EF432: **Introduction to spagetti and meatballs**

CSC 418/2504: Computer Graphics

Course web site (includes course information sheet):

https://github.com/alecjacobson/computer-graphics-csc418

Instructors:

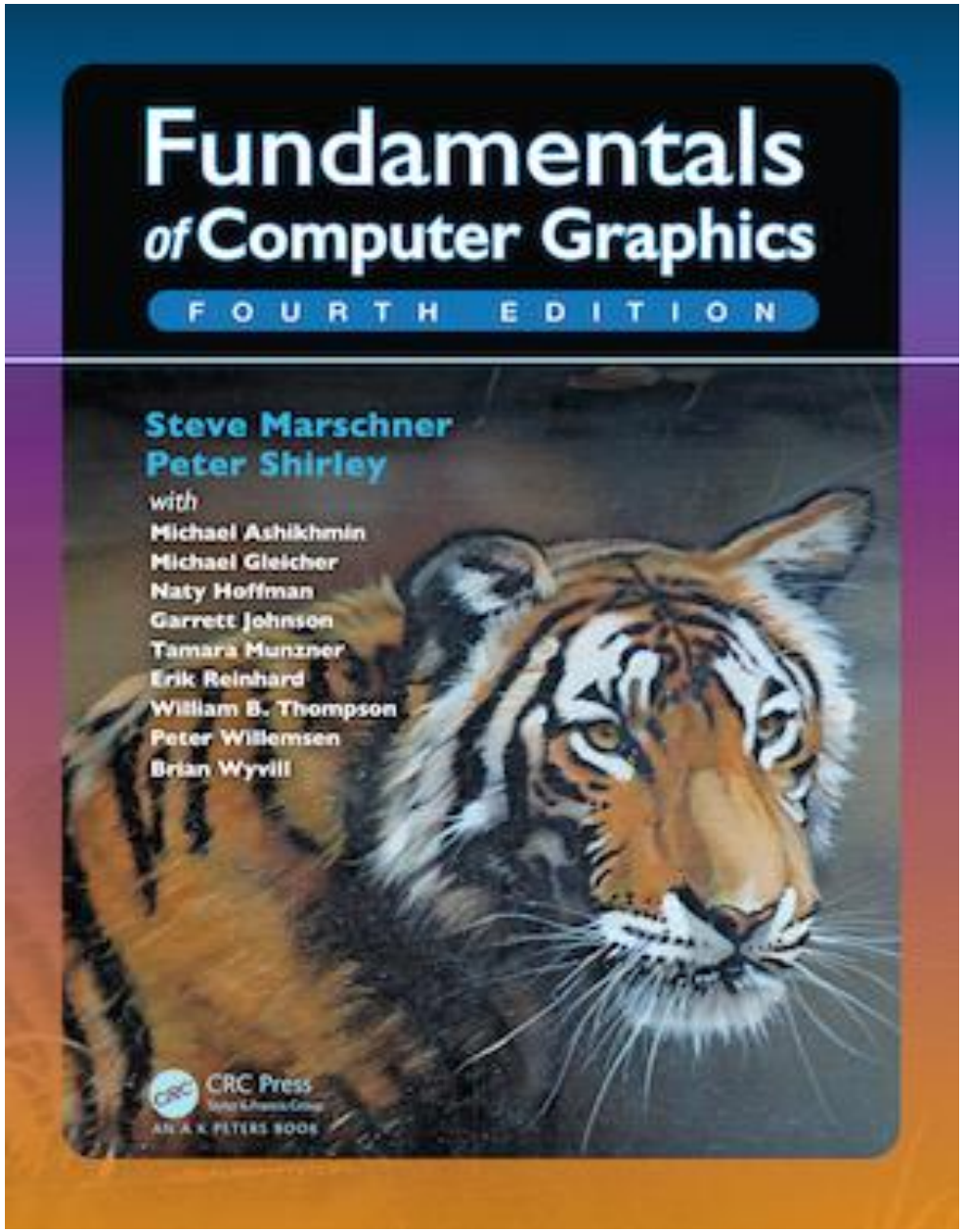
LEC0201 Tuesdays 15:00-17:00 in SF 3202
Prof. Karan Singh *karan@dgp.toronto.edu*
+1 416-978-7201
Office hours Tuesdays 14:00-15:00 in BA 5258
or by appointment.

LEC0101 Wednesdays 15:00-17:00 in GB 244
Prof. Alec Jacobson *jacobson@cs.toronto.edu*
+1 416-946-8630
Office hours Wednesdays 17:00-19:00 in BA 5266
or by appointment.

Tutorial for both sections will be held *together* on Mondays 15:00-16:00 in GB 244.

Textbooks: Fundamentals of Computer Graphics
OpenGL Programming Guide & Reference

Required Textbook



Topics

1. Introduction: What is Computer Graphics?
2. Raster Images (image input/output devices and representation)
3. Scan conversion (pixels, lines, triangles)
4. Ray Casting (camera, visibility, normals, lighting, Phong illumination)
5. Ray Tracing (shadows, supersampling, global illumination)
6. Spatial Data Structures (AABB trees, OBB, bounding spheres, octree)
7. Meshes (connectivity, smooth interpolation, uv-textures, subdivision, Laplacian smoothing)
8. 2D/3D Transformations (Translate, Rotate, Scale, Affine, Homography, Homogeneous coordinates)
9. Viewing and Projection (matrix composition, perspective, Z-buffer)
10. Shader Pipeline (Graphics Processing Unit)
11. Animation (kinematics, keyframing, Catmull-Romm interpolation, physical simulation)
12. 3D curves and objects (Hermite, Bezier, cubic curves, curve continuity, extrusion/revolve surfaces)
13. Advanced topics overview

Topic 1.

Introduction:
What Is Computer Graphics?

What is Computer Graphics?

Computers:

accept, process, transform and present information.

Computer Graphics:

accept, process, transform and present information

in a visual form.

Ok but… what is the course really about?

The science of turning the rules of geometry, motion and physics into (digital) pictures.

What its not about?

Photoshop, AutoCAD, Maya, Renderman, Graphics APIs.

…**wow**, heavy math and computer science!!

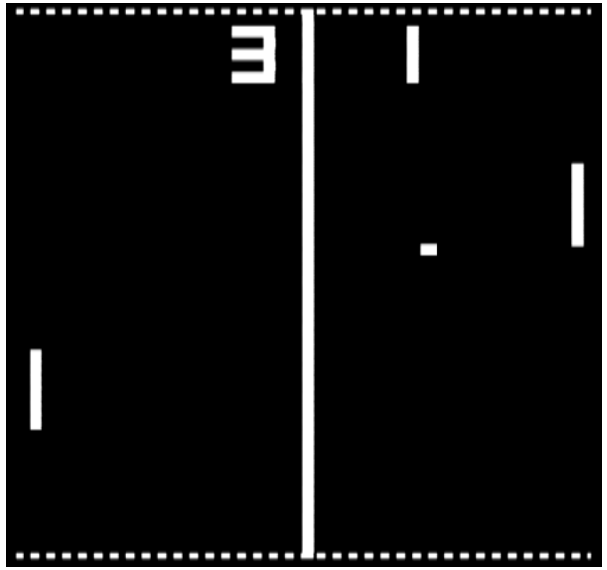
Movies

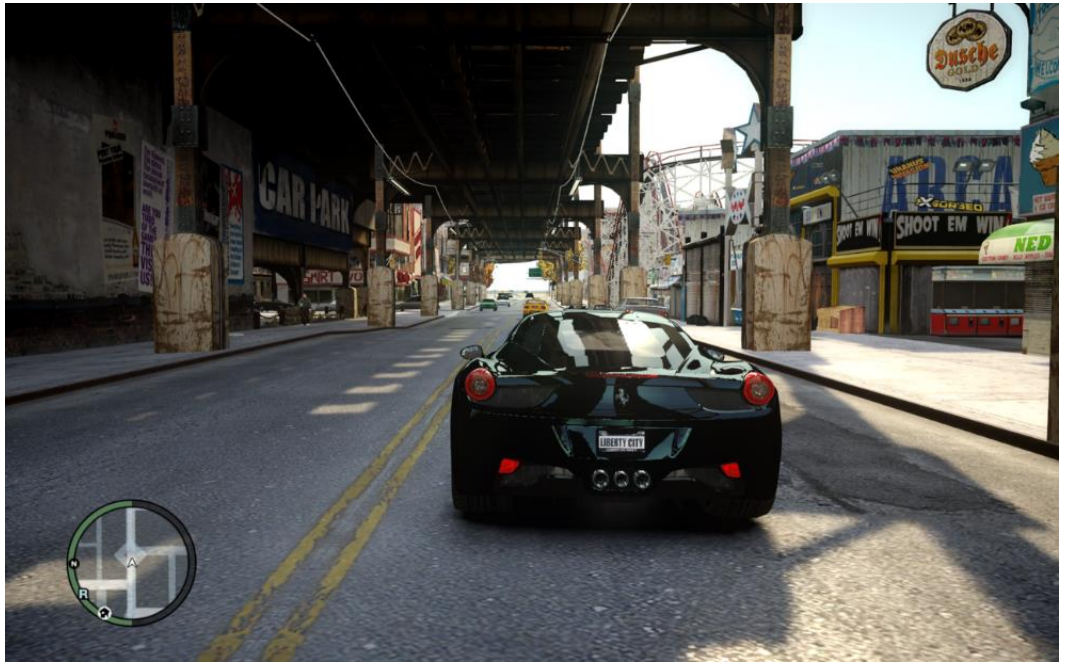
- Drive new directions in CG
- Set quality standards for CG

Games

Drive interactivity and AI in CG

Push CG hardware to its limits

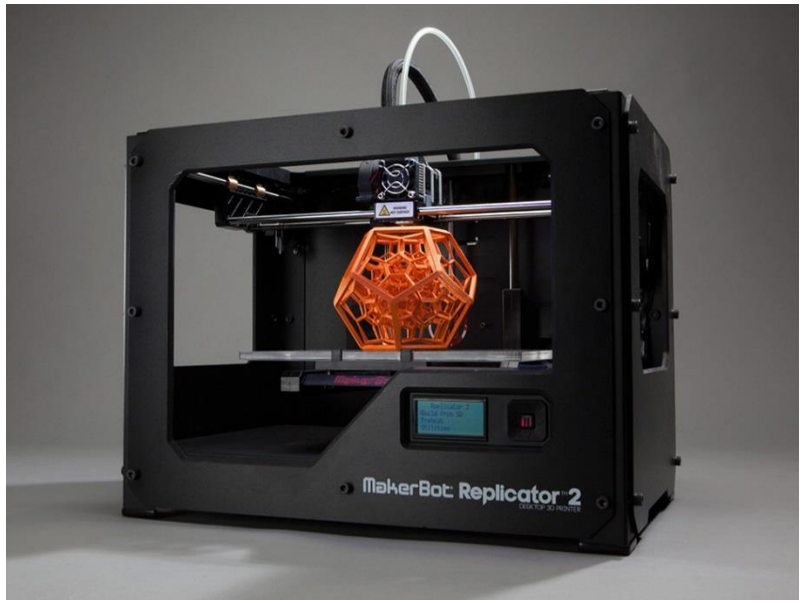


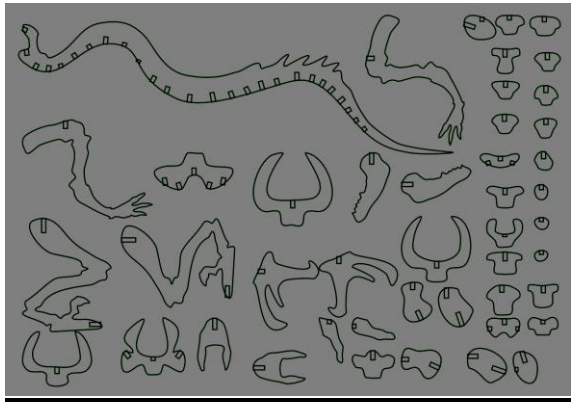


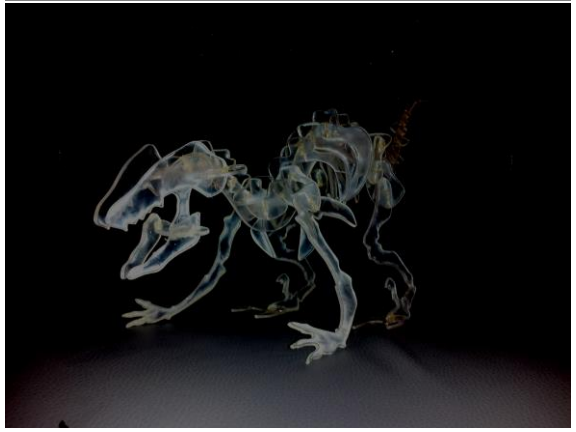
Design

CG for prototyping and fabrication

Drives precision modeling and engineering visualization





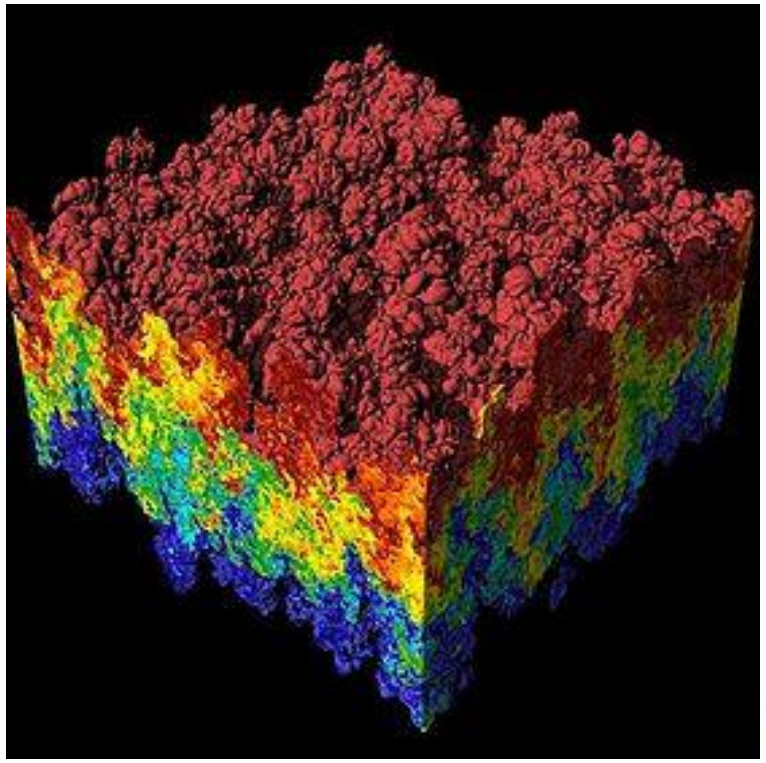


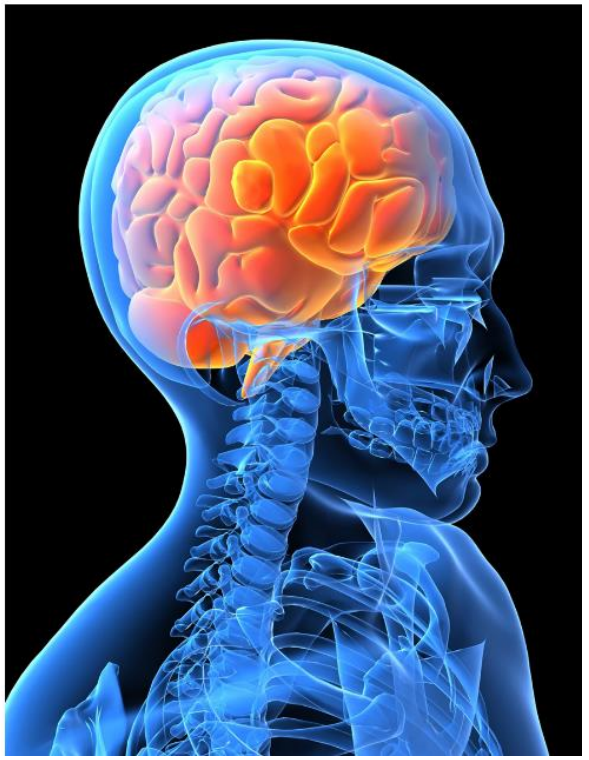
Scientific and Medical Visualization, Operation

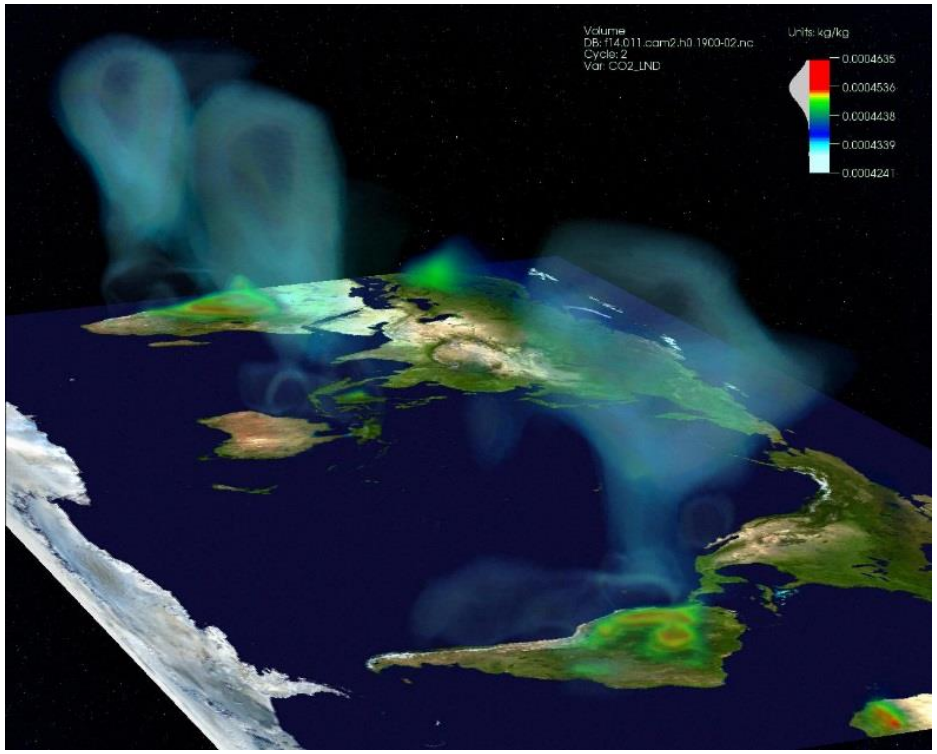
Drives the rendering of large datasets

May need device integration

Real-time and interactive



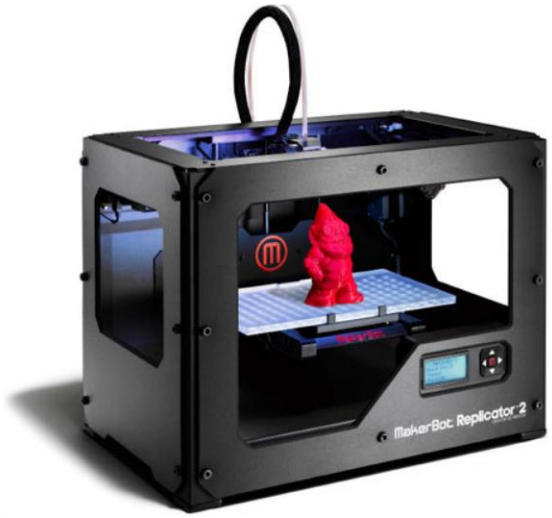




GUIs, AR/VR, scanners…

I/O of 3D data in CG

Drives interaction and usability

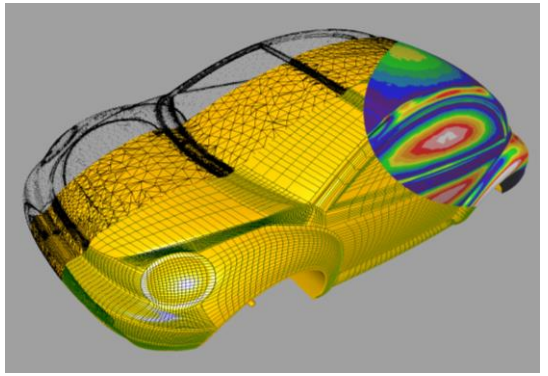


Computer Graphics: Basic Questions

- Form (modeling)

  How do we represent (2D or 3D) objects & environments?

  How do we build these representations?



- Function, Behavior (animation)

  How do we represent the way objects move?

  How do we define & control their motion?

- Appearance (rendering)

  How do we represent the appearance of objects?

  How do we simulate the image-forming process?

What You Will Take Away …

#1: yes, math IS useful in CS !!

#2: how to turn math & physics into pictures.

#3: how to code CG tools

Administrivia

Grading:

%	Item
70%	Assignments
20%	Monday, October 29, in-tutorial exam
10%	Monday, November 26, in-tutorial exam

Tutorial sessions:

- Math refreshers, tutorials on graphic libraries, additional topics.
- Attendance STRONGLY encouraged since I will not be lecturing on these topics in class.

Lecture slides will be posted prior to the lecture.

Topic 2.

Raster Images

What is an Image?

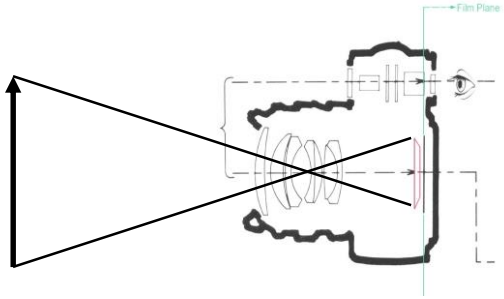
Image = distribution of light energy on 2D “film”

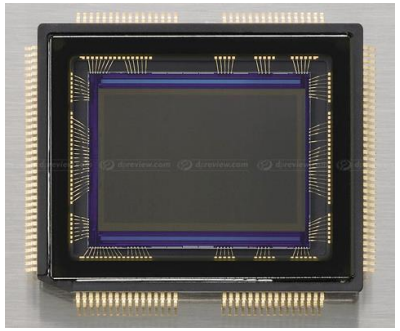
Digital images represented as rectangular arrays of pixels

Raster Devices

Input (scanners, cameras)

2D array sensor: digital camera

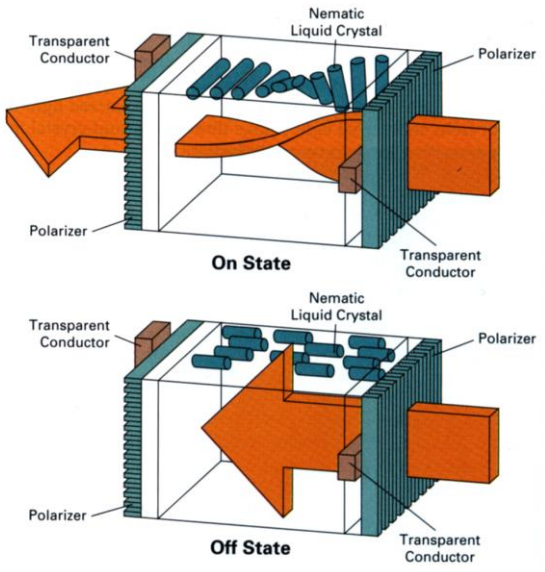




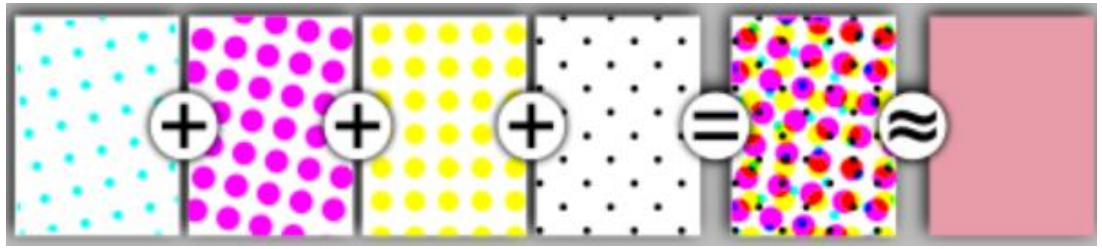
Output (printers, displays)

Emissive: light-emitting diode (LED)

Transmissive: liquid crystal display (LCD)

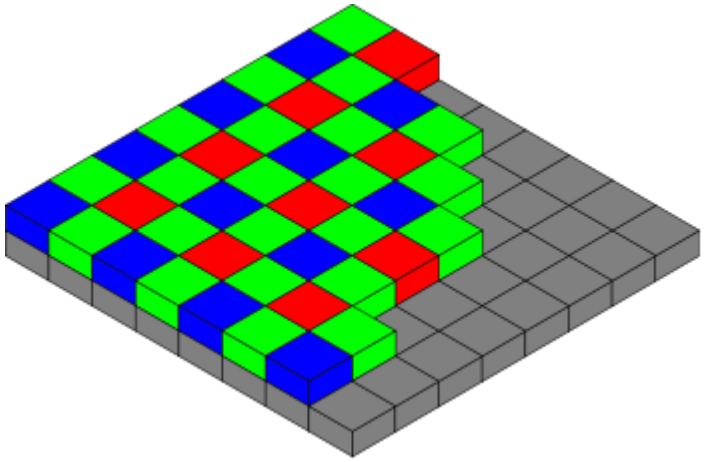


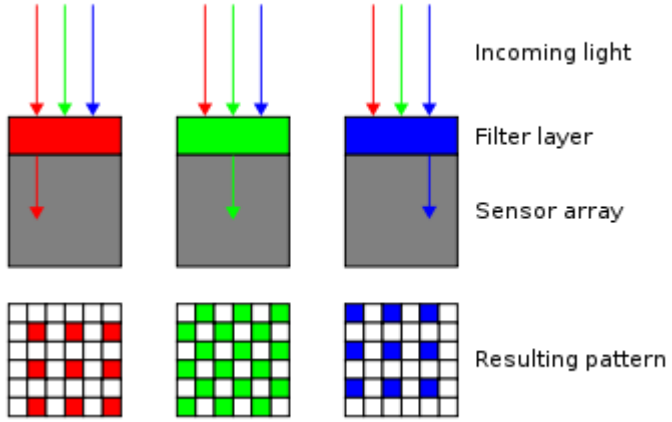
Ink-jet printer



Camera Sensor Array

Bayer color mosaic





Raster image representation

All these devices suggest 2D arrays of numbers

Bitmaps: boolean per pixel (1 bpp):

- interp. = black and white; e.g. fax

Grayscale: integer per pixel:

- interp. = shades of gray; e.g. black-and-white print
- precision: usually byte (8 bpp); sometimes 10, 12, or 16 bpp

Color: 3 integers per pixel:

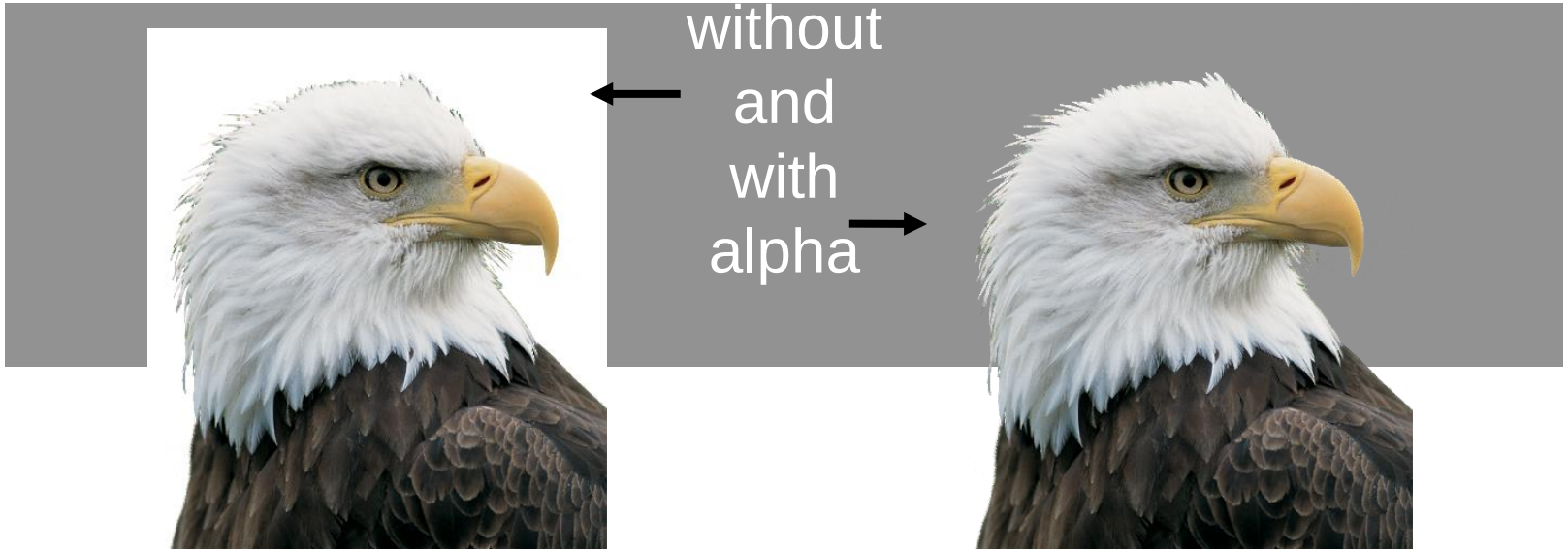
- interp. = full range of displayable color; e.g. color print
- precision: usually byte[3] (24 bpp)
- sometimes 16 (5+6+5) or 30 or 36 or 48 bpp

Floating point:

- more abstract, because no output device has infinite range
- provides *high dynamic range* (HDR)
- represent real scenes independent of display
- becoming the standard intermediate format in graphics processor

Datatypes for raster images

- Transparency (add *alpha* channel)



- Storage for 1024x1024 image (1 megapixel)
  - bitmap: 128KB
  - grayscale 8bpp: 1MB
  - grayscale 16bpp: 2MB
  - color 24bpp: 3MB
  - floating-point HDR color: 12MB

Converting pixel formats

Color to gray

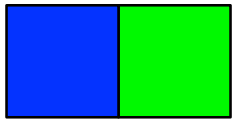
- could take one channel (blue, say)

  leads to odd choices of gray value

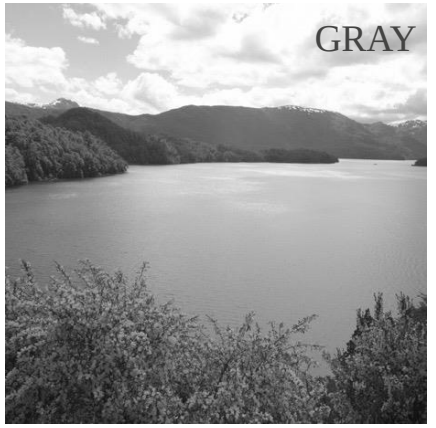
- combination of channels is better

  but different colors contribute differently to lightness

  which is lighter, full blue or full green?

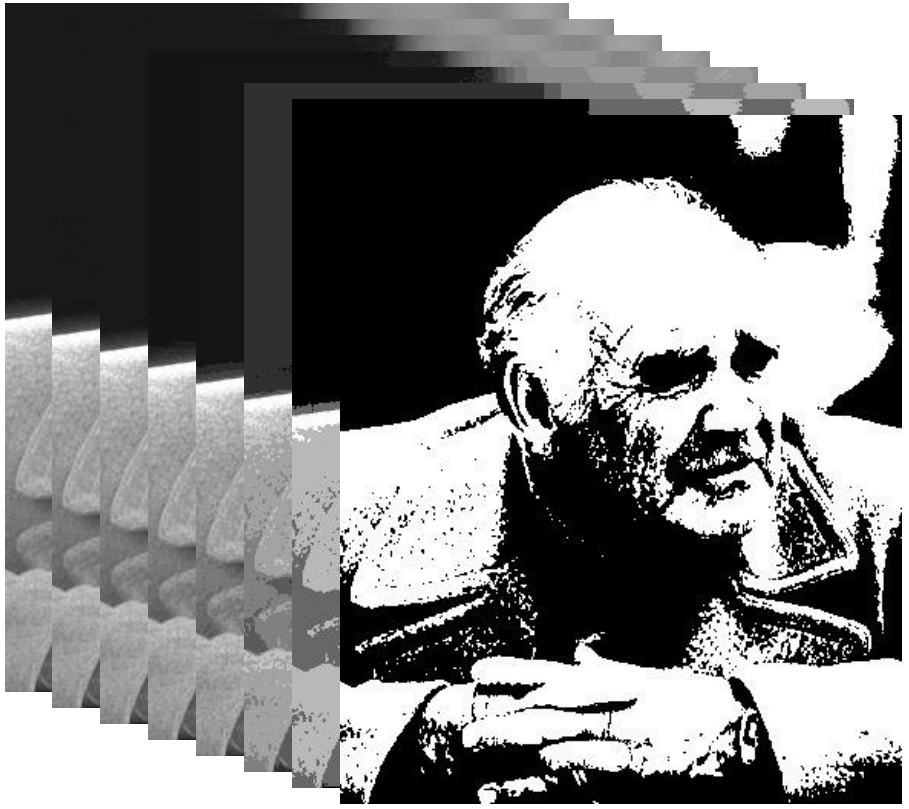


good choice: gray = 0.2 R + 0.7 G + 0.1 B



Converting pixel precision

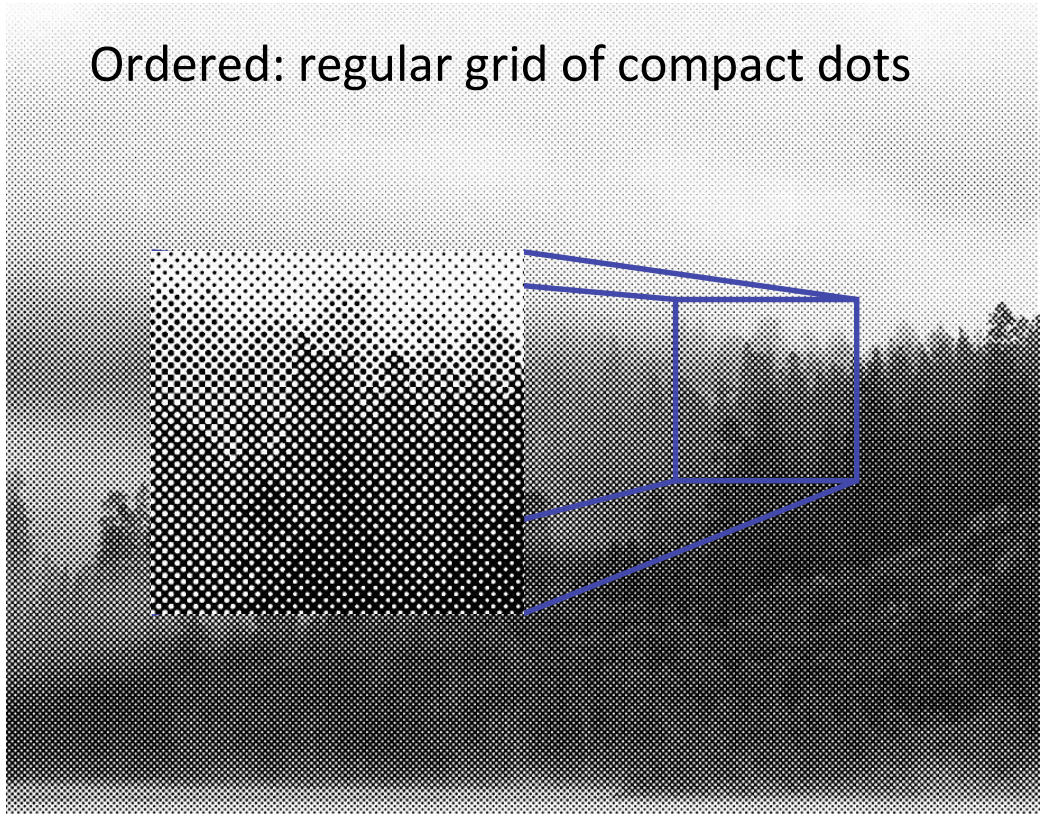
Up is easy, down loses information…

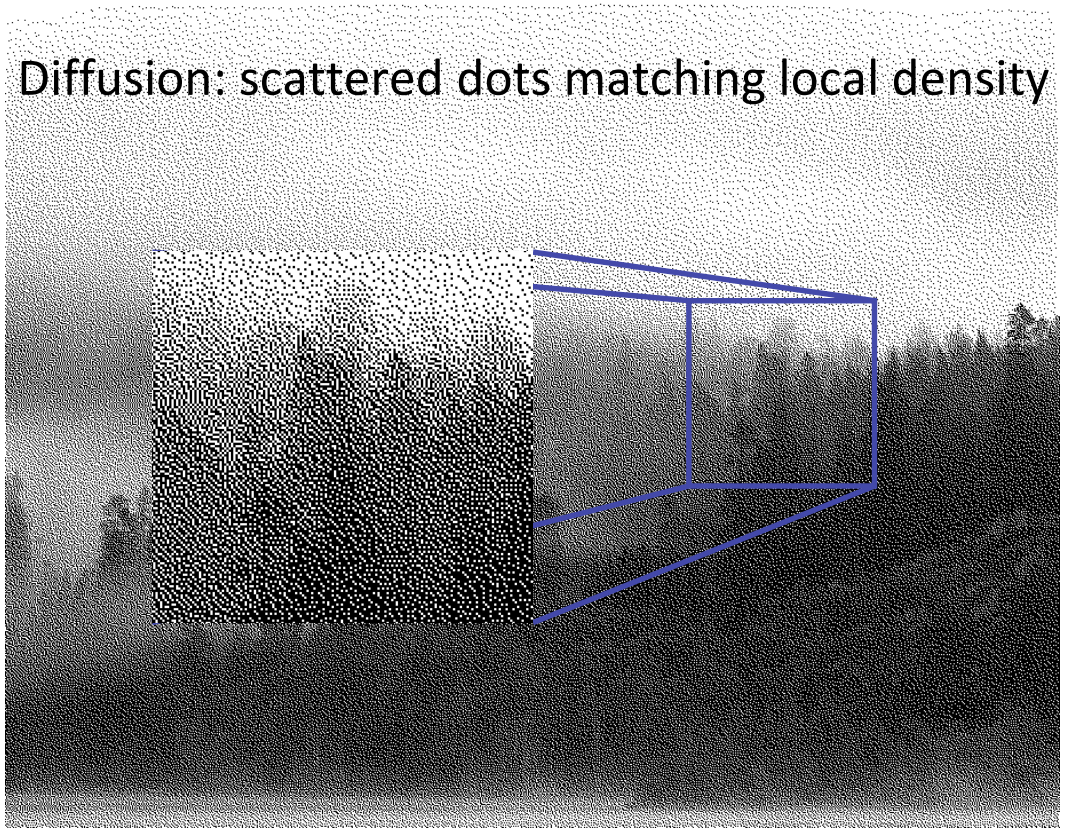


Dithering

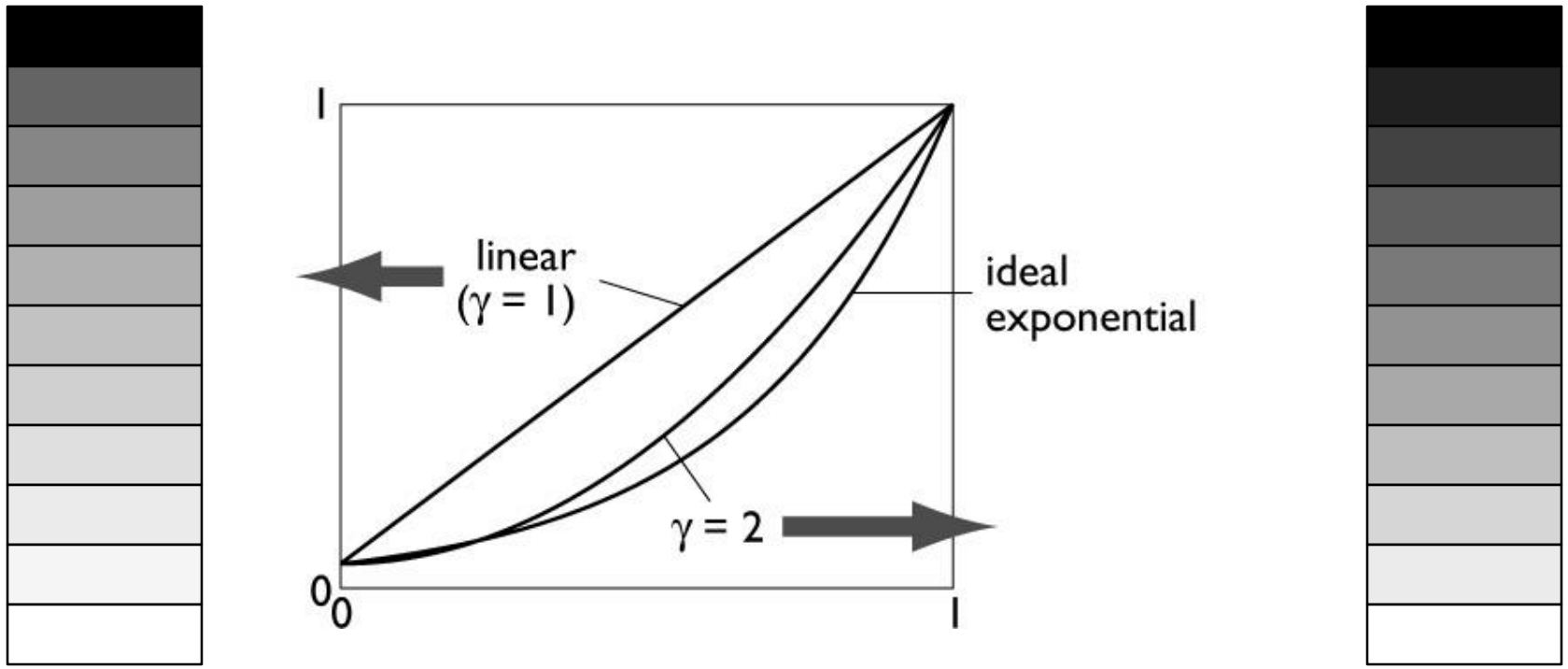
Decreasing bpp => quantize/threshold

- Consistent quantization causes banding.
- Trade spatial for tonal resolution: Dither.



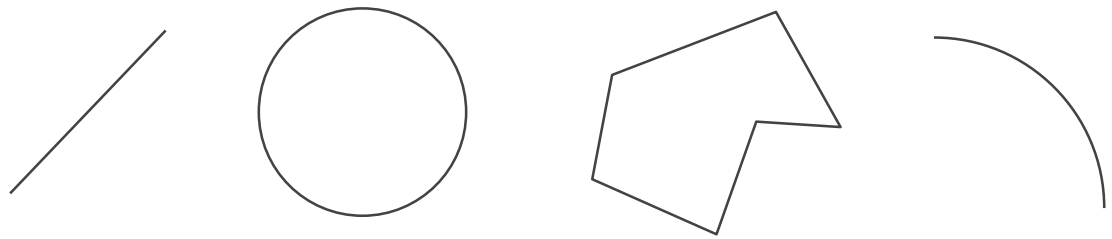


Gamma correction



2D Drawing

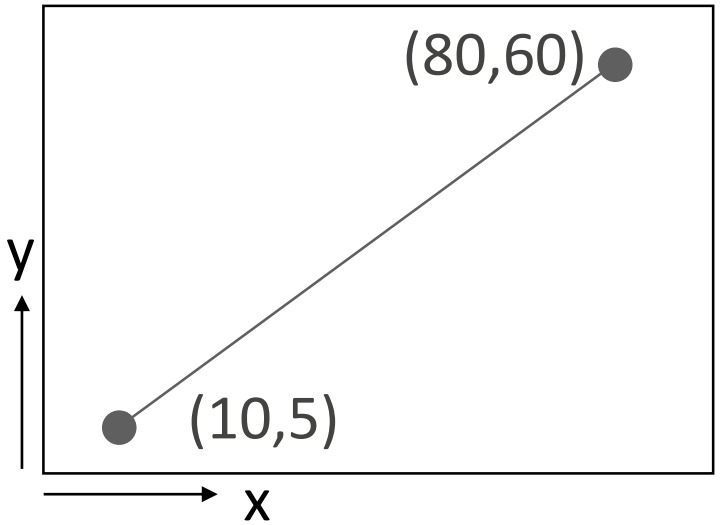
Common geometric objects:



When drawing a picture, 2D geometric shapes are specified as if they are drawn on a continuous plane

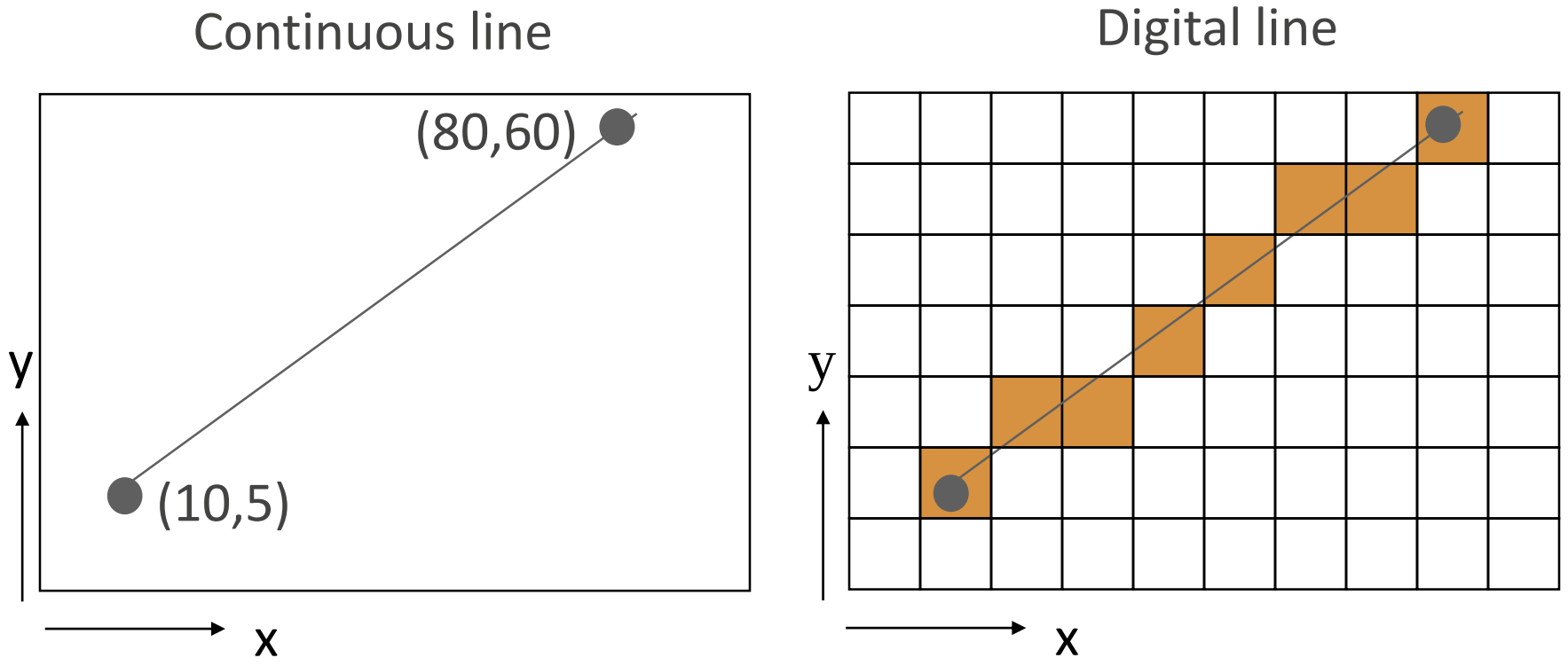
Drawing command:

Draw a line from point (10,5)
to point (80,60)



2D Drawing

In reality, computer displays are arrays of pixels, not abstract mathematical continuous planes



In graphics, the conversion from continuous to discrete 2D primitives is called scan conversion or rasterization

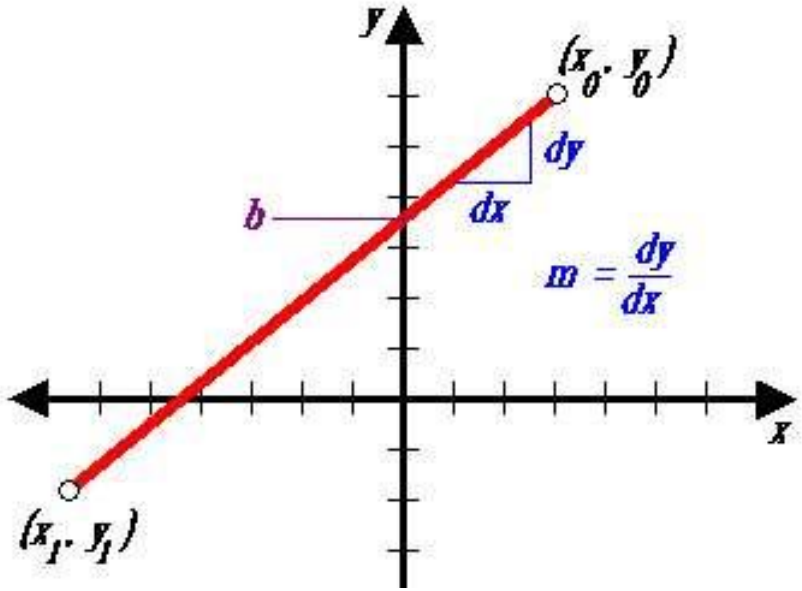
Equation of a Line

Line between (x_0,y_0) *and* (x_1,y_1)

$dx = x_1 - x_0, dy = y_1 - y_0$

Explicit : $y = mx + b$

$m = dy/dx,\ b = y_0 - mx_0$



Parametric :

$x(t) = x_0 + dx*t$

$y(t) = y_0 + dy*t$

$P = P_0 + (P_1-P_0)*t$

$P = P_0*(1-t) + P_1*t$ (weighted sum)

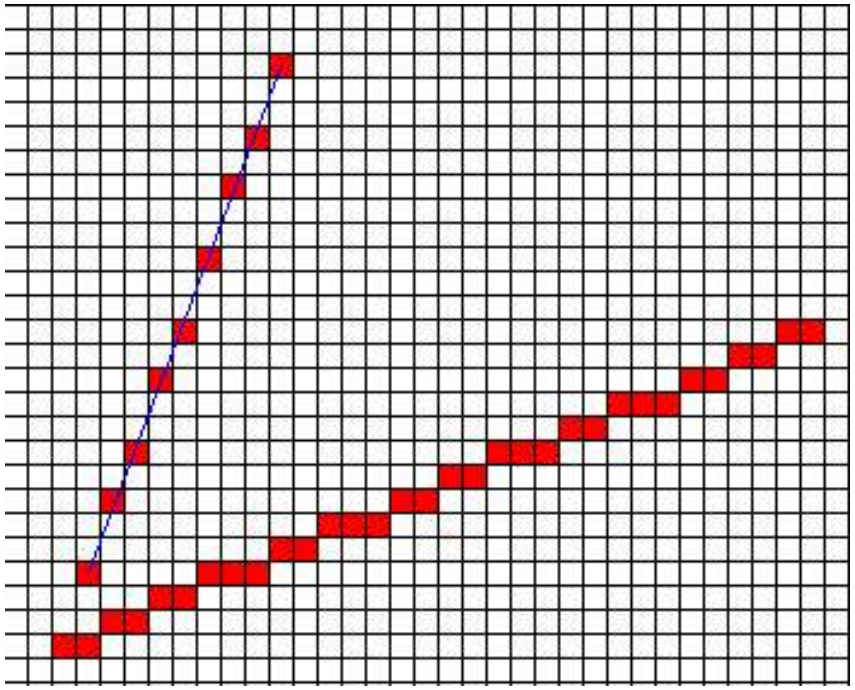
Implicit : $(x-x_0)dy - (y-y_0)dx = 0$

DDA Algorithm

Explicit form:

y= dy/dx * (x-x0) + y0

```
dx = x1-x0; dy = y1 – y0;
m = dy/dx;
for ( x=0; x<=x1-x0; x++)
{
    setpixel (x+x0, round(m*x+y0));
}
```

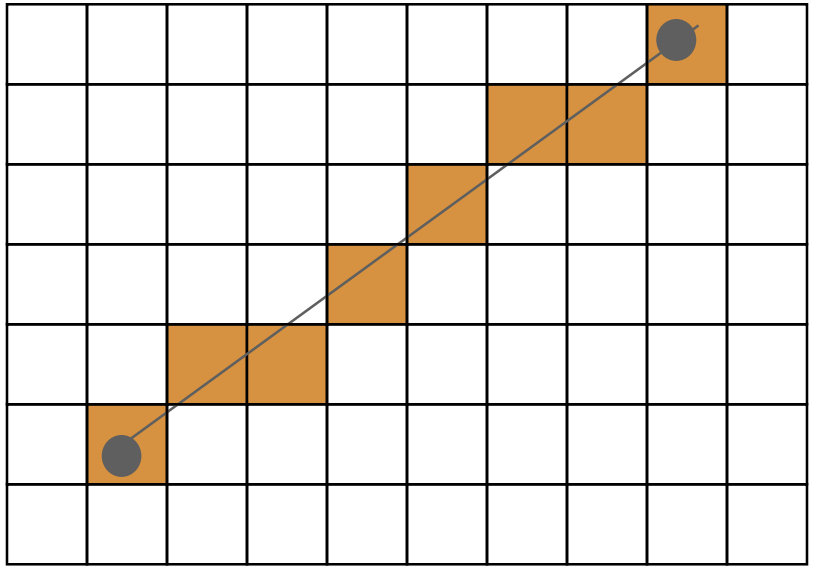


Anti-Aliasing

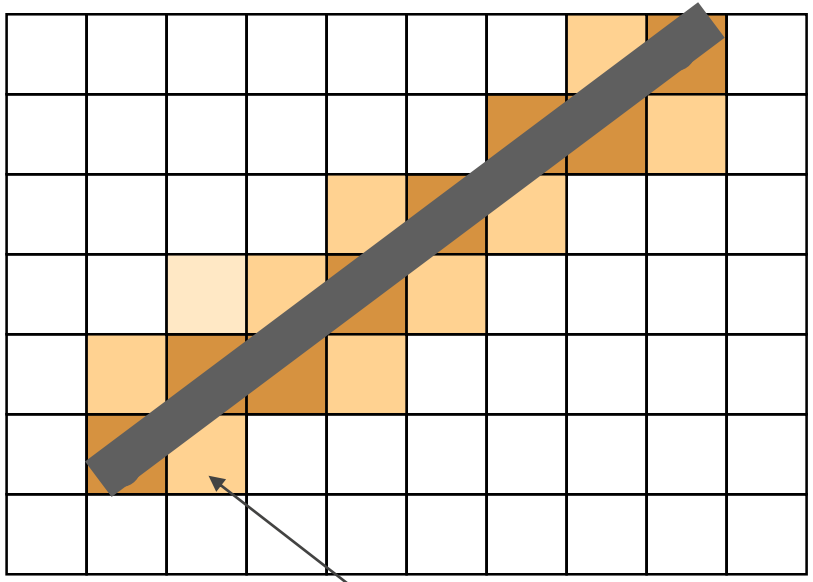
Raster line drawing can make look jaggy!

How can we make a digital line appear less jaggy?

Aliased line



Anti-aliased line



Rasterization or Scan Conversion of triangles

Rasterize horizontal span of pixels between a pair of triangle edges.

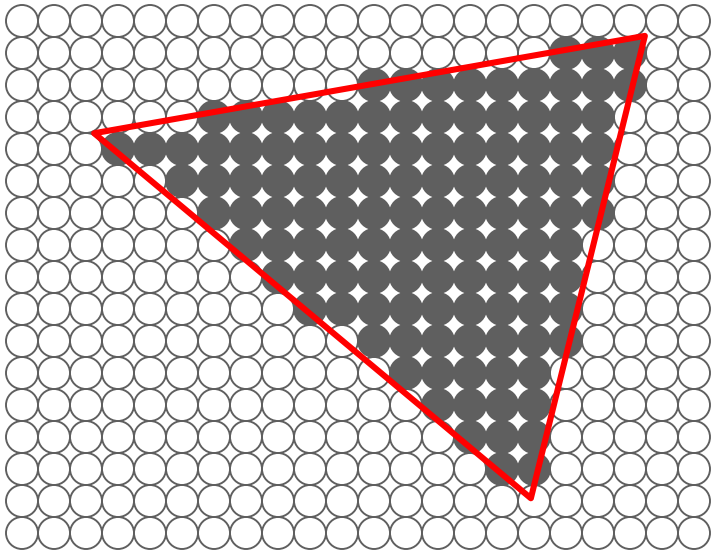


Image making algorithms

Object-Order

for-each ***object***

update the ***pixels*** the ***object*** influences;

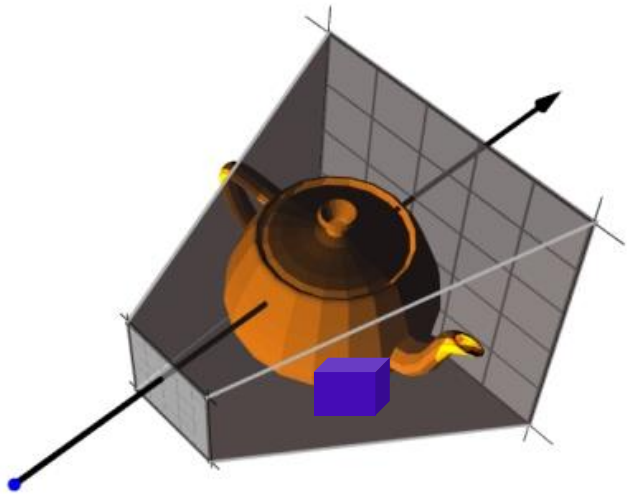


Image-Order

for-each ***pixel***

set the ***pixel*** based on the ***objects*** that influence it;