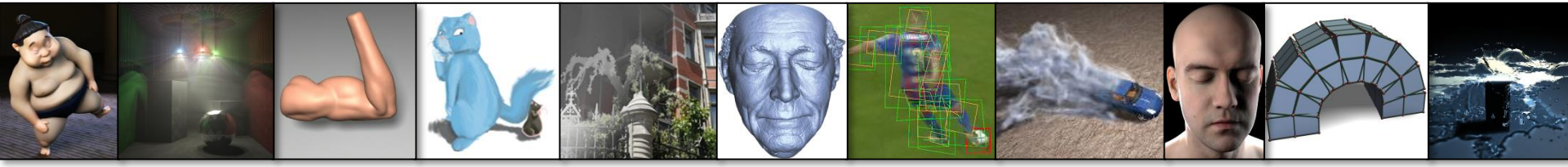
Visual Computing:
Welcome to Computer Graphics!



Prof. Dr. Stelian Coros

ETH zürich

cgl

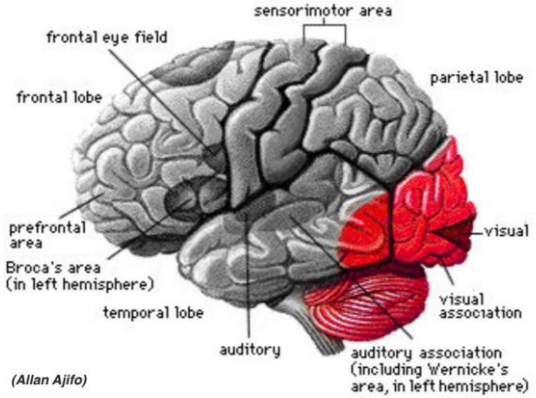
What is Computer Graphics?

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

Why visual information?

Humans are visual creatures!

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



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

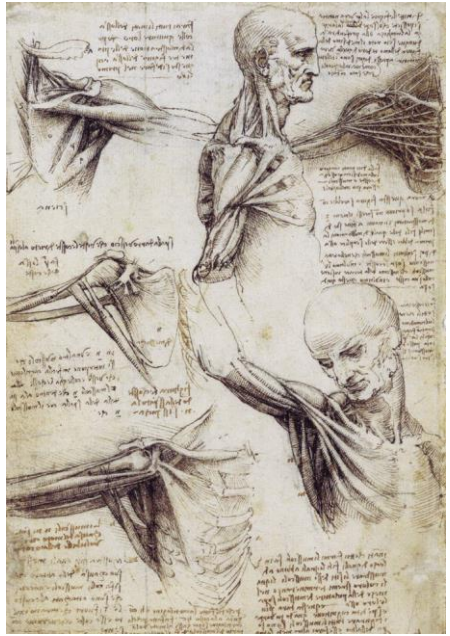
History of visual depiction

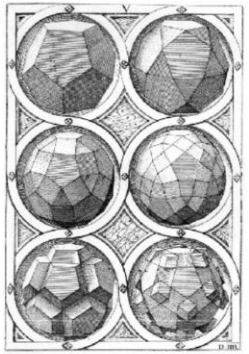
Humans have always been visual creatures!

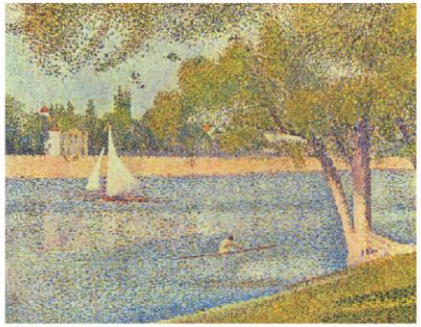
Indonesian cave painting (~38,000 BCE)

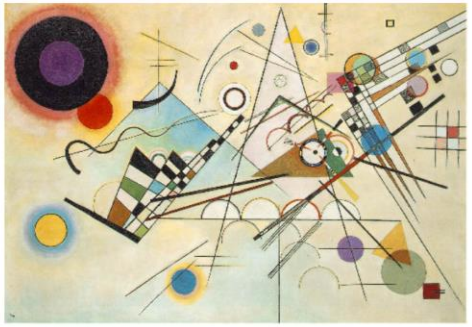
History of visual depiction

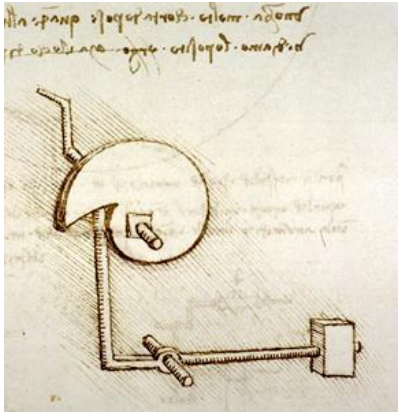
Useful to represent MANY different types of information…











History of visual depiction

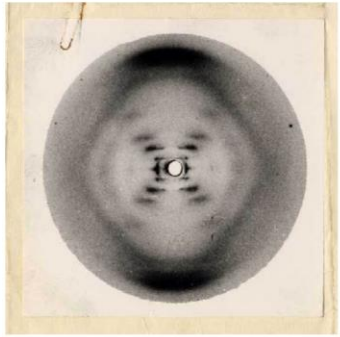
Of course, not restricted to “flat” representations

History of visual depiction: photography / imaging

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

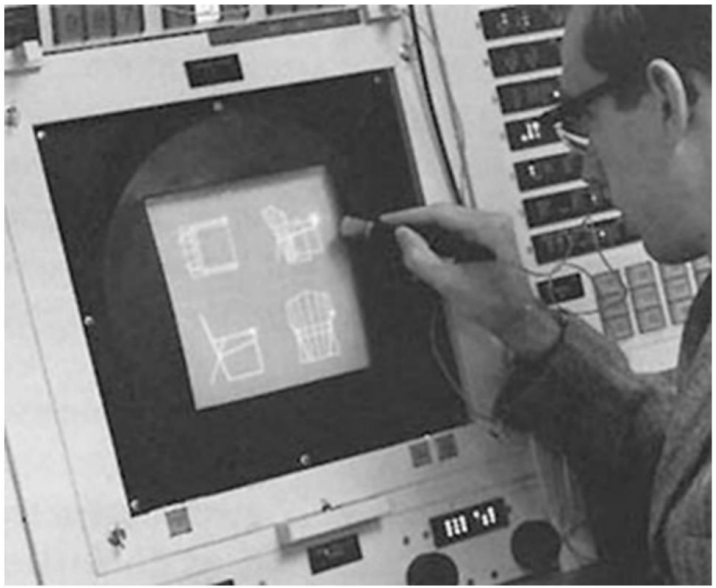
Joseph Niépce, "View from the Window at Le Gras" (1826)

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

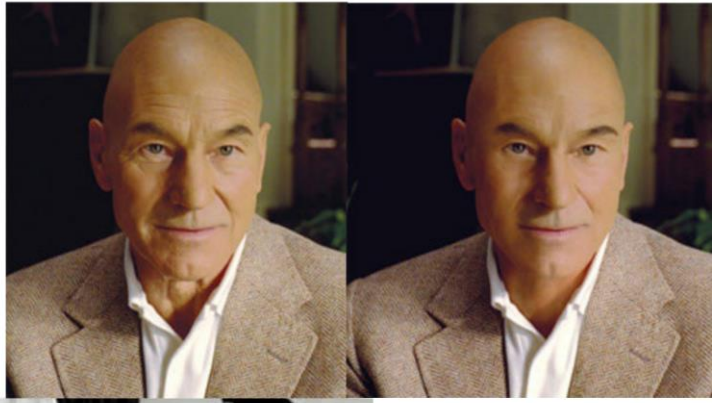
Nowadays, Computer Graphics is **everywhere**!

Applications

Entertainment

Entertainment – not just cartoons!

Entertainment – not just cartoons!



Sadly, also for Fake News…

https://www.youtube.com/watch?v=AmUC4m6w1wo

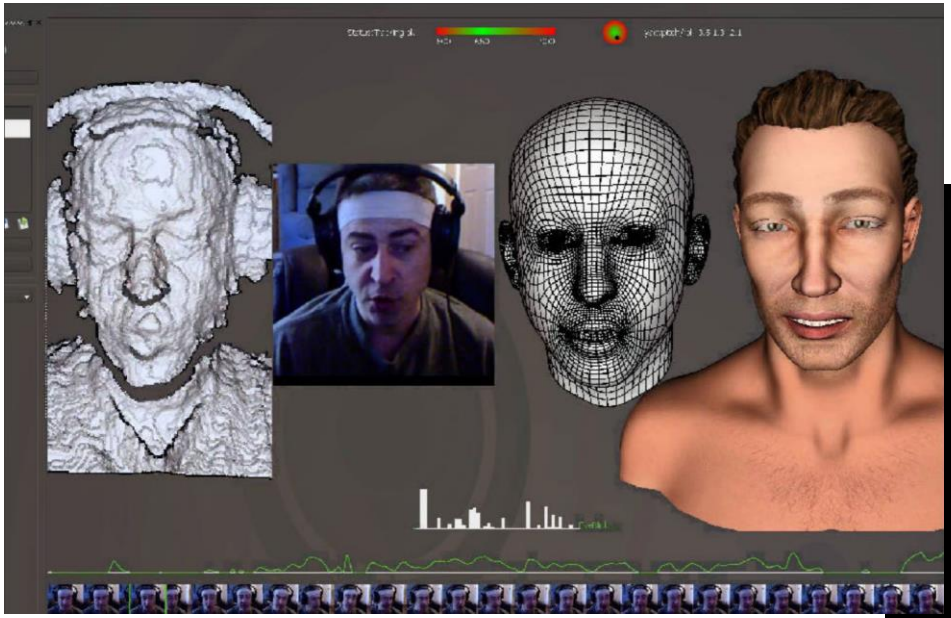
Applications

Video games – not just entertainment!

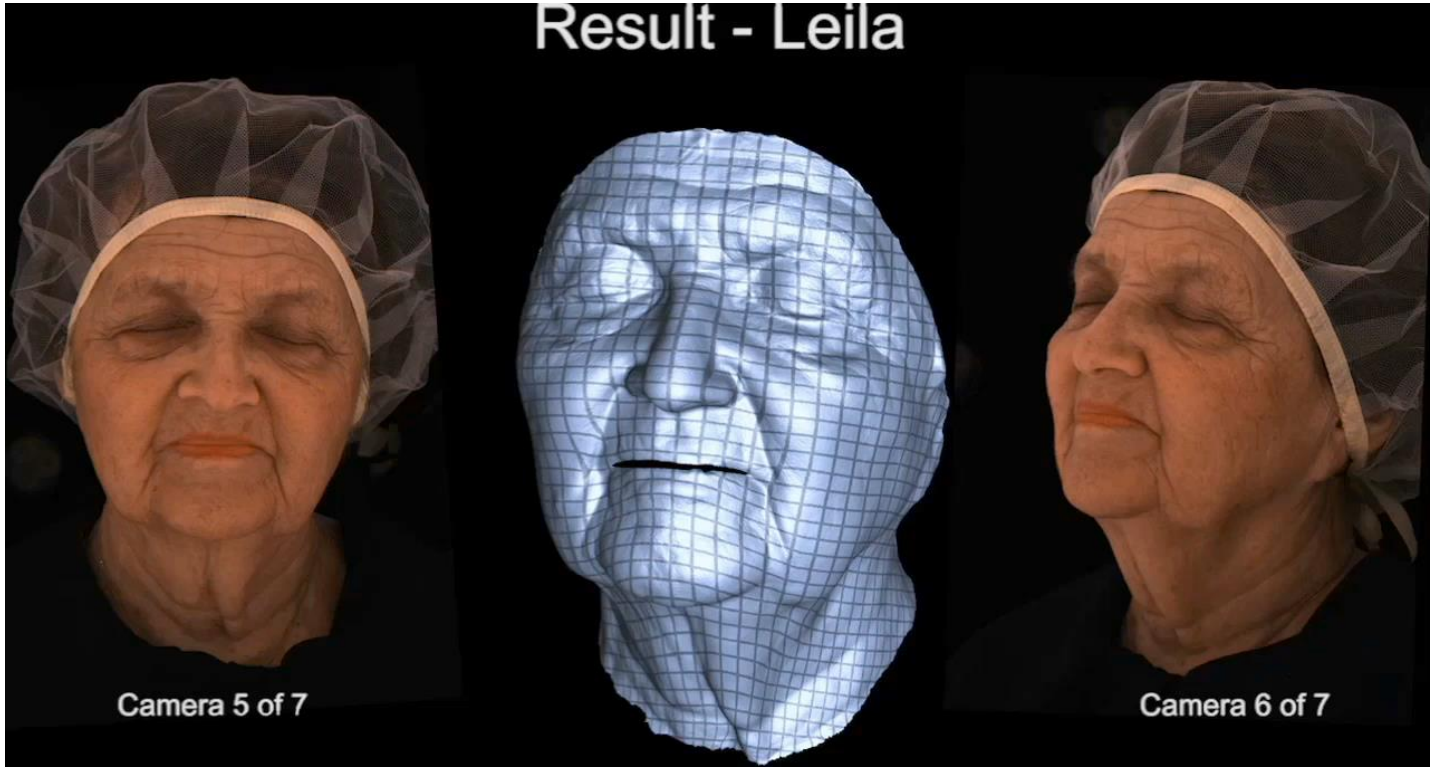
https://youtu.be/zKu1Y-LlfNQ?t=118

Applications

Communication and social interactions

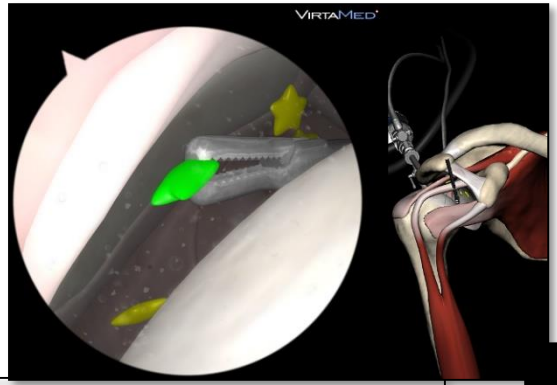


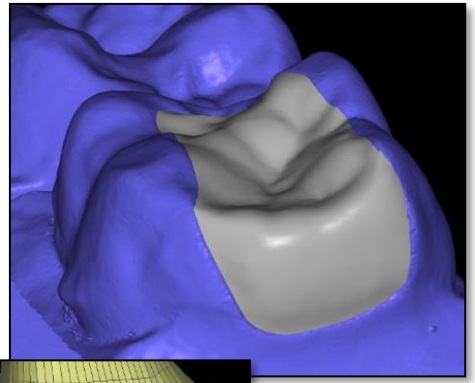
Performance Capture

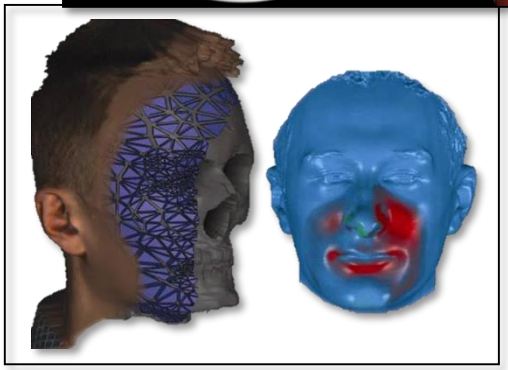


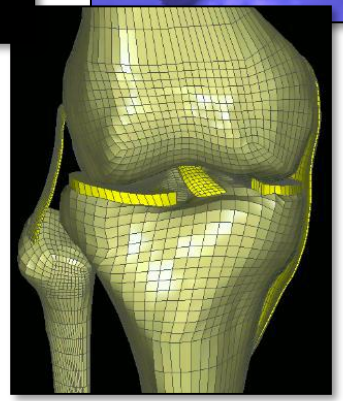
Applications

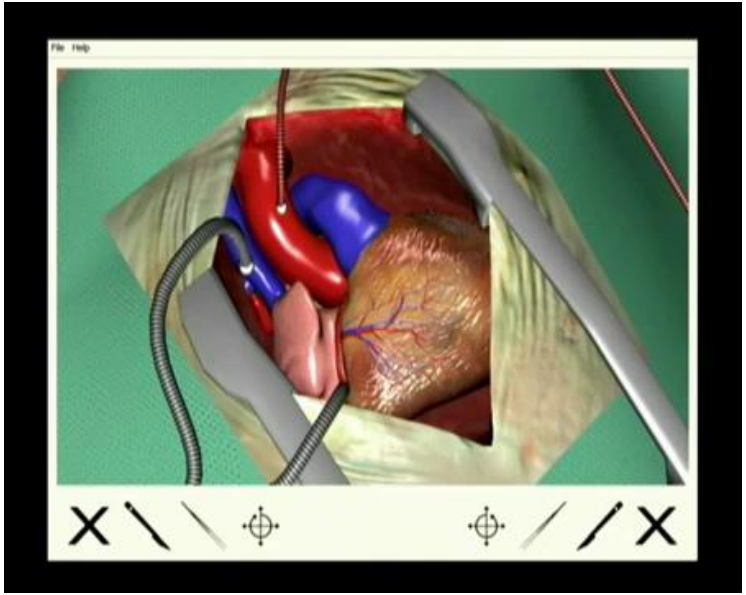
Medical Simulations





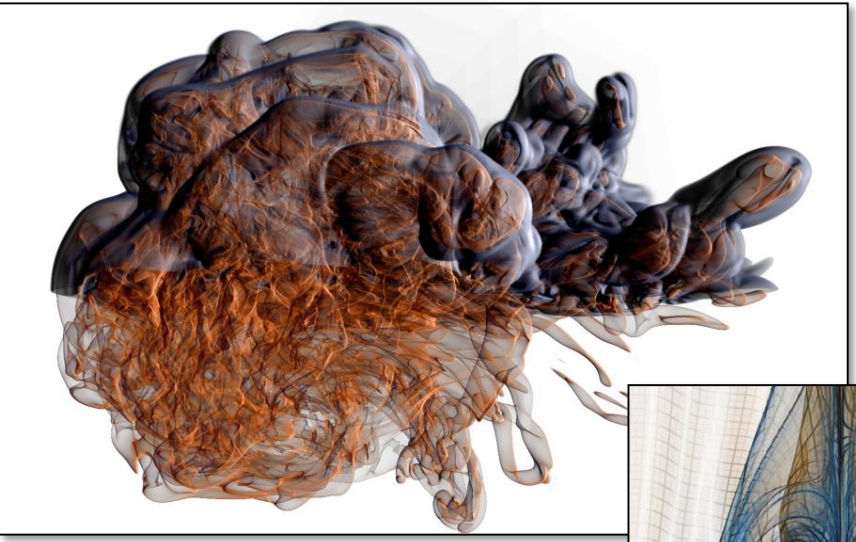


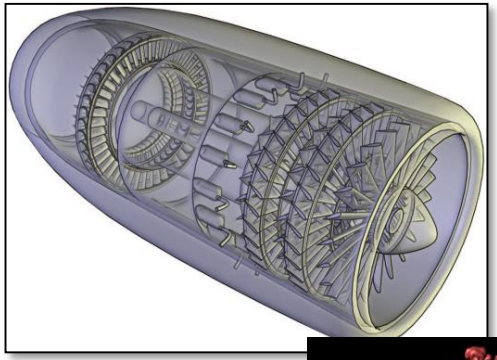


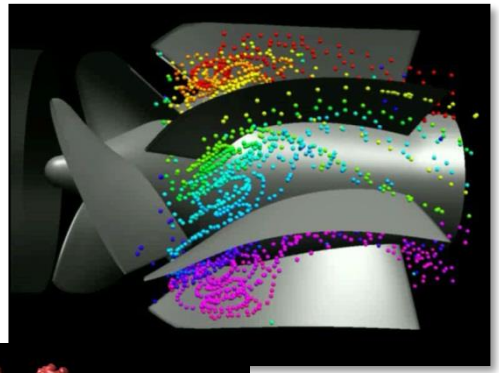


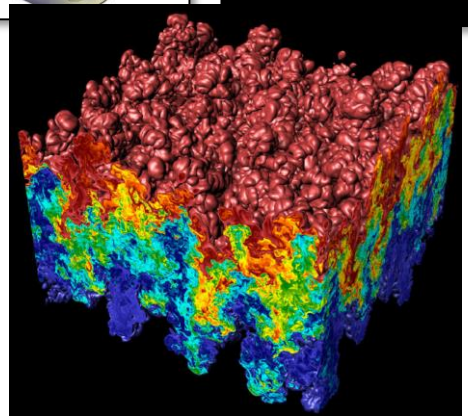
Applications

Scientific Visualization



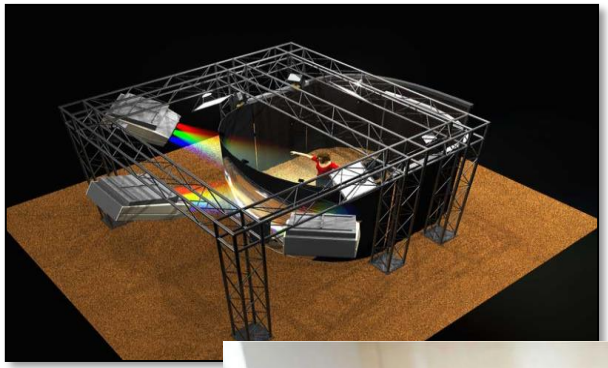


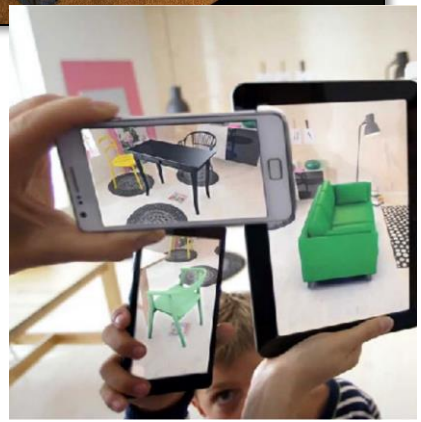




Applications

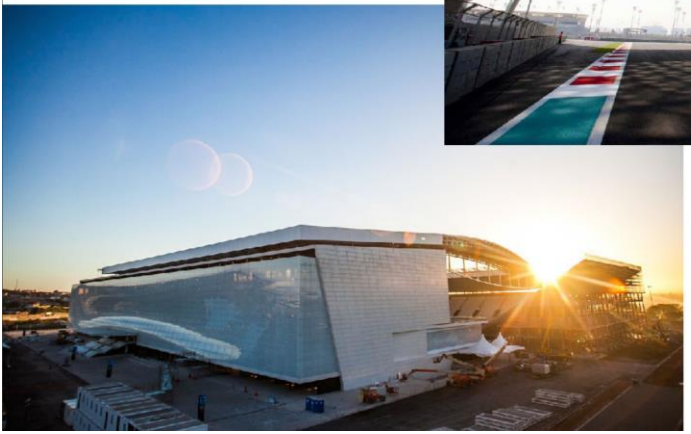
Virtual/Augmented Reality

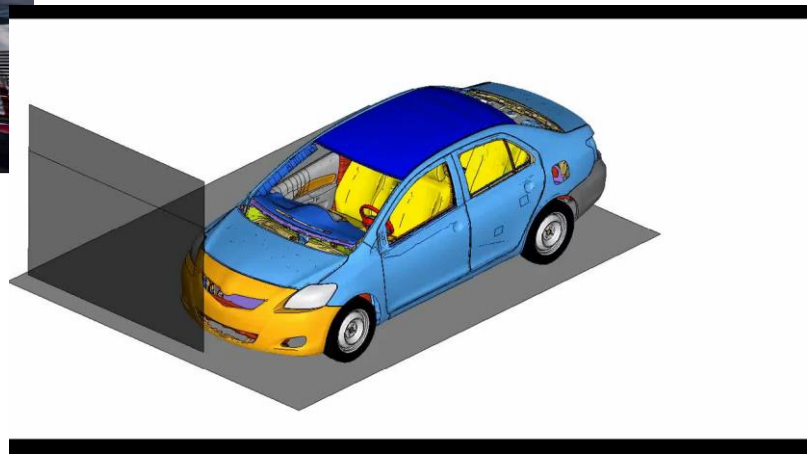




Applications

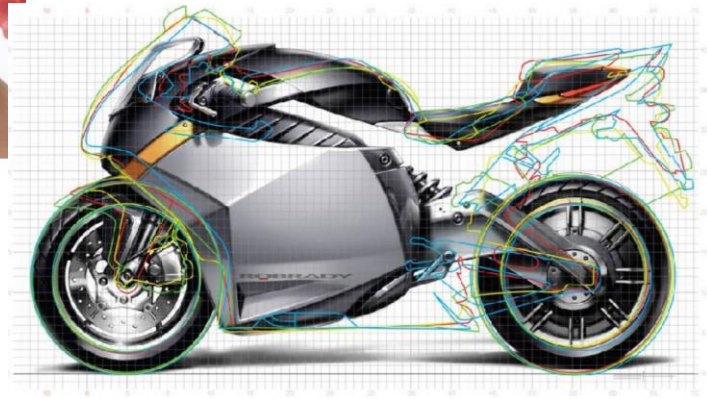
Computer-aided engineering



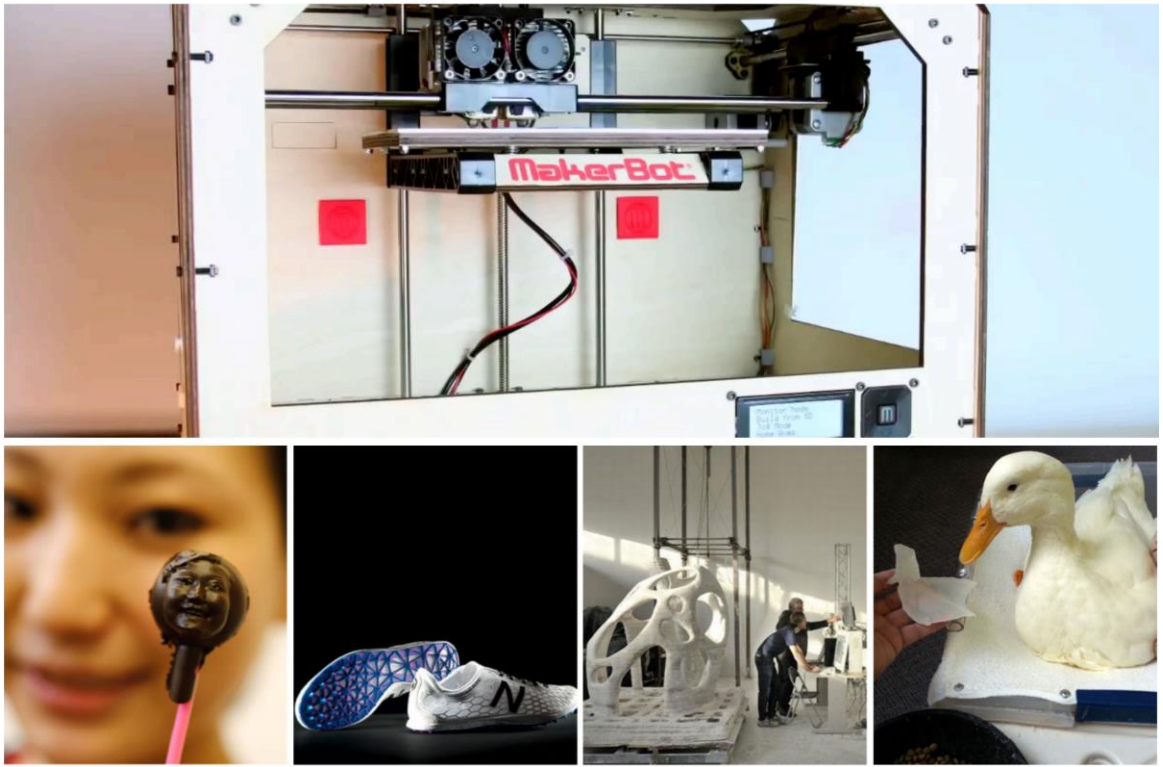


Applications

Art and design



Visual Technology: digital fabrication



Recent research highlights…

What is Computer Graphics?

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

Computer Graphics is about building computational models of the real world!

Foundations of computer graphics

- Building models of the world demands **sophisticated** theory & systems
- Theory
  - geometric representations
  - sampling theory
  - integration and optimization
  - perception
  - physics-based modeling
  - etc
- Systems
  - parallel, heterogeneous processing
  - graphics-specific programming languages

Foundations of computer graphics

- We will explore some of these concepts over the next seven weeks

- Let’s get started…

Class activity: modeling and drawing a cube

Let’s draw this 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 )
```

Does this list of numbers suffice to describe the cube?

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:

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 (e.g. screen)?

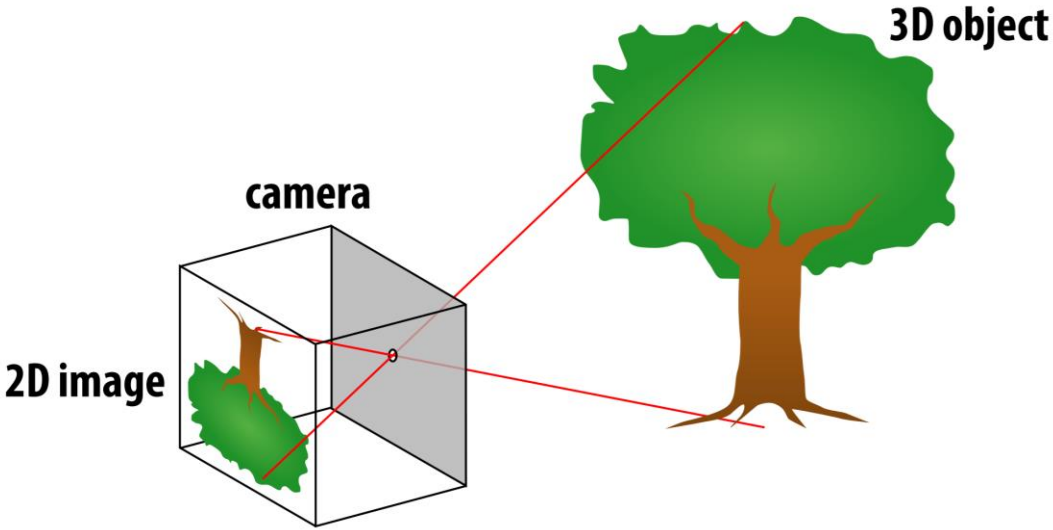
Basic strategy:

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

Ok, but how?

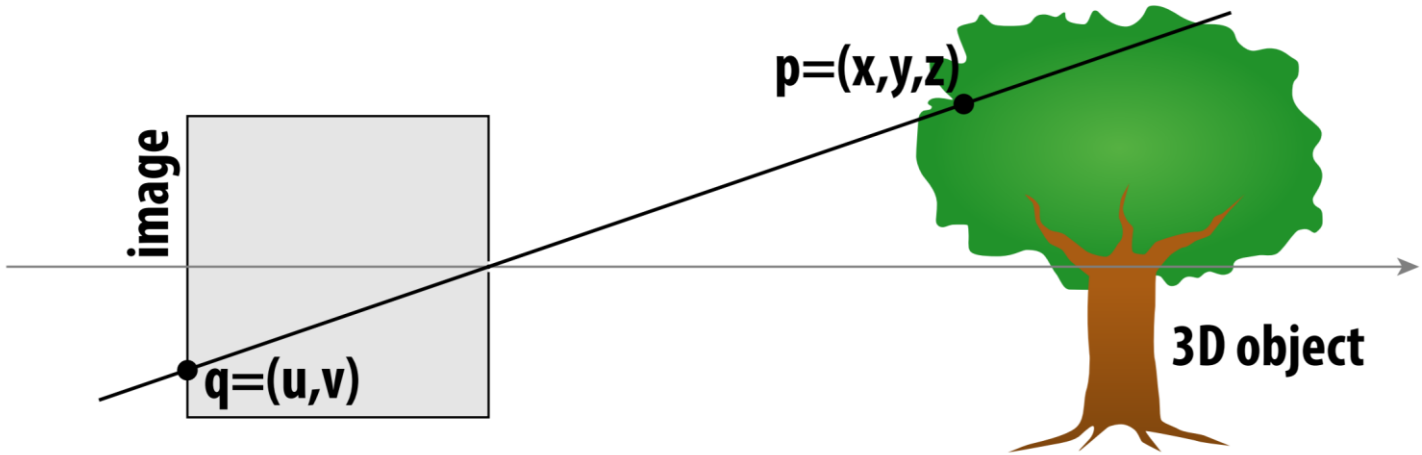
Perspective projection

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



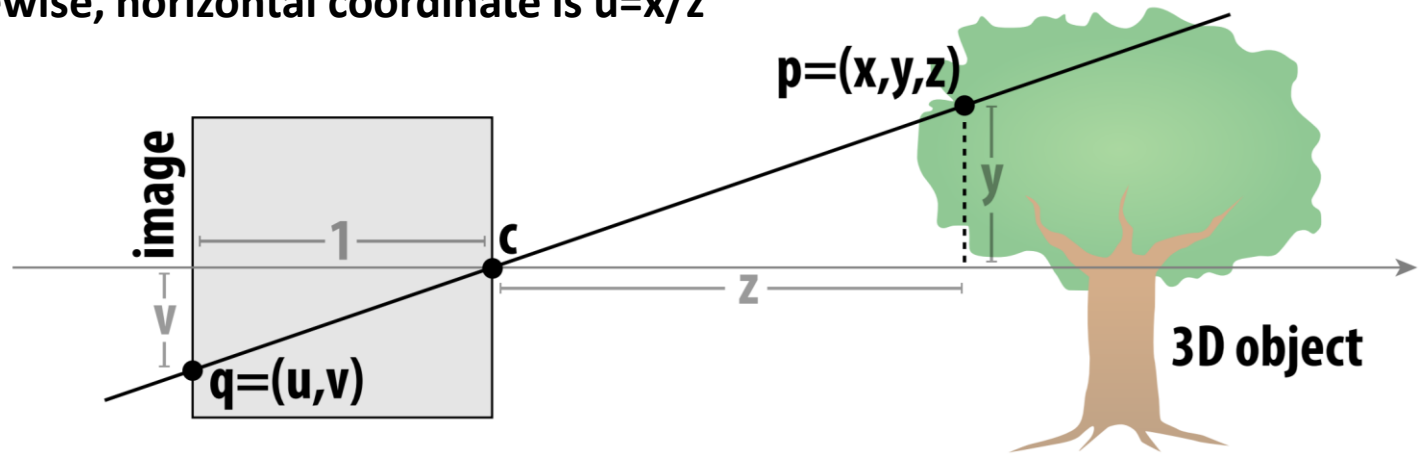
Perspective projection: side view

- 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 v 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)
- for each edge, 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)

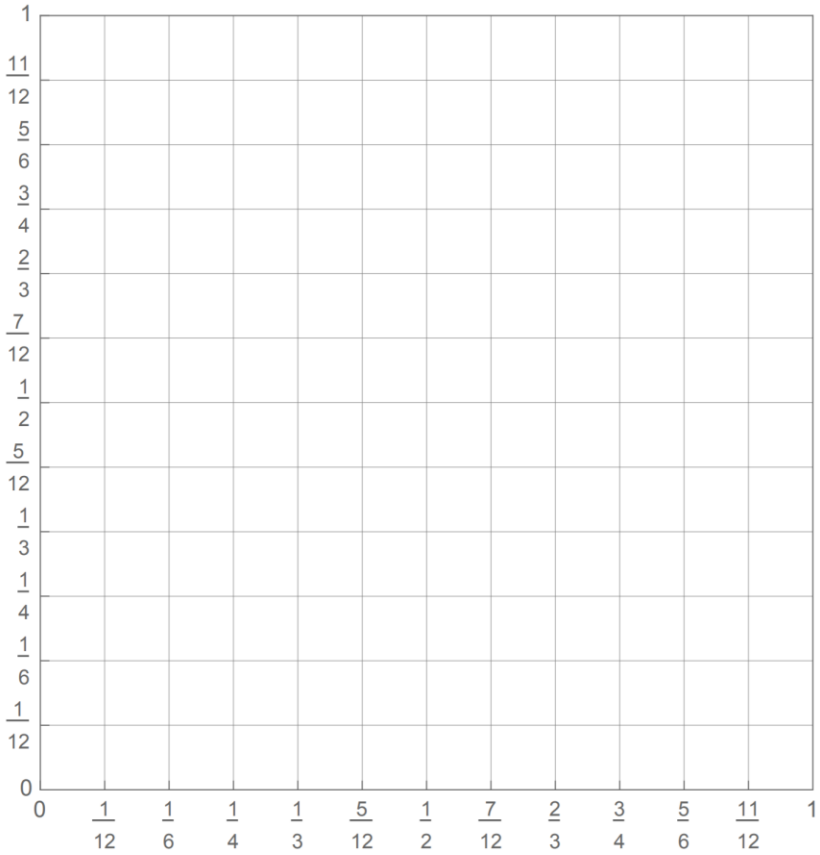
Vertices

```
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 )
```

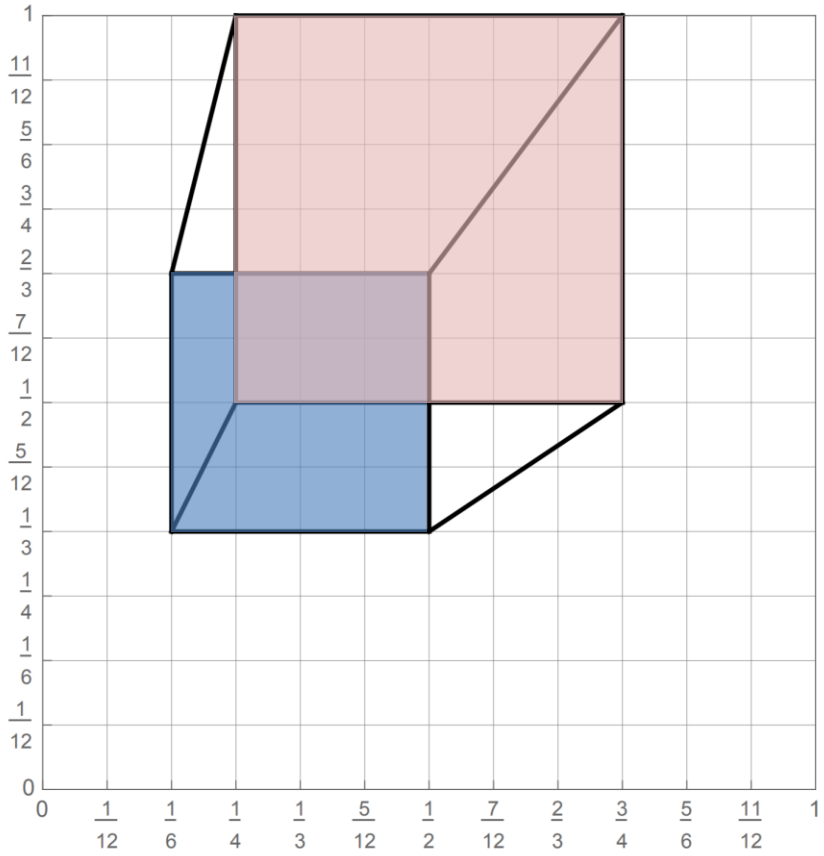
Edges

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

Class activity: Let's draw that cube!



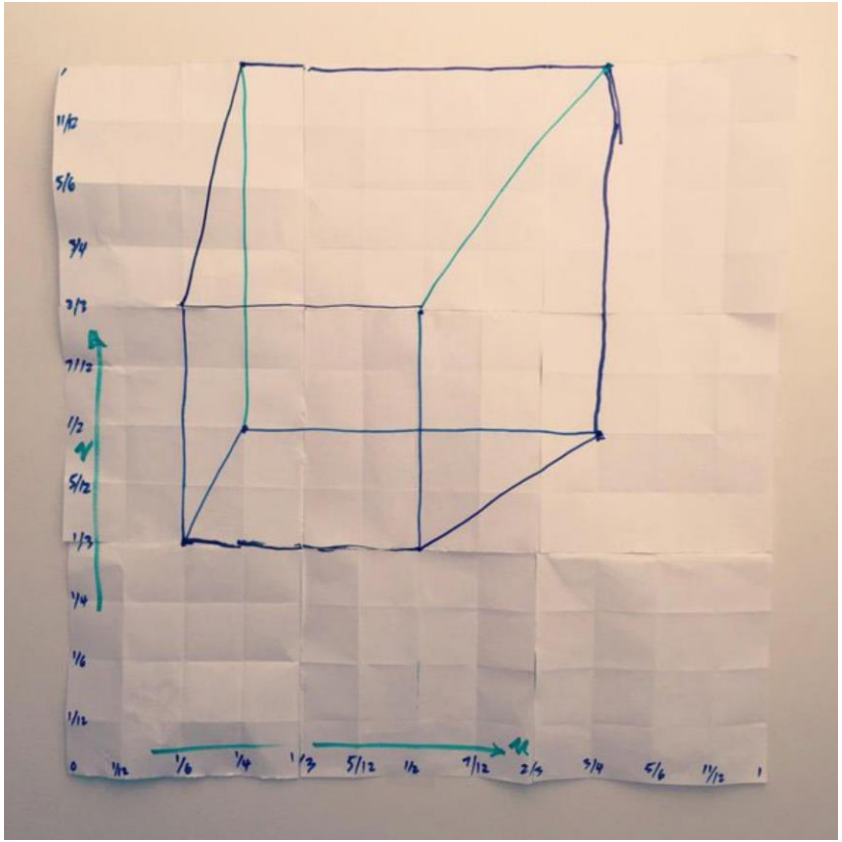
Class activity: How did we do?



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

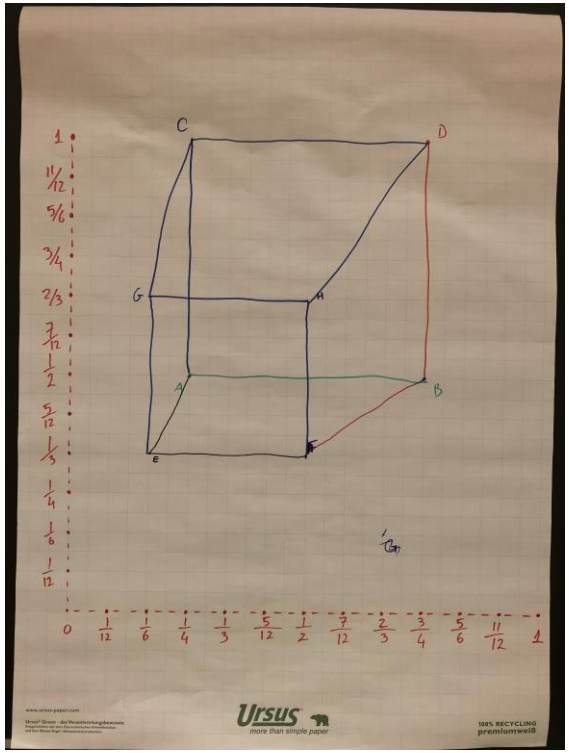
Class activity: Previous year's result



computer [k*uh* m-**pyoo**-ter]

- *noun*
- a programmable electronic device designed to accept data, perform prescribed mathematical and logical operations at high speed, and display the results of these operations. Mainframes, desktop and laptop computers, tablets, and smartphones are some of the different types of computers.Compare analog computer, digital computer.
- a person who computes; computist.

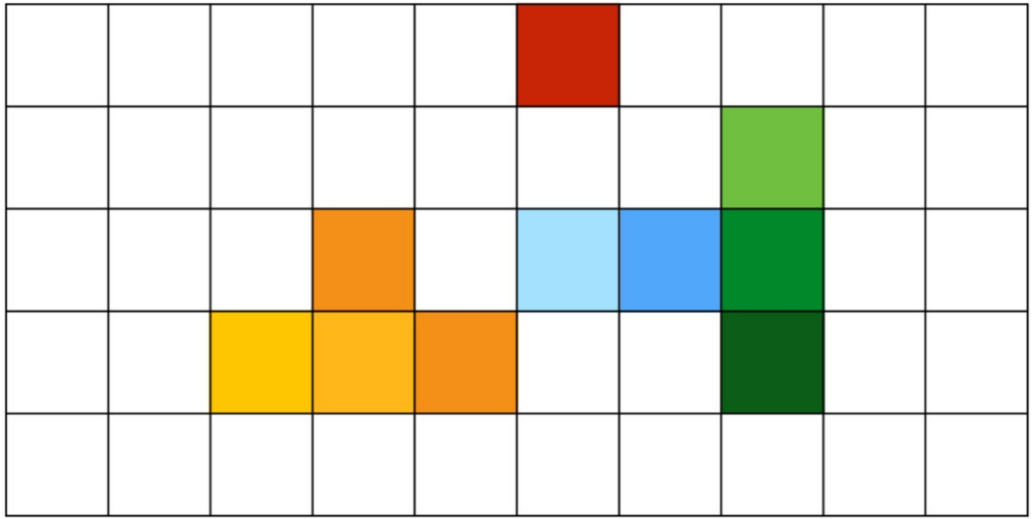
But wait... how does a (digital) computer draw lines?



Close up photo of pixels on a modern display

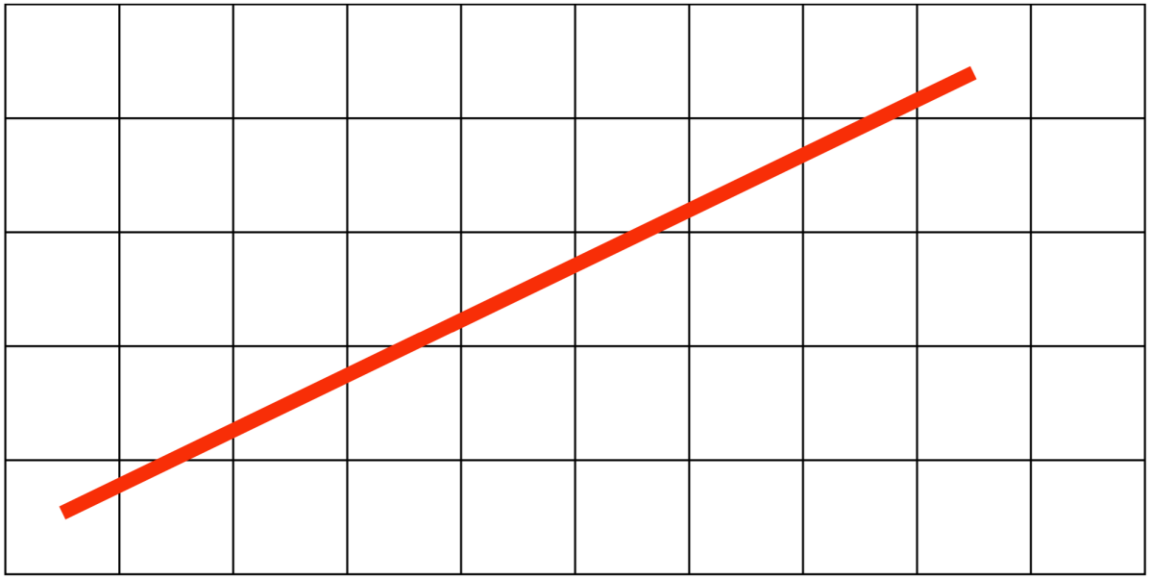
Drawing on a raster display

- Common abstraction:
  - Image represented as a 2D grid of “pixels”
  - Each pixel can take on a unique color value



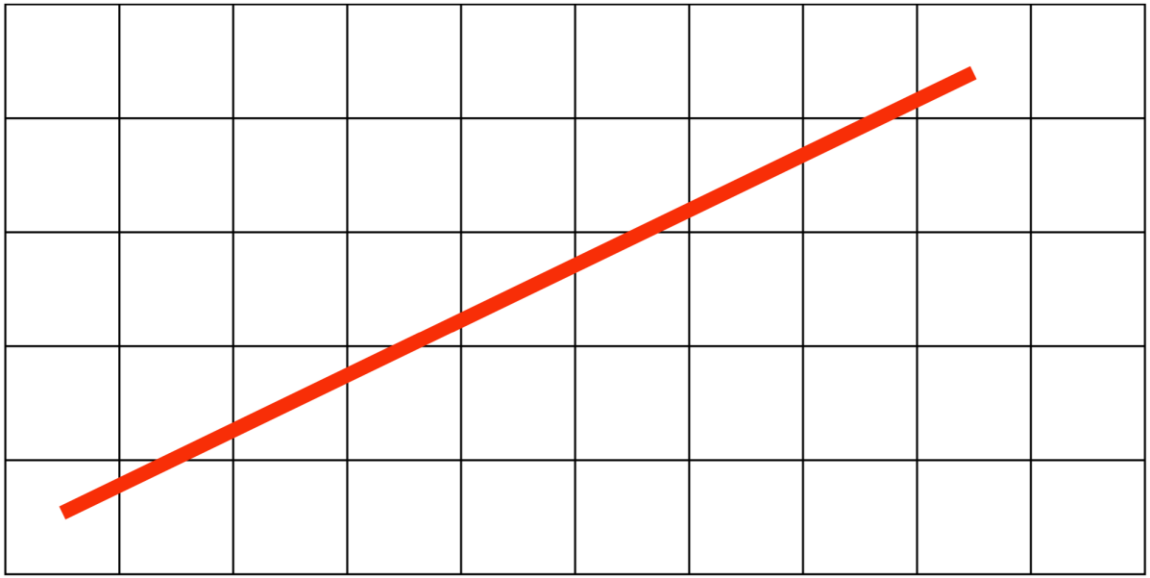
Which pixels should we color in to depict the line?

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



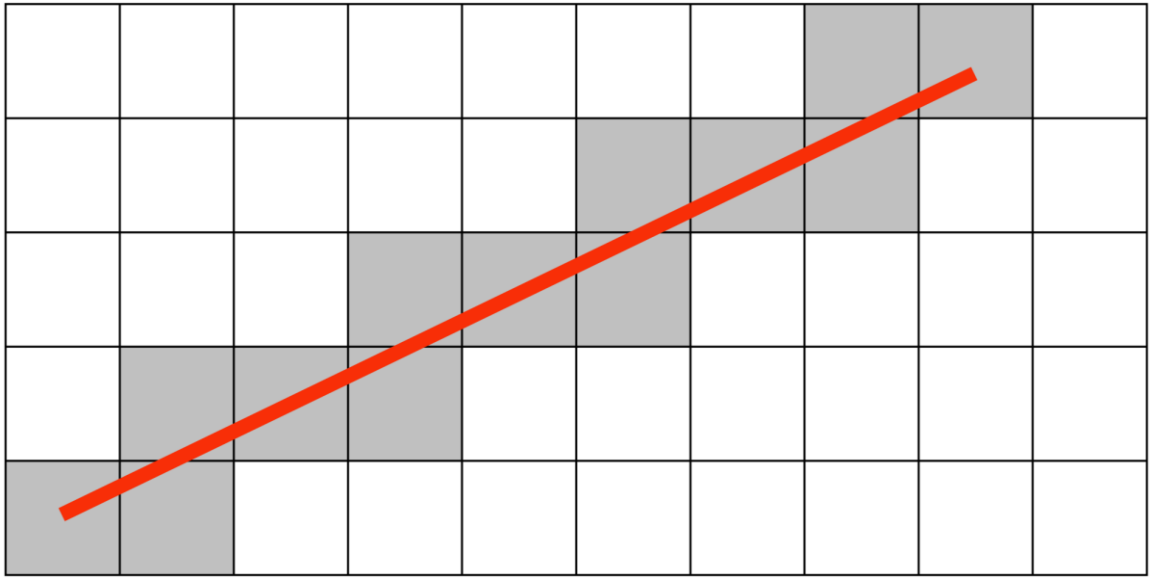
Which pixels should we color in to depict the line?

- Light up all pixels intersected by the line?



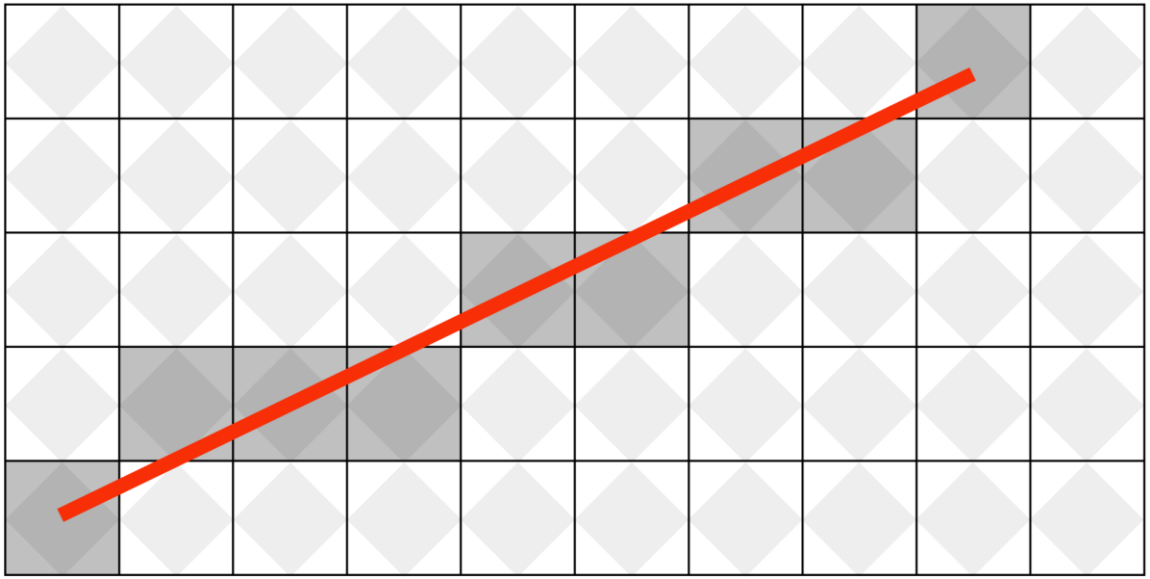
Which pixels should we color in to depict the line?

- Light up all pixels intersected by the line?



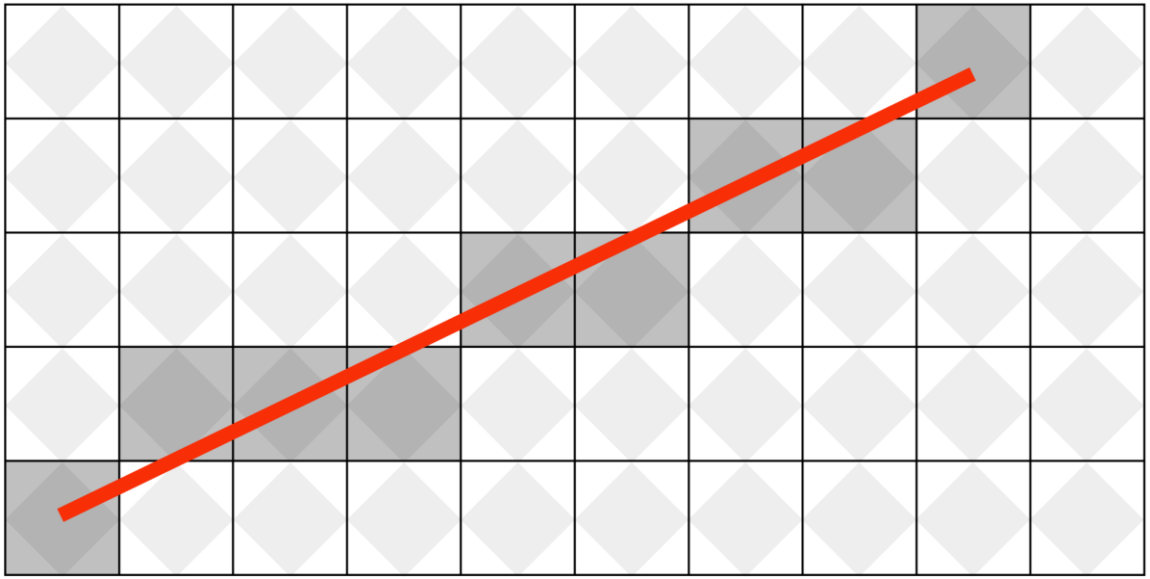
Which pixels should we color in to depict the line?

- 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?

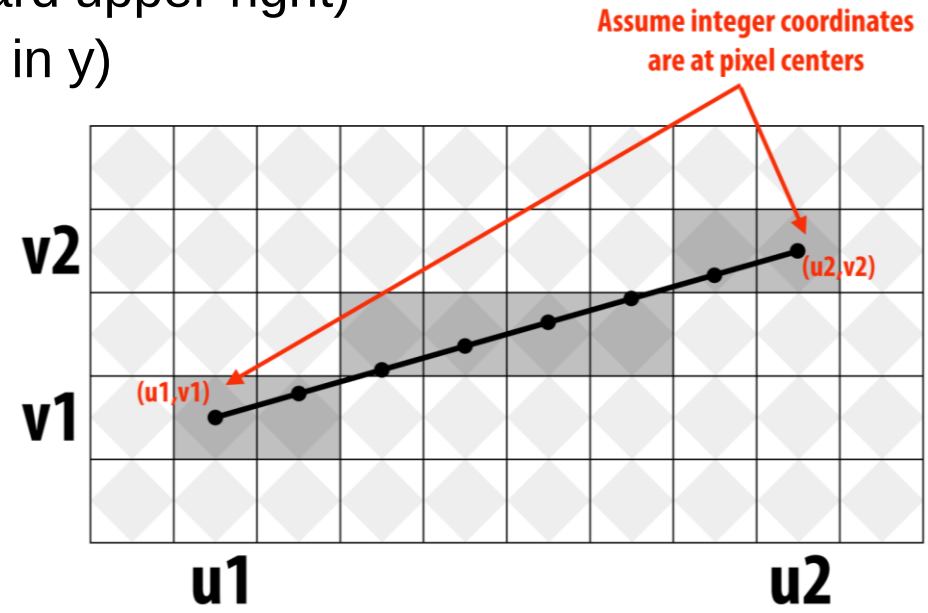
- Could check every single pixel in the image to see if it meets the condition…
  - $O(n^2)$ pixels in image vs at most O(n) “lit up” pixels
  - Must be able to do better (work proportional to the number of pixels in the drawing of the line)

Incremental line rasterization

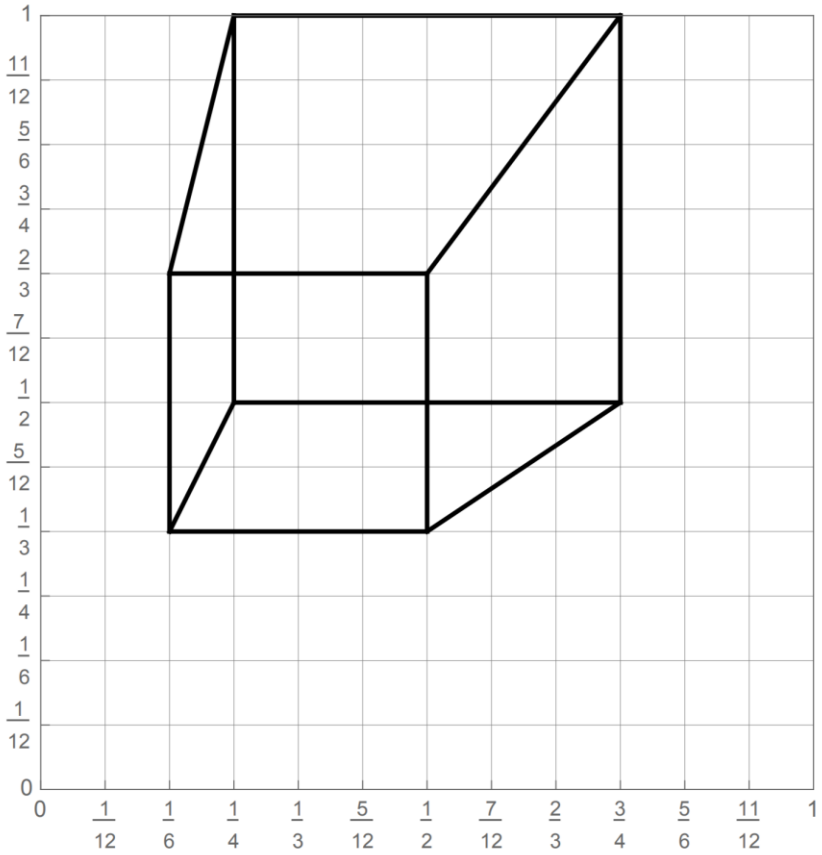
- Let's say a line is represented with integer endpoints (u1, v1) and (u2, v2)
- Slope of line: s = (v2-v1)/(u2-u1)
- Consider a very easy special case:
  - u1 < u2, v1 < v2 (line points toward upper-right)
  - 0 < s < 1 (more change in x than in y)

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

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



The cube, now drawn on a computer screen…



But… we rendered only a simple line drawing…

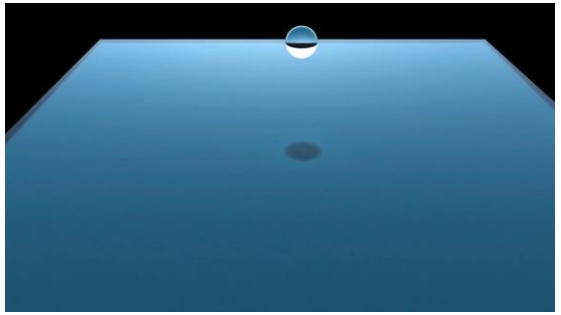
- We want much richer models of the world…

Shapes and surfaces

Materials

Lighting environments

Motion



Physics-based simulation

Schedule

Lecture	Topic	Exercise
Week 1	Introduction, drawing triangles	Ex. 6: Intro to WebGL
Week 2	Transformations, Geometry and Textures	Ex. 7: Transformations
Week 3	Rendering Pipeline	Ex. 8: Meshes and Textures
Week 4	Lighting and Shading, Visibility and Shadows	Ex. 9: Shaders
Week 5	Animation, Rigging	Ex. 10: Keyframing
Week 6	Physically-based simulation, PDEs	Ex. 11: Rigid body dynamics
Week 7	On to more advanced topics	Q&A

Advanced Courses

- **Computer Graphics**
  - Materials, rendering, raytracing
- **Shape Modeling and Geometry Processing**
  - Geometric modeling, splines, meshes, processing
- **Game Programming Laboratory**
  - Design & development of a game in small groups
- **Physically-based Simulation**
  - Animation, deformation, fracture, collision detection
- **Seminar: Advanced Topics in CG & Vision**
  - State-of-the-art research papers
- **Computational models of motion for character animation and robotics**
  - Forward and inverse kinematics, dynamics and control, motion capture, sim 2 real

That’s all for today…