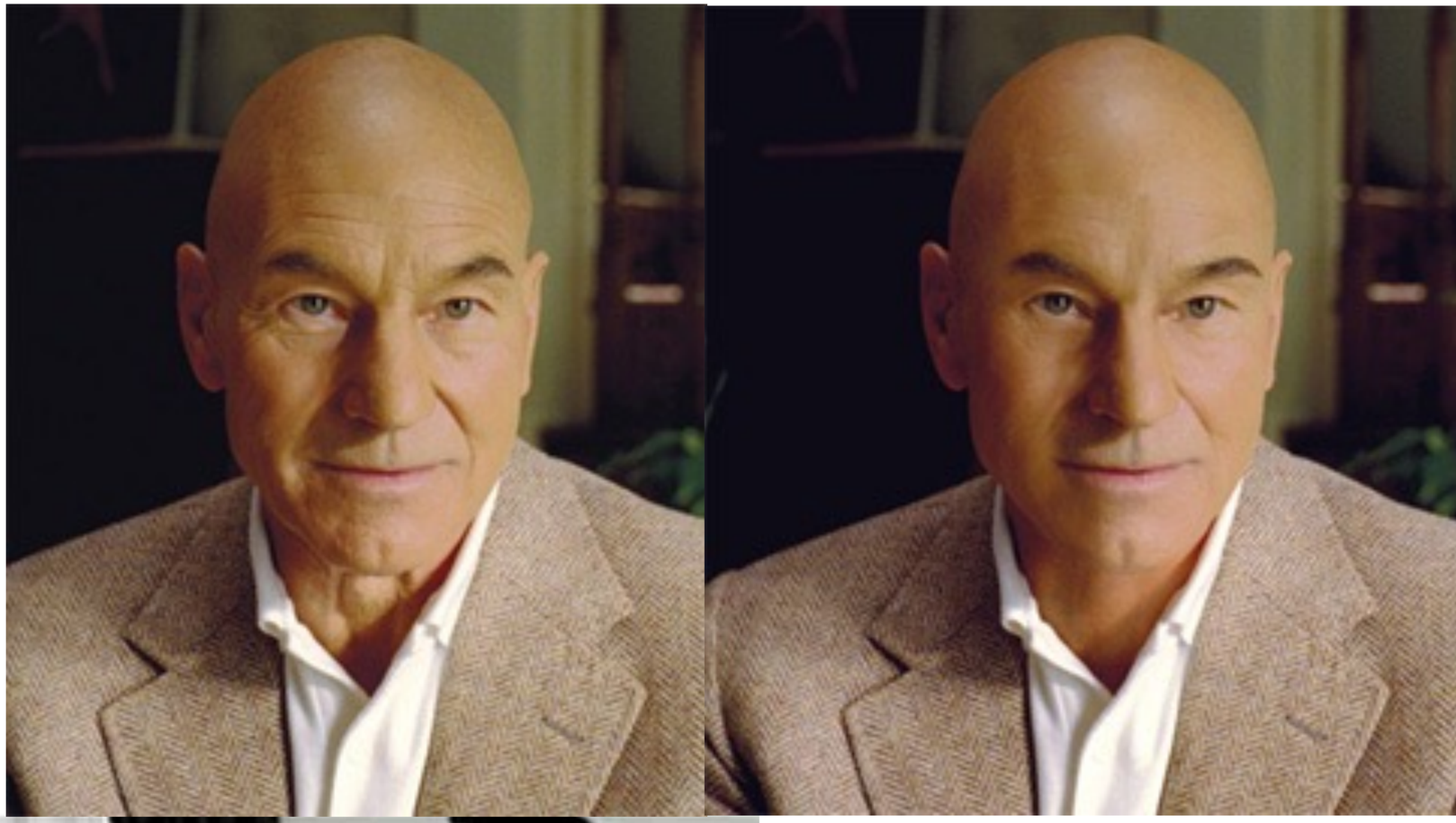
- Drawing/painting/illustration (~40,000 BCE)
- Sculpture (~40,000 BCE)
- Photography (~1826)
- *Digital Imagery (~1963)*
- *3D Fabrication (~1979)*

Computer graphics is *everywhere!*

Entertainment (movies, games)

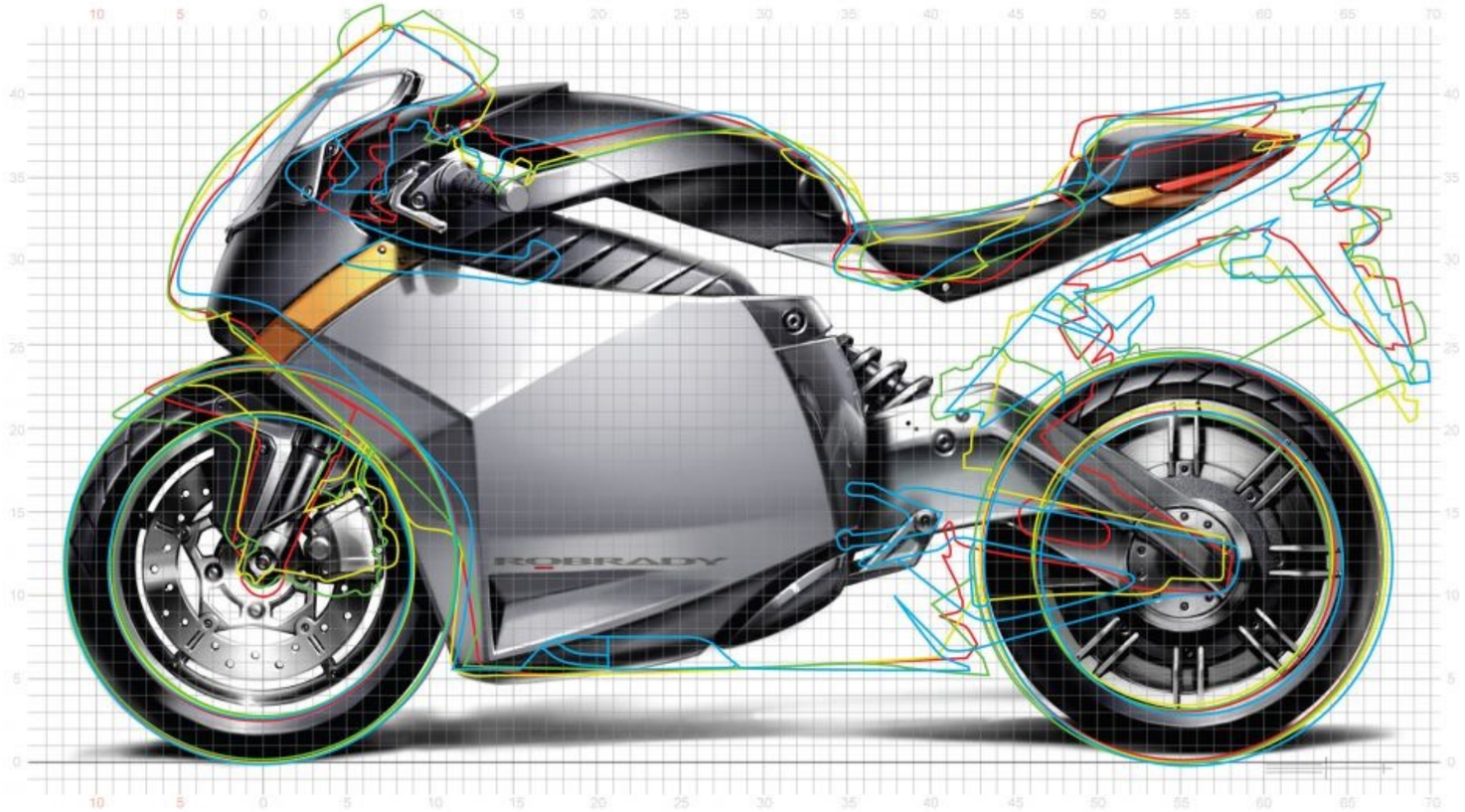
Entertainment

- Not just cartoons!

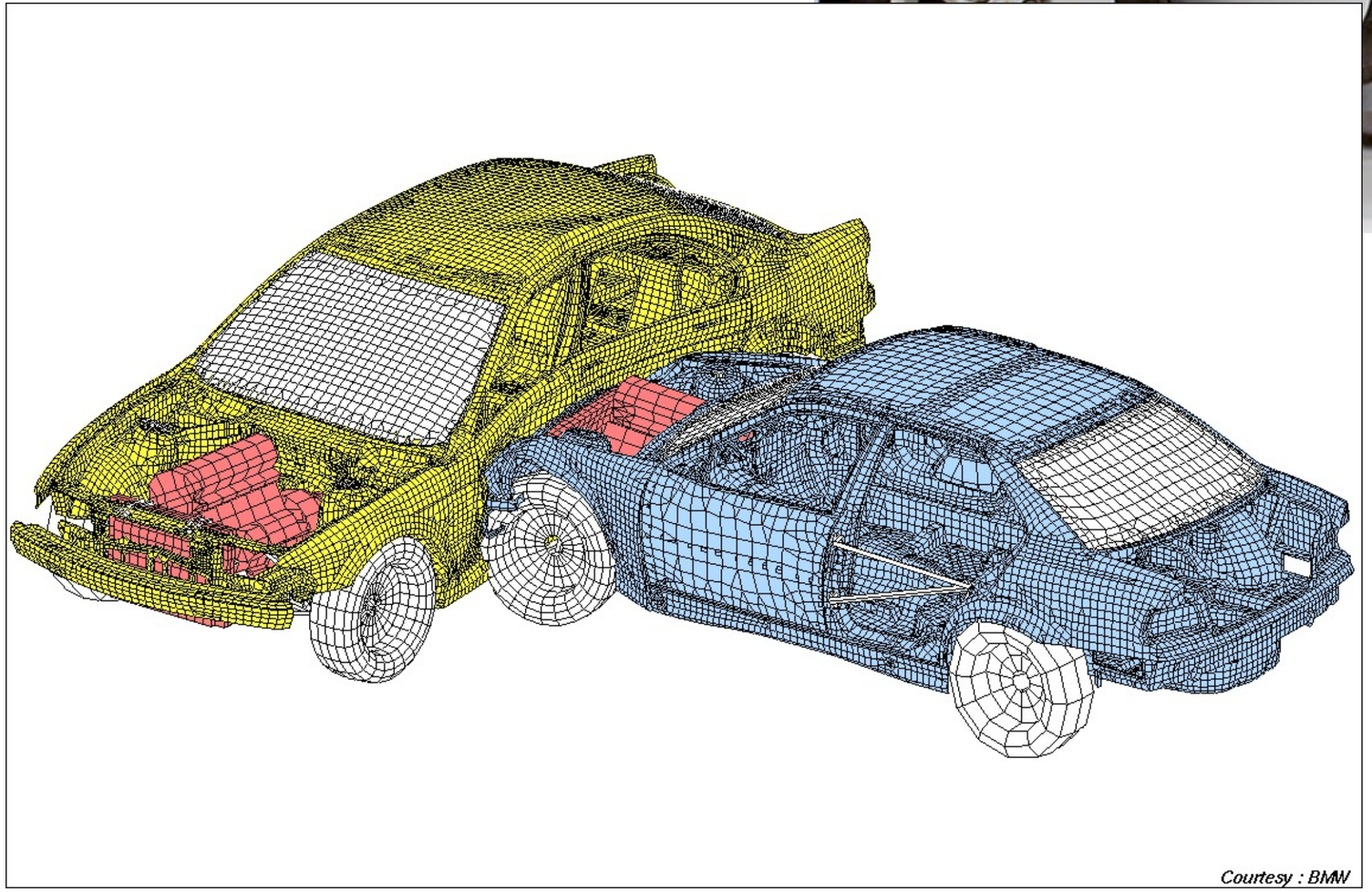


Art and design

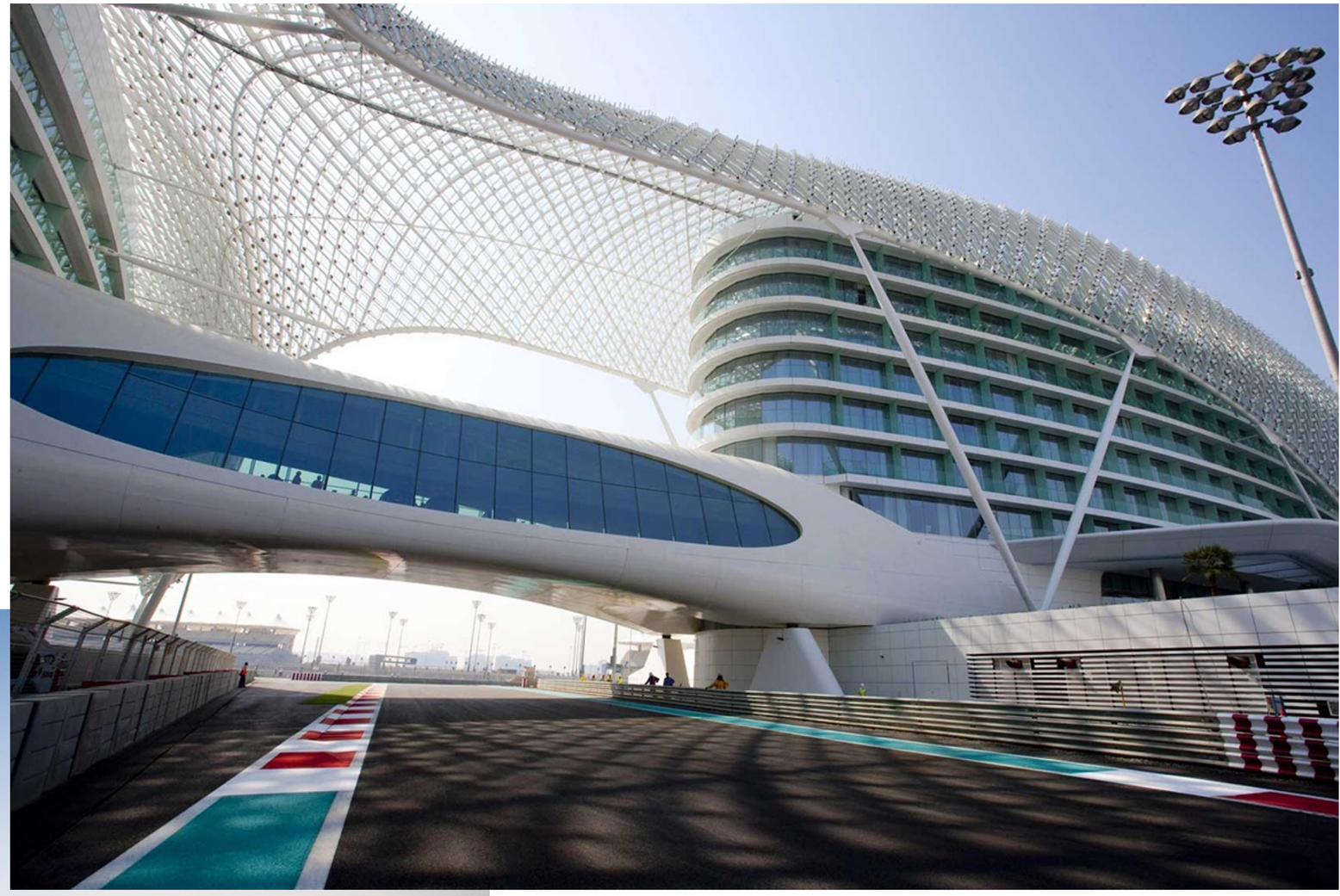
Industrial design

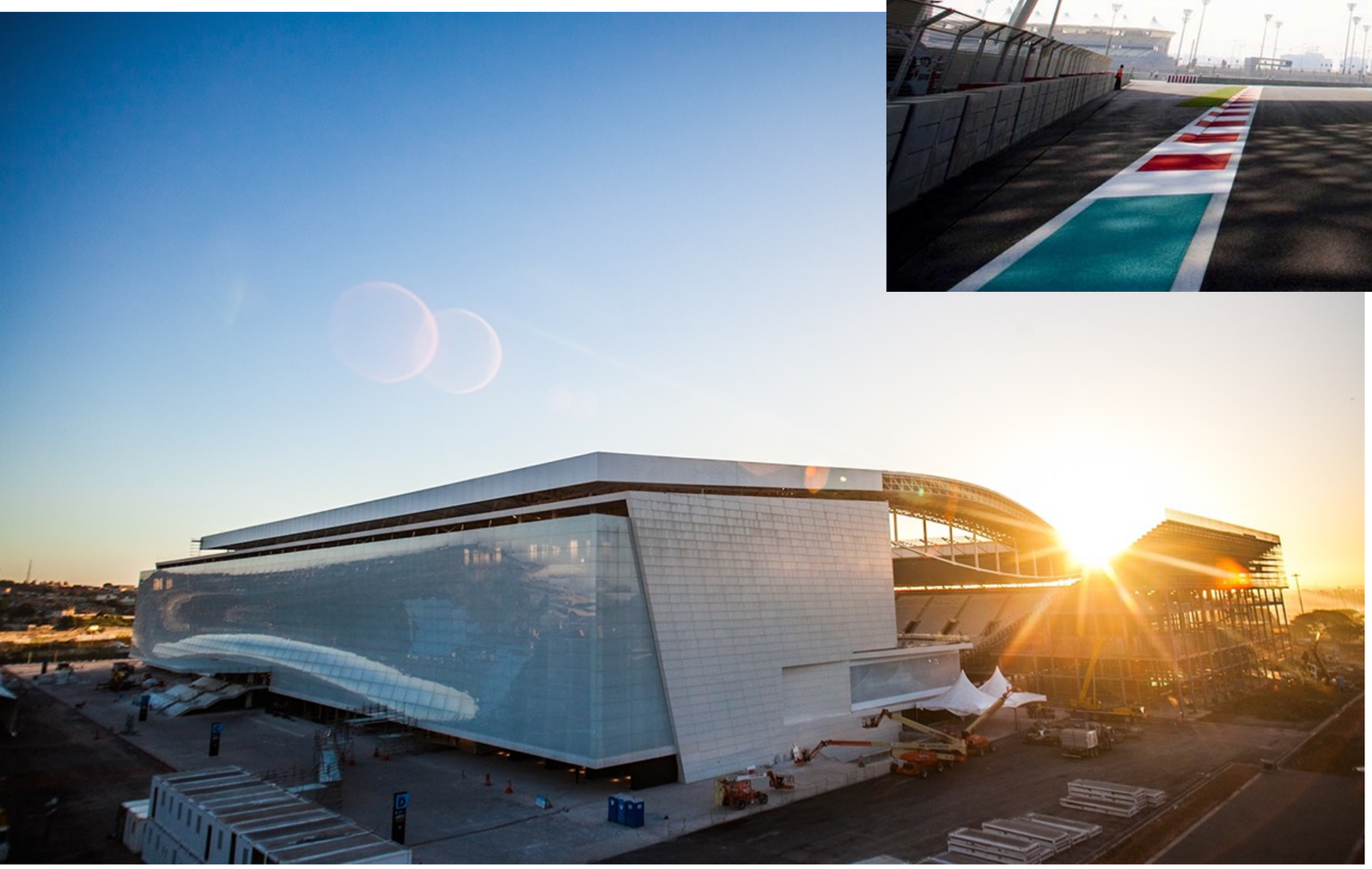


Computer aided engineering (CAE)

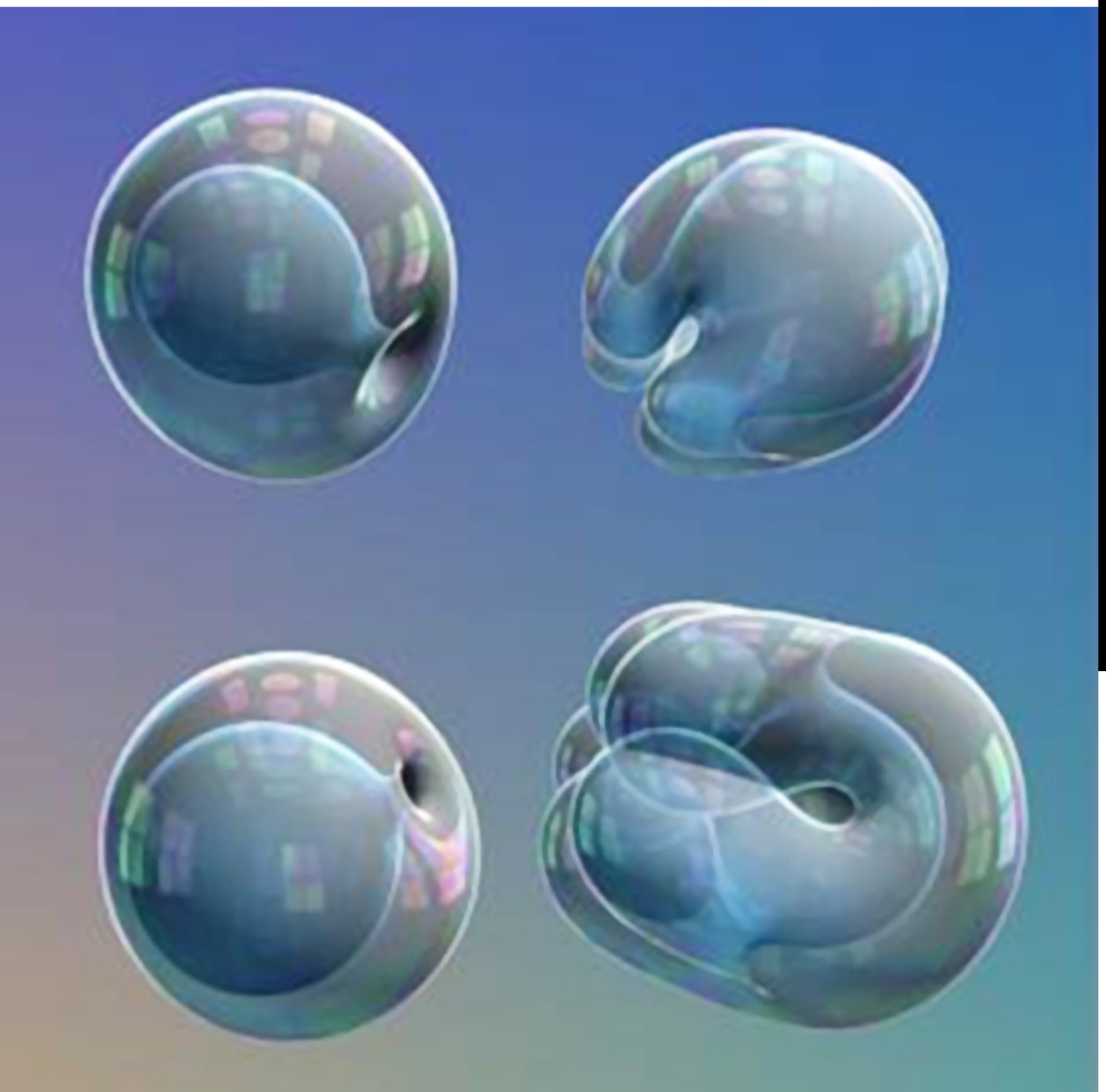


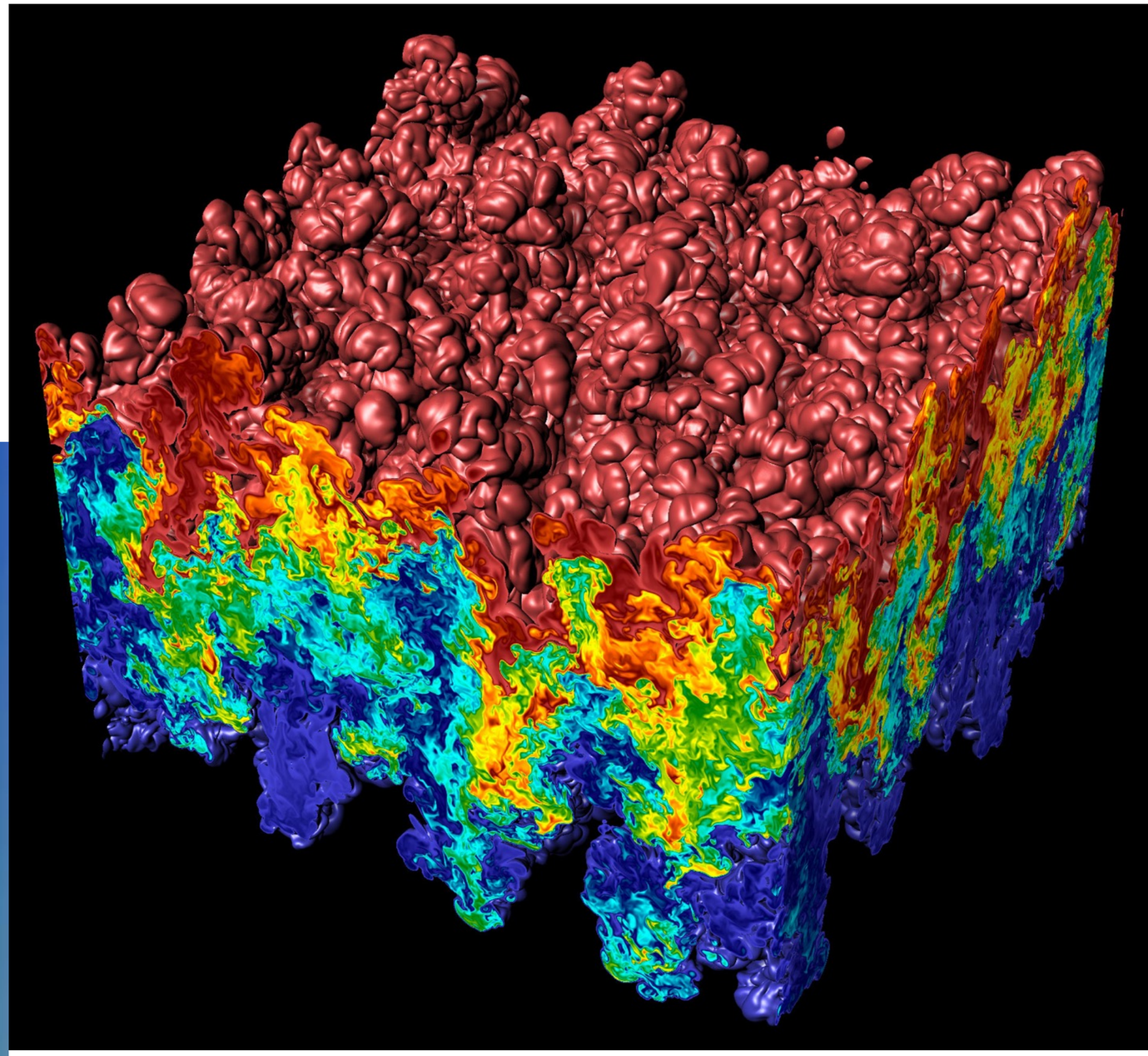
Architecture



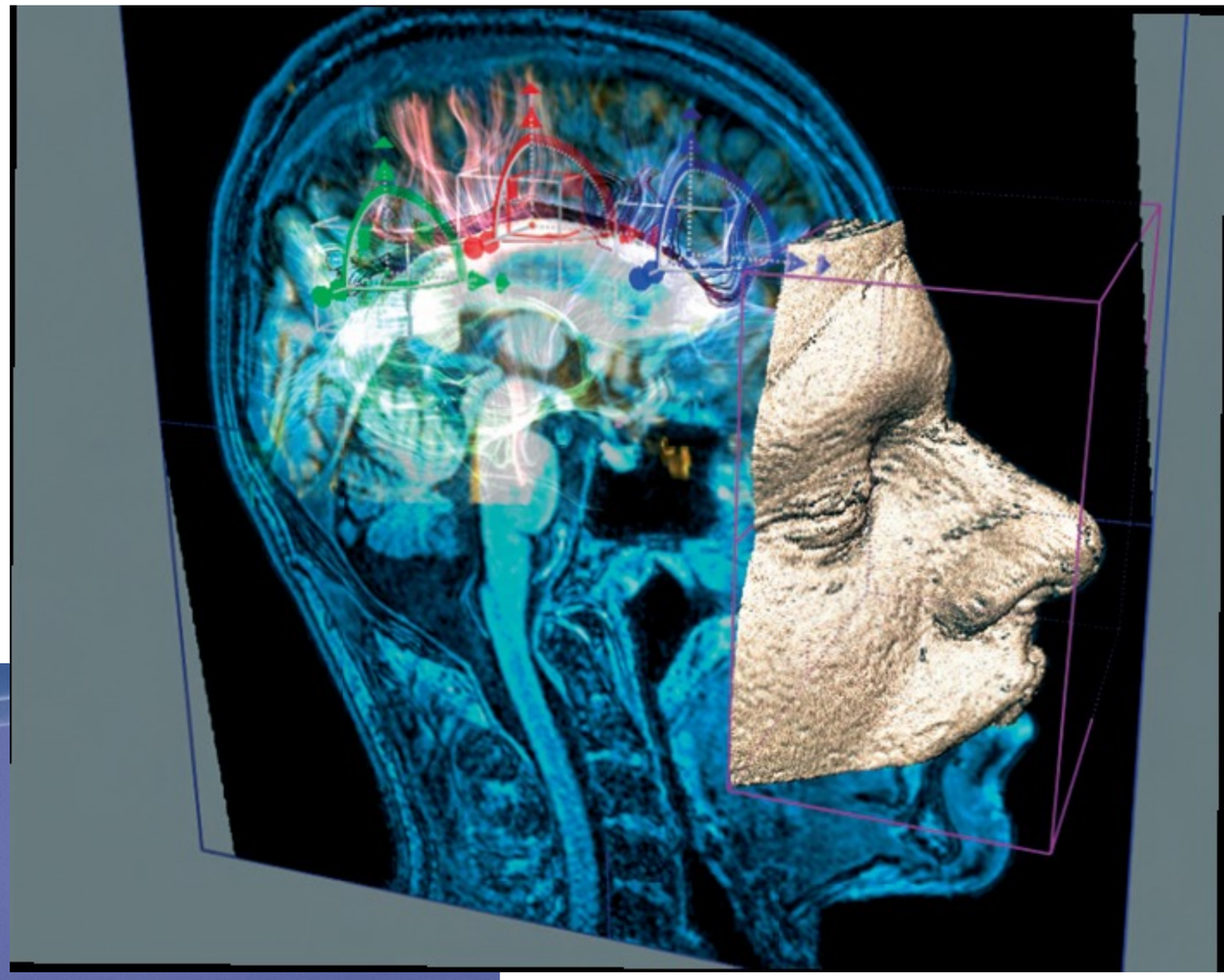


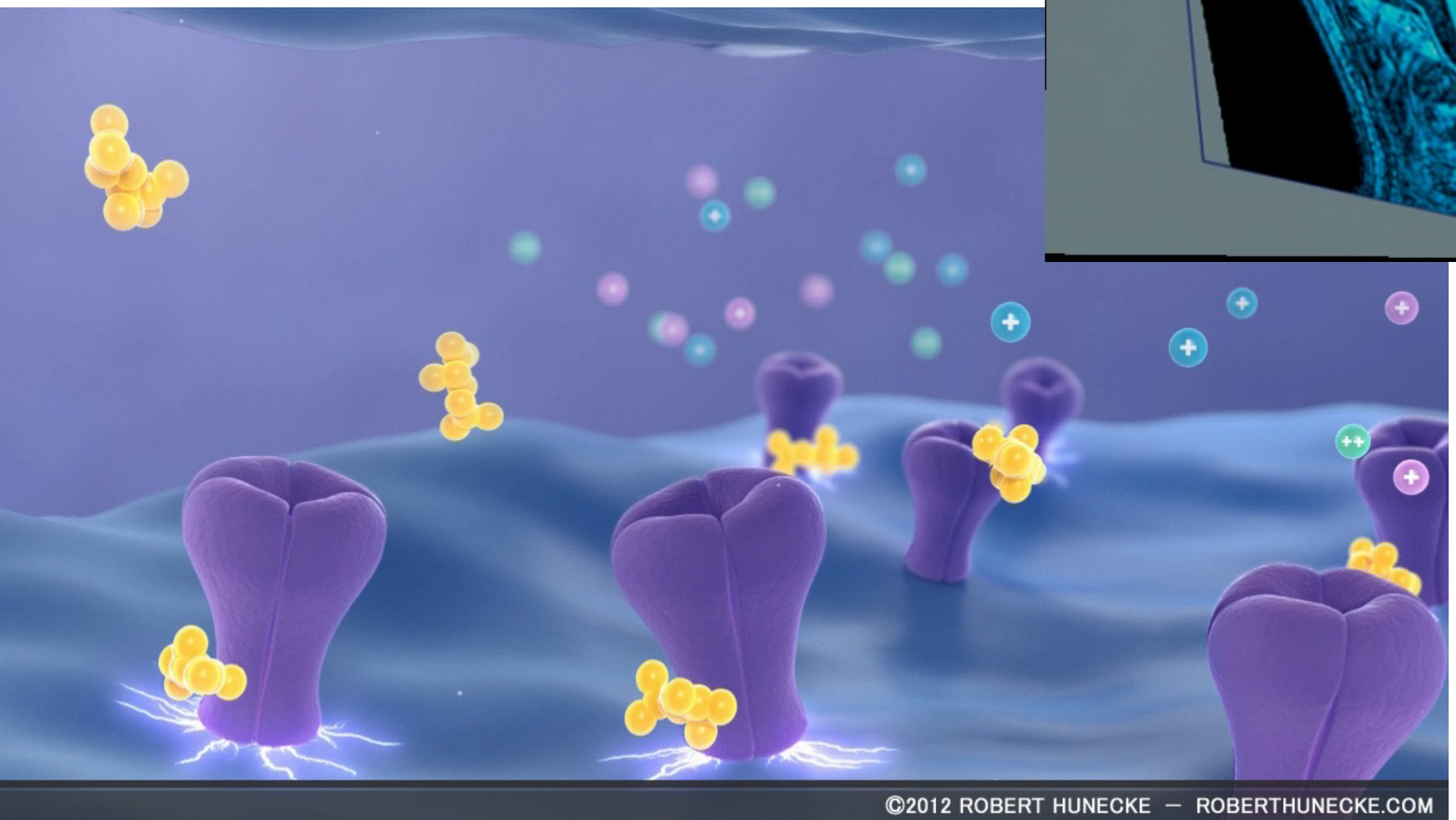
Scientific/mathematical visualization



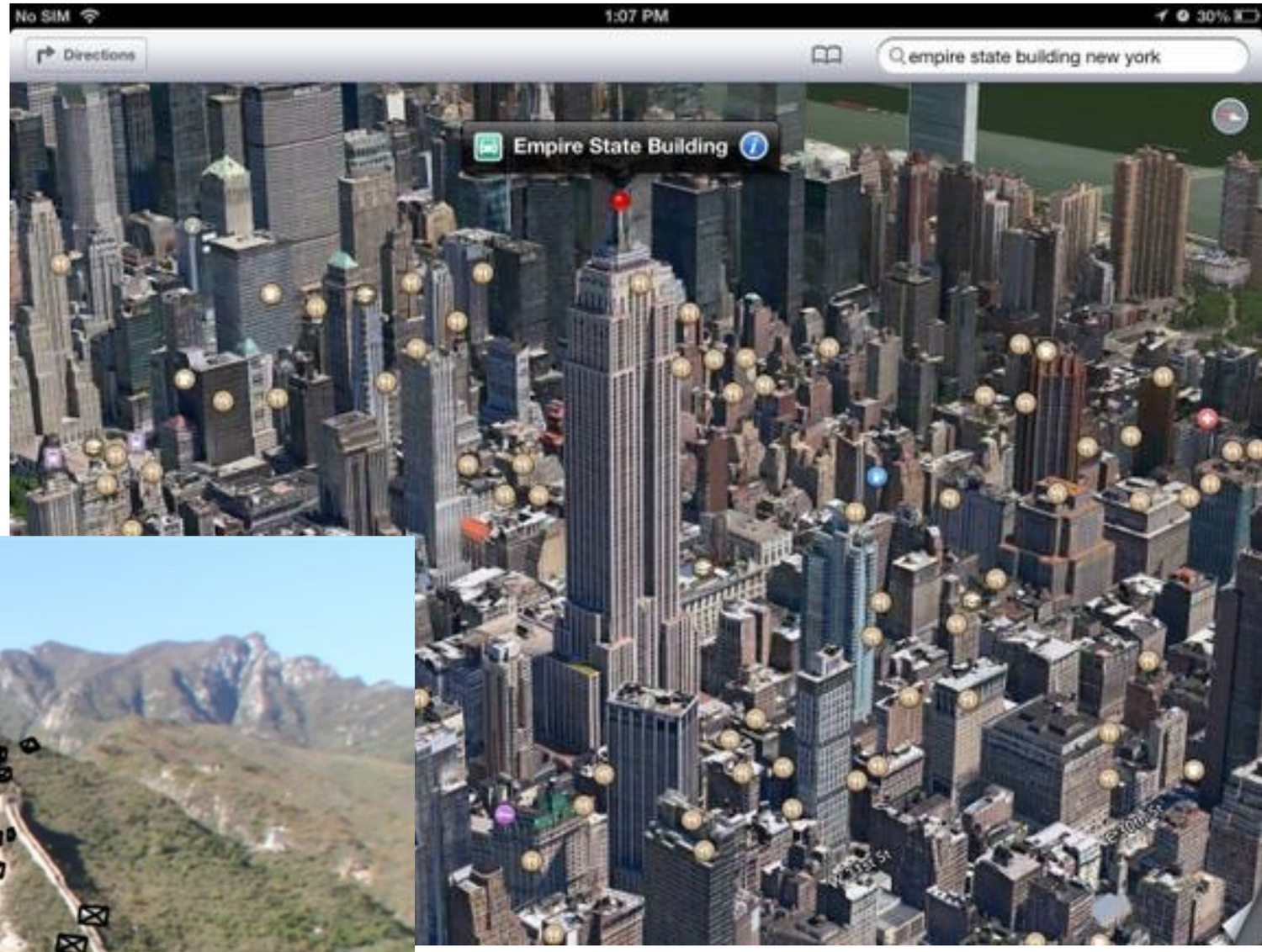


Medical/anatomical visualization

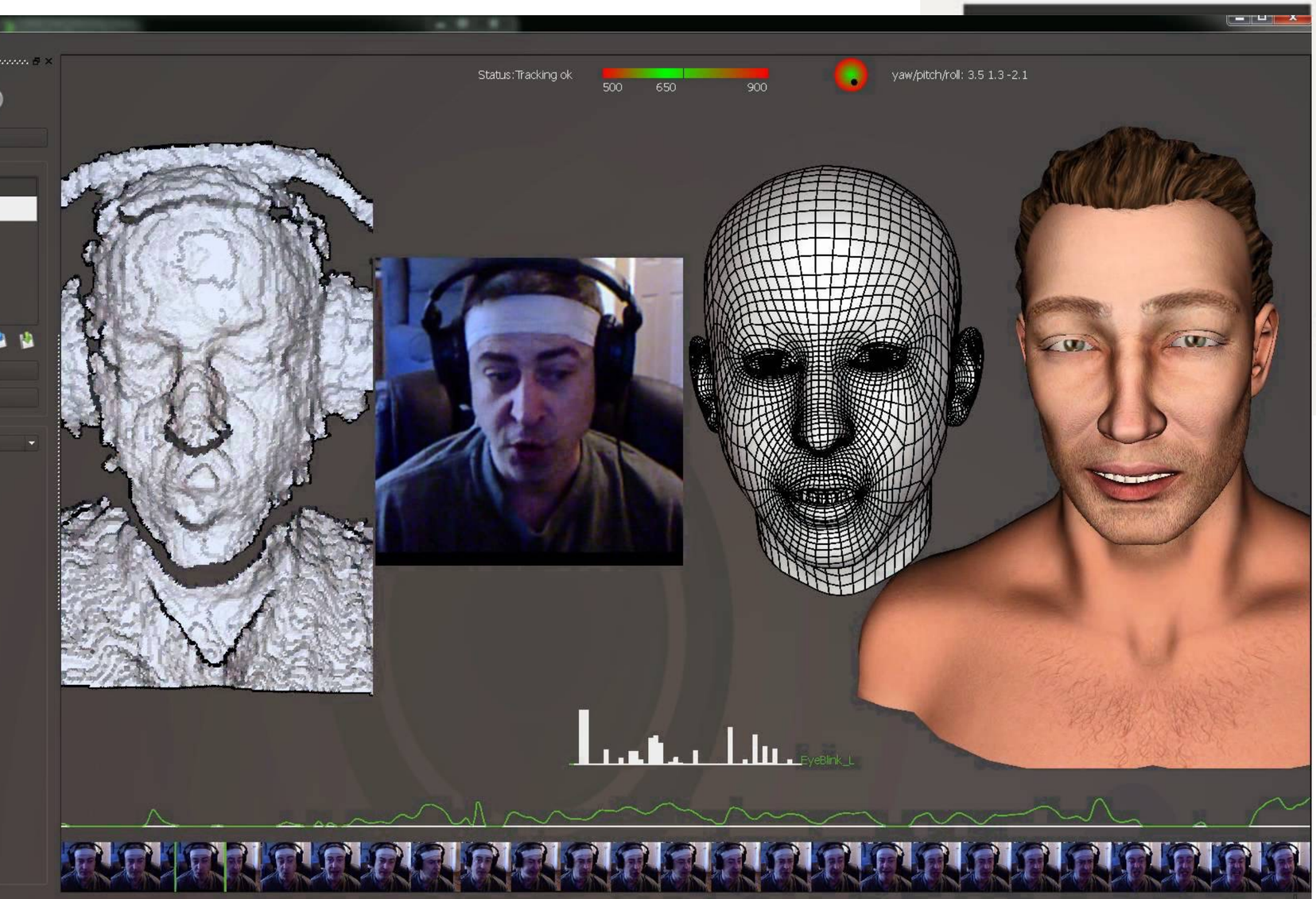


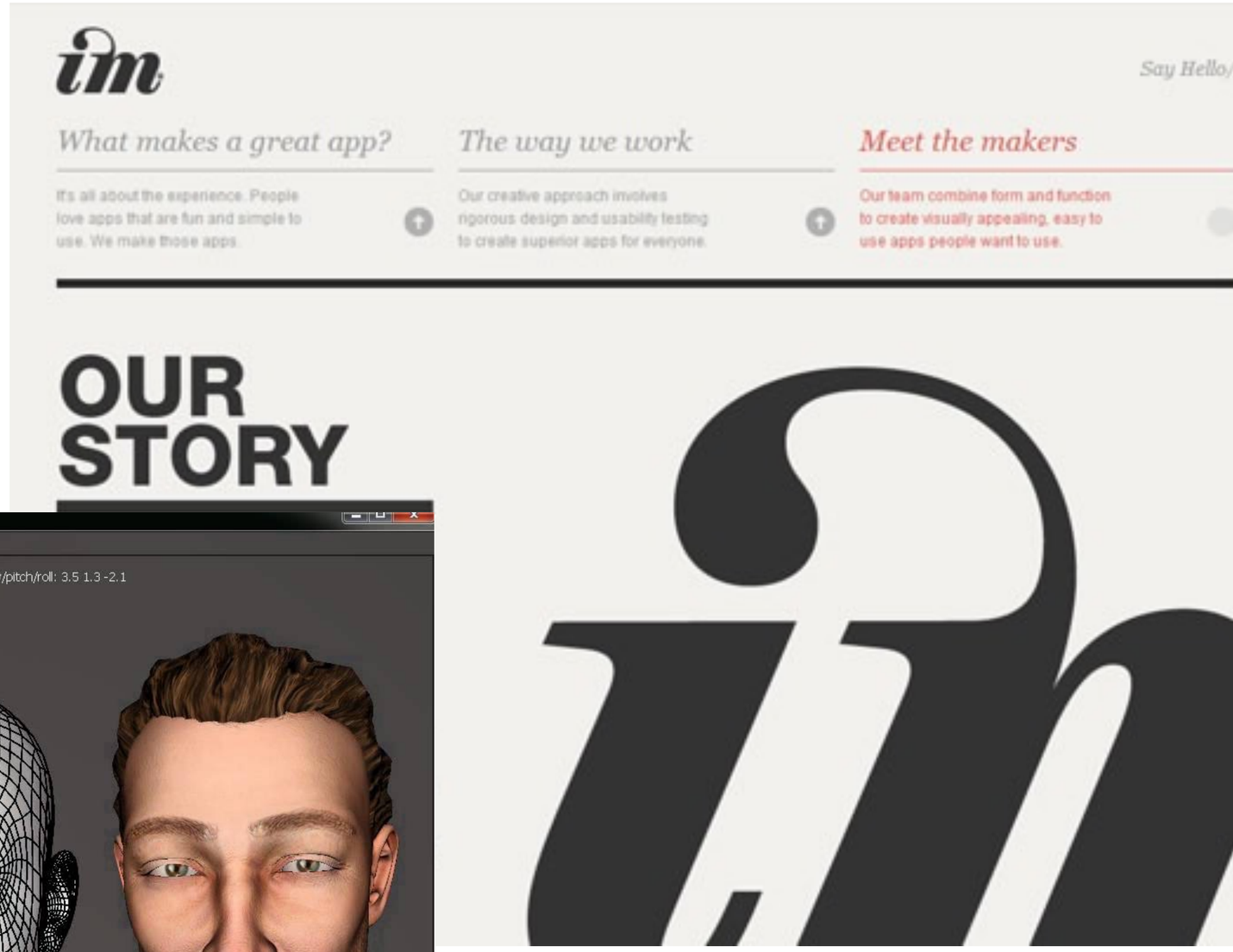


Navigation


No SIM
1:07 PM
30%
Directions
empire state building new york
Empire State Building

Communication





Foundations of computer graphics

- All these applications demand *sophisticated* theory & systems
- Theory
  - geometric representations
  - sampling theory
  - integration and optimization
  - radiometry
  - perception and color
- Systems
  - parallel, heterogeneous processing
  - graphics-specific programming languages

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 2x2x2
- 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

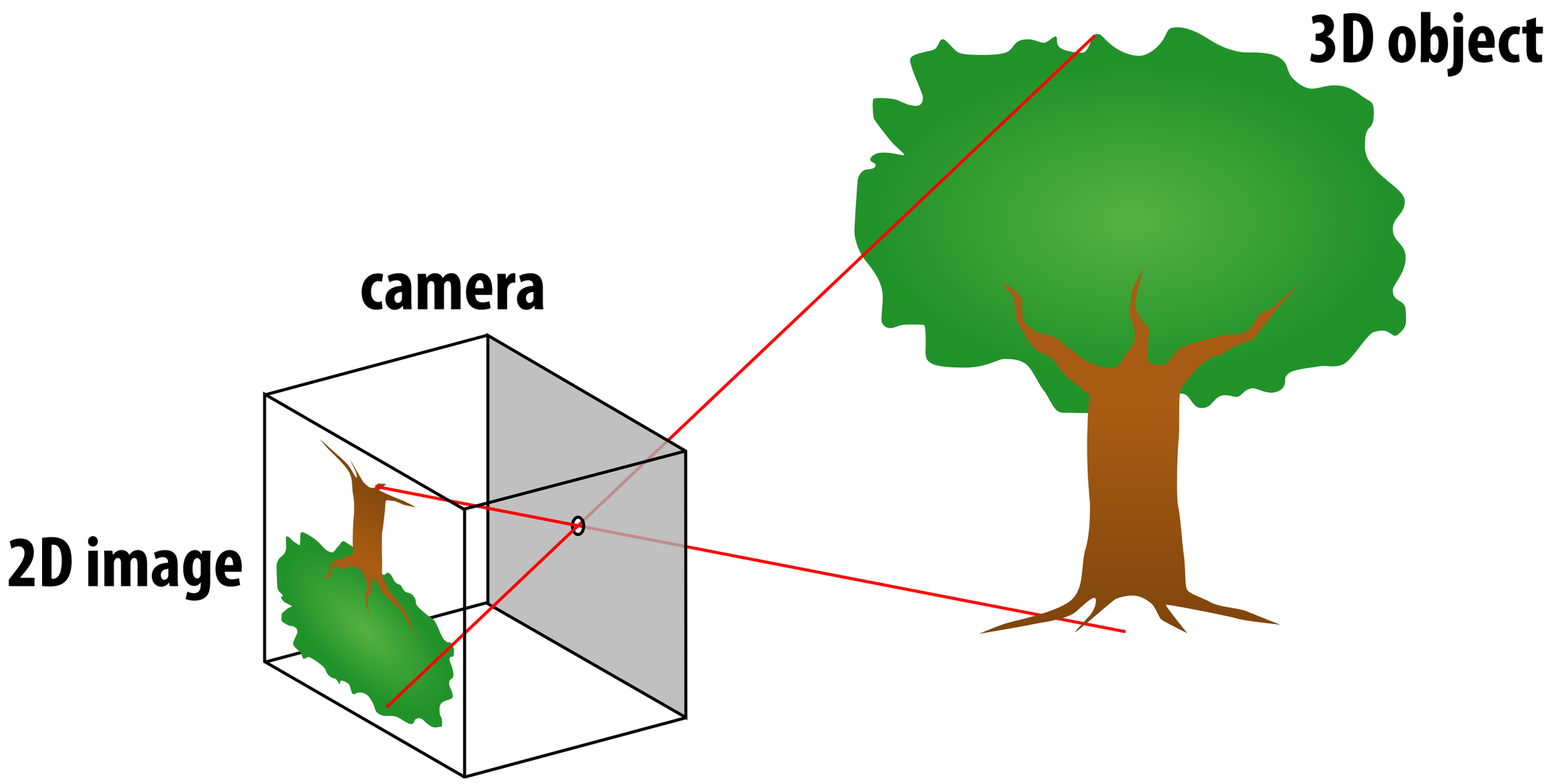
- **Now have a digital description of the cube:**

```
VERTICES                                    EDGES
A: ( 1, 1, 1 )   E: ( 1, 1,-1 )
B: (-1, 1, 1 )   F: (-1, 1,-1 )             AB, CD, EF, GH,
C: ( 1,-1, 1 )   G: ( 1,-1,-1 )             AC, BD, EG, FH,
D: (-1,-1, 1 )   H: (-1,-1,-1 )             AE, CG, BF, DH
```

- **How do we draw this 3D cube as a 2D (flat) image?**
- **Basic strategy:**
  1. **map 3D vertices to 2D points in the image**
  2. **connect 2D points with straight lines**
- **...Ok, but how?**

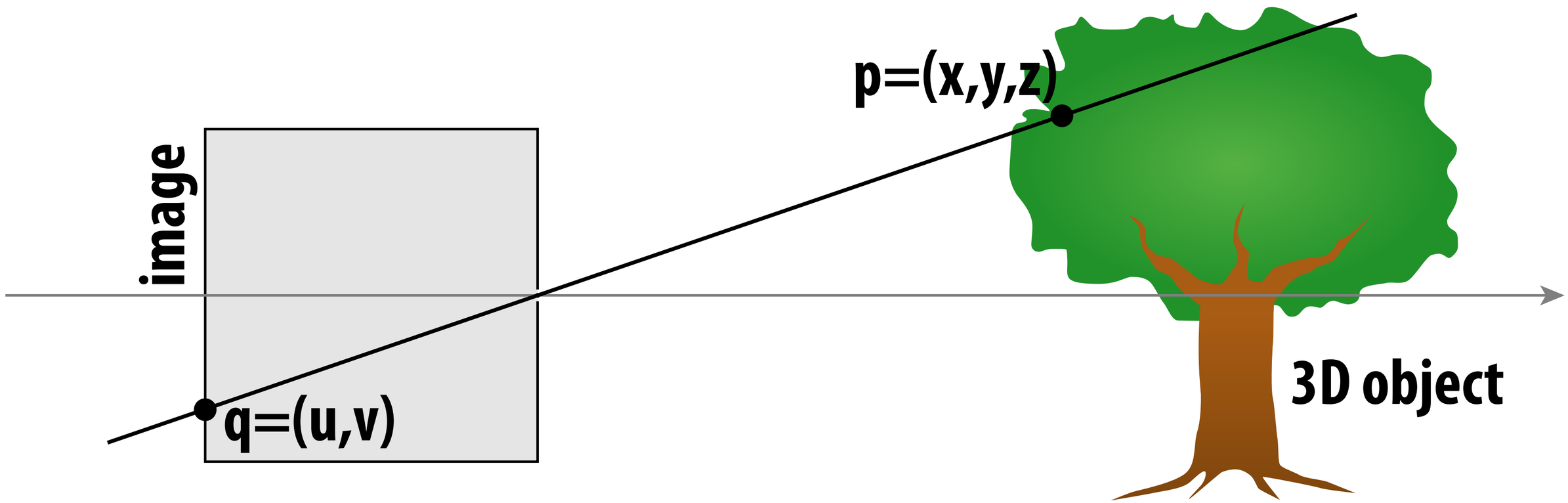
Perspective projection

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



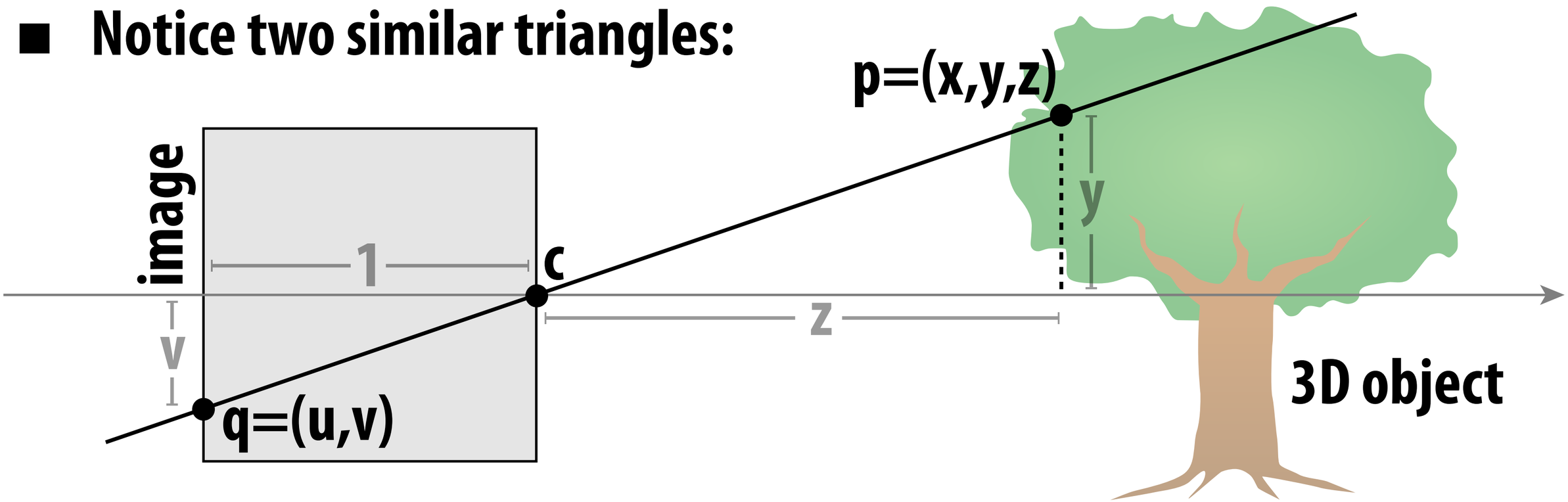
Perspective projection: side view

- 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:



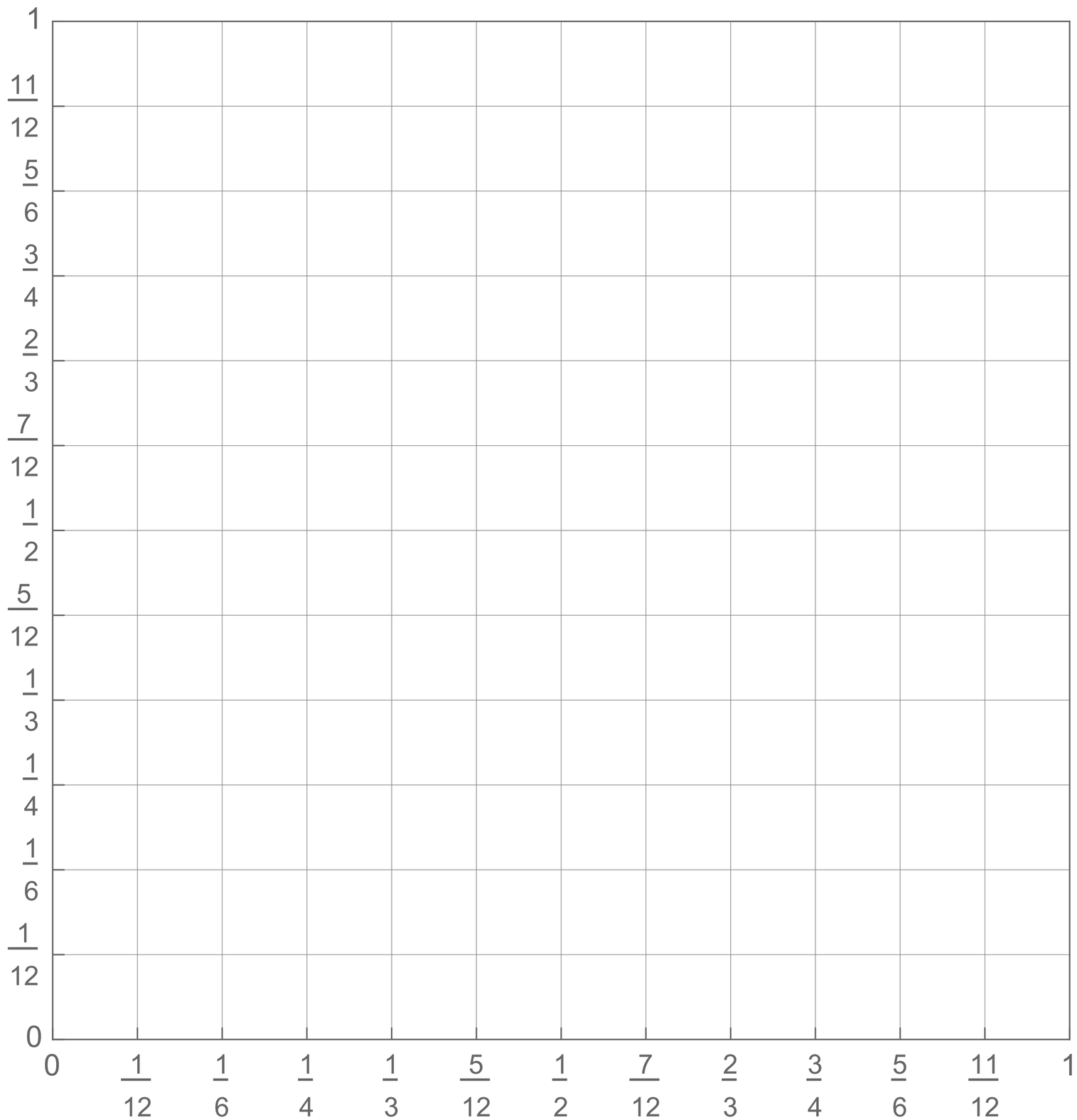
- Assume camera has unit size, coordinates relative to pinhole c
- Then v/1 = y/z, i.e., vertical coordinate is just the slope y/z
- Likewise, horizontal coordinate is u=x/z

ACTIVITY: now draw it!

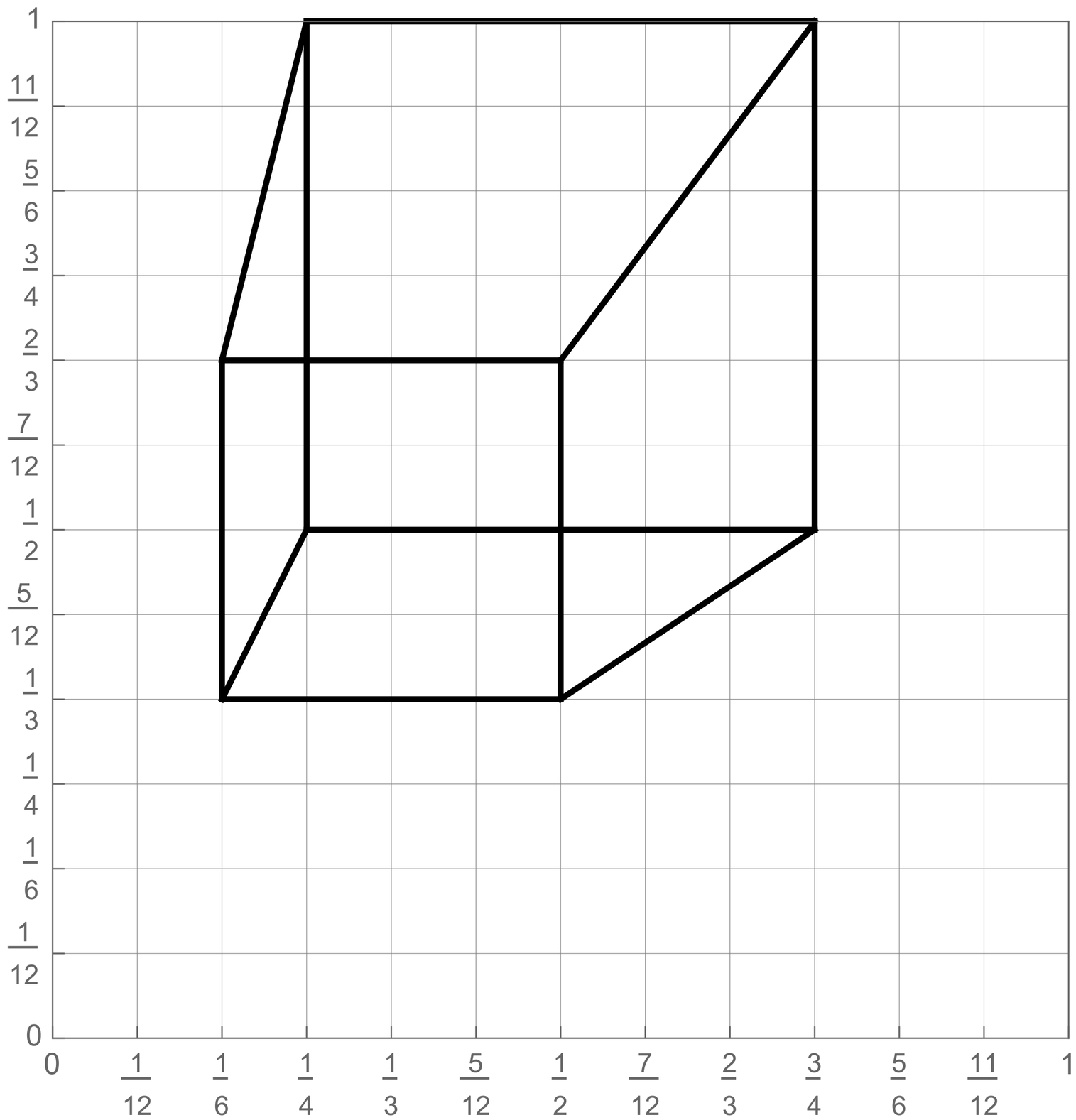
- Need 12 volunteers
  - each person will draw one cube edge
  - assume camera is at c=(2,3,5)
  - convert (x,y,z) of both endpoints to (u,v):
    1. subtract camera location
    2. divide x and y by z
  - draw line between (u1,v1) and (u2,v2)

```
VERTICES                                    EDGES
A: ( 1, 1, 1 )   E: ( 1, 1,-1 )
B: (-1, 1, 1 )   F: (-1, 1,-1 )             AB, CD, EF, GH,
C: ( 1,-1, 1 )   G: ( 1,-1,-1 )             AC, BD, EG, FH,
D: (-1,-1, 1 )   H: (-1,-1,-1 )             AE, CG, BF, DH
```

ACTIVITY: output on graph paper



ACTIVITY: how did we do?



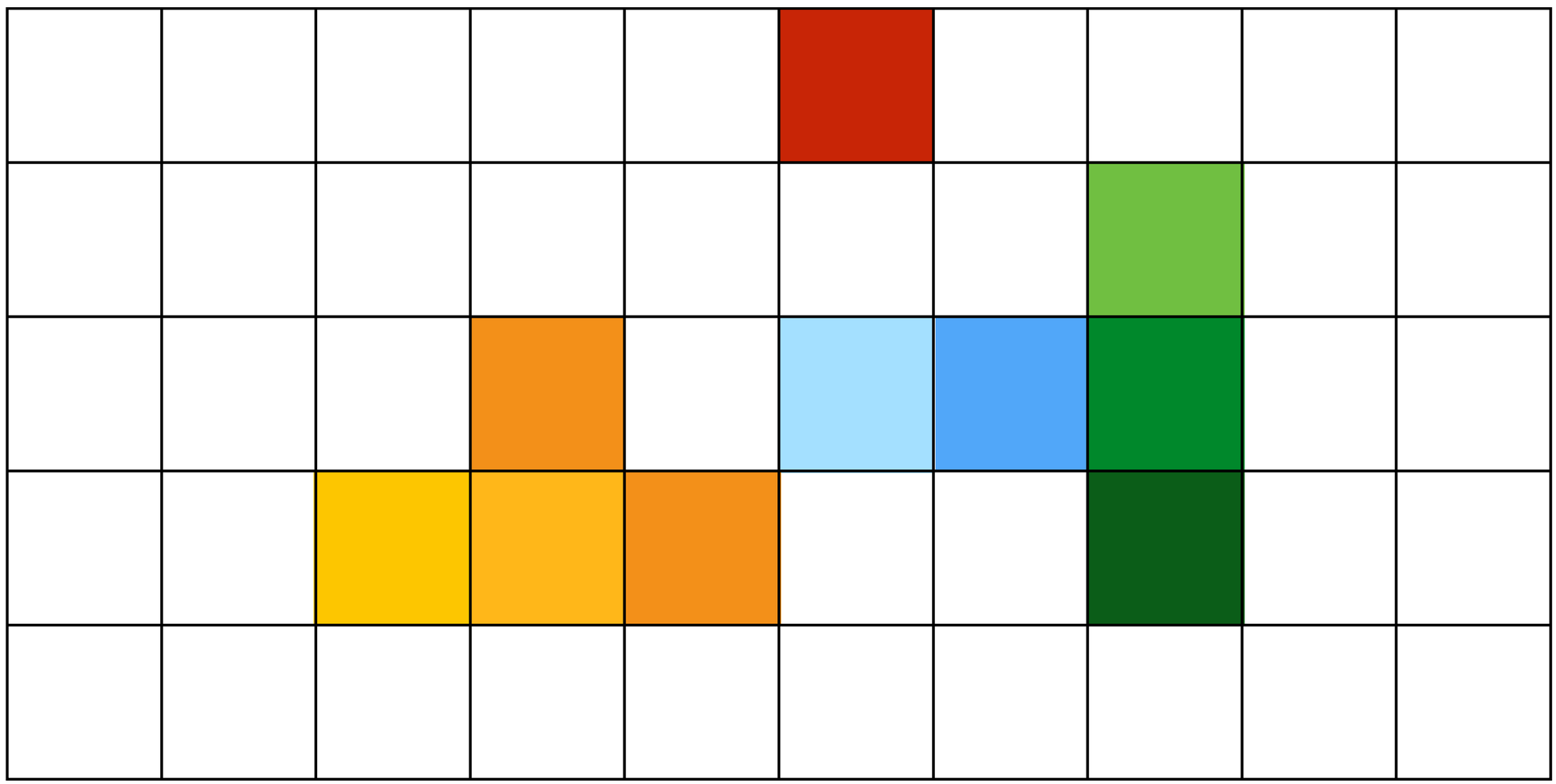
2D coordinates:

```
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
```

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 can take on a unique color value

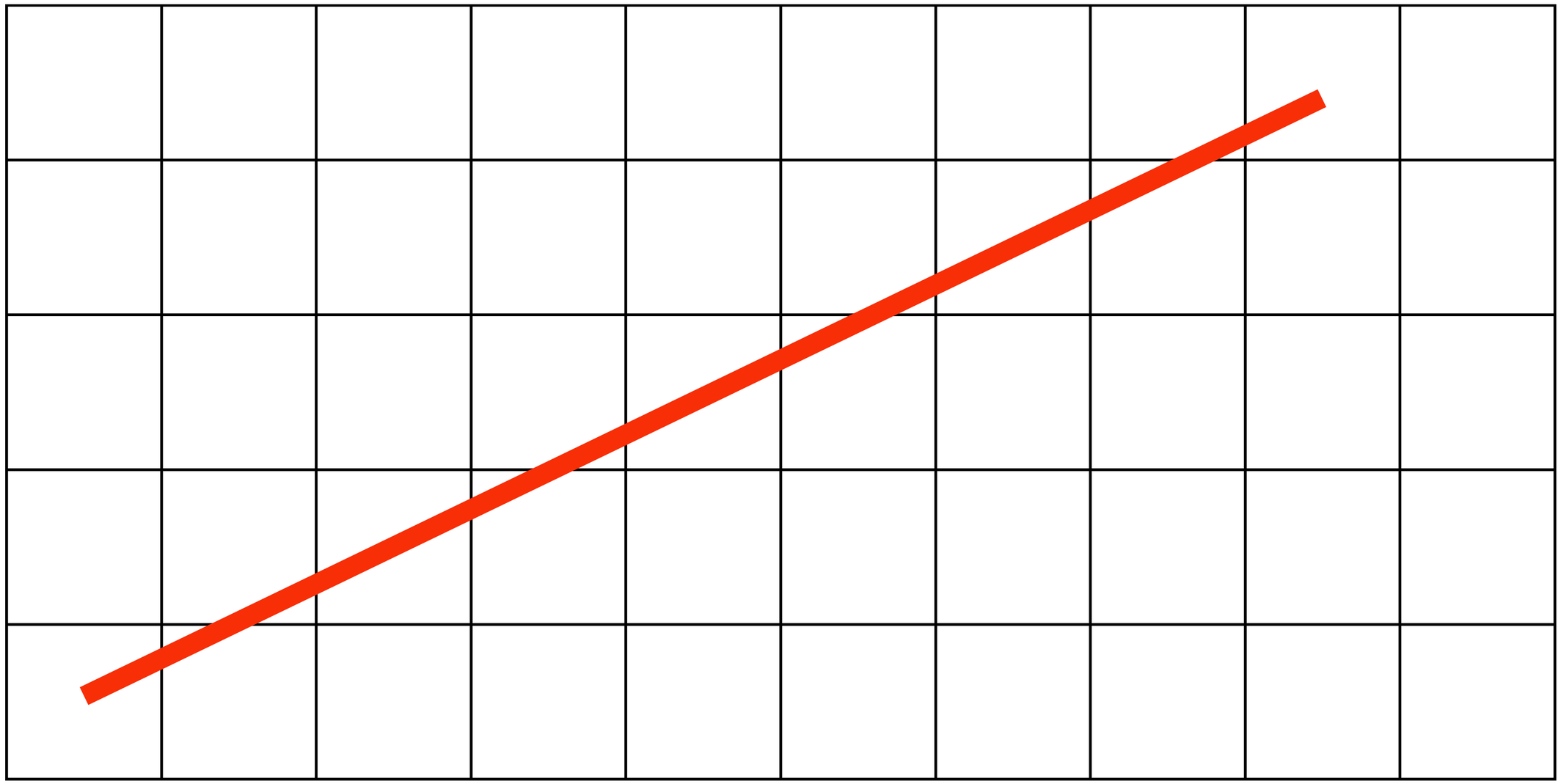


** Kayvon will strongly challenge this notion of a pixel "as a little square" next class.
But let's go with it for now. ;-)

Close up photo of pixels on a modern display

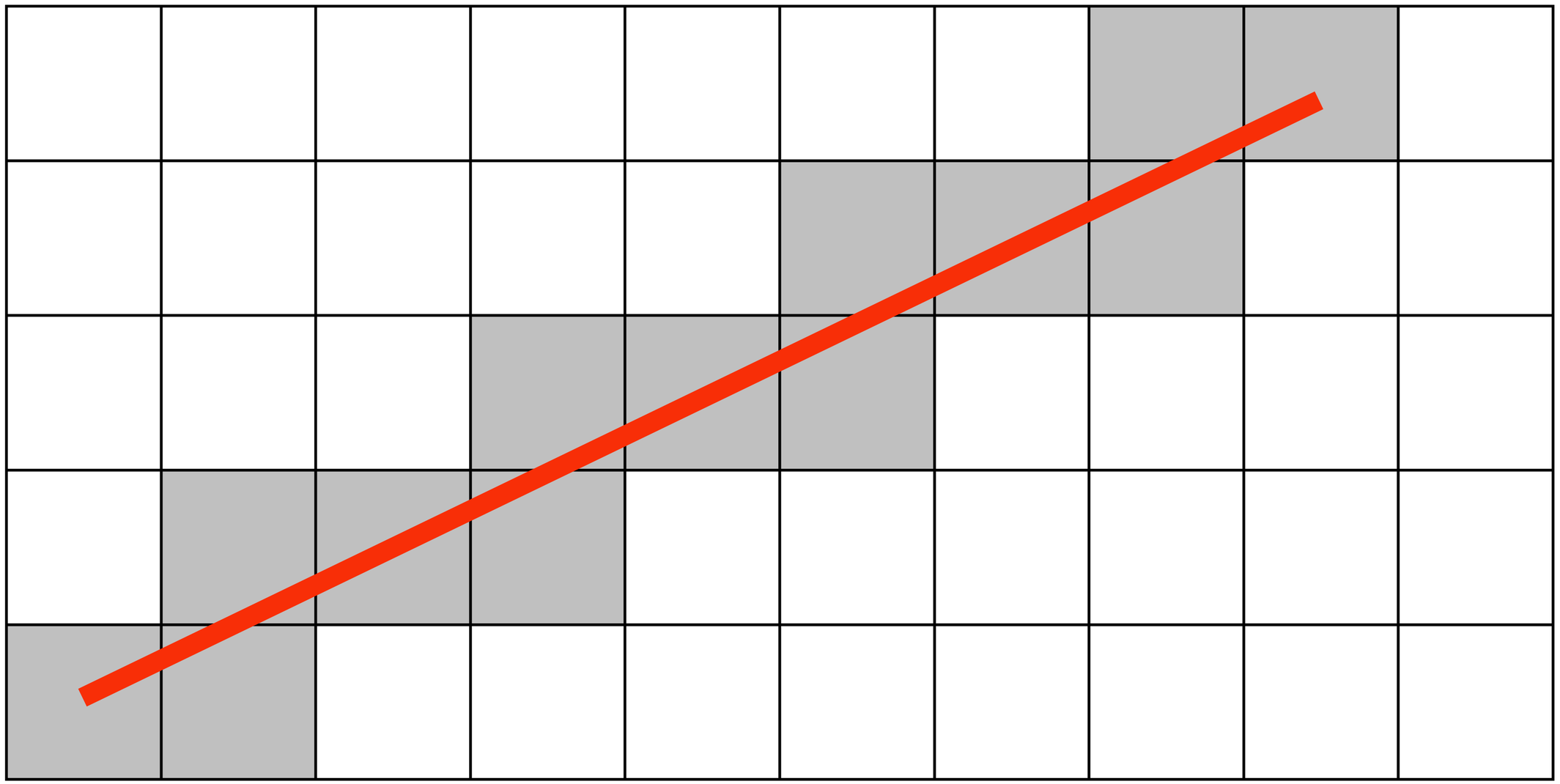
What pixels should we color in to depict a line?

"Rasterization": process of converting a continuous object to a discrete representation on a raster grid (pixel grid)



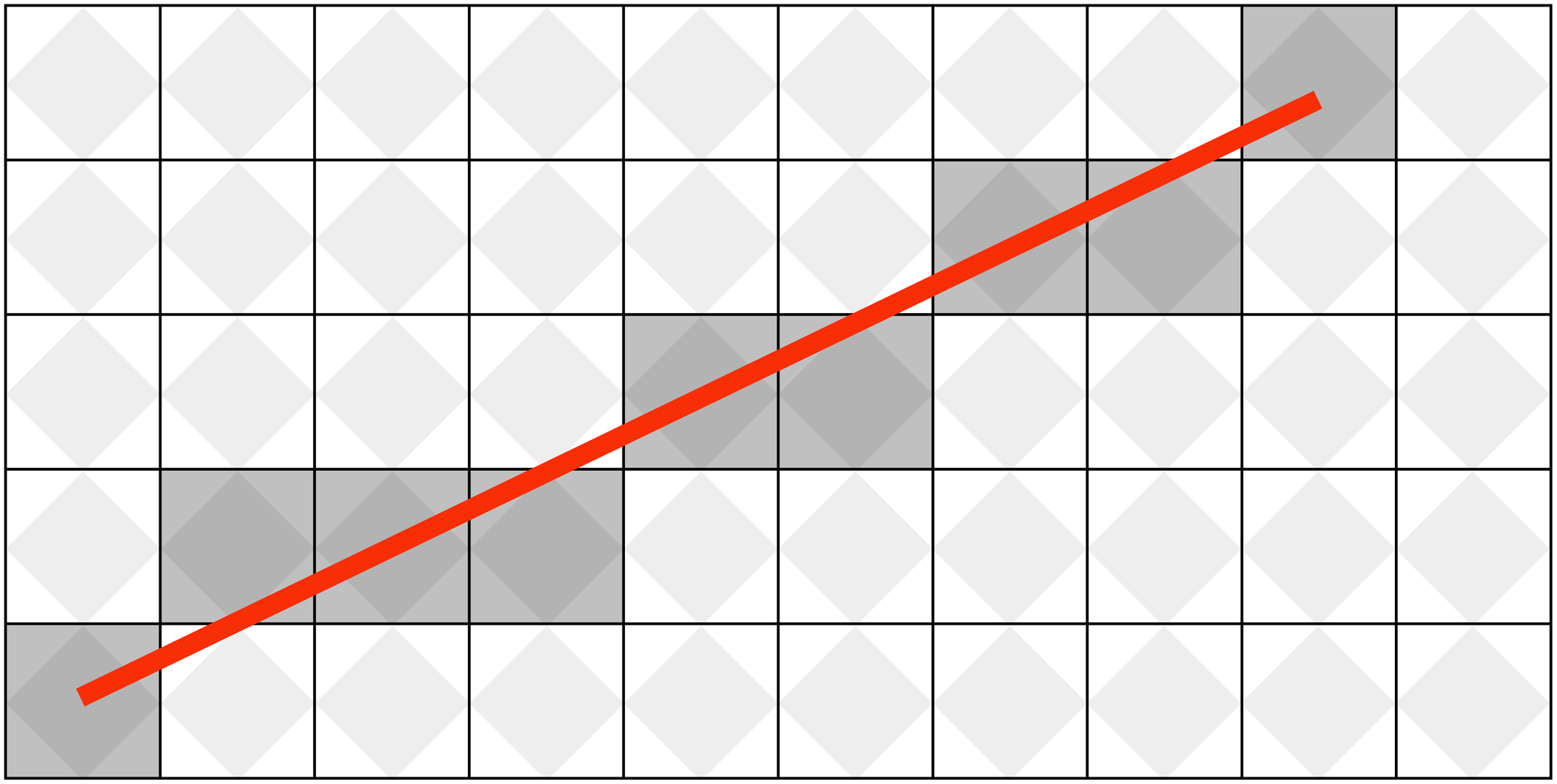
What pixels should we color in to depict a line?

Light up all pixels intersected by the line?



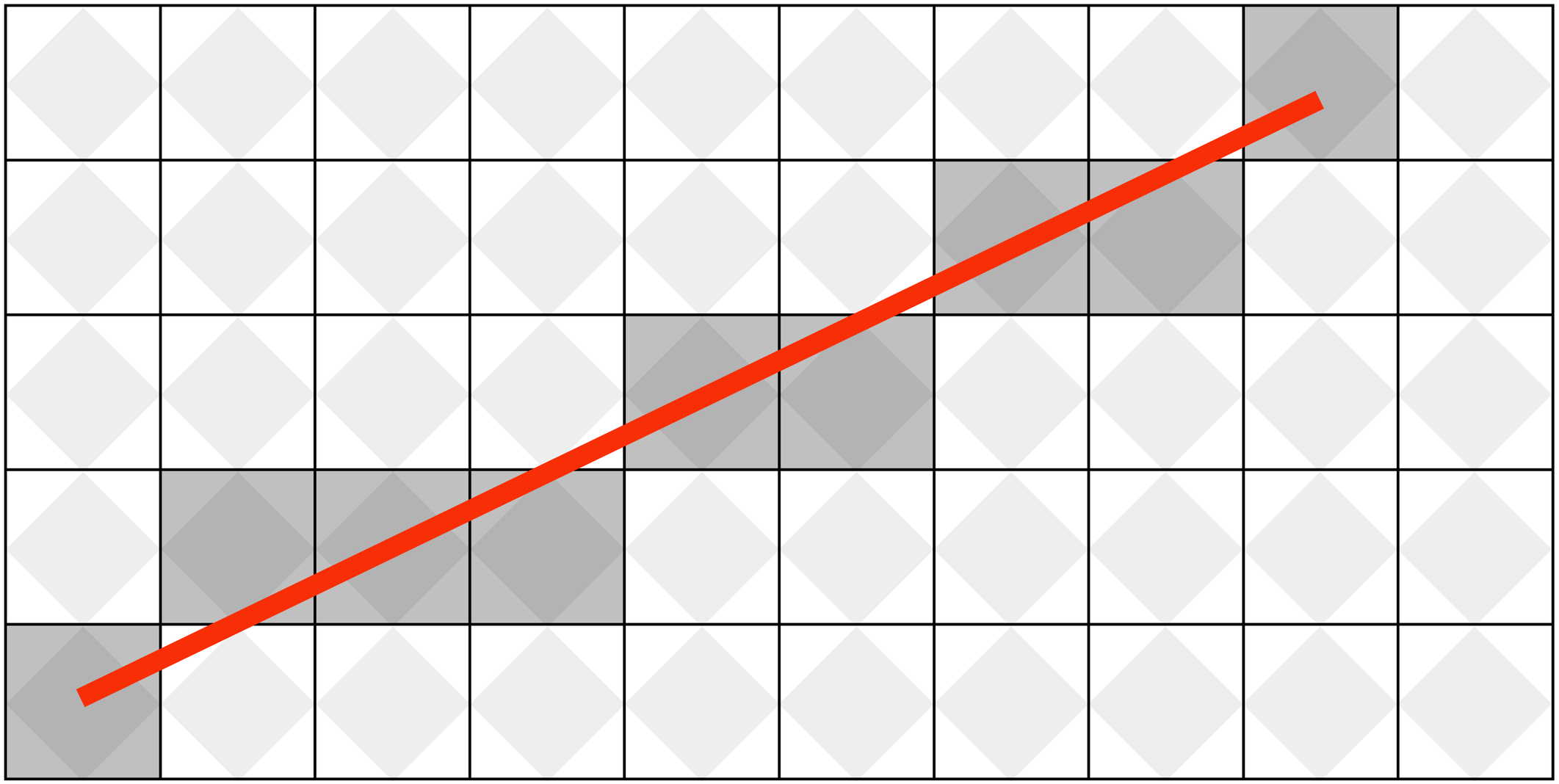
What pixels should we color in to depict a line?

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)



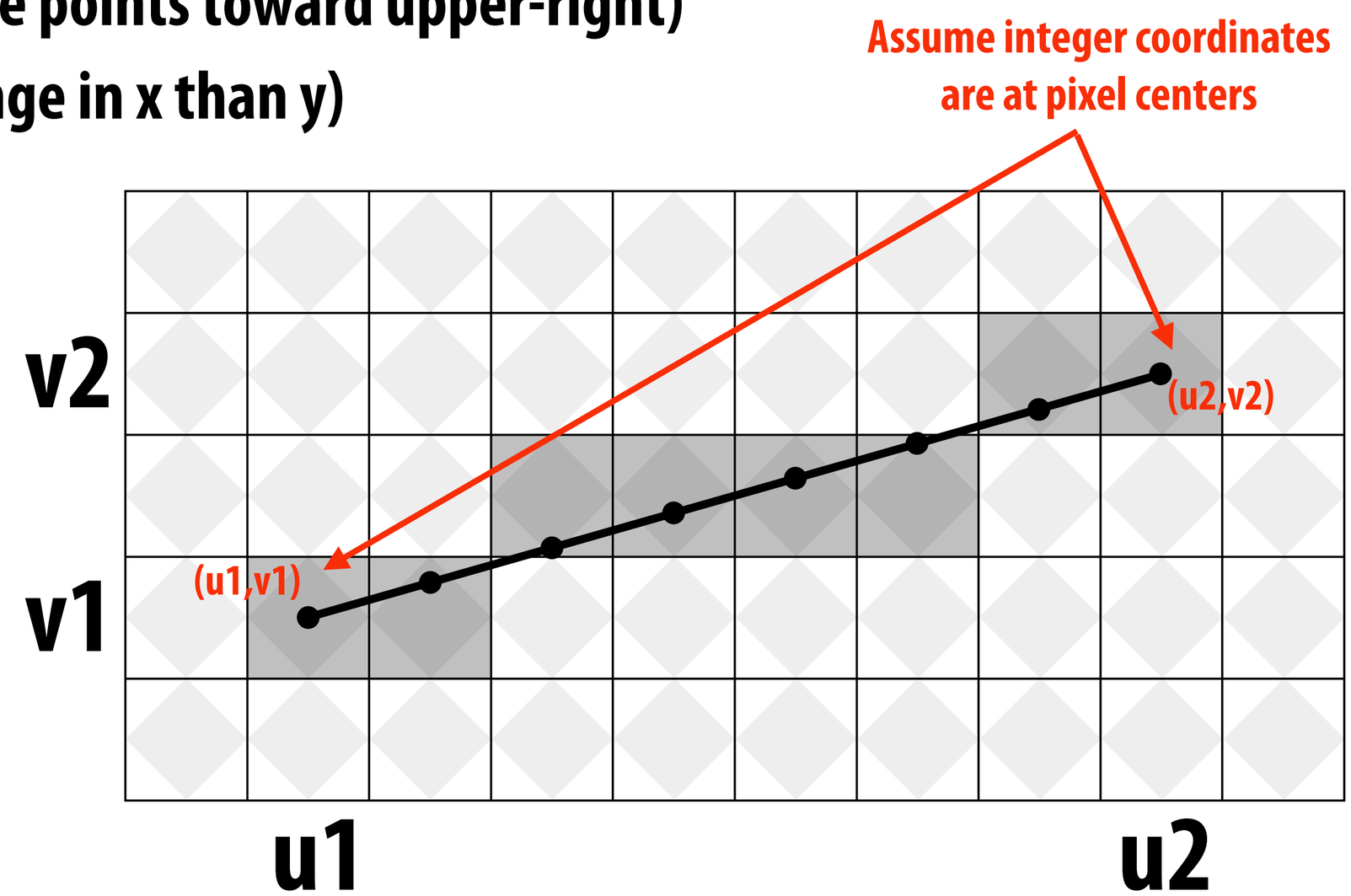
How do we find the pixels satisfying a chosen rasterization rule?

- 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., work proportional to number of pixels in the drawing of the line)

Incremental line rasterization

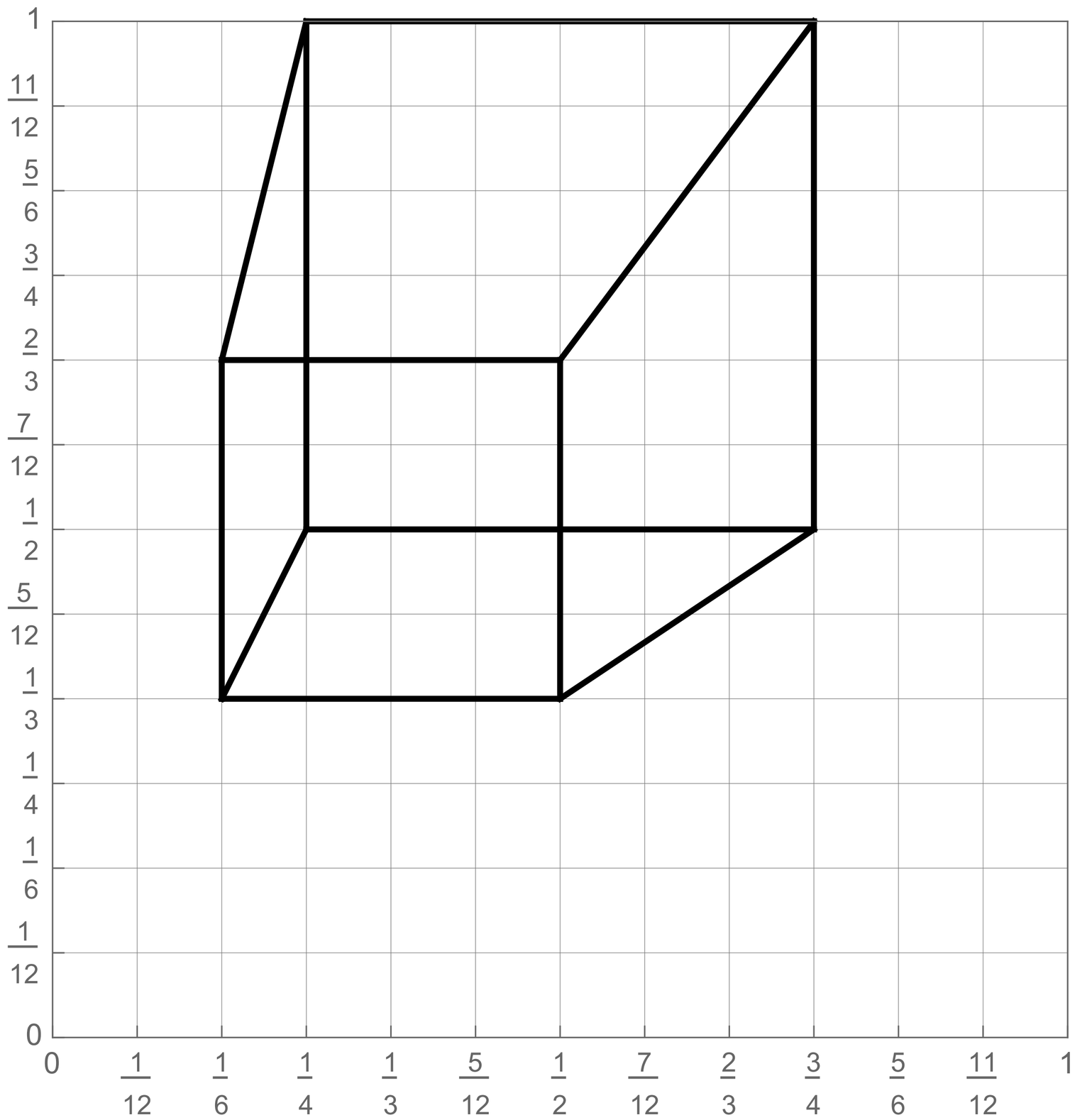
- Let's say a line is represented with integer endpoints: (u1,v1), (u2,v2)
- Slope of line: s = (v2-v1) / (u2-u1)
- Consider a very easy special case:
  - u1 < u2, v1 < v2 (line points toward upper-right)
  - 0 < s < 1 (more change in x than y)

```
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)

Our line drawing!



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
```

We just rendered a simple line drawing of a cube.

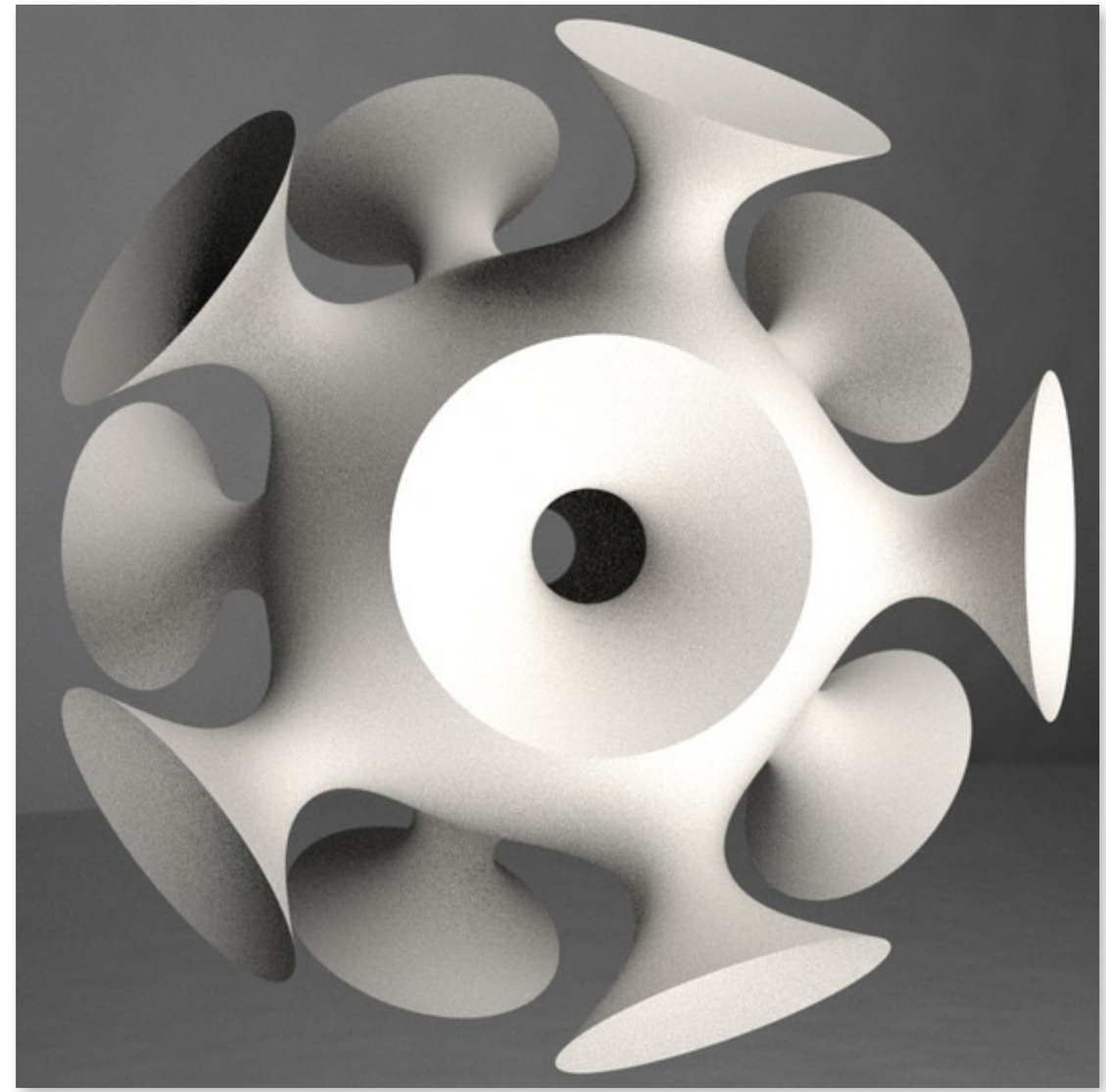
But to render more realistic pictures (or animations) we need a much richer model of the world.

surfaces
motion
materials
lights
cameras

2D shapes

[Source: Batra 2015]

Complex 3D surfaces



Platonic noid

Modeling material properties


[Wann Jensen 2001]


[Jakob 2014]


[Zhao 2013]

Realistic lighting environments

Realistic lighting environments

Realistic lighting environments

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

Unreal Engine Kite Demo (Epic Games 2015)

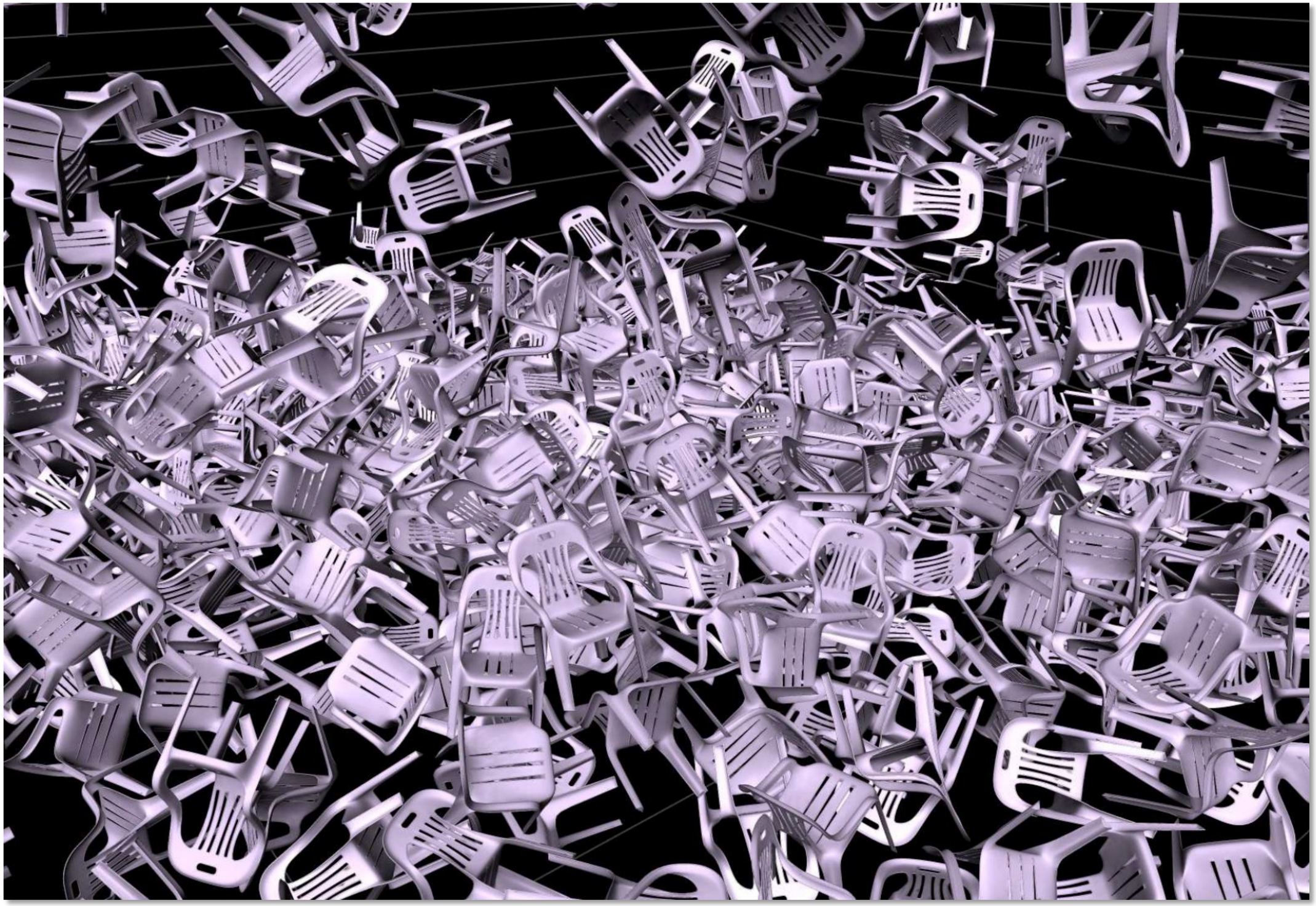
So is this.

[Mirror's Edge 2008]

Animation: modeling motion

https://www.youtube.com/watch?v=wYfYtV_2ezs

Physically-based simulation of motion



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

[James 2004]

Course Logistics

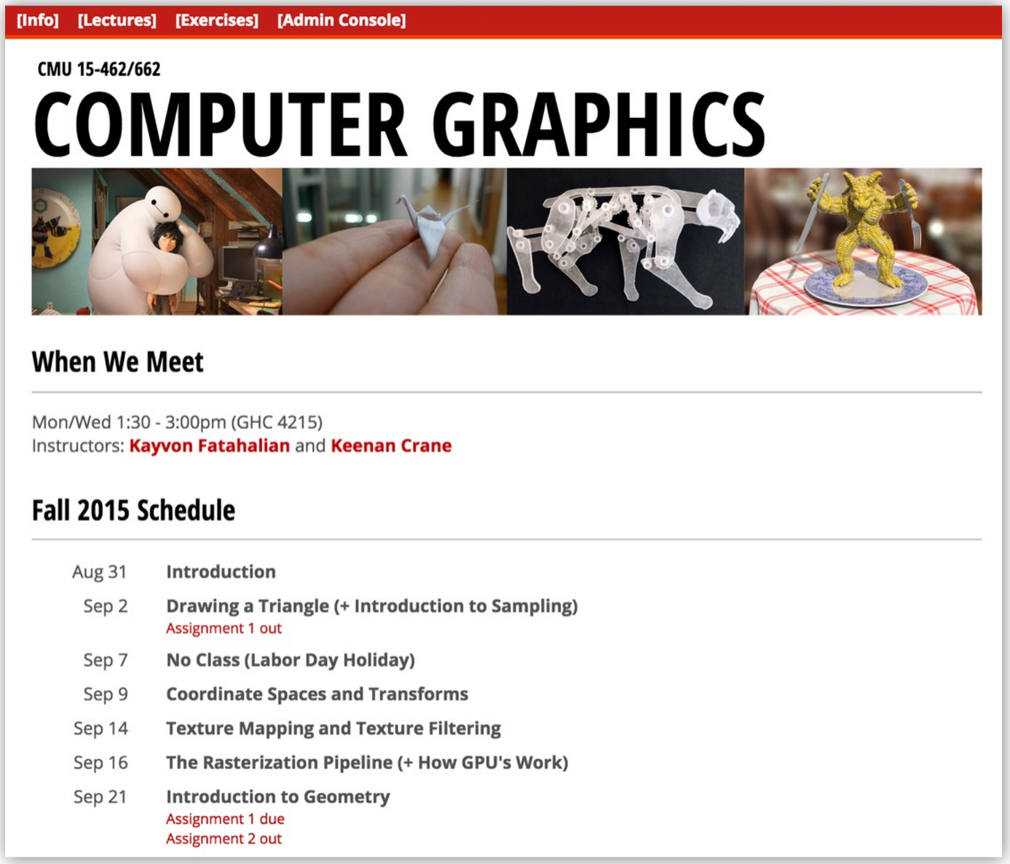
About this course

- A broad overview of major topics and techniques in computer graphics: geometry, rendering, animation, imaging
- This year:
  - All new lectures
  - All new assignments (bear with us)
  - Increased focus on fundamental data structures and algorithms that are reused across all areas of graphics

Getting started

- Create an account on the course web site:
  - http://15462.courses.cs.cmu.edu

- Sign up for the course on Piazza
  - http://piazza.com/cmu/fall2015/15462



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

Assignments / Grading

- (65%) Five programming assignments
  - Four programming assignments (individually)
  - One "go further" final assignment (in pairs)
- (7%) Take home quizzes
  - One per lecture, graded on effort
  - Must be turned in BY YOU at the beginning of the next lecture
- (25%) Midterm / final
  - Both cover cumulative material seen so far
- (3%) Class participation
  - In-class/website comments, other contributions to class

Late hand-in policy

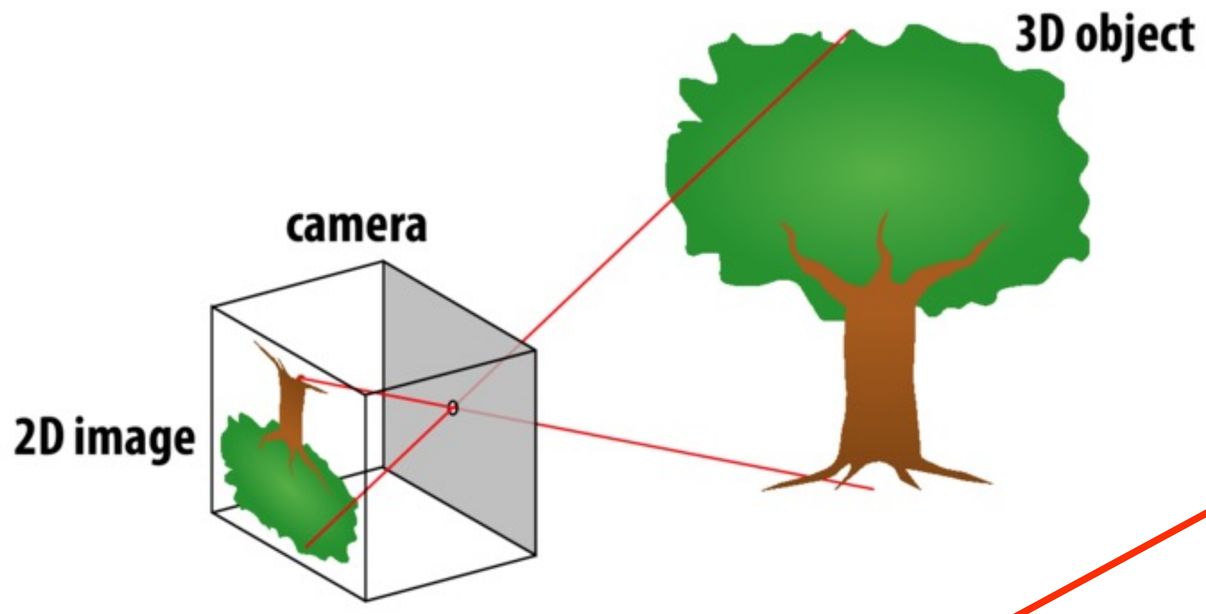
- **Programming assignments**
  - Five late day points for the semester
  - First four programming assignments only
  - No more late points? 10% penalty per day
  - No assignments will be accepted more than 3 days past the deadline
- **Daily Quizzes**
  - You can skip up to 6 with no penalty

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:



CMU 15-462/662, Fall 2015

Previous | Next --- Slide 30 of 65 Back to Lecture Thumbnails

Add Private Note

kayvonf about an hour ago

Question: During class Keenan asked a question about why do objects look smaller when hey 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?

Prompt Edit Delete Archive [0 Upvote Downvote]

"Add private note" button:
You can add notes to yourself about this slide here.

Slide comments and discussion

Our philosophy

- We want a very active class: come to class, participate in the class, contribute to the web site
- Challenging assignments (with tons of "going further" opportunities: see what you can do!)
- Challenging exams (see what you can do!)
- Very reasonable grading (at least the instructors think so)

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?