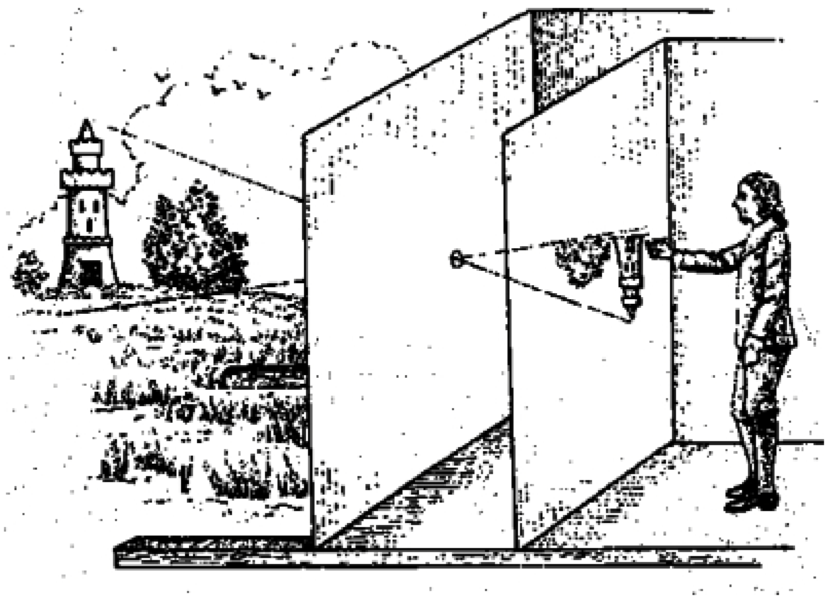
Computer Vision and Computer Graphics: Two sides of a coin

COS 116: Apr 19, 2011

Sanjeev Arora

Brief history of image-making



Camera Obscura
(known to the Chinese, 5th c. BC)

19th c.: replace hole with lens; coat paper with light-sensitive paper; "Camera."

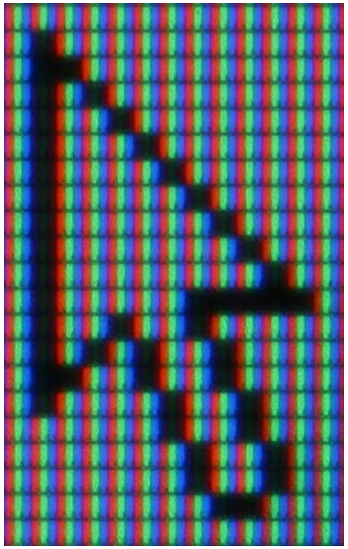
Late 20th c.: replace light-sensitive paper with digital sensor + memory card: "digital camera."

Theme 1: What is an image?

What is an image?

- Rectangular (2D) array of pixels

Continuous image



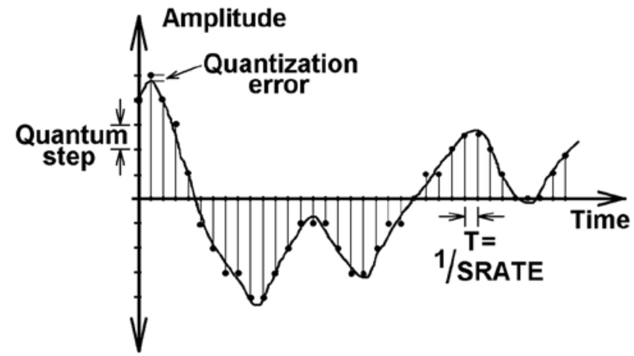
Digital image

"Pixels"

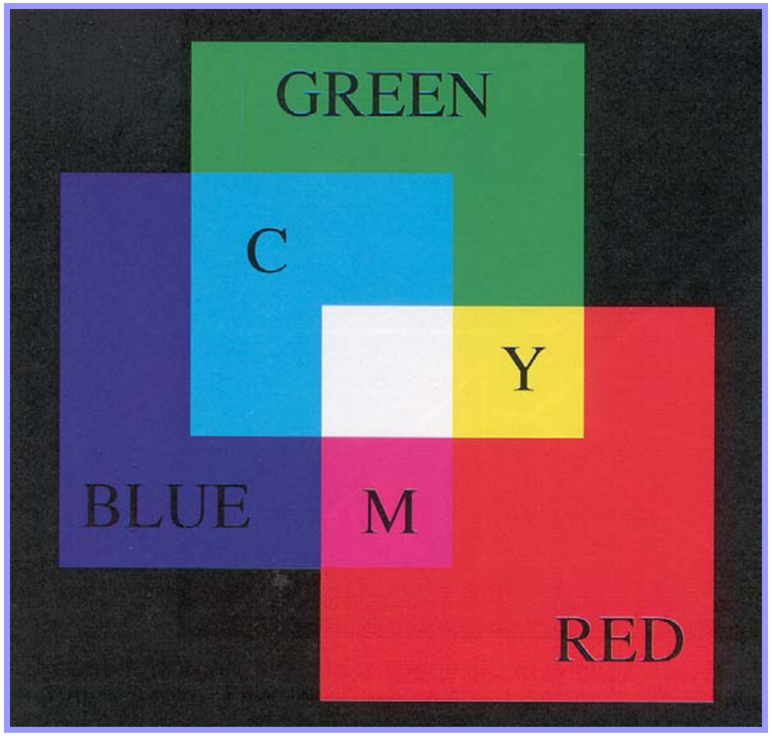
“Pixel” is a sample; need not be square

(Many choices for “rendering” the same information)

(Remember music lecture:



RGB Color Model



Colors are additive

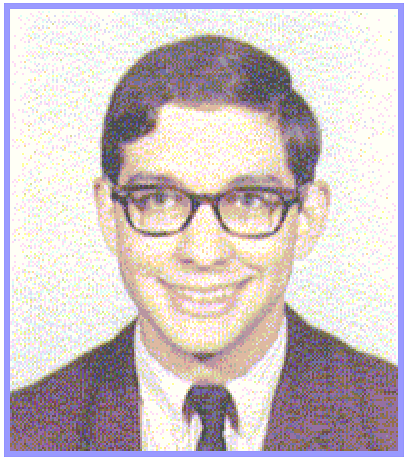
Plate II.3 from FvDFH

R	G	B	Color
0.0	0.0	0.0	Black
1.0	0.0	0.0	Red
0.0	1.0	0.0	Green
0.0	0.0	1.0	Blue
1.0	1.0	0.0	Yellow
1.0	0.0	1.0	Magenta
0.0	1.0	1.0	Cyan
1.0	1.0	1.0	White
0.5	0.0	0.0	?
1.0	0.5	0.5	?
1.0	0.5	0.0	?
0.5	0.3	0.1	?

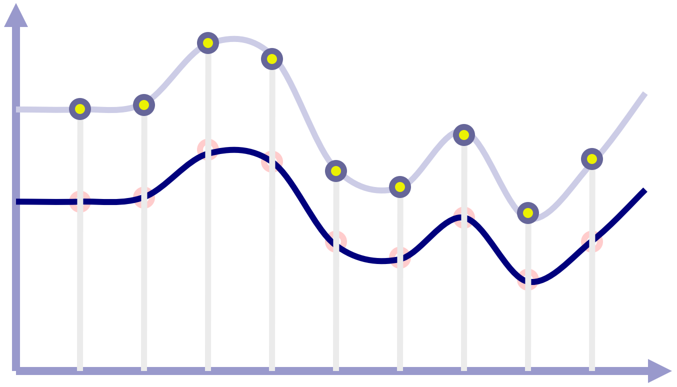
Adjusting Brightness

- Simply scale pixel components
  - Must clamp to range (e.g., 0 to 1)

Original



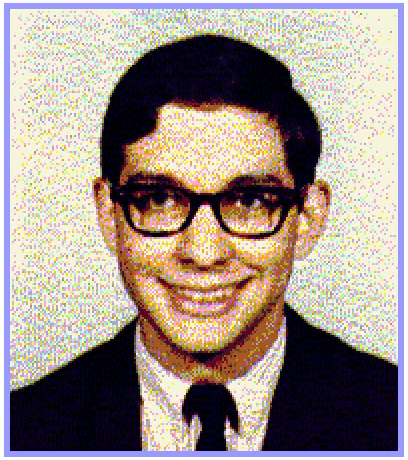
Brighter



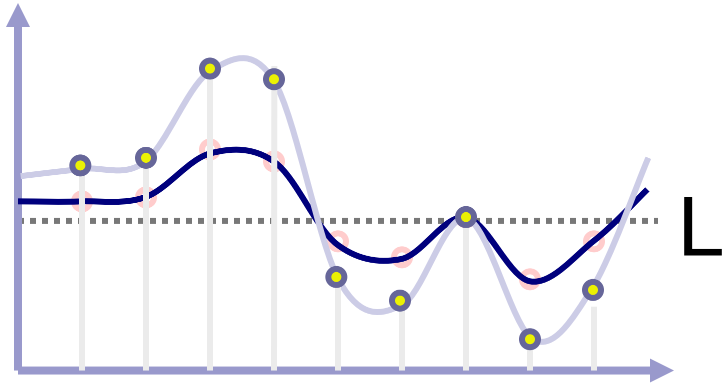
Adjusting Contrast

- Compute average luminance L for all pixels
  - luminance = 0.30*r + 0.59*g + 0.11*b
- Scale deviation from L for each pixel
  - Must clamp to range (e.g., 0 to 1)

Original



More Contrast



Scaling the image

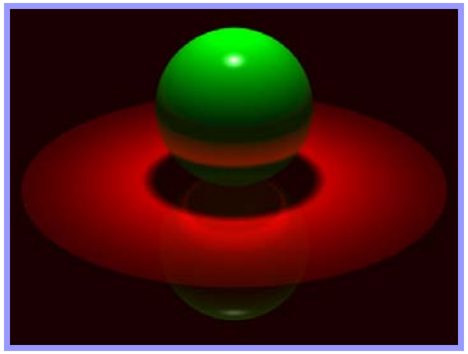
- Resample with fewer or more pixels (mathy theory…)

Original

1/4X resolution

4X resolution

Theme 2: Computer vision
vs
Computer Graphics
(and why they get mathy)

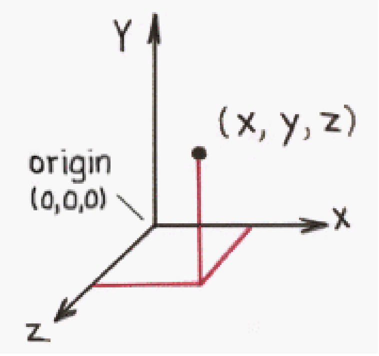


Computer vision: understanding "content" of an image (end result: "model" of the depicted scene)

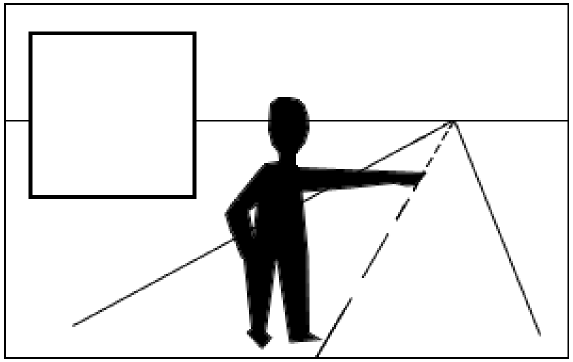
Computer graphics: Start with a computer "model", create an image using it.

Math needed to describe images

(1) Coordinate geometry (turns geometry into algebra)

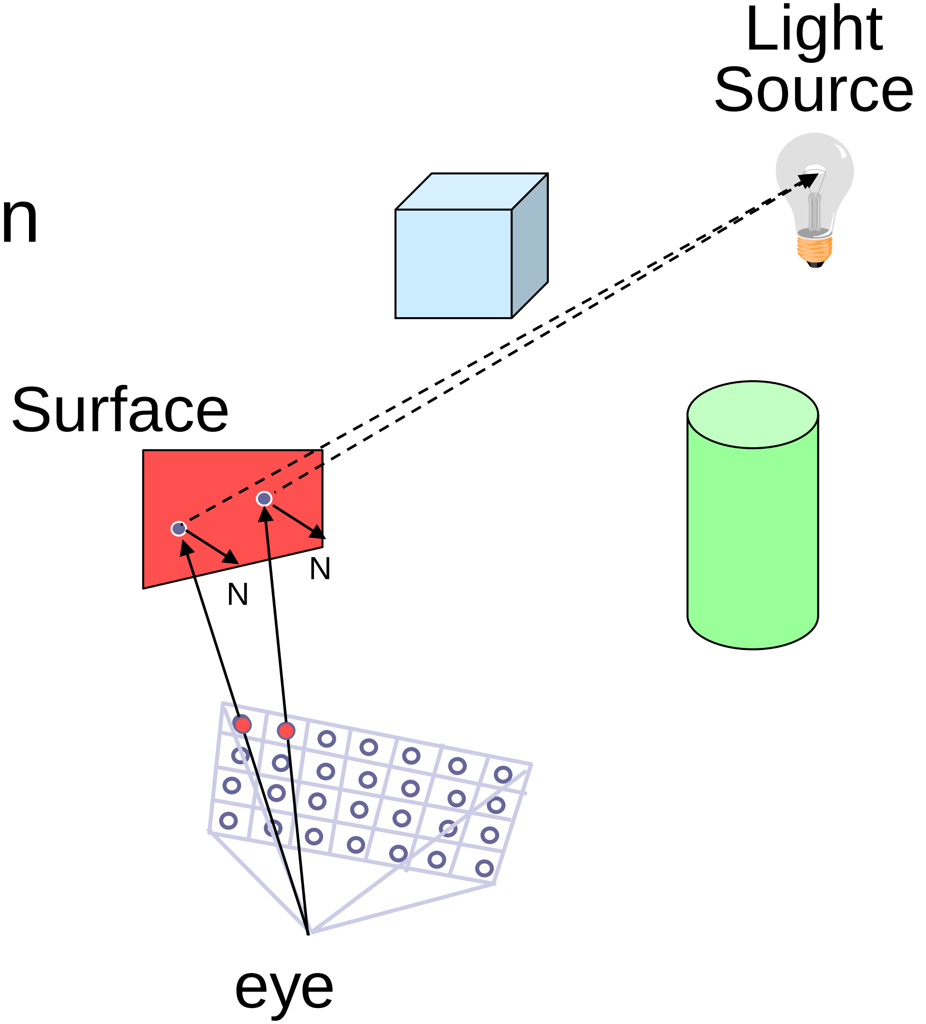


2) Laws of perspective



(Math needed..) Physics of light

- Lighting parameters
  - Light source emission
  - Surface reflectance



Math needed in the design of algorithms

Example: Image Morphing

[Beier & Neeley]

Image_0 Warp_0

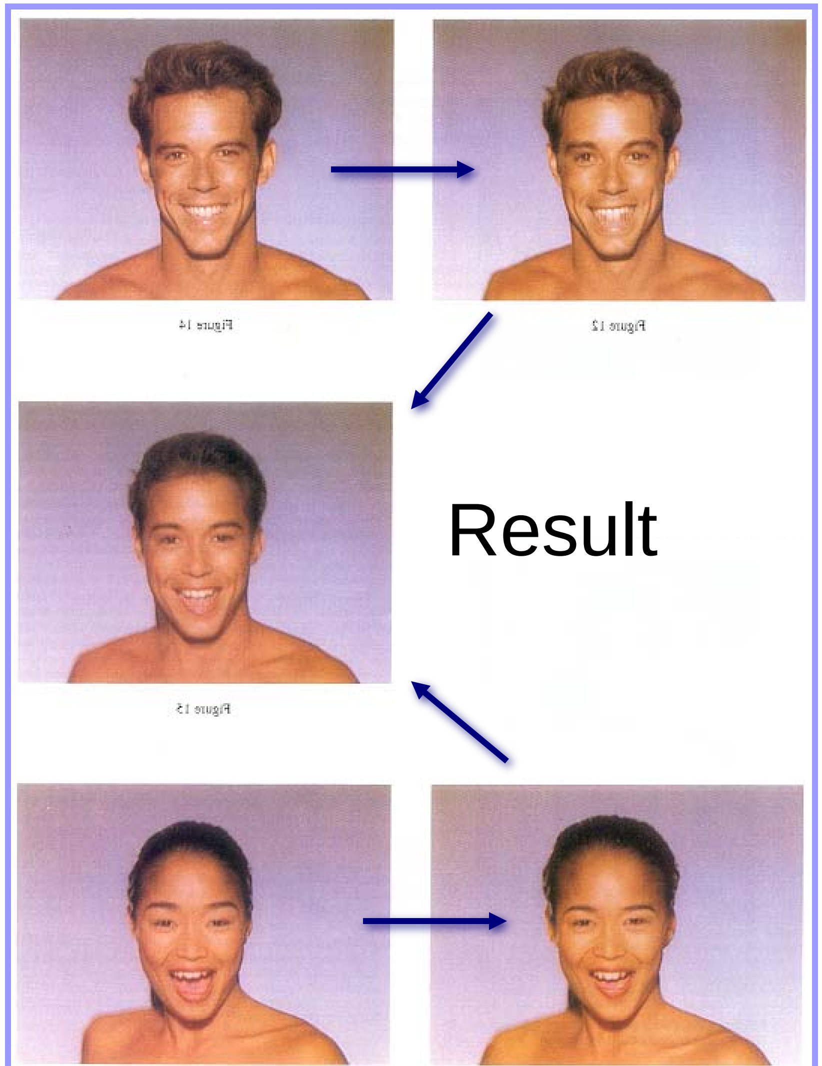


Image_1 Warp_1

Intro to computer vision

What is depicted in the image?

Edge detection

What is an “edge”?

Place where image “changes” suddenly

How to identify edges?

A very simple edge detection idea

$$A[i,j] \leftarrow 5A[i,j] - A[i+1,j] - A[i-1,j] - A[i,j+1] - A[i,j-1]$$

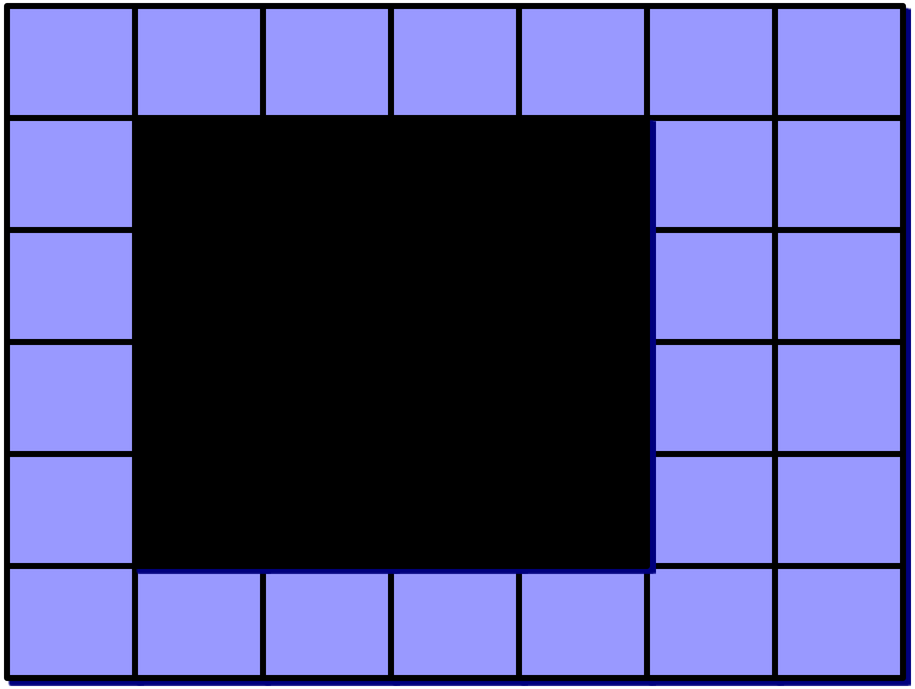
More sophisticated edge-detection uses smarter versions of this; use Gaussian filters, etc.

Human eye does some version of edge detection

Edge information is still too “low level.”

What does the pseudocode on prev. page do on this image?

$$A[i,j] \leftarrow 5A[i,j] - A[i+1,j] - A[i-1,j] - A[i,j+1] - A[i,j-1]$$

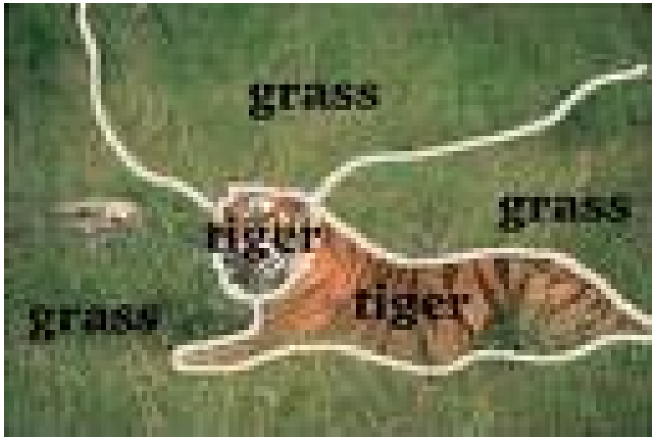


More high level understanding: Image Segmentation

What are the regions in this image?

Uses many many algorithmic ideas; still not 100% accurate

High level vision: Object recognition

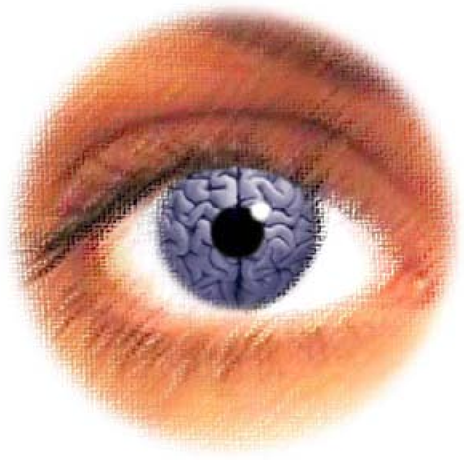


What do you see in this picture?

Much harder task than it may seem. Tiger needs to be recognized from any angle, and under any lighting condition and background.

Aside

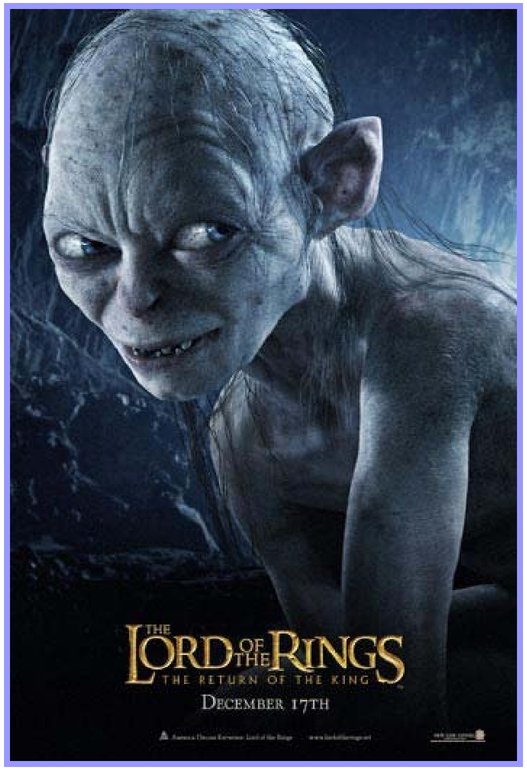
At least 8 “levels” in human vision system. Object recognition seems to require transfer of information between levels, and the highest levels seem tied to rest of intelligence.

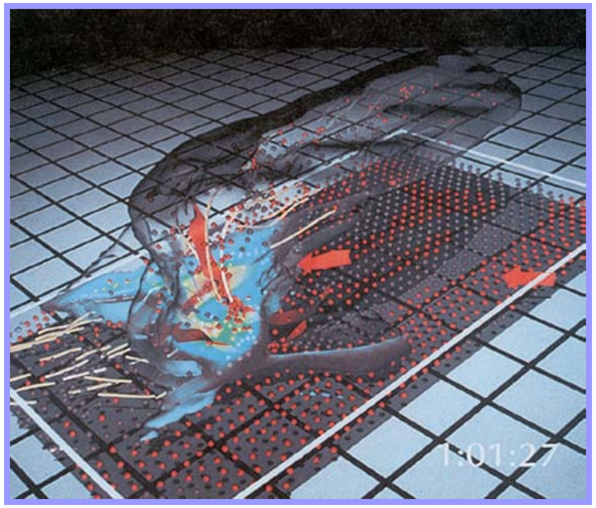


Next: Computer Graphics

Applications:

- Entertainment
- Computer-aided design
- Scientific visualization
- Training
- Education
- E-commerce
- Computer art





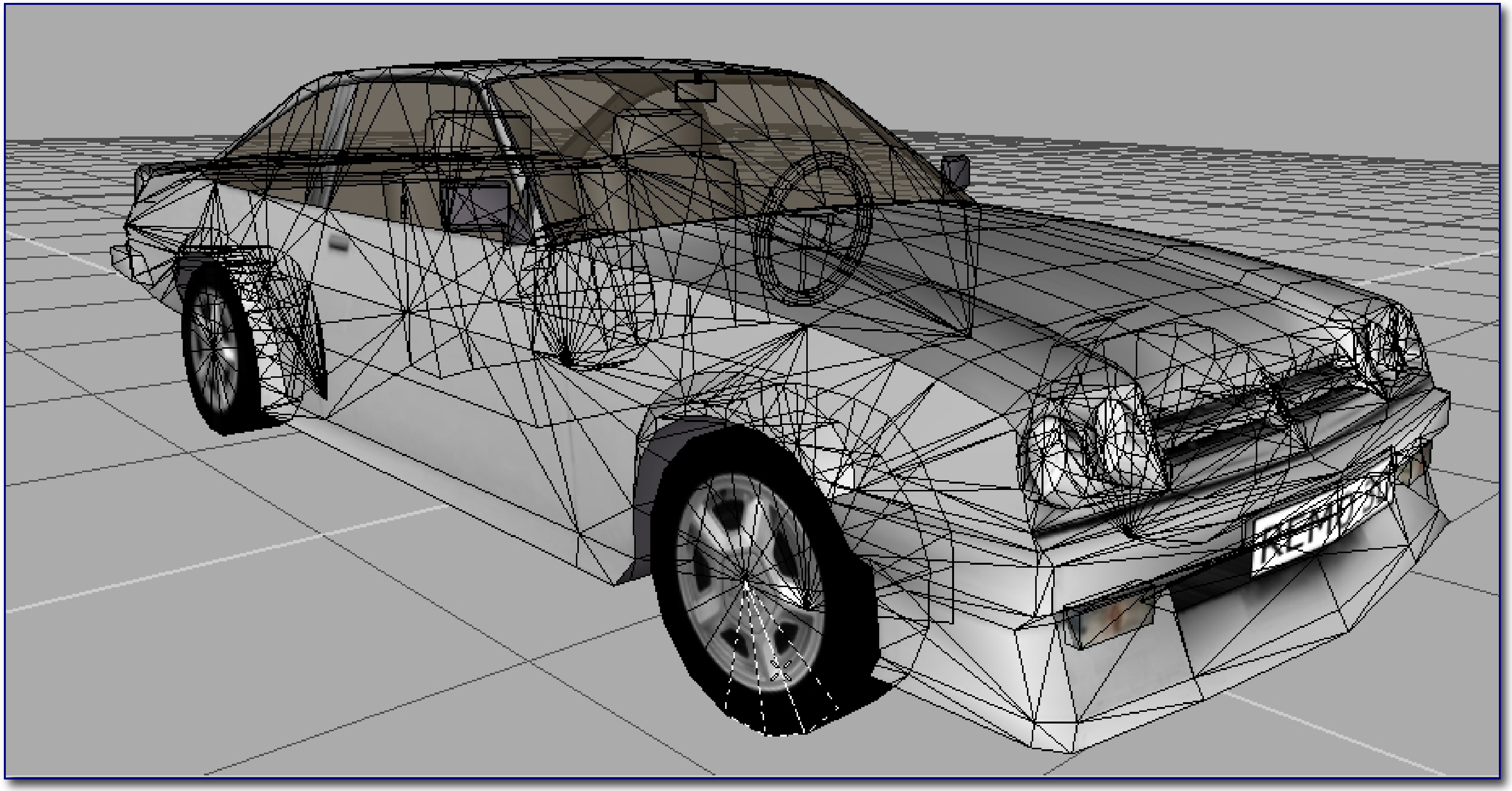
Inside a Thunderstorm

(Bob Wilhelmson, UIUC)

Boeing 777 Airplane

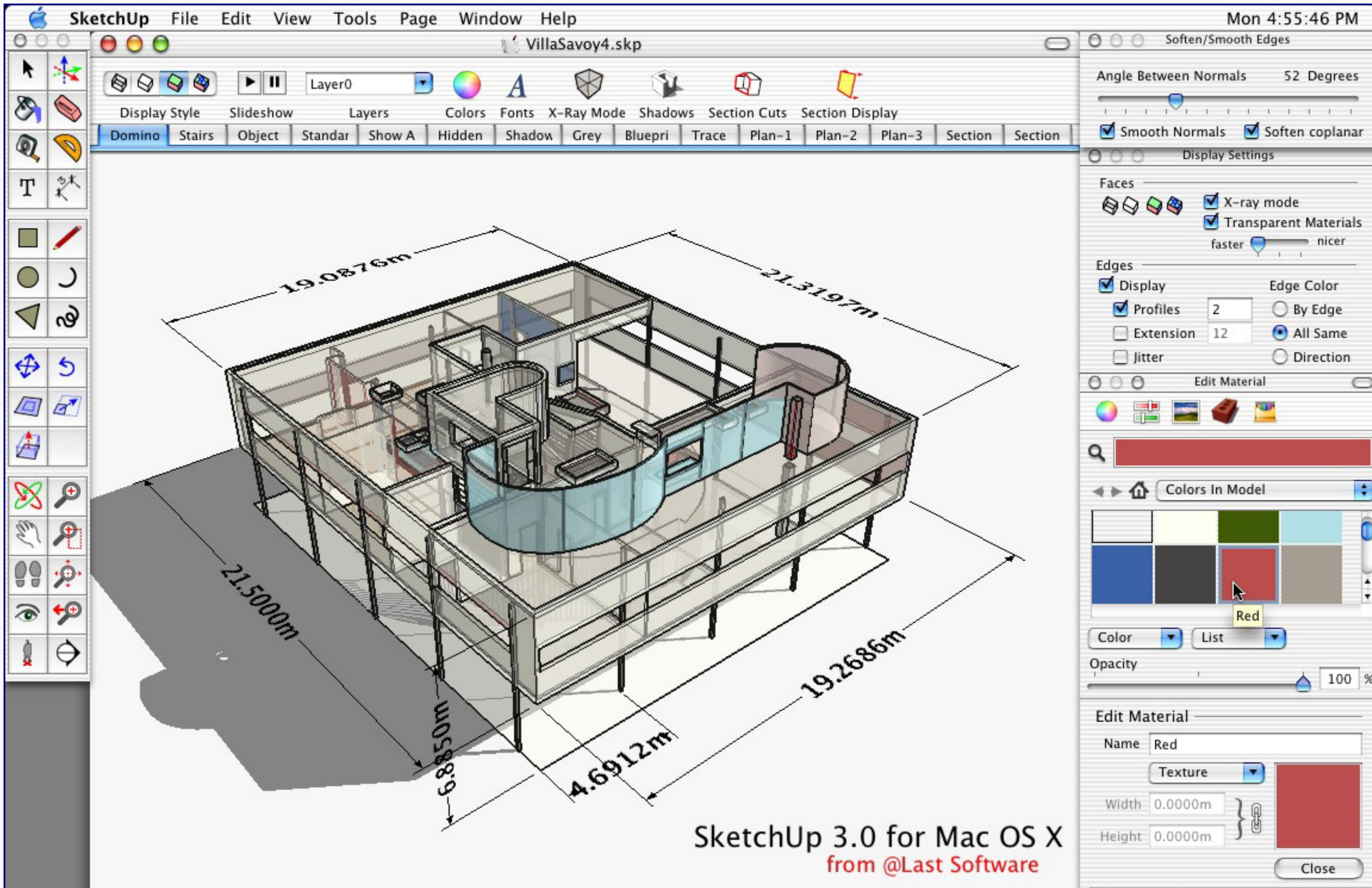
Step 1: Modeling

- How to construct and represent shapes (in 3D)



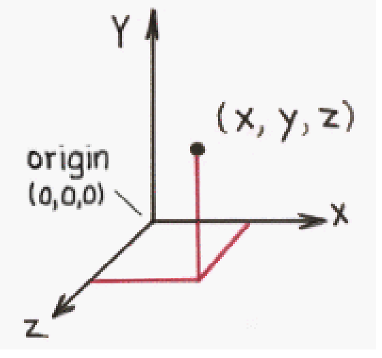
(Remo3D)

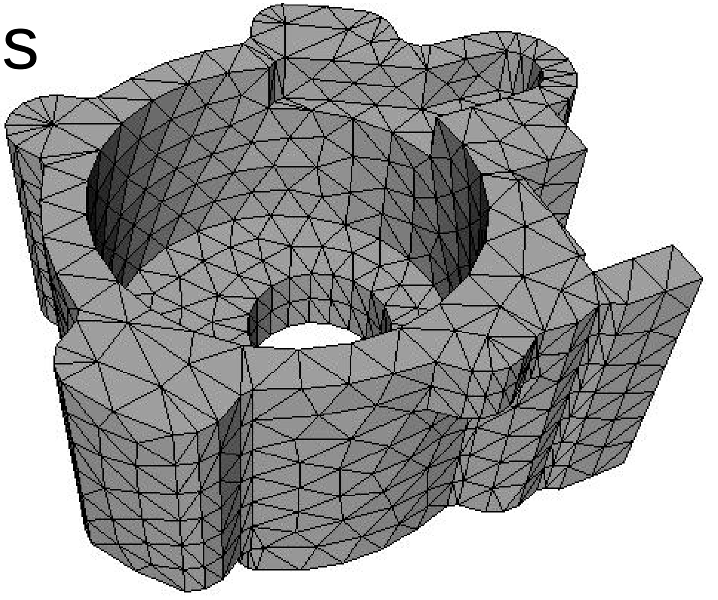
Modeling in SketchUp (demo)

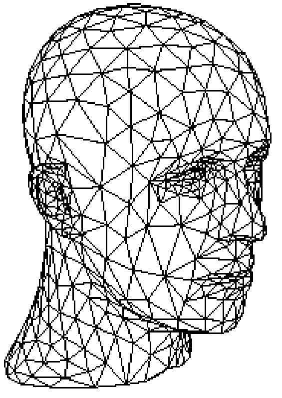


Example of “model”: wireframe

- Most common: list of triangles
  - Three vertices in 3D
    (x_1, y_1, z_1)
    (x_2, y_2, z_2)
    (x_3, y_3, z_3)



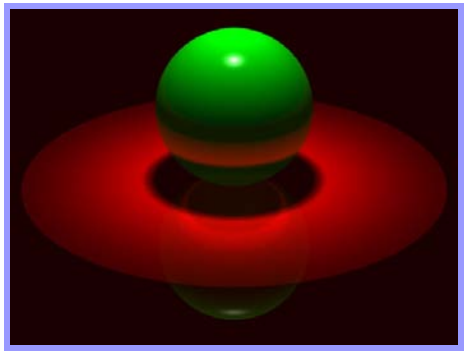




Usually would be augmented with info about texture, color etc.

Step 2: Rendering

- Given a model, a source of light, and a point of view, how to render it on the screen?



Rendering (contd)

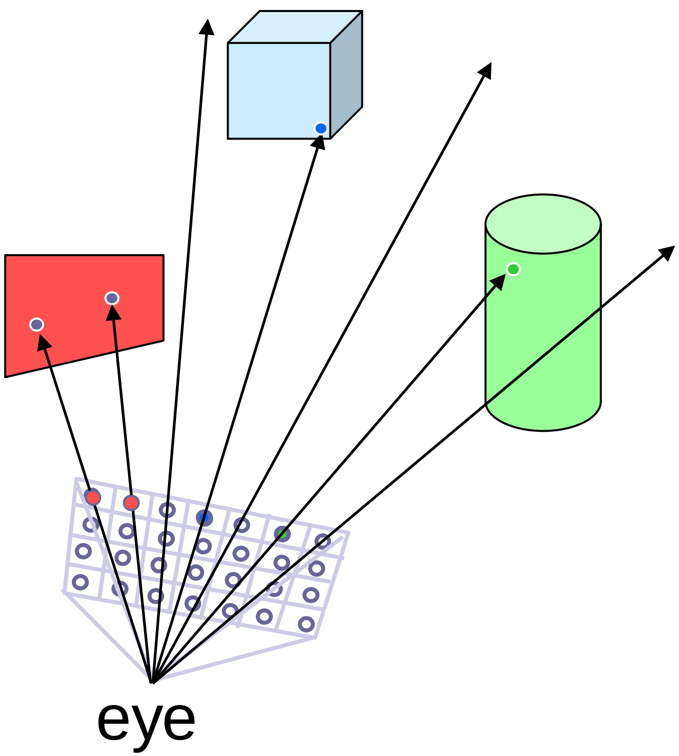
- Direct illumination
  - One bounce from light to eye
  - Implemented in graphics cards
  - OpenGL, DirectX, …
- Global illumination
  - Many bounces
  - Ray tracing

Direct Illumination
(Chi Zhang, CS 426, Fall99)

Ray Tracing
(Greg Larson)

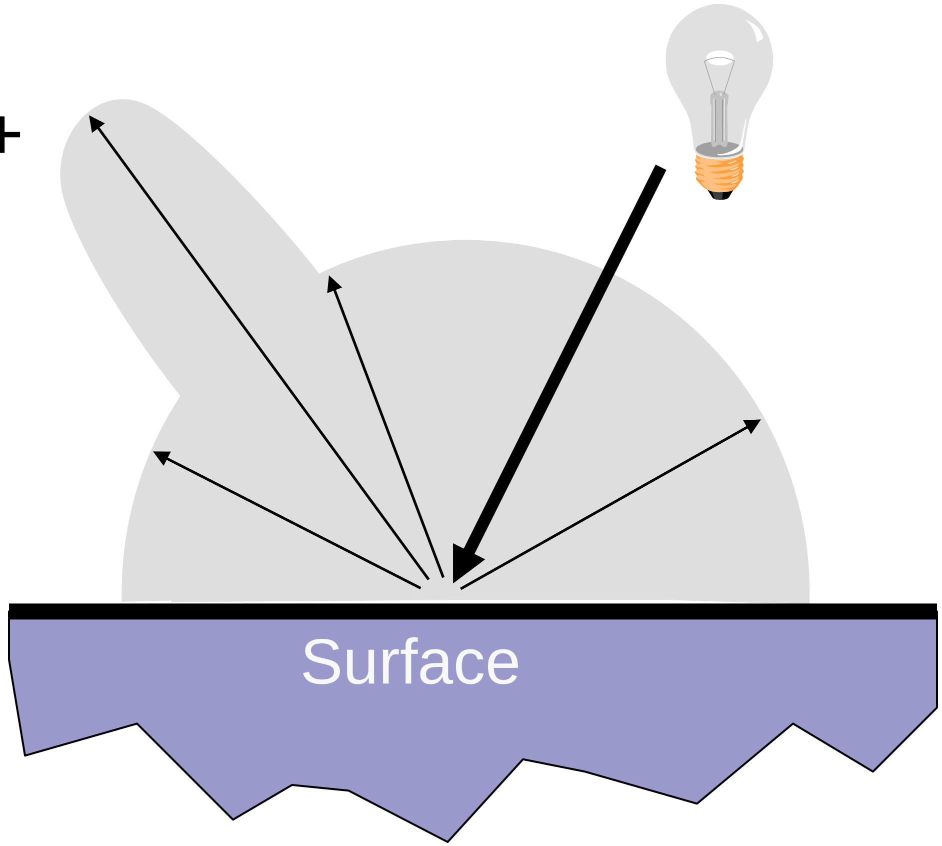
Ray Casting

- A (slow) method for computing direct illumination
- For each sample:
  - Construct ray from eye through image plane
  - Find first surface intersected by ray
  - Compute color of sample based on surface properties



Simple Reflectance Model

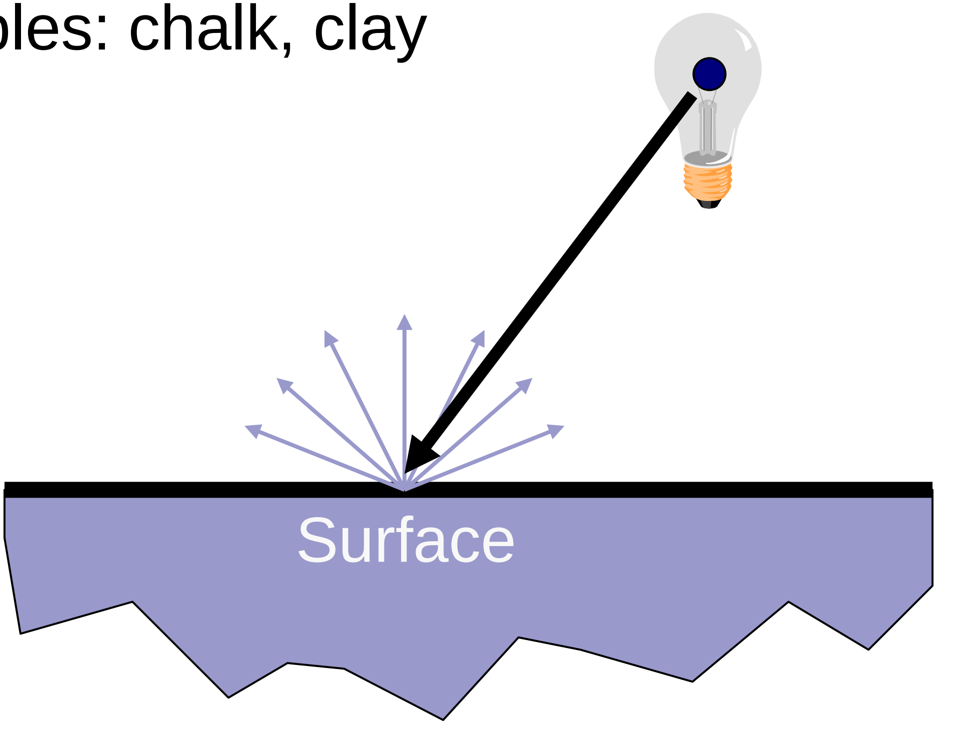
- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - ambient lighting



Based on model proposed by Phong

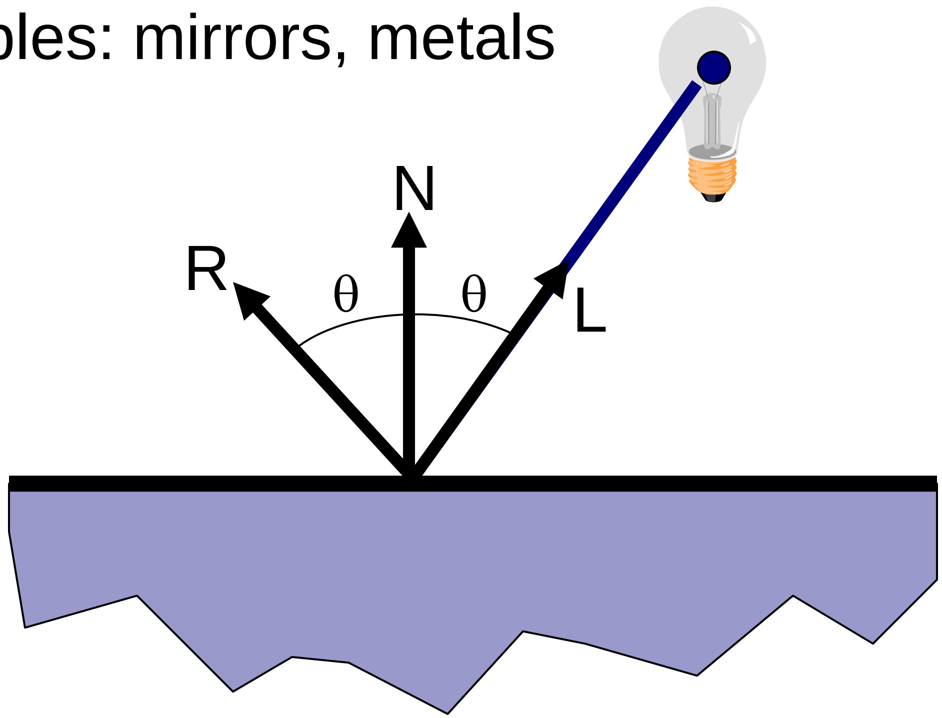
Diffuse Reflection

- Assume surface reflects equally in all directions
  - Examples: chalk, clay



Specular Reflection

- Reflection is strongest near mirror angle
  - Examples: mirrors, metals



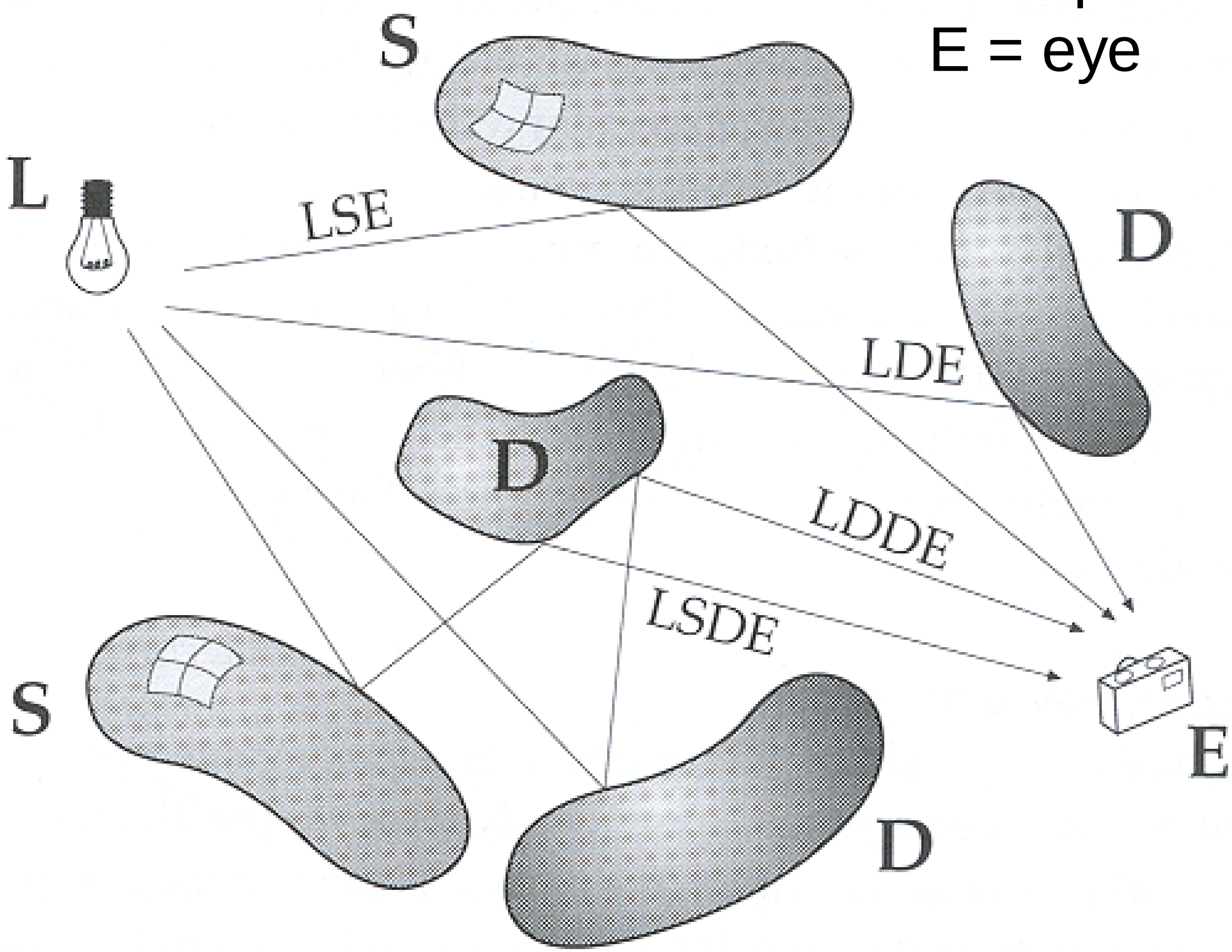
Ambient Lighting

- Represents reflection of all indirect illumination

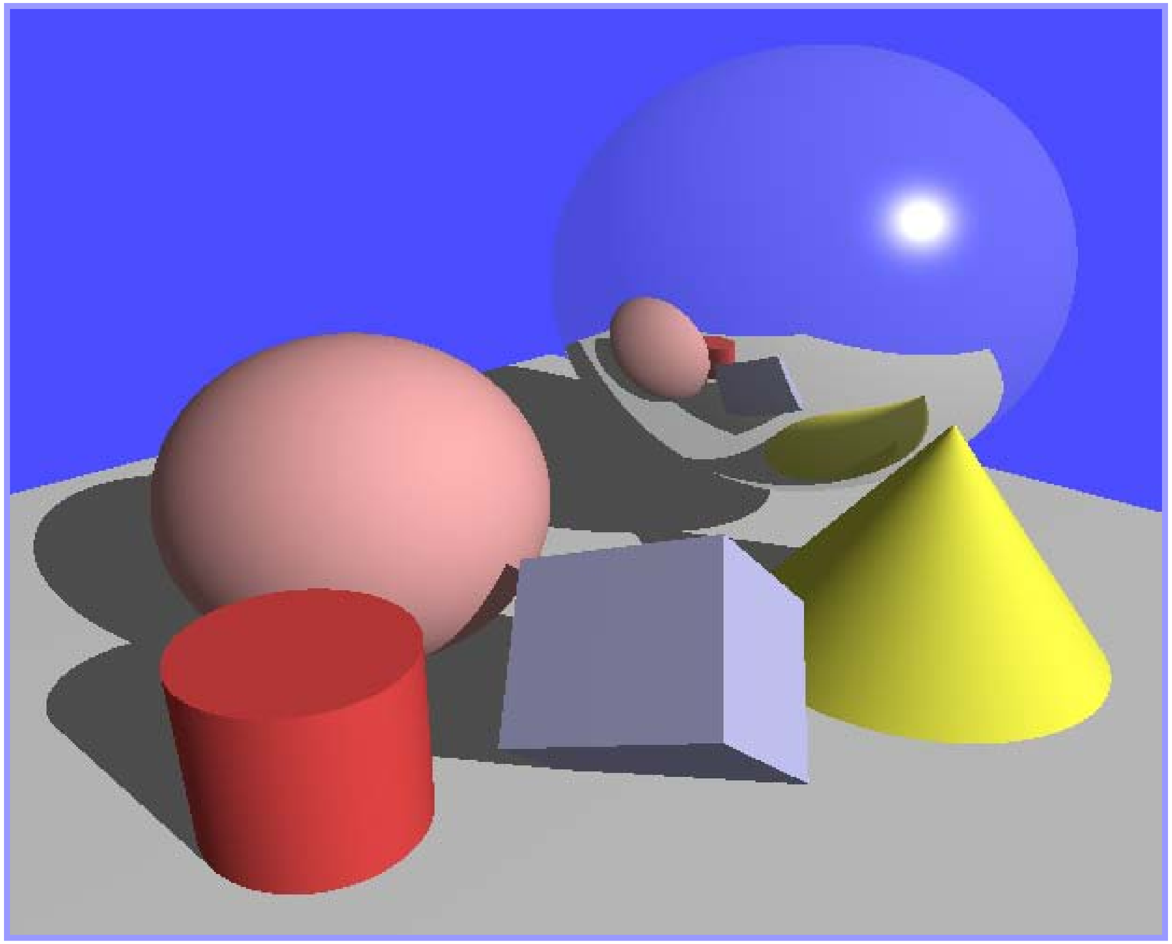
This is a total cheat (avoids complexity of global illumination)!

Path Types

L = light
D = diffuse bounce
S = specular bounce
E = eye

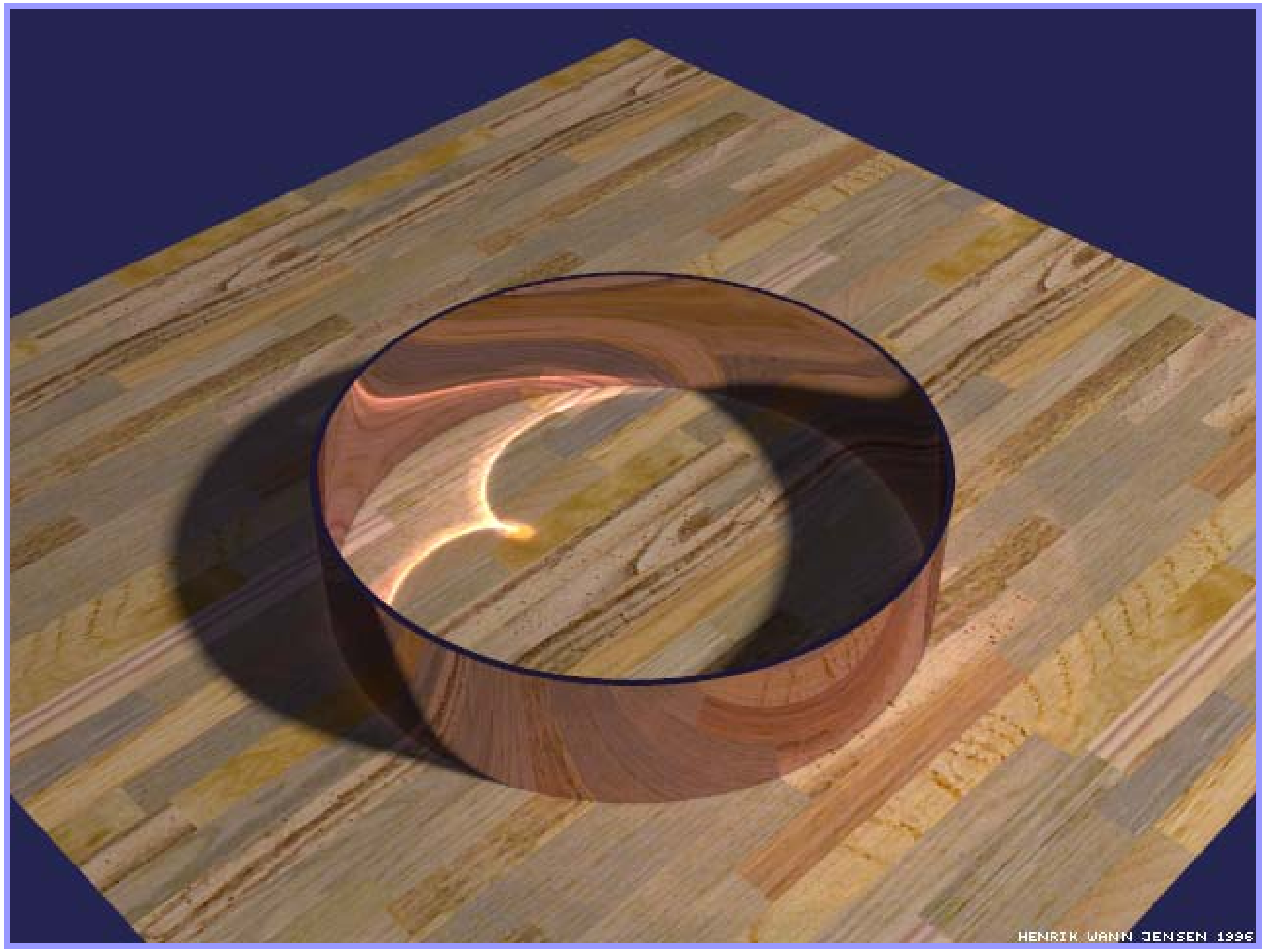


Path Types?



Henrik Wann Jensen

Ray Tracing



Henrik Wann Jensen

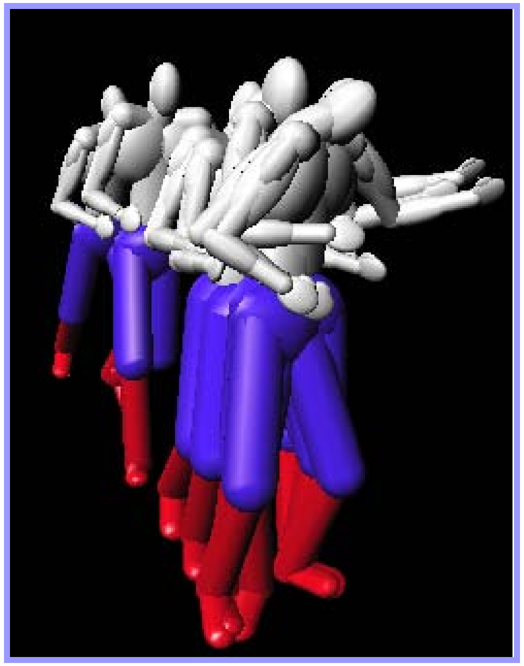
Ray Tracing

Ray Tracing

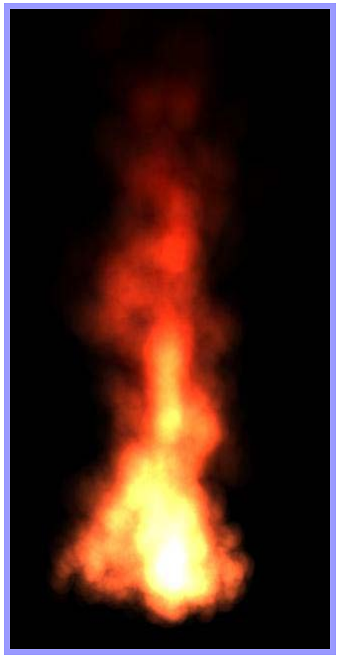
Terminator 2

Step 3: Animation

- Keyframe animation
  - Articulated figures
- Simulation
  - Particle systems



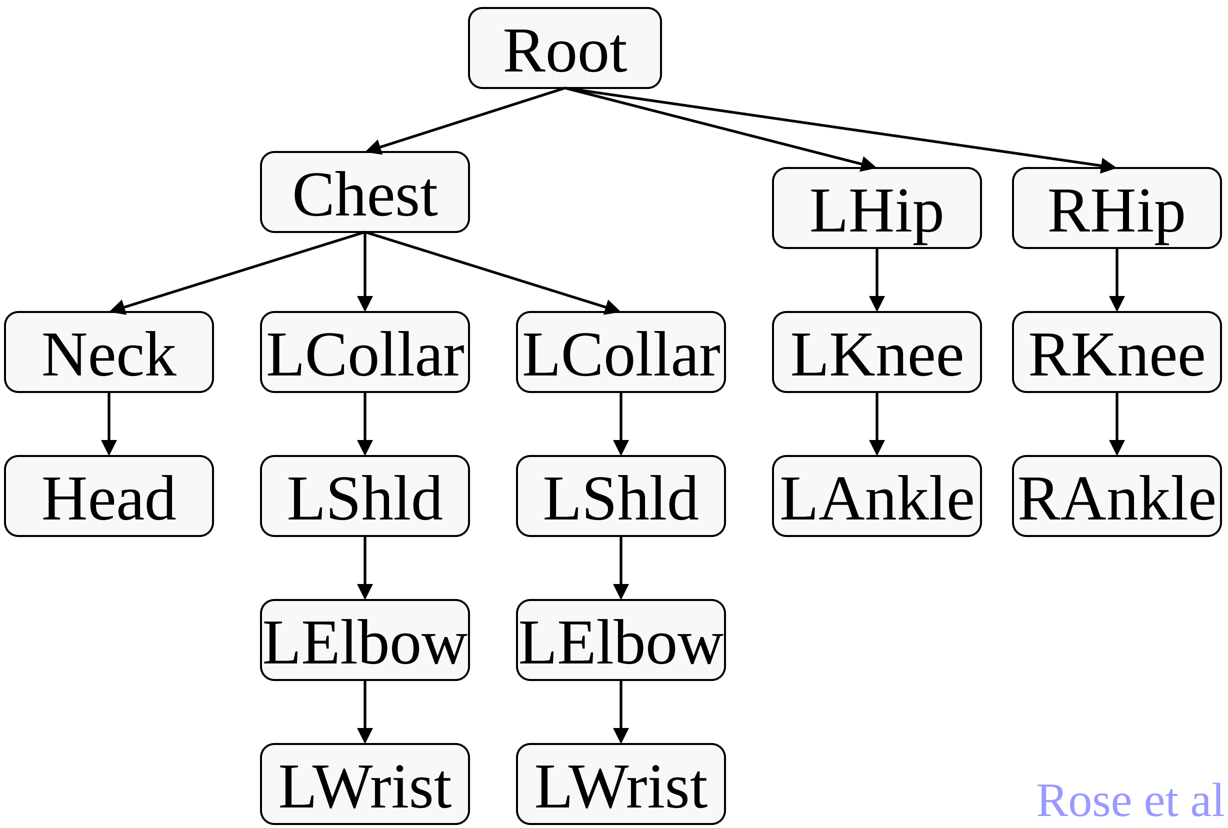
Animation
(Jon Beyer, CS426, Spring04)



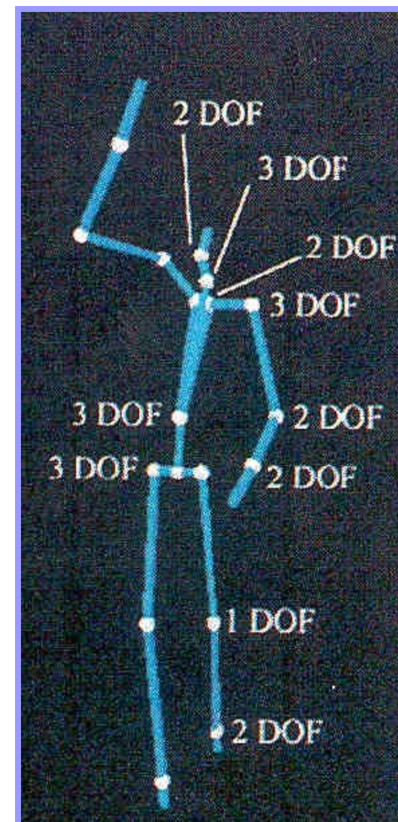
Simulation

Articulated Figures

- Well-suited for humanoid characters



Rose et al. `96

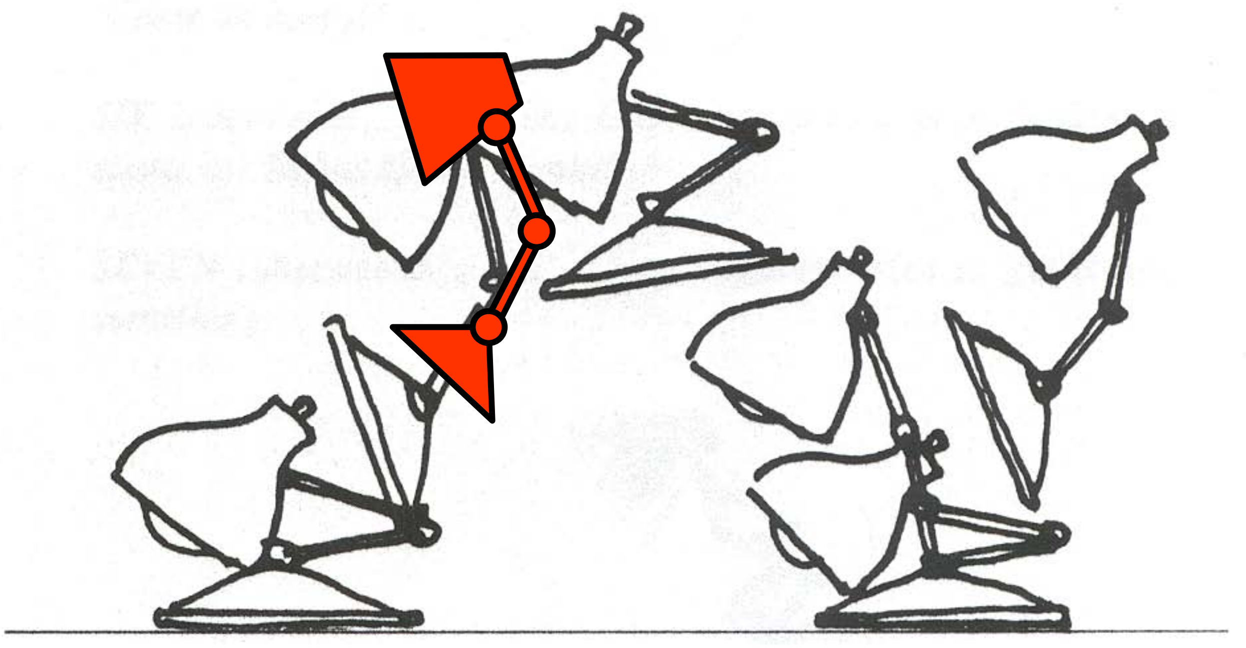


Keyframe Animation: Luxo Jr.

Pixar

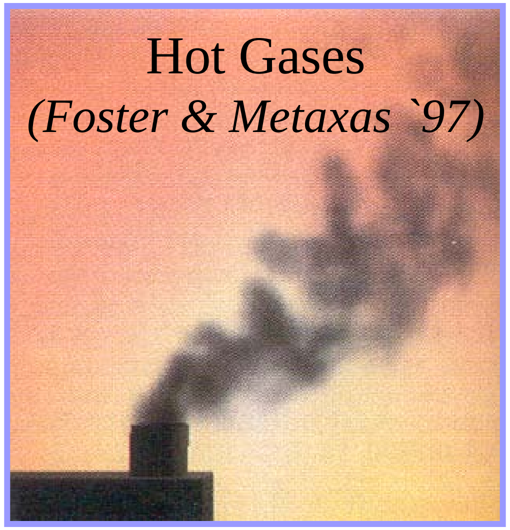
Keyframe Animation

- Define character poses at specific times: “keyframes”
- “In between” poses found by interpolation

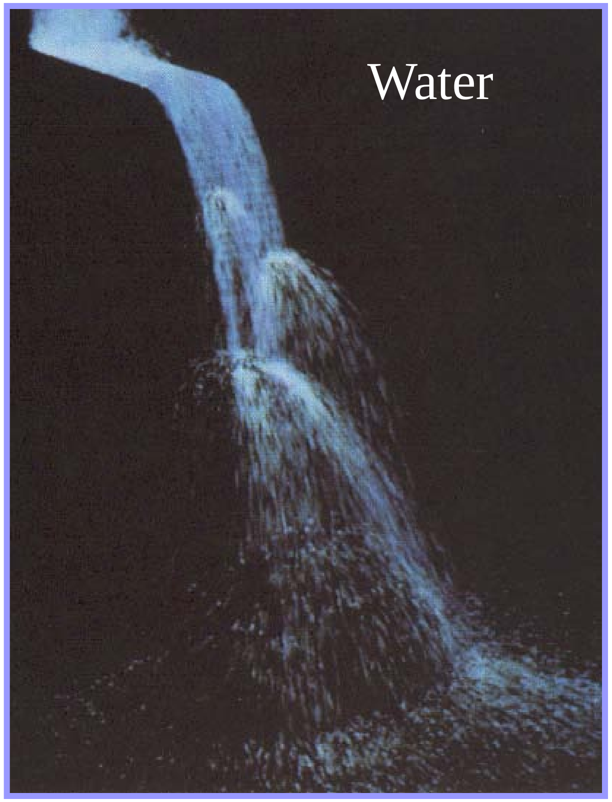


Lasseter `87

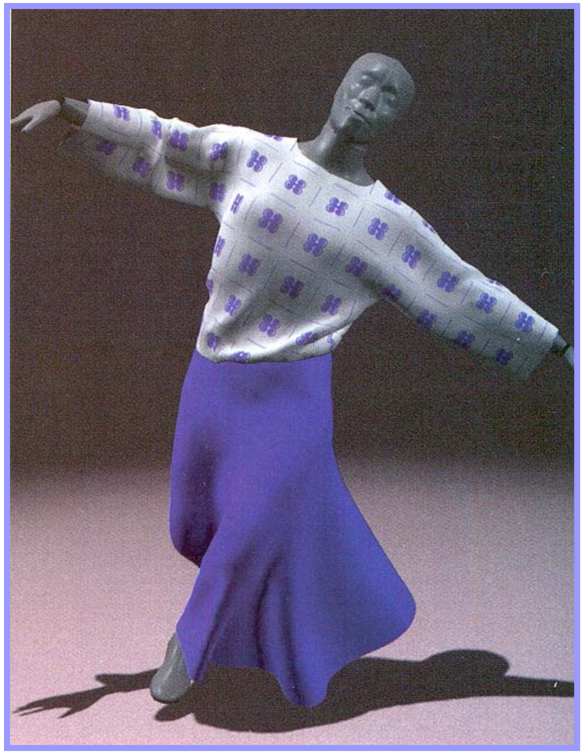
- But, animator cannot specify motion for:
  - Smoke, water, cloth, hair, fire
  - Soln: animation!



Hot Gases
(Foster & Metaxas `97)



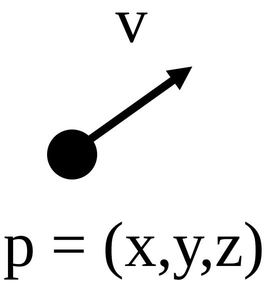
Water



Cloth
(Baraff & Witkin `98)

Particle Systems

- A particle is a point mass
  - Mass
  - Position
  - Velocity
  - Acceleration
  - Color
  - Lifetime
- Many particles to model complex phenomena
  - Keep array of particles



Particle Systems

- Recall game of life, weather etc.…
- For each frame (time step):
  - Create new particles and assign attributes
  - Delete any expired particles
  - Update particles based on attributes and physics
    Newton's Law: f=ma
  - Render particles