Introduction to Computer Graphics

CMSC - 23700

What is Computer Graphics?

You are probably thinking about this:

More fundamentally — why do we need Computer Graphics?

How does Computer Graphics shows up in Computer Science?

What is the function of CG in digital computation?

Why do we need it?

Computer Graphics

Computer graphics is a sub-field of computer science which studies methods for digitally *synthesizing* and manipulating visual content.

Computer graphics **refers to a fairly wide field of *digital creation* that primarily regards the use of computers and software programs to create visual media.**

Creating anything that is visually engaging and created through the use of computer

I asked Gemini...

Computer graphics is essentially the creation and manipulation of visual images using computers. It's a broad field that encompasses a wide range of techniques and applications. Here's a breakdown of key aspects:

Core Concept:
It involves using computers to generate and display images, whether they are 2D or 3D.

It's a blend of computer science and design, requiring both technical knowledge and artistic understanding.

PS - the rest of the answer wasn't so great for various reasons. But the above is an OK summary with some small caveats!

SIGGRAPH 2022 Technical Papers Trailer

About Me (Rana Hanocka)

- Joined UChicago in July 2021
- Research is in Computer Graphics & Machine Learning
  - Geometry Processing & Deep Learning
- Lived in 3 countries (USA, Israel, Denmark)
  - speak a few languages
- 11 month old daughter - MINA
- I like martial arts (Muay Thai/Thai Boxing, Boxing)

https://people.cs.uchicago.edu/~ranahanocka/

https://3dl.cs.uchicago.edu/

Course Staff

Rana Hanocka

Nam Anh Dinh

Jacob Serfaty

Ron Kiehn

All Course Info on 3DL Website

https://3dl.cs.uchicago.edu/courses/CMSC23700-S25

- Let's walk through the syllabus and all the course logistics on the website together

In case I forgot…

- Join the slack and add a photo (It helps us remember your face & names!)
- Ask questions during the lectures!
  - It helps me get to know how you are thinking when you are hearing about this material for the first time
  - Helps me know what is or isn't clear
    - if you are confused likely others are too
  - … and keeps everyone engaged!
  - Please, please, please do not plagiarize. If you are desperate for help EMAIL ME!! We will help you!

Computer graphics is ***everywhere***!

Most Pics In IKEA Catalogues Aren't Photos, They're 3D Renders

To ensure everything looked photorealistic, the company's photographers and 3D modelers swapped places.

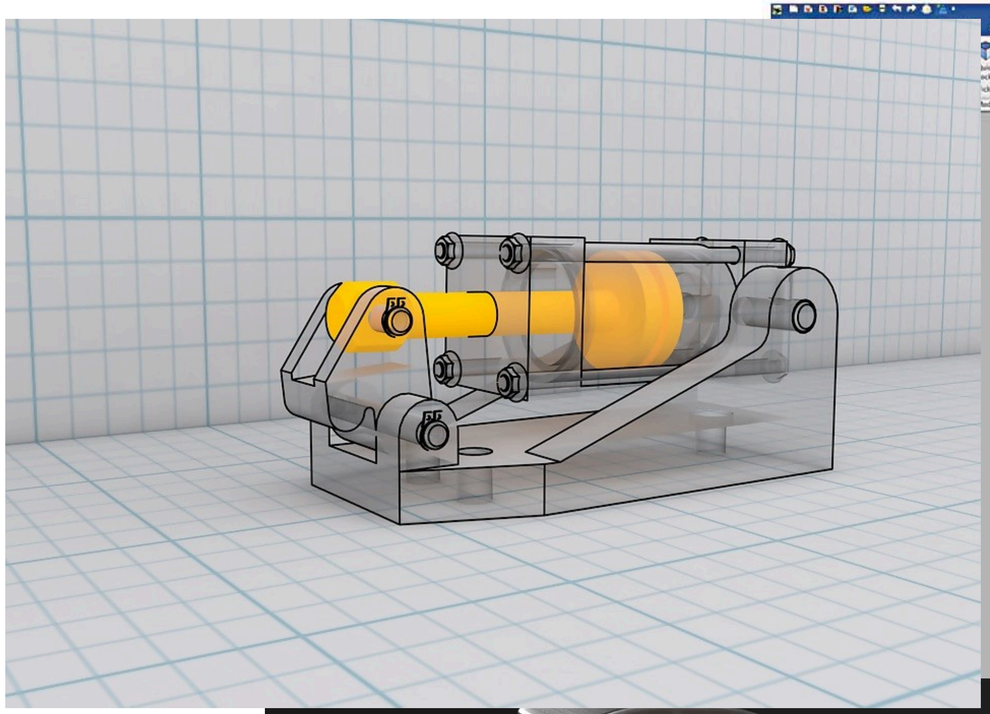
They learned the ins-and-outs of each others' roles so that they could create better visuals, and it worked.

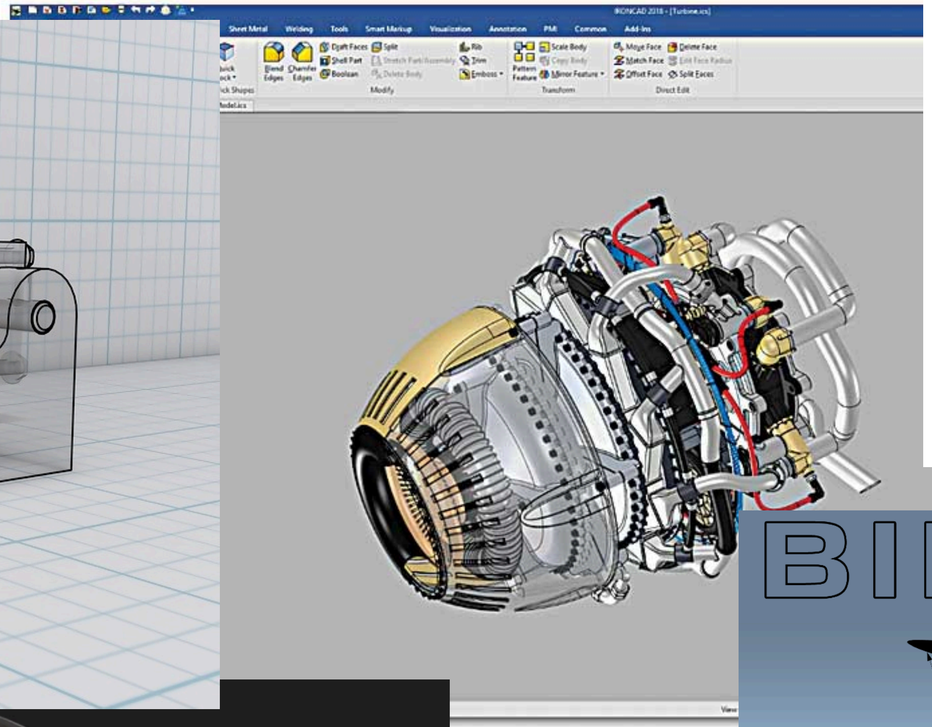
Alan: So that's kind of pervasive now. So when you're looking at the kitchen catalogue, most of those renters are all in 3D. It's funny, because Helen Papagiannis — author of Augmented Human — she's got this game, "Augmented Reality Or Real?" And I've gone through the magazine, I can't tell. I really can't tell what's real and what's 3D.

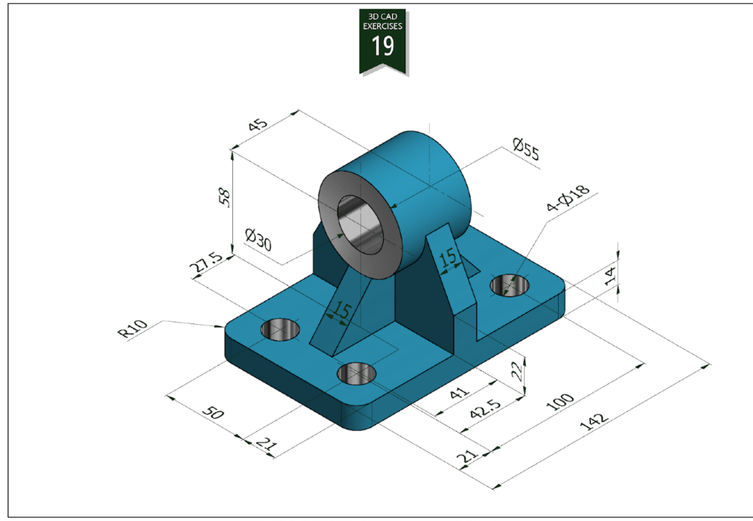
https://xrforbusiness.io/podcast/delivering-digital-swedish-meatballs-in-ar-with-ikeas-martin-enthed/

Entertainment: Animated Films & Video Games

CAD: Computer Aided Design







Simulations

Action

Precondition: The mug is full of liquid

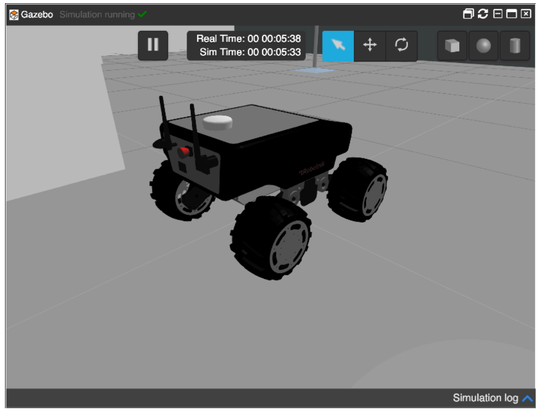
Action: The robot empties the mug into the sink

Symbolic action:EmptyLiquidFromObject object=Mug

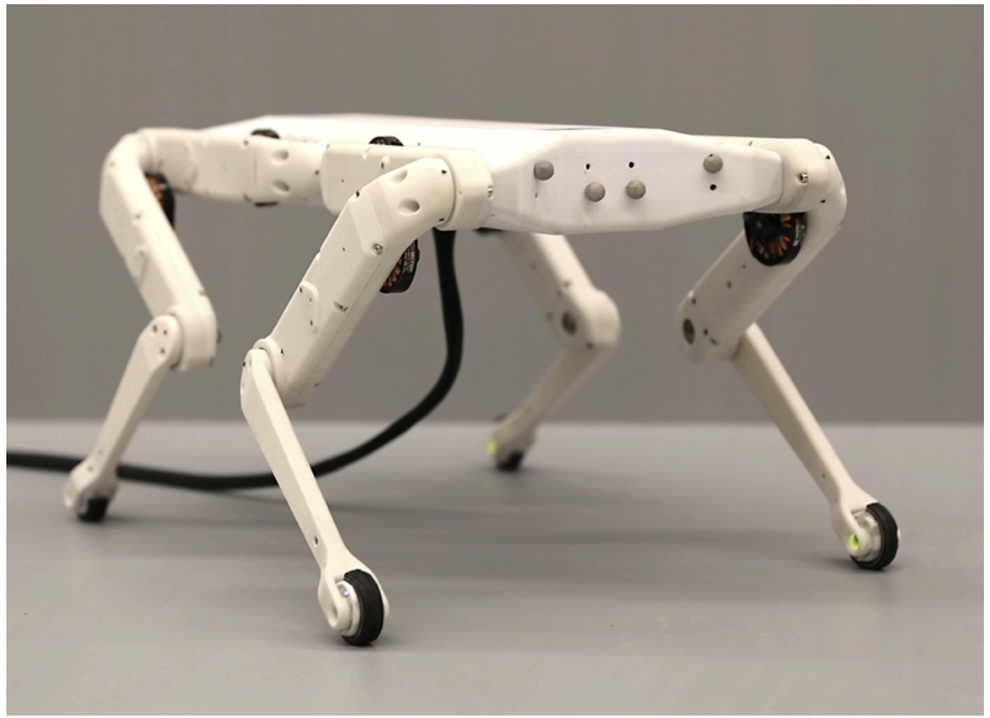
Initial State

Result

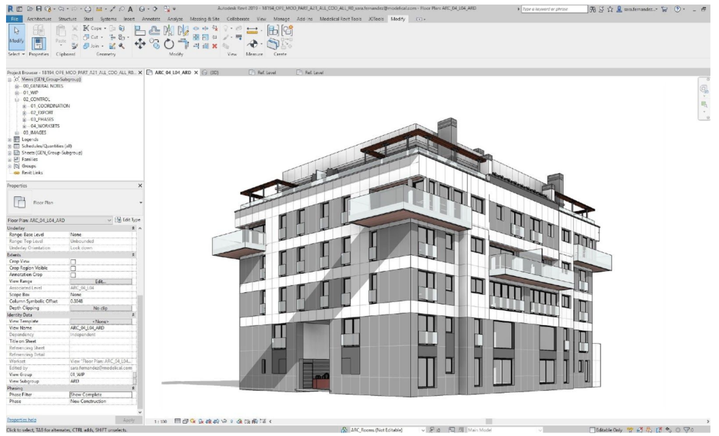
Train an AI Agent



Robotics



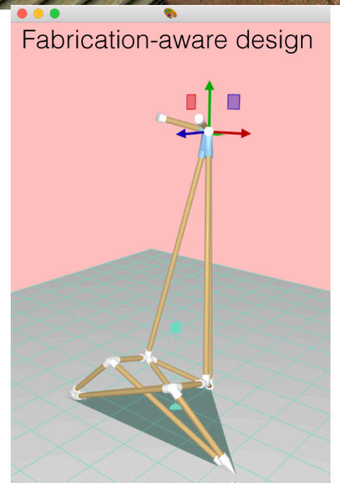
Architecture

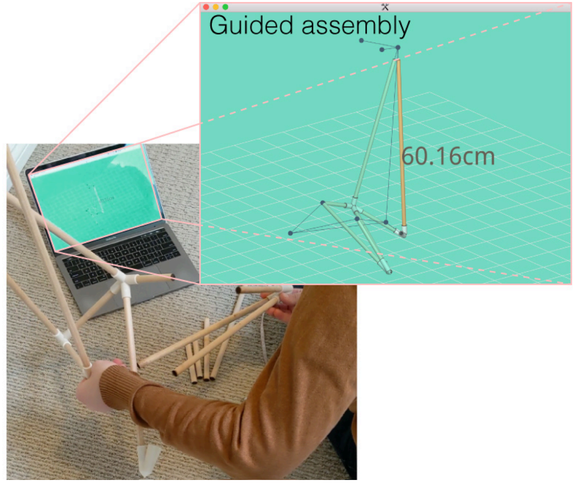


Fabrication

Parts-based model

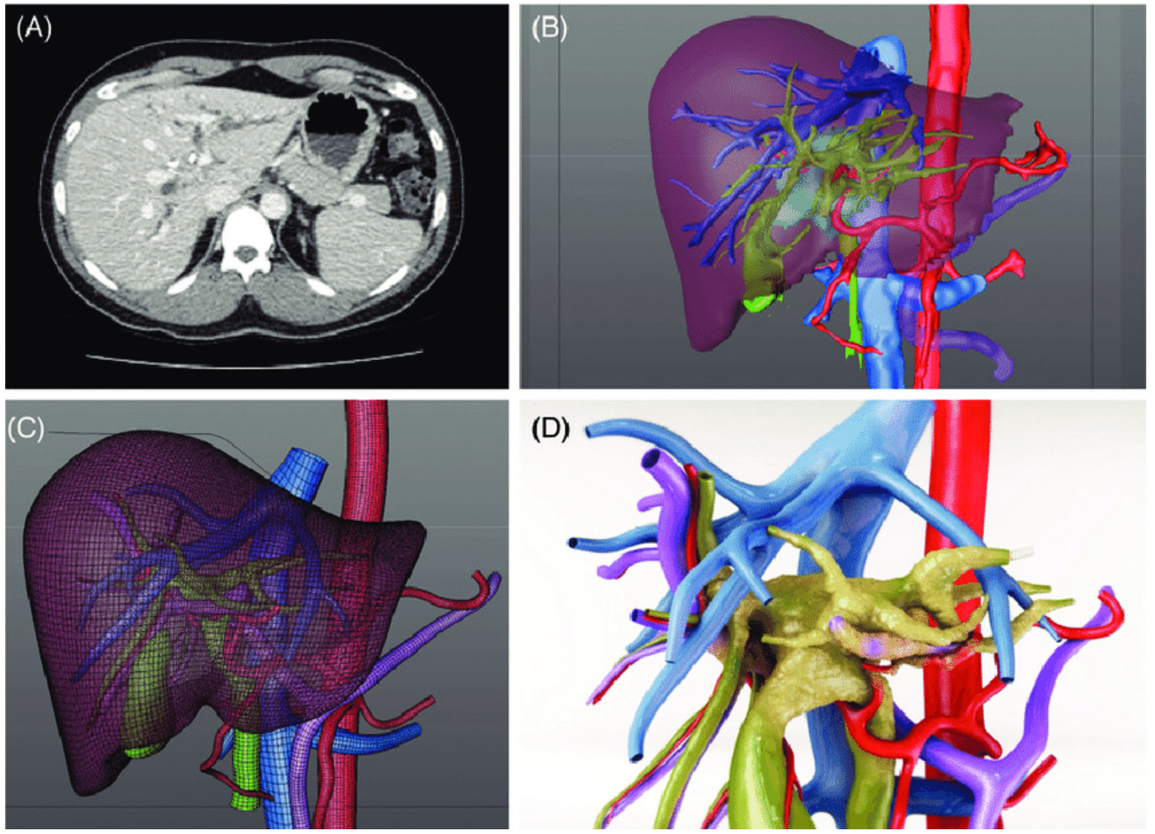
Fabricated result



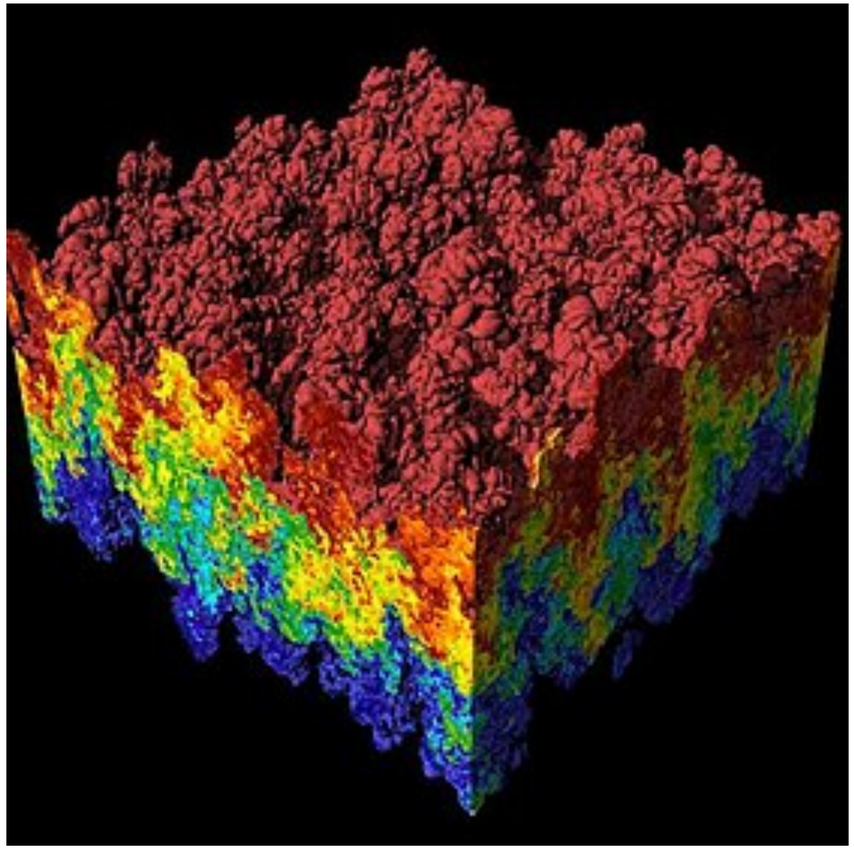


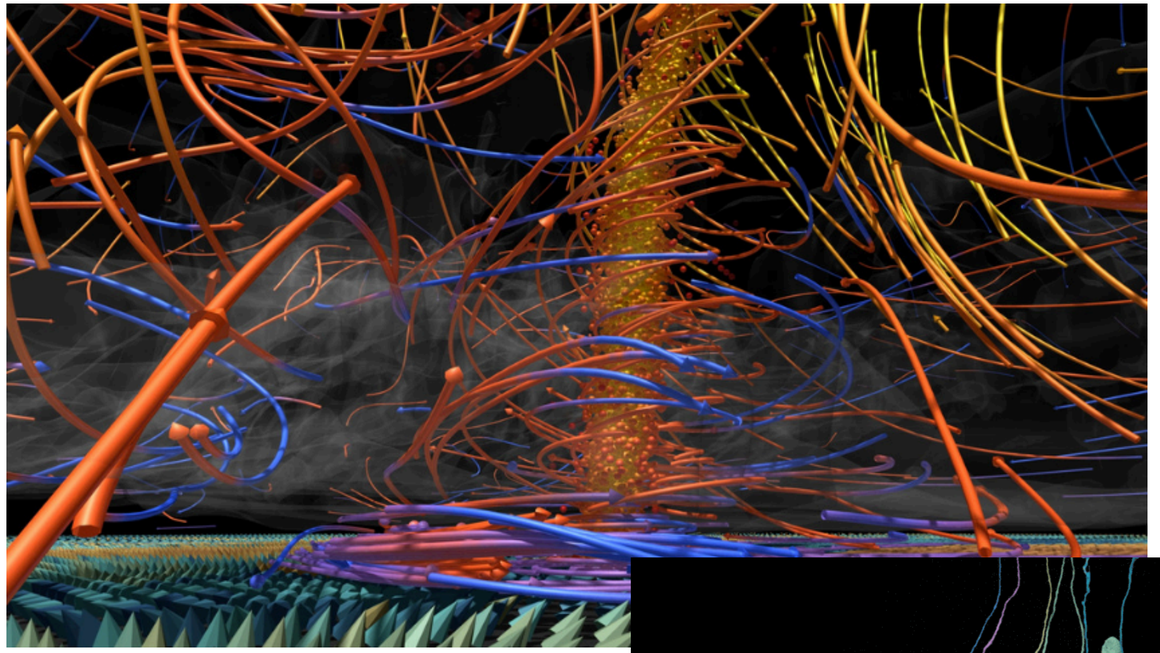
VR/AR

Biomedicine



Scientific Visualization





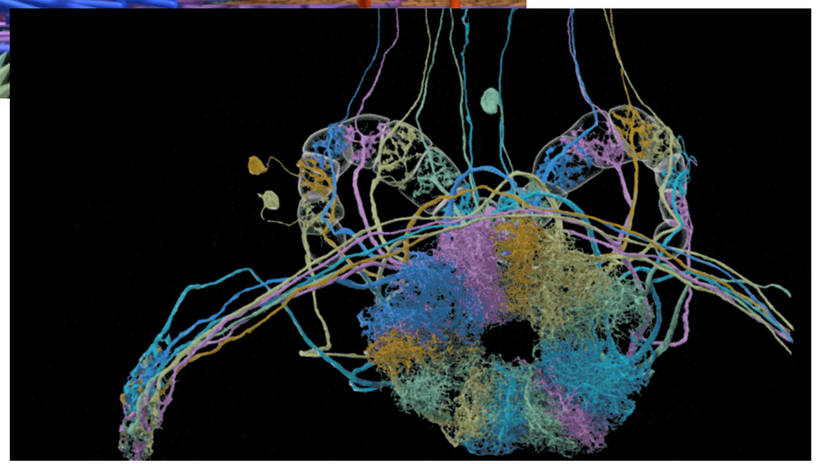
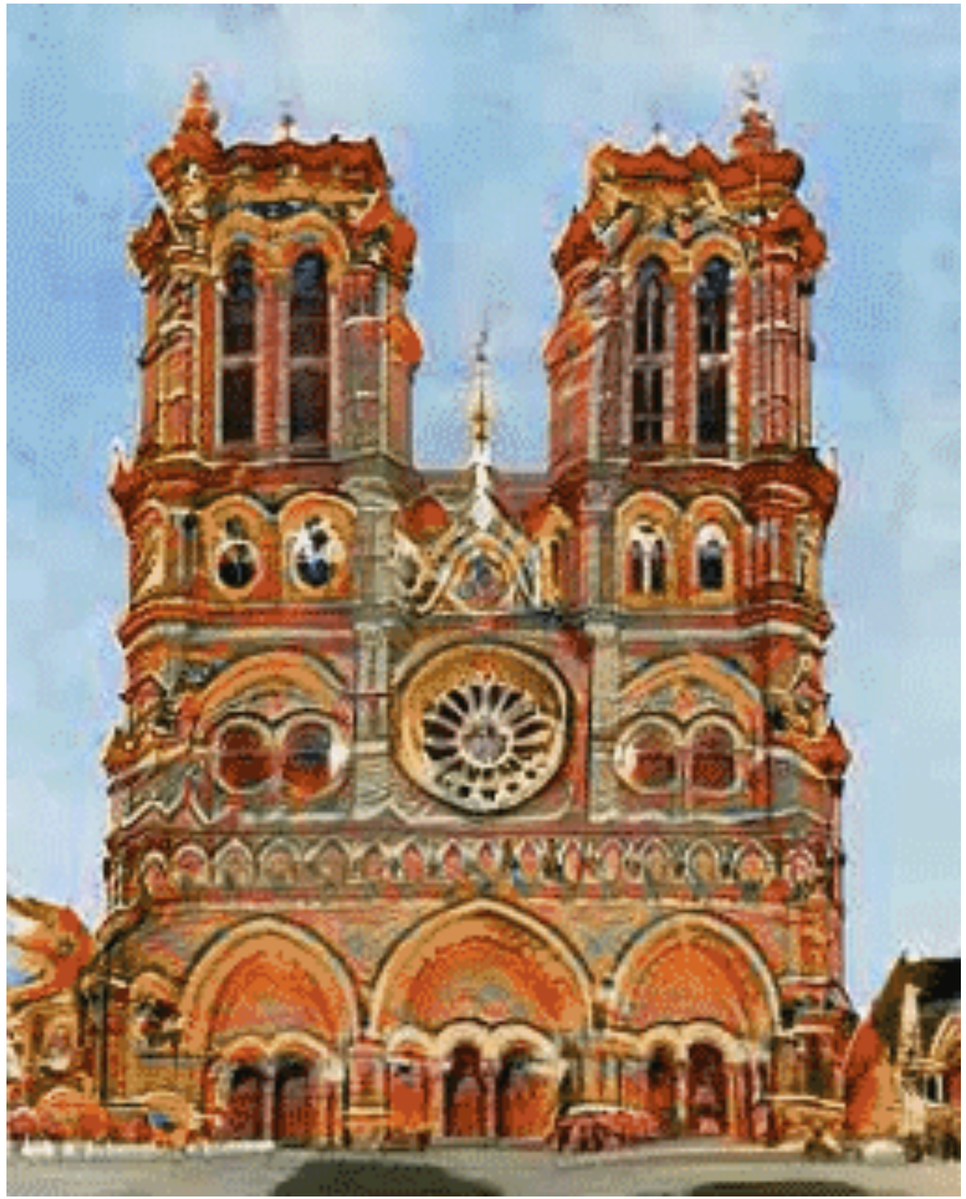
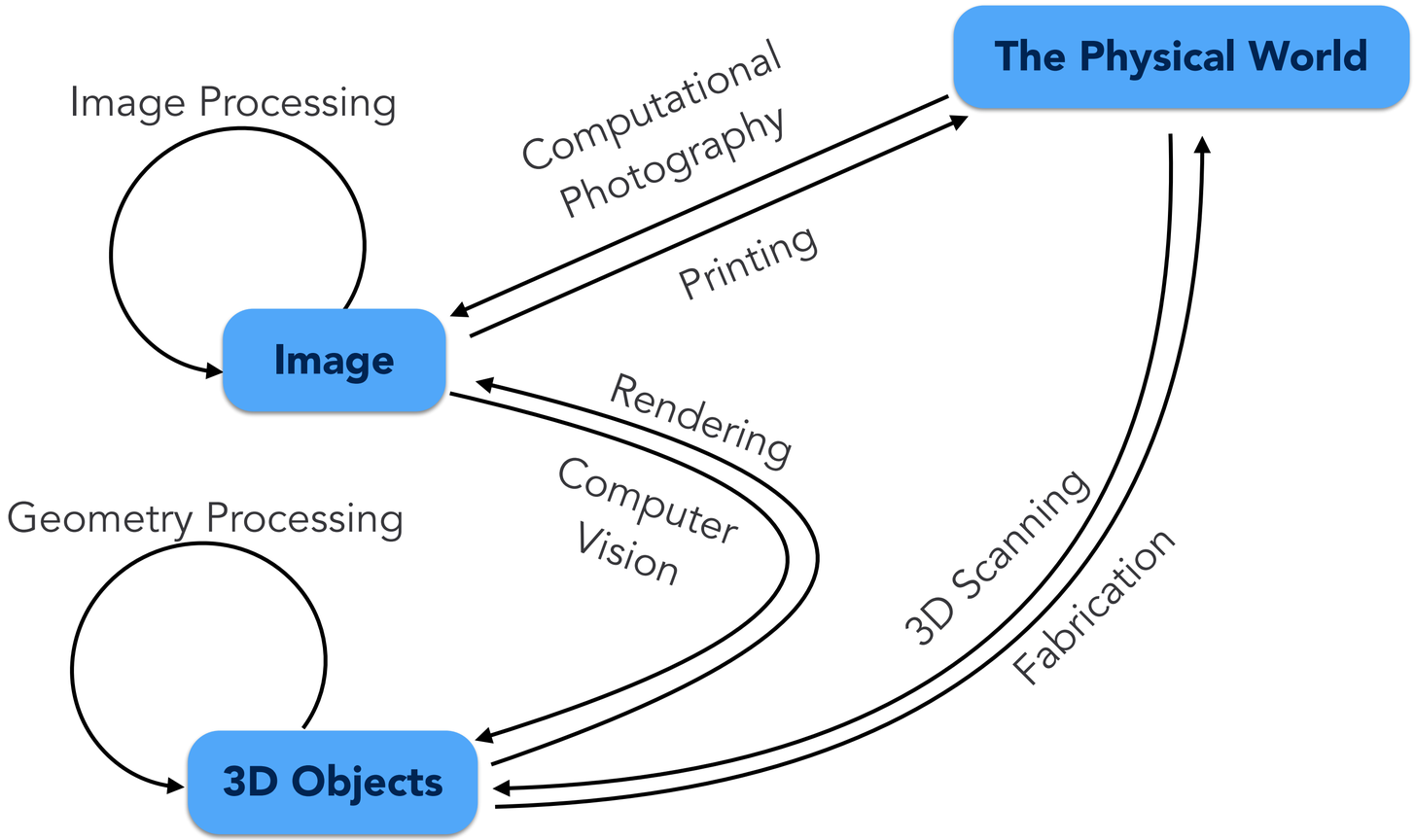


Image manipulation tools



“Visual Computing”



Computer Graphics Different Subfields

- 2D Imaging (still or video)
  - Creating, representing, editing
- 3D Objects
  - Creating, representing, editing, fabricating
- Rendering
- Computational Photography
- Fabrication
- Visualization
- Perception
- Animation
- Physics-based animation
- Geometry processing
- Physics-based simulations

What you will learn in this course

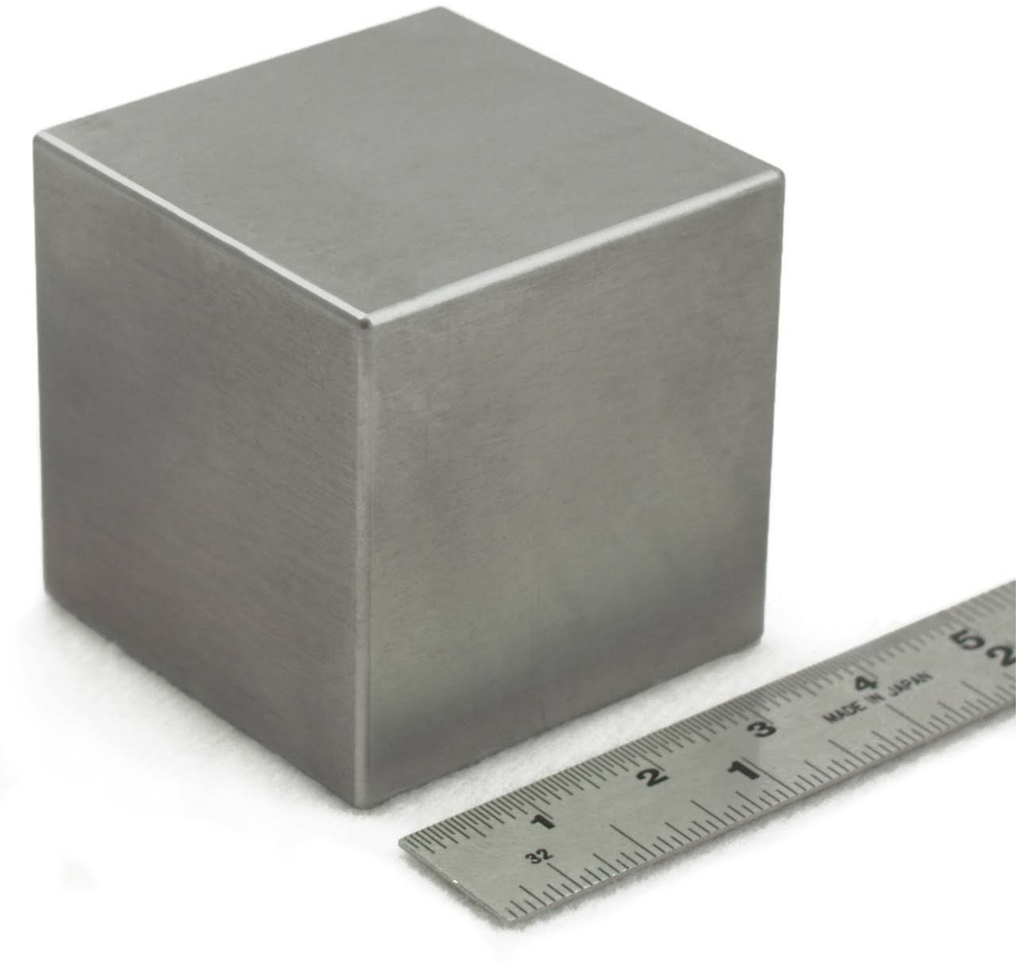
- Not a particular application, rather the fundamentals of computer graphics
- ‘Mathematical/algorithmic foundations’ for approaching problems
- ‘Systems’ view on how to implement things (fast, etc)

“Hello World” of Graphics

How to have a computer draw a cube?

- *Modeling*: how to describe the geometry of the cube

- *Rendering*: how to visualize (create an image of) the model

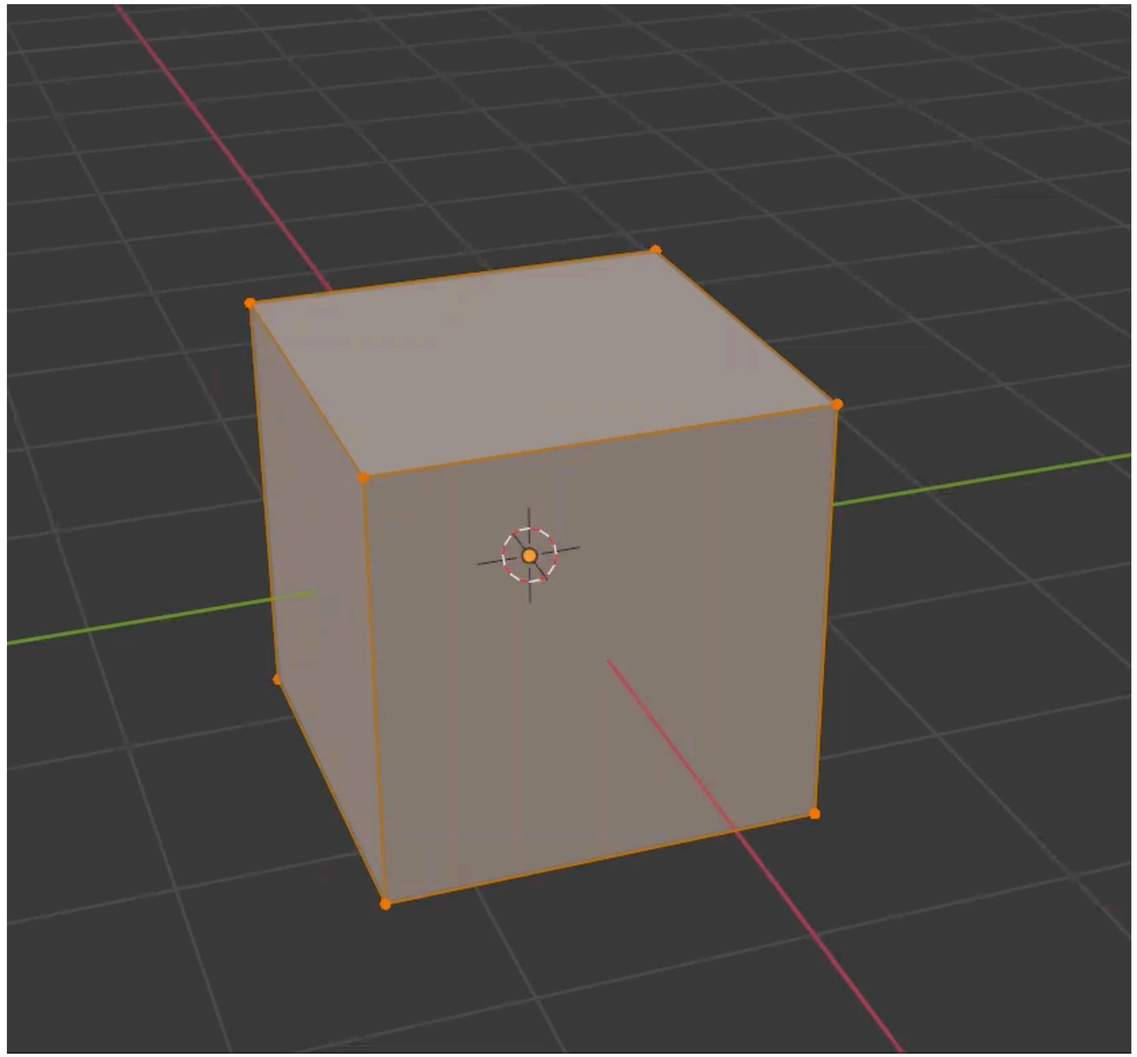


First: consider the geometry of a cube

- Suppose the cube is:
  - Centered at the origin
  - Has dimensions 2x2x2
  - Edges are aligned with x/y/z axis

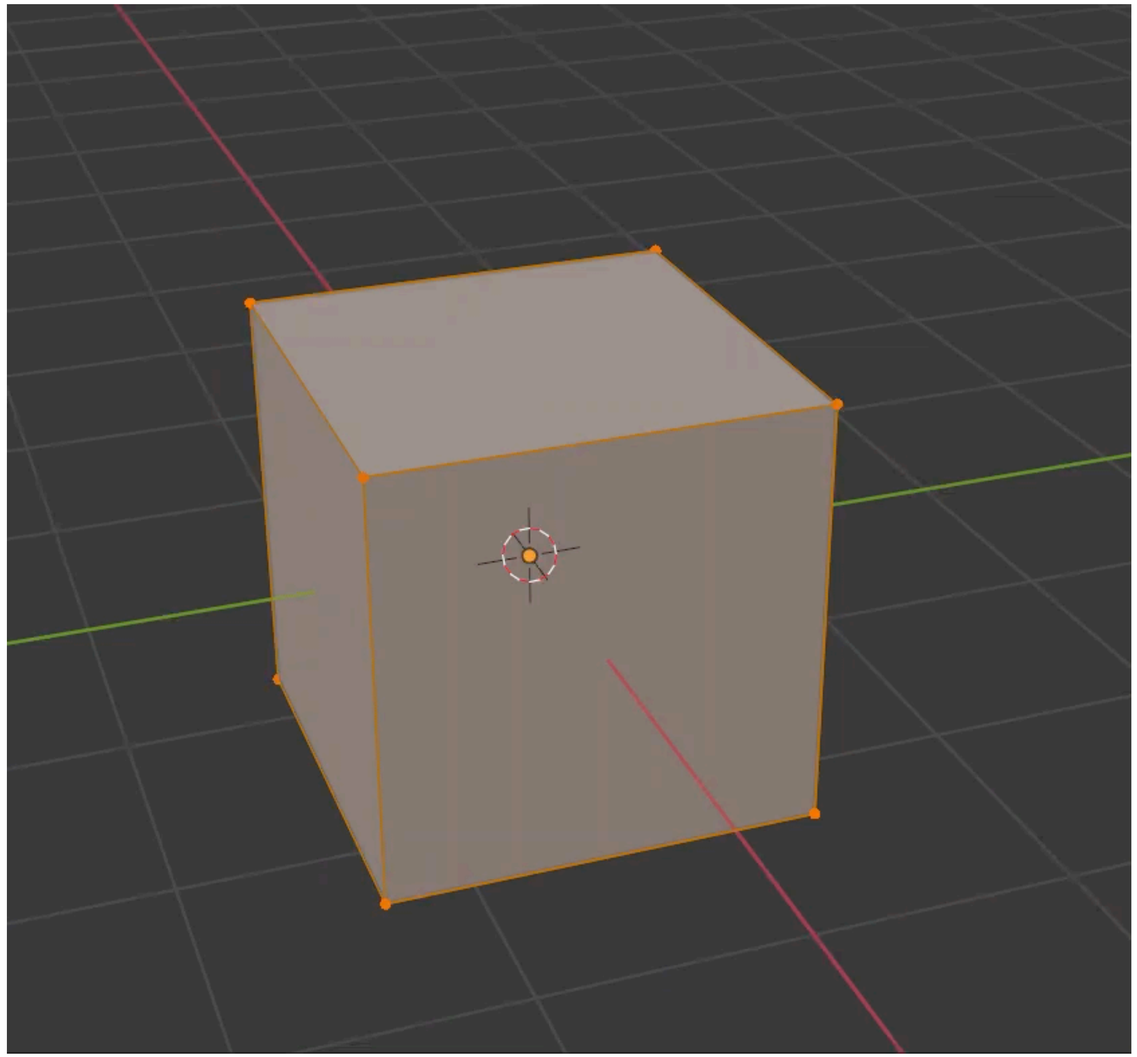
How to explicitly represent this in a computer?

- What would be a more generalizable representation?



First: consider the geometry of a cube

- Suppose the cube is:
  - Centered at the origin
  - Has dimensions 2x2x2
  - Edges are aligned with x/y/z axis



- What are the coordinates of the corners?

```
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 are the edges of cube?

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

How to draw this 3D cube as an Image

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
```

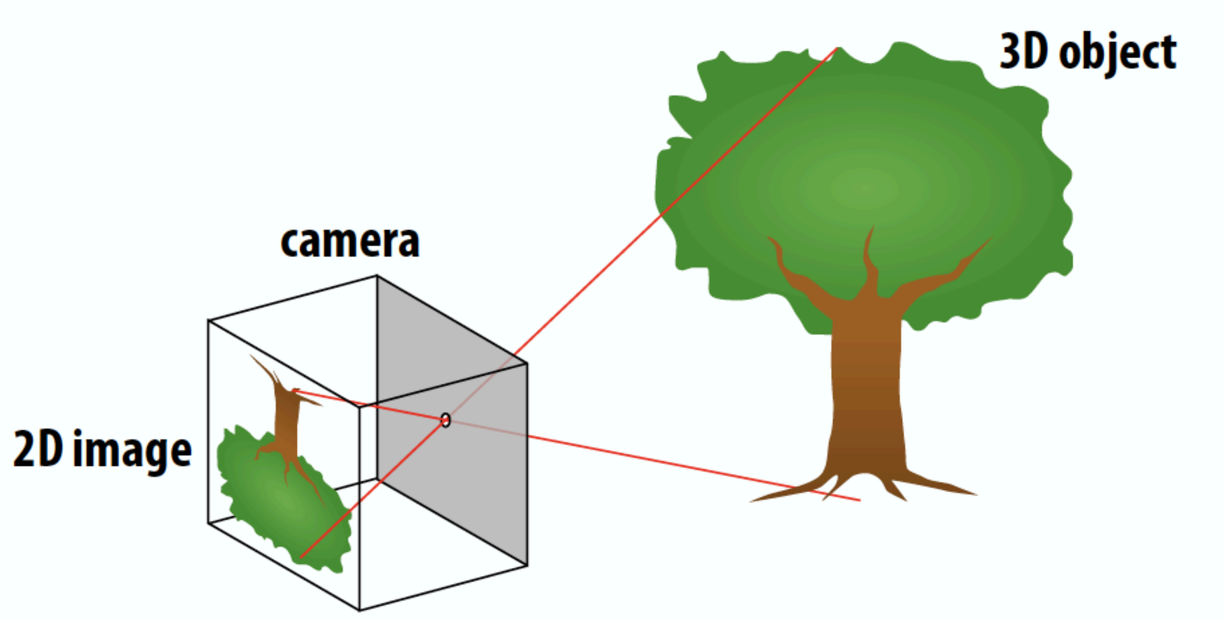
How would you approach it?

- you could throw away the z coordinate...
  - but that would not create interesting geometric perspective

- General Idea:
  - 1. Map 3D vertices to 2D points in the image
  - 2. Connect 2D points with straight lines

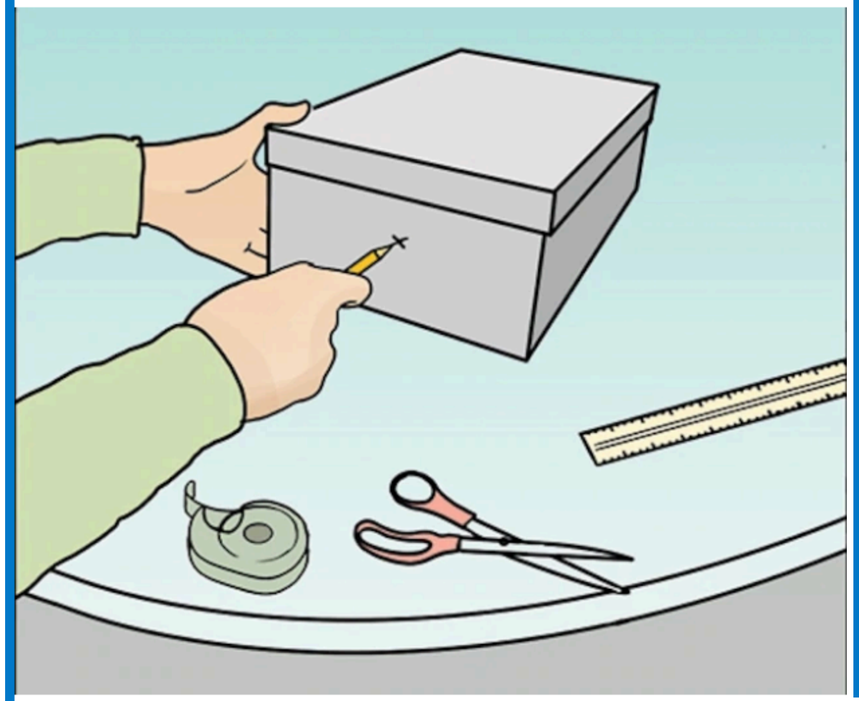
Perspective Projection

- Objects look smaller as they get further away “perspective”
- Parallel lines intersect eventually
- Why does this happen?
- Consider a simple “pinhole” model of a camera:

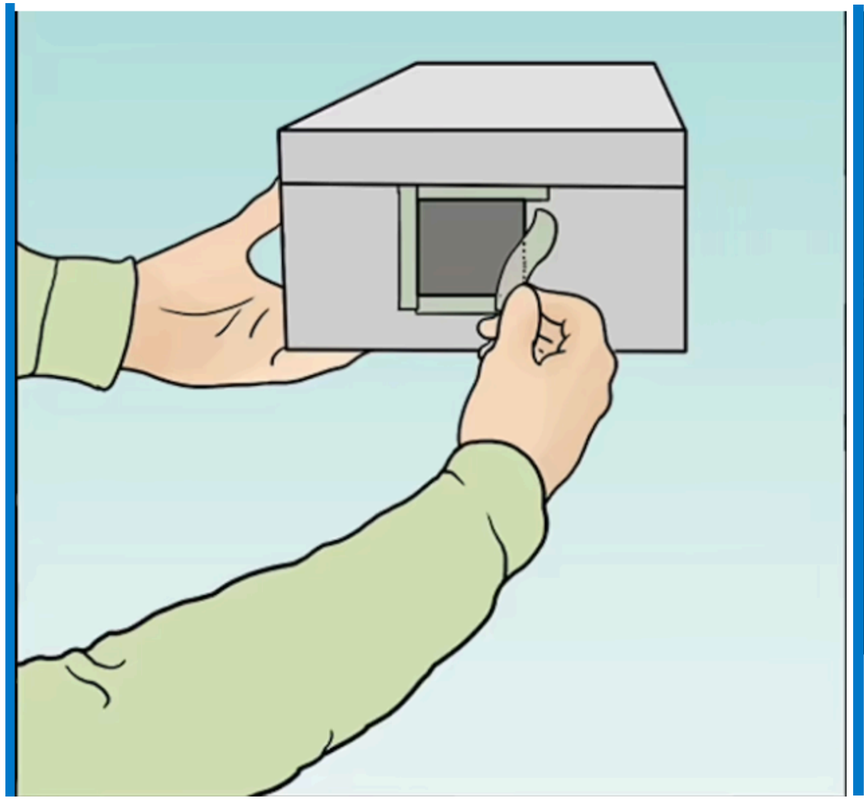


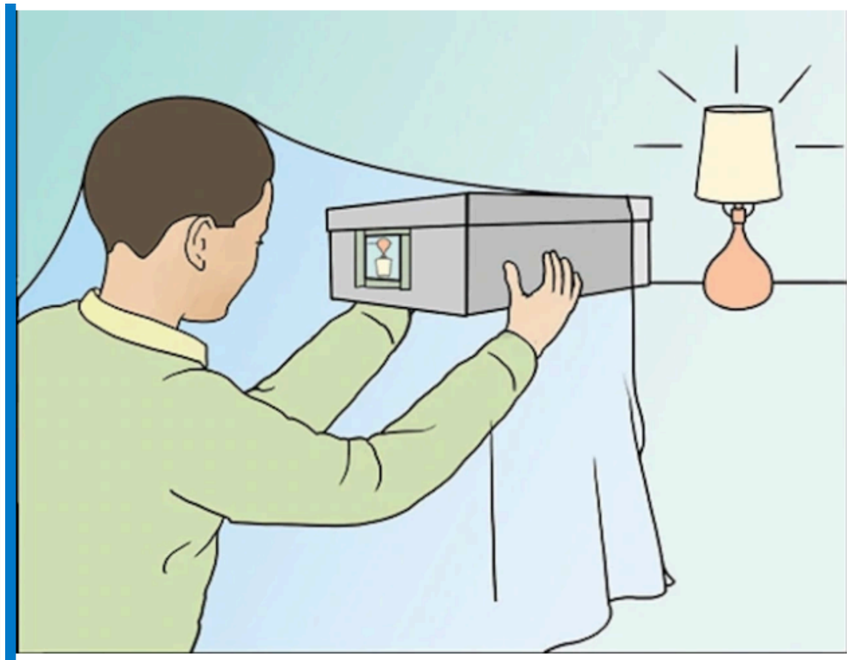
Pinhole camera

STEP ONE



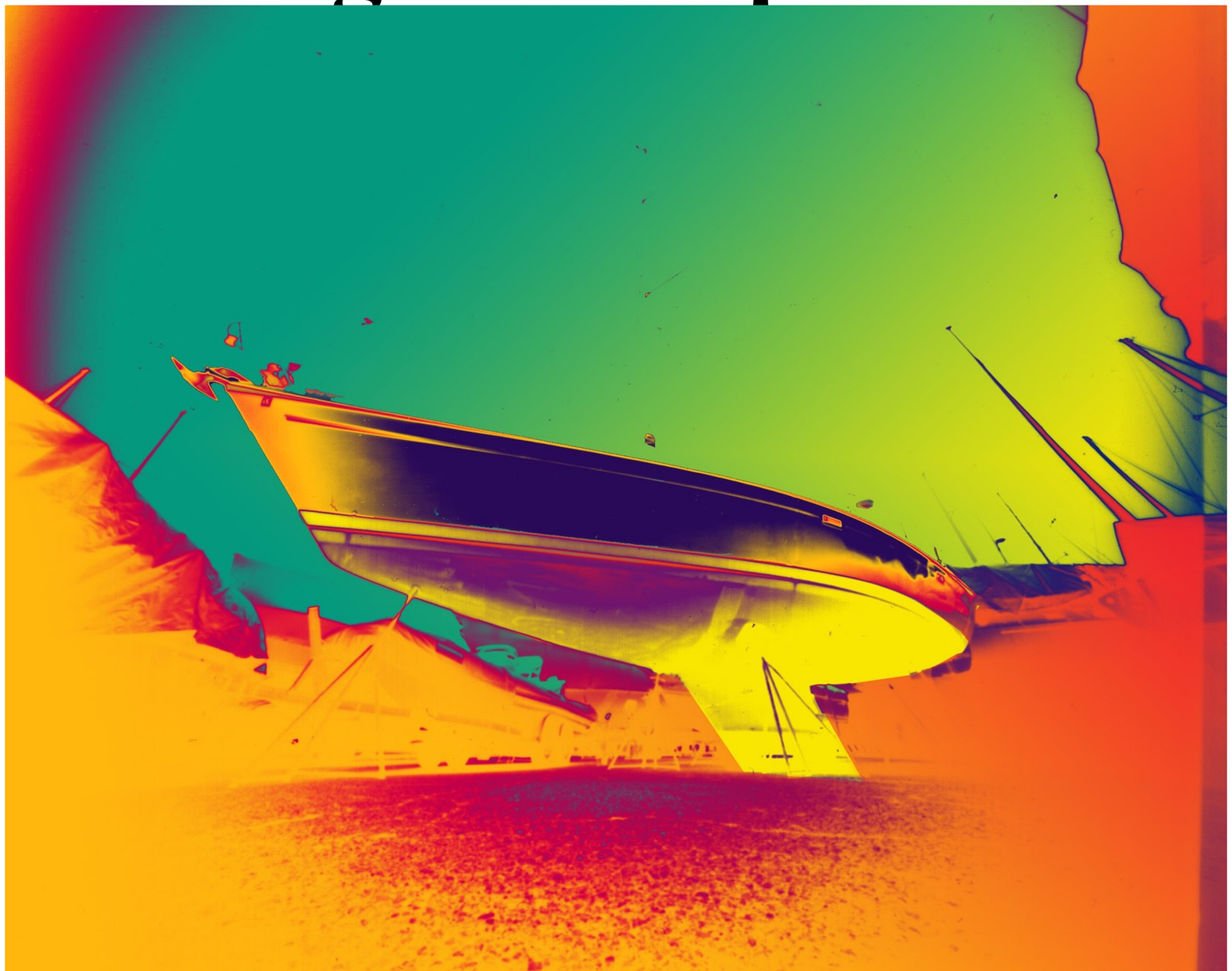
Use the point of a sharp pencil to punch a hole in one of the shorter ends of the shoe box.



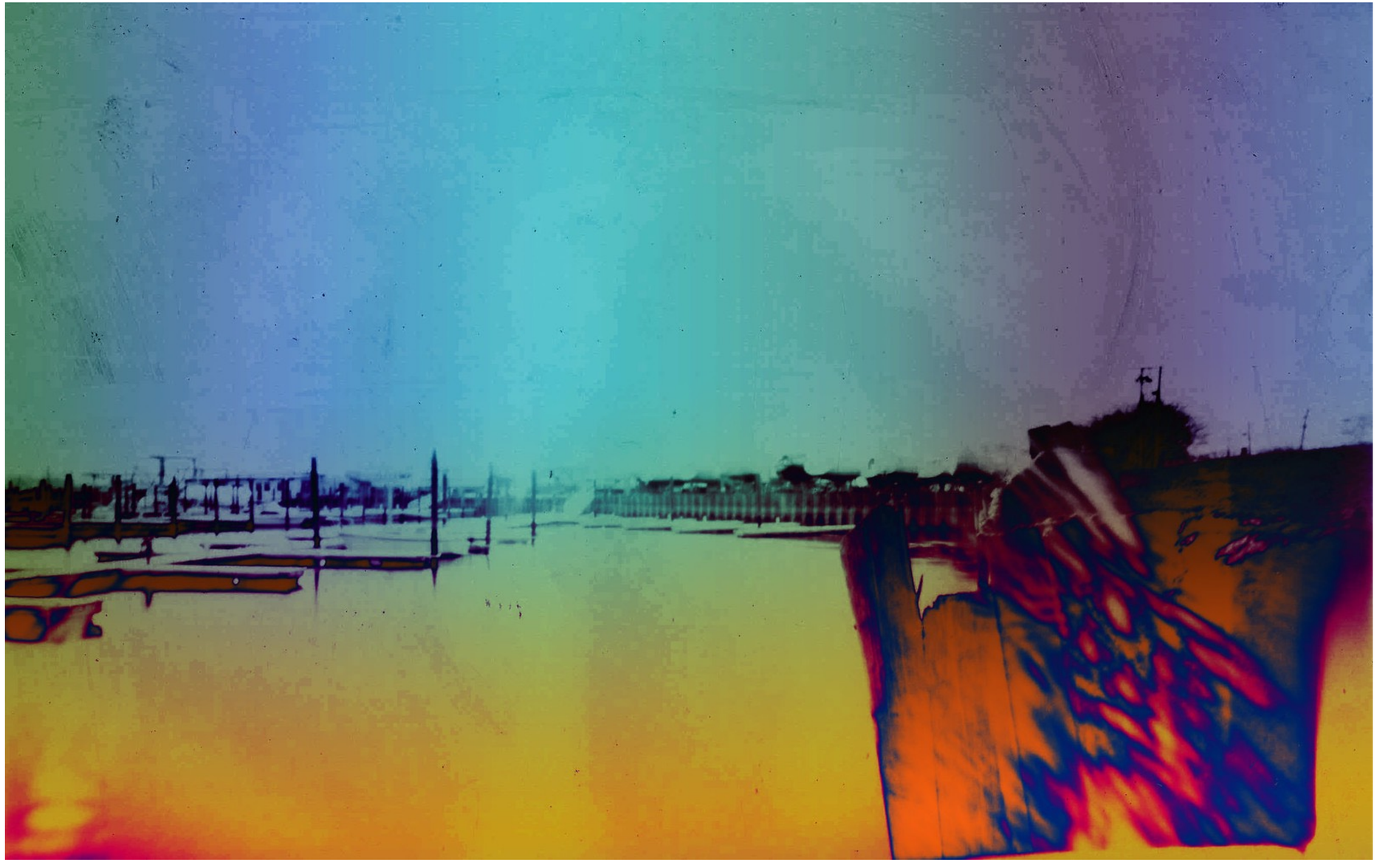


Hold your pinhole camera at arms length from your face and aim it at the lamp. Keep it steady until you see an upside-down image of the lamp.

Images from a pinhole



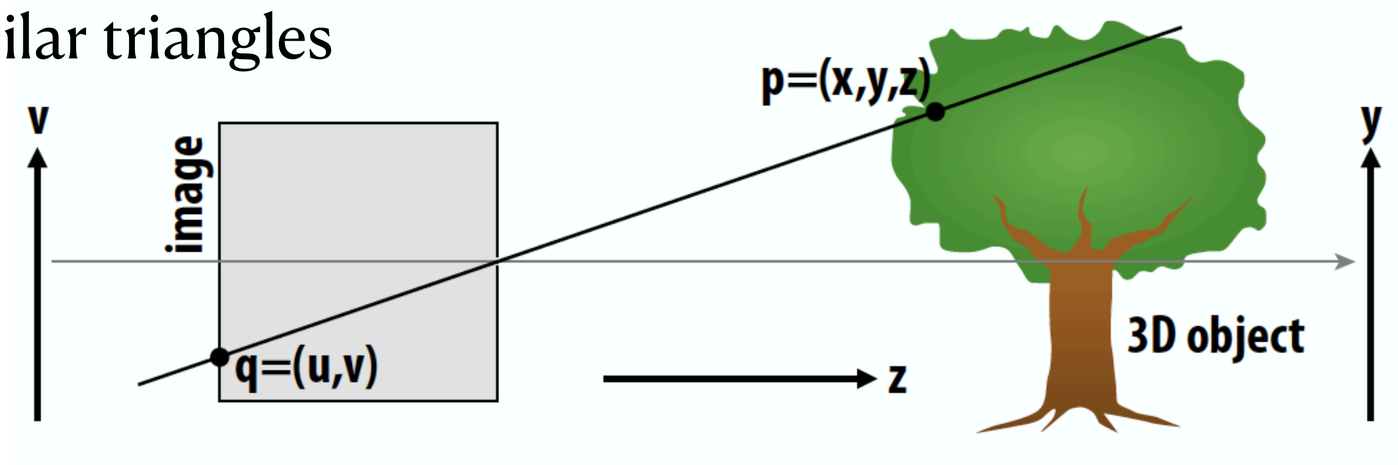
Images from a pinhole



Images from a pinhole

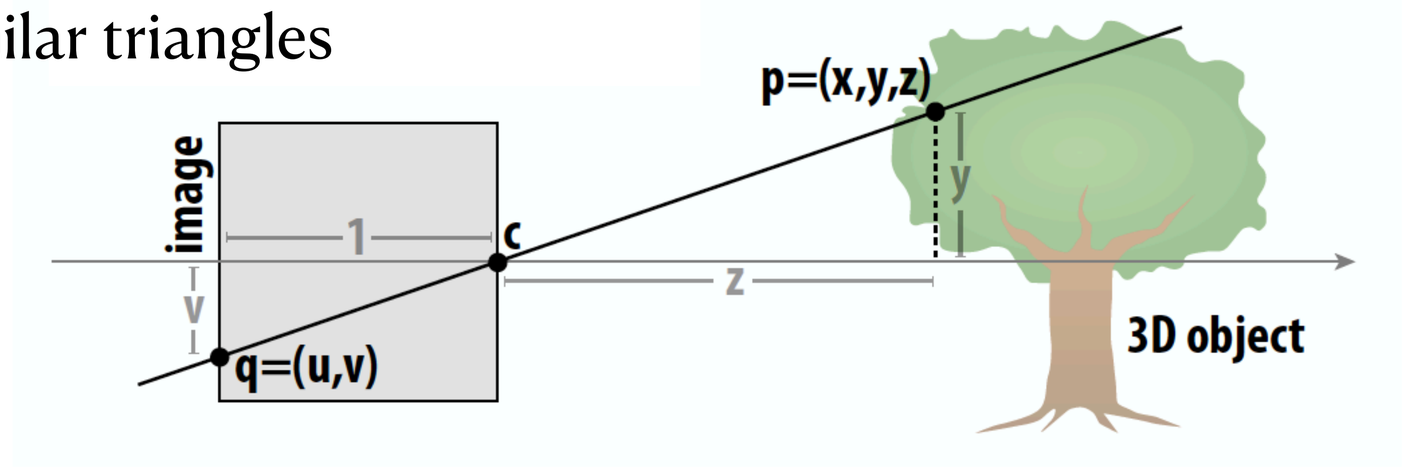
Perspective projection: side view

- Where exactly does a point p = (x,y,z) end up on the image?
- Call the image point q = (u, v)
- Notice two similar triangles



Perspective projection: side view

- Where exactly does a point p = (x, y, z) end up on the image?
- Call the image point q = (u, v)
- Notice two similar triangles

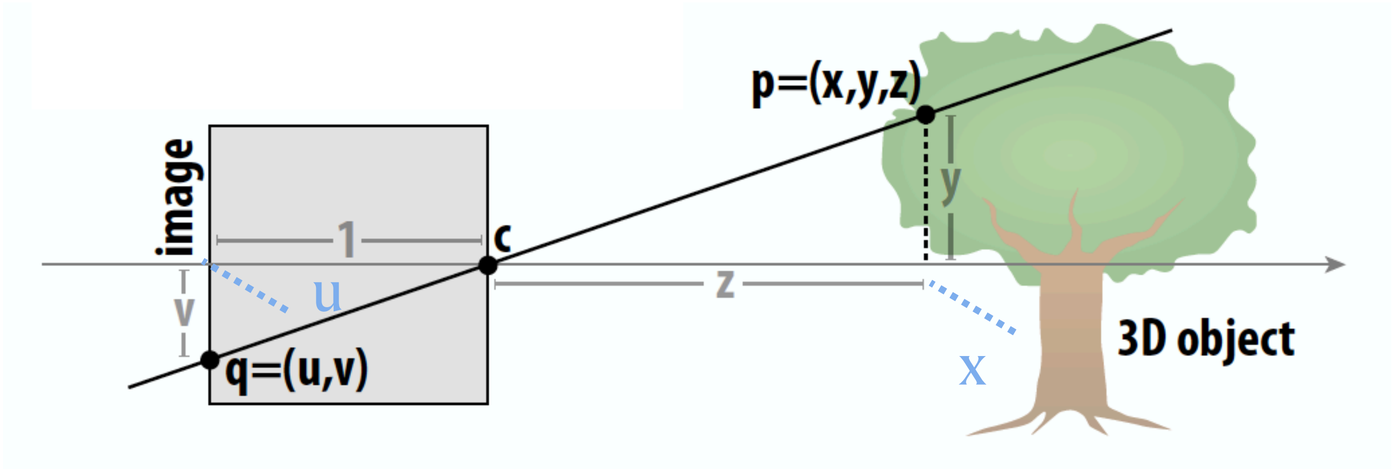


- The camera has unit size, origin is at pinhole c
- Then $\frac{v}{1} = \frac{y}{z}$, vertical coordinate is just the slope y/z
- Likewise, horizontal coordinate is u=x/z

Things look smaller as they get far away!

As Z gets bigger: (u,v) get smaller

How to calculate (u, v) explicitly



Gives 3D coordinates relative to the camera

Step 1: $p = (x, y, z) => (X, Y, Z) - c$

$$\frac{v}{1} = \frac{y}{z}$$

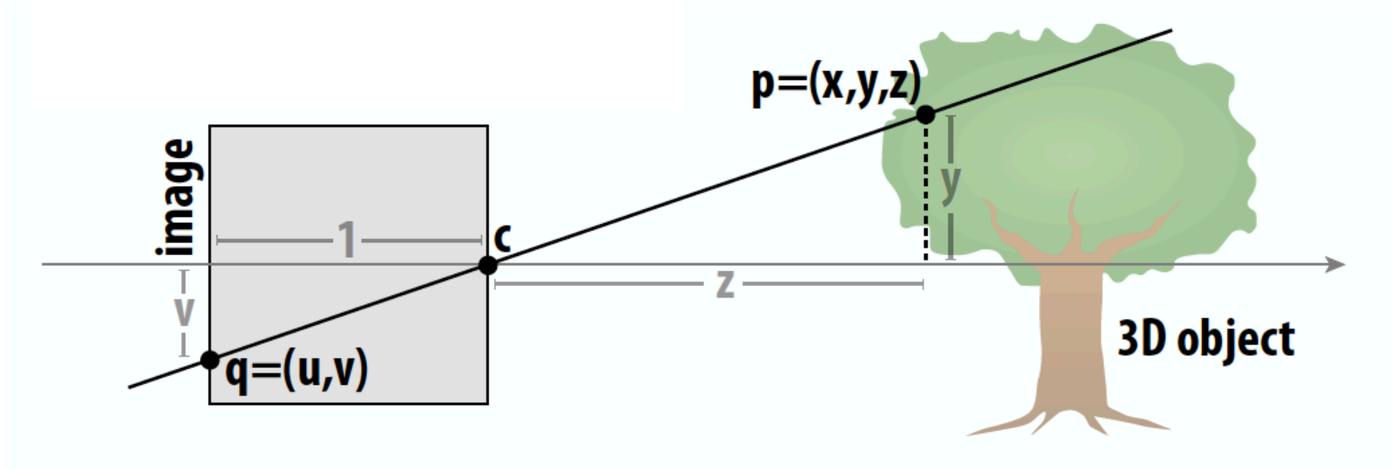
$$v = 1 \times \frac{y}{z}$$

$$\frac{u}{1} = \frac{x}{z}$$

$$u = 1 \times \frac{x}{z}$$

Visualize looking down the other direction

An example of projecting 3D to 2D



Assume 3D point: Assume camera at:

$(X, Y, Z) = (1,1,1)$ $c = (2,3,5)$

$$p => (1,1,1) - (2,3,5) = (-1, -2, -4)$$

$$u = \frac{-1}{-4} = \frac{1}{4} \qquad v = \frac{-2}{-4} = \frac{1}{2}$$

Simple Cube Drawer Routine

- Produce an algorithm to draw each vertex (8x), and then connect each edge
  - Run it once per vertex
  - Assume camera is at c=(2,3,5)
  - Convert (X,Y,Z) of both endpoints to (u,v)
  - Subtract camera c from vertex (X, Y, Z) to get (x, y, z)
  - Divide (x,y) by z to get (u,v)
- Go over all 12 edges, draw lines 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
```

Simple Cube Drawer Routine

- Produce an algorithm to draw each vertex (8x), and then connect each edge
  - Run it once per vertex
  - Assume camera is at c=(2,3,5)
  - Convert (X,Y,Z) of both endpoints to (u,v)
  - Subtract camera c from vertex (X, Y, Z) to get (x, y, z) (1,1,1) - (2,3,5) =(-1, -2, -4)
  - Divide (x,y) by z to get (u,v) (-1/-4, -2/-4) =(1/4, 1/2)
- Go over all 12 edges, draw lines 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
```

Simple Cube Drawer Routine

- Produce an algorithm to draw each vertex (8x), and then connect each edge
  - Run it once per vertex
  - Assume camera is at c=(2,3,5)
  - Convert (X,Y,Z) of both endpoints to (u,v)
  - Subtract camera c from vertex (X, Y, Z) to get (x, y, z)
  - Divide (x,y) by z to get (u,v)
- Go over all 12 edges, draw lines between (u1,v1) and (u2, v2)

(1,1,1) - (2,3,5) =(-1, -2, -4)

(-1/-4, -2/-4) =(1/4, 1/2)

(-1,1,1) - (2,3,5) =(-3, -2, -4)

(-3/-4, -2/-4) =(3/4, 1/2)

VERTICES

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

EDGES

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

Activity: YOU calculate it!

LEFT SIDE OF ROOM: calculate CDE
RIGHT SIDE OF THE ROOM: calculate FGH

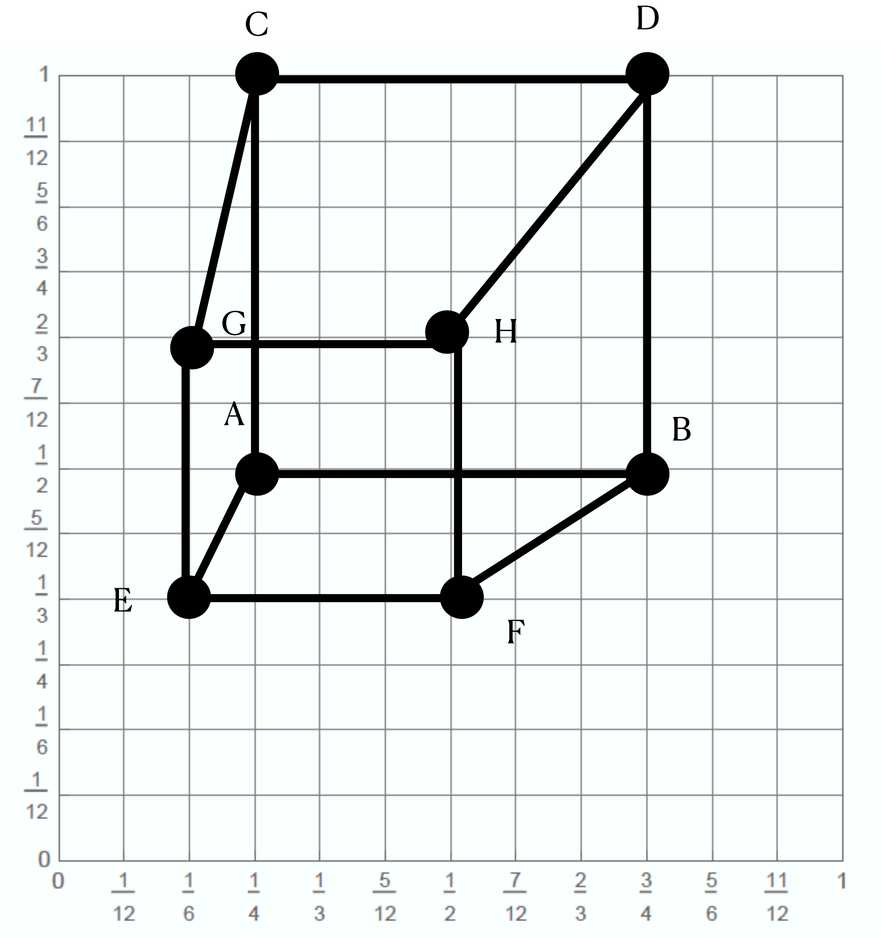
Camera center c=(2,3,5)

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



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)

First graphics algorithm!

Digital information

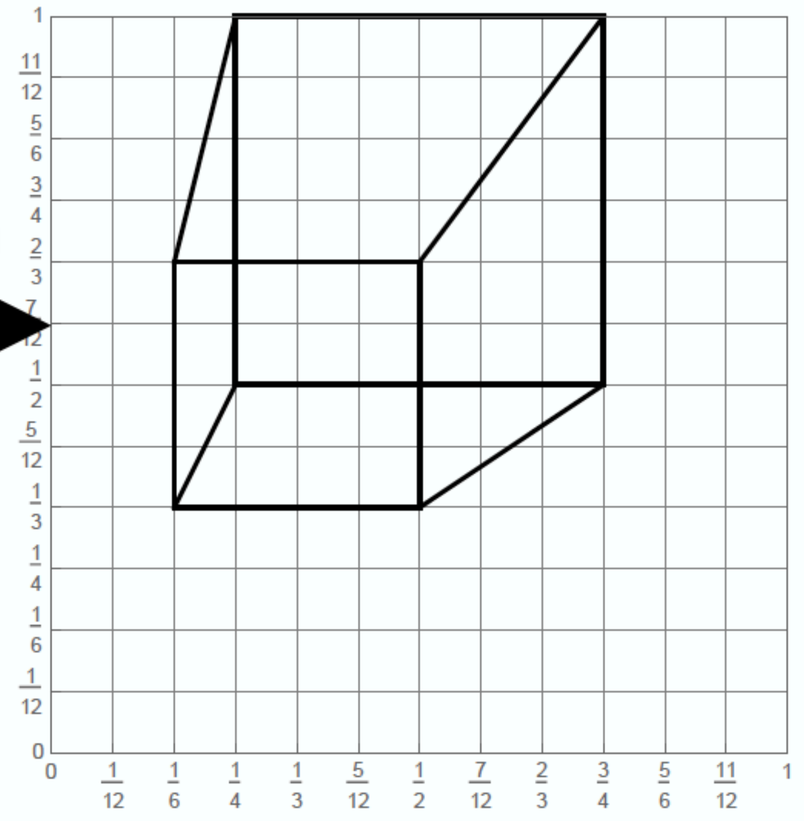
```
VERTICES
A: ( 1, 1, 1 )
B: (-1, 1, 1 )
C: ( 1,-1, 1 )
D: (-1,-1, 1 )
E: ( 1, 1,-1 )
F: (-1, 1,-1 )
G: ( 1,-1,-1 )
H: (-1,-1,-1 )

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

CAMERA
C = (2,3,5)
```

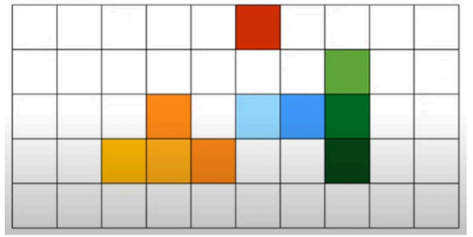
computation

Visual information



But how do we draw lines on a computer?

...what is an image?



- Image represented as a 2D grid of pixels
- Each pixel takes on a unique color value

BTW:
Pixel = Pixel element

Digital Images

Basically a matrix/array

I(0,0)	I(1,0)	I(2,0)	I(3,0)
I(0,1)	I(1,1)	...	...
I(0,2)	...	I(2,2)	...
I(0,3)	...	..	...
I(0,4)	...	...	...

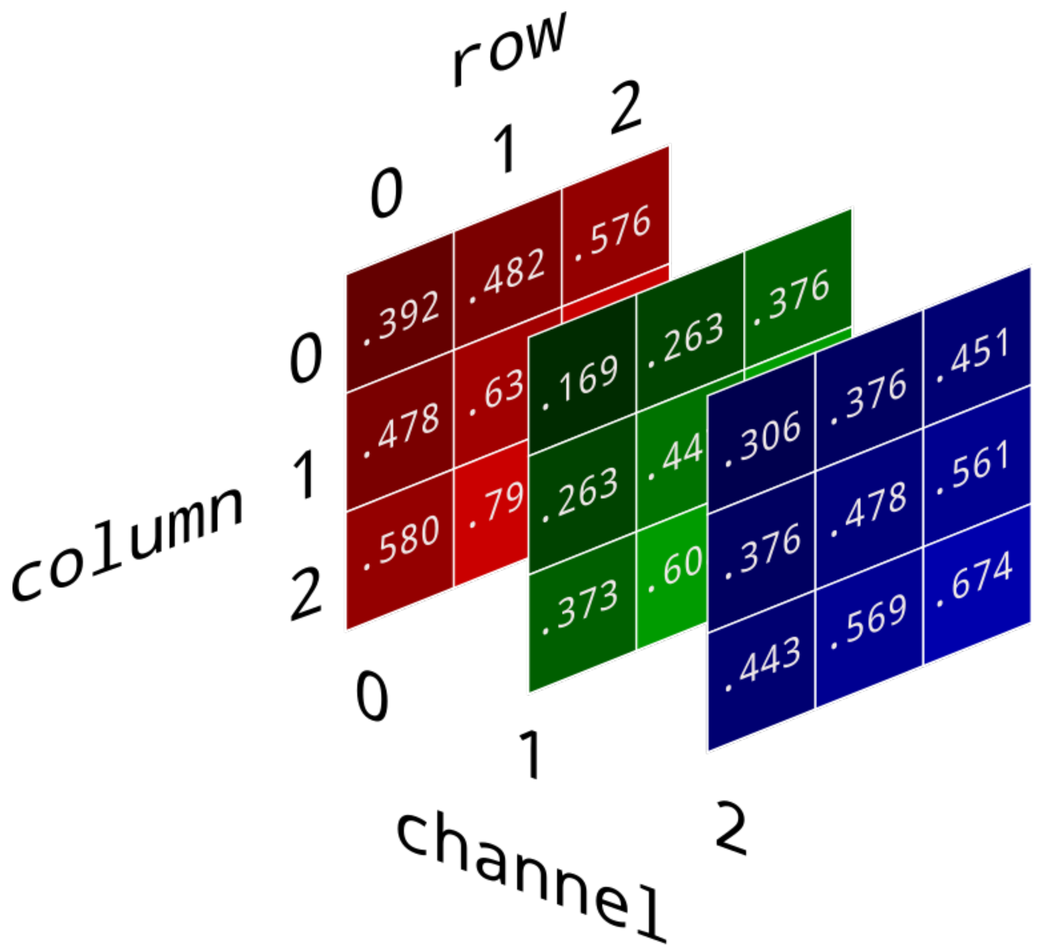
I(x, y)

I(column, row)

Digital Images

Basically a matrix:

HxWx3



I(x, y)

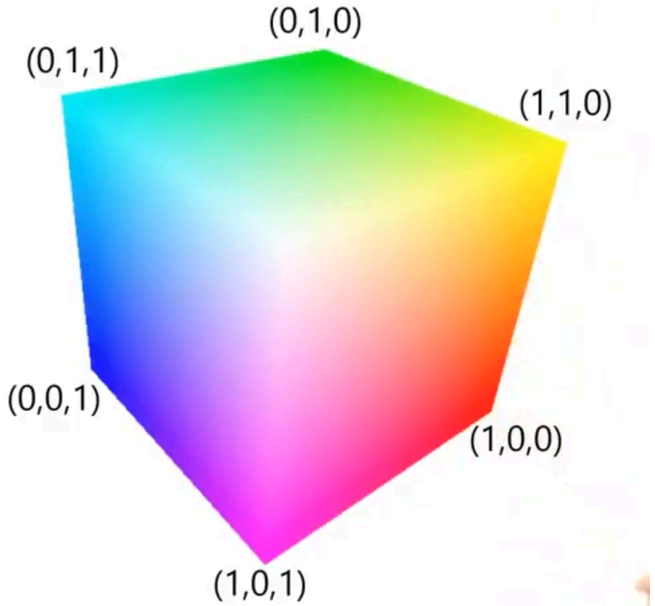
I(column, row)

Color

Color

RGB values in [0, 1]

black = (0,0,0)
red = (1,0,0)
green = (0,1,0)
blue = (0,0,1)
yellow = (1,1,0)
magenta = (1,0,1)
cyan = (0,1,1)
white = (1,1,1)



Color

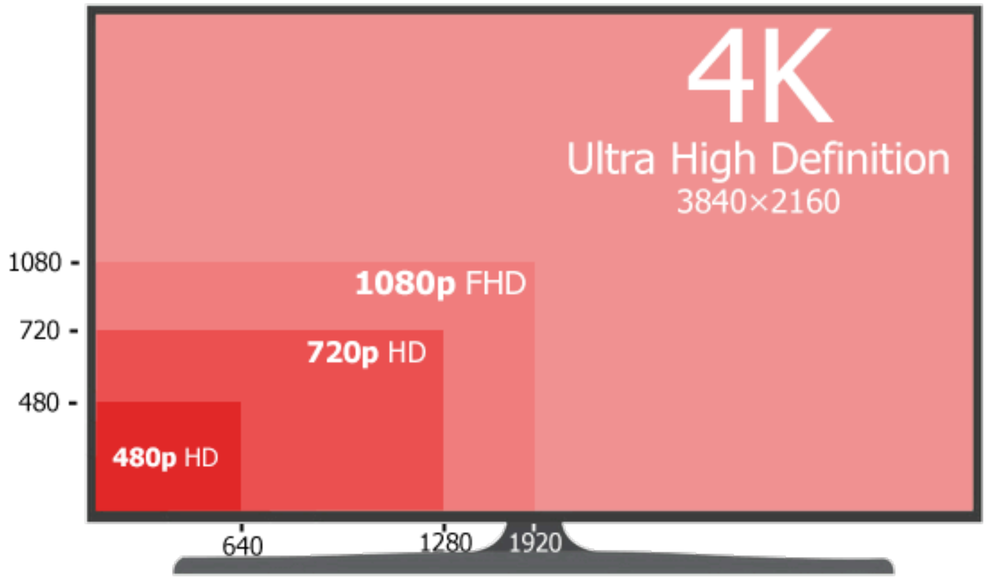
RGB values in [0, 1]

LDR Storage:

- **8 bits / color channel (0 to 255)**
- 16 bits / color channel (0 to 65536)

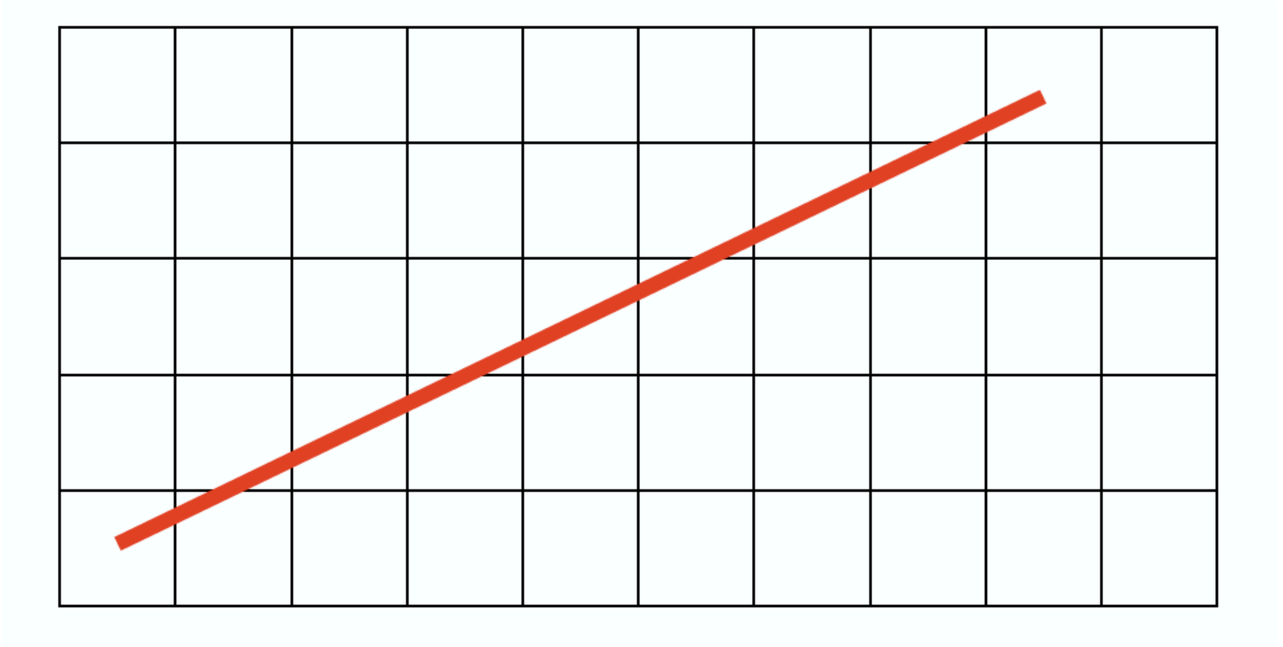
Image files

- Image resolution is the number of pixels
- Common image formats:
- JPEG (Joint Photographic Experts Group)
  - -> lossy compression
- GIF (Graphics Interchange Format)
  - -> lossless compression
- PNG (Portable Network Graphics) “ping”
  - -> lossless compression



What pixels should we color in to depict a line?

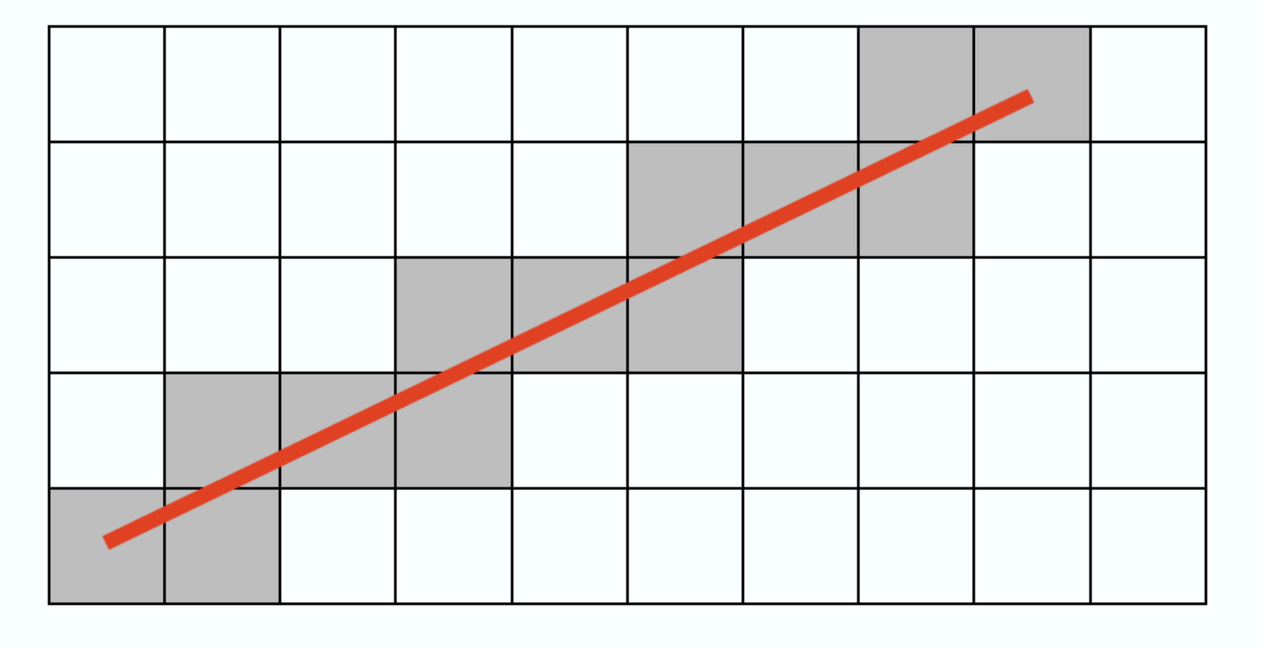
- **Rasterization**: process of converting a continuous object to a discrete representation on a raster grid (pixel grid)



What do you think?

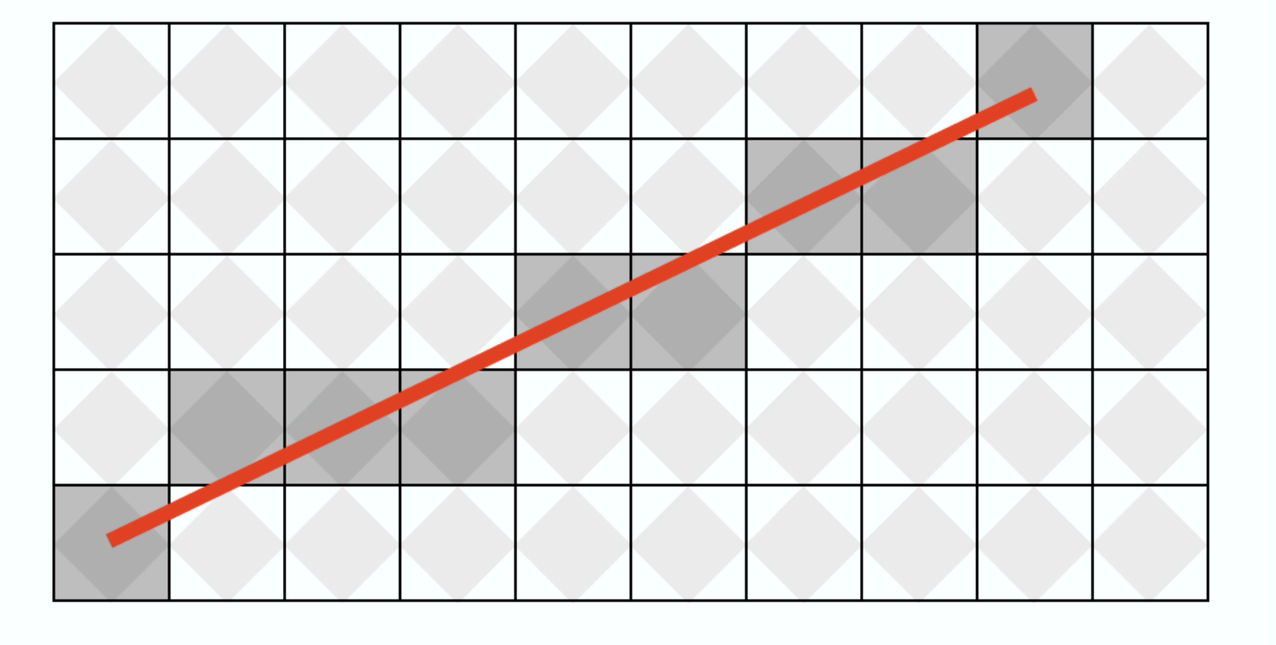
What pixels should we color in to depict a line?

- 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 about thickness?



Is there a right answer?

No!

- In this case: many different ways to discretize a line (*different answers*)
  - Thickness, etc
- In general in graphics & in your assignments:
- different ways of arriving at the *same answer*

How do we implement this idea algorithmically?

- Given a particular rule (e.g., diamond rule) what's the algorithmic pseudo code?
- Could check every single pixel in the image (n width) to see if it meets the condition...
- What's the complexity?
- O(n^2) in image vs. at most O(n) "activated" pixels
- In principle: should be able to reduce complexity
  - think: different "acceleration" strategies

Fundamentally the idea of computer graphics

Digital information

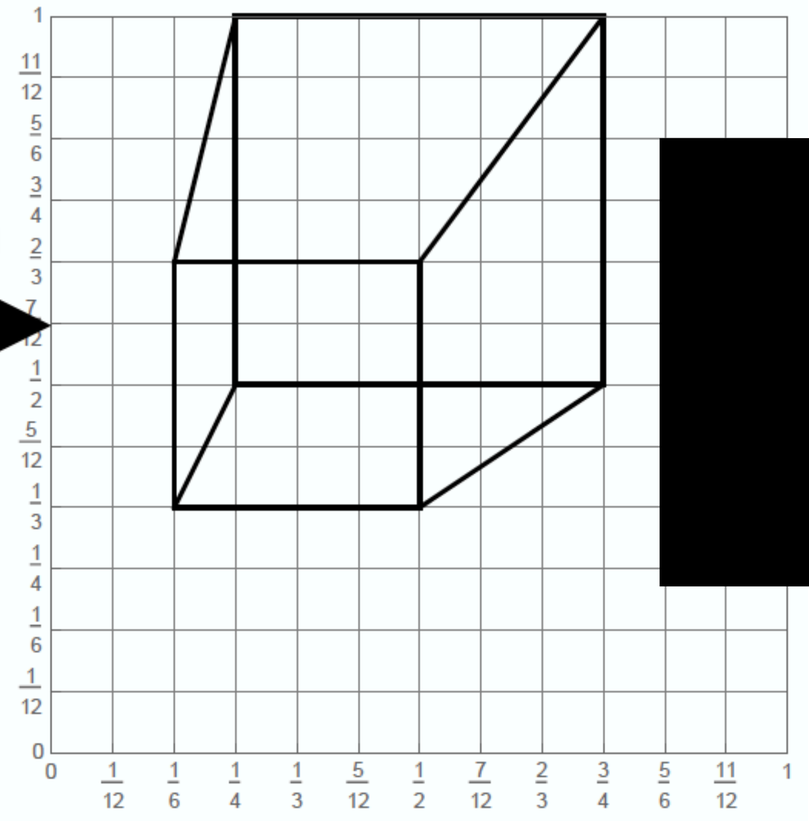
```
VERTICES
A: ( 1, 1, 1 )
B: (-1, 1, 1 )
C: ( 1,-1, 1 )
D: (-1,-1, 1 )
E: ( 1, 1,-1 )
F: (-1, 1,-1 )
G: ( 1,-1,-1 )
H: (-1,-1,-1 )

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

CAMERA
C = (2,3,5)
```

computation

Visual information



Geometry
Materials
Lights
Cameras
Motion
....

Logistical Note

- I am traveling for SIGGRAPH technical papers committee meeting, so:

- my PhD student Dale will give the lecture this Thursday

- I will not hold office hours this week

- I might be slow to respond to emails Wed-Fri

Course Assignments

A note on programming Graphics in Python

- If you go work in the CG industry it is very possible you will not use Python

BUT... I decided to use it for this class because:

- Python IS the default language when combining CG & ML

- It is a high-level language that will help you focus on the **CONCEPTS** of CG

- We can still think about algorithmic acceleration strategies... you will see that in your assignments

- I don't want you to be too focused on low-level programming/memory management etc. rather the high level concepts of Graphics!

Surveys

Please fill out a short google anonymous survey after you complete each assignment.

It will help us improve

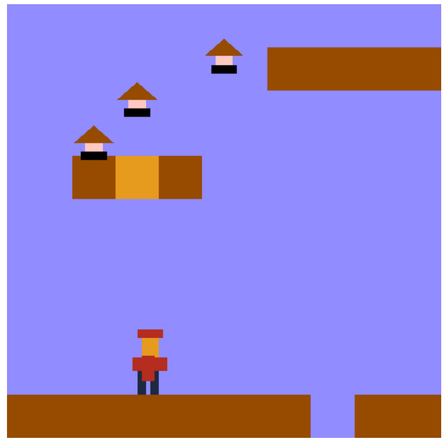
Keep / add more things you like

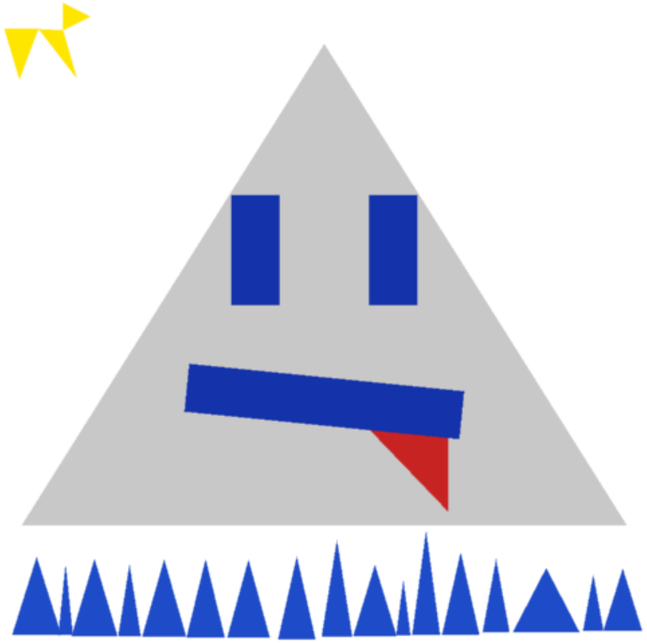
etc

Assignment 1

- 2D Rasterization

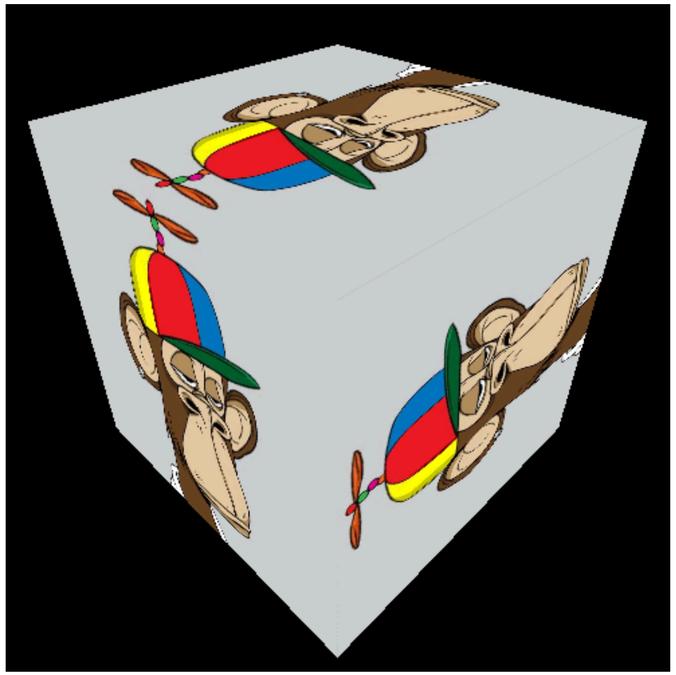
- SVG (vector graphics)





Assignment 2

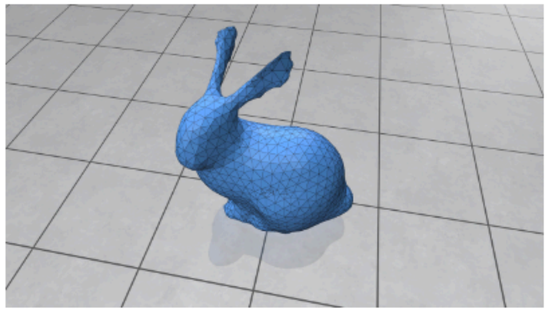
- camera viewing
- 3D geometry
- 3D rasterization



Assignment 3

- 3D Mesh editing
- 3D mesh data structures

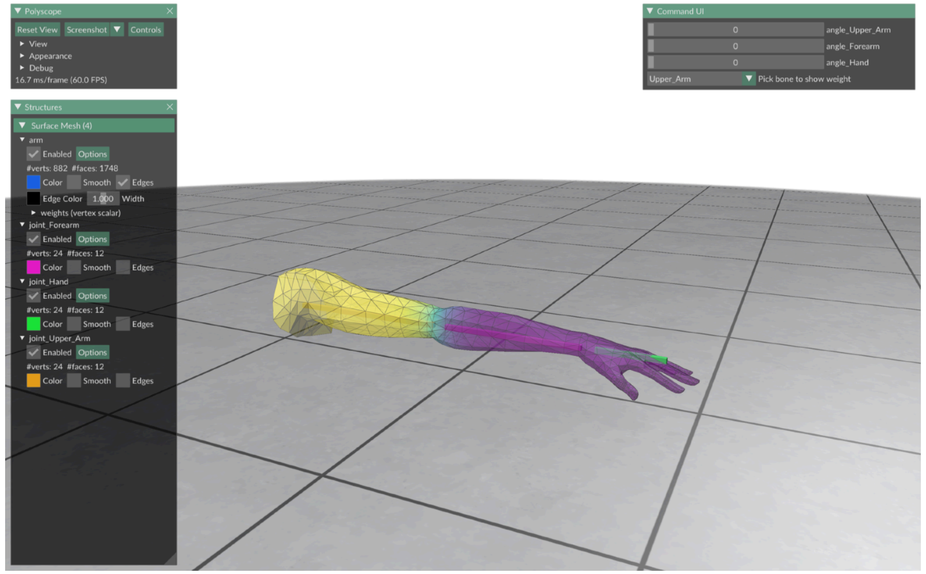
(a) Original Mesh (b) After 1 smoothing iteration



(c) After 2 smoothing iterations

Assignment 4

- animation of 3D shapes
- skinning and rigging



Final Project (Part A)

- Familiarity with rendering package

- Bsplines and color interpolation

Final Project

Demos