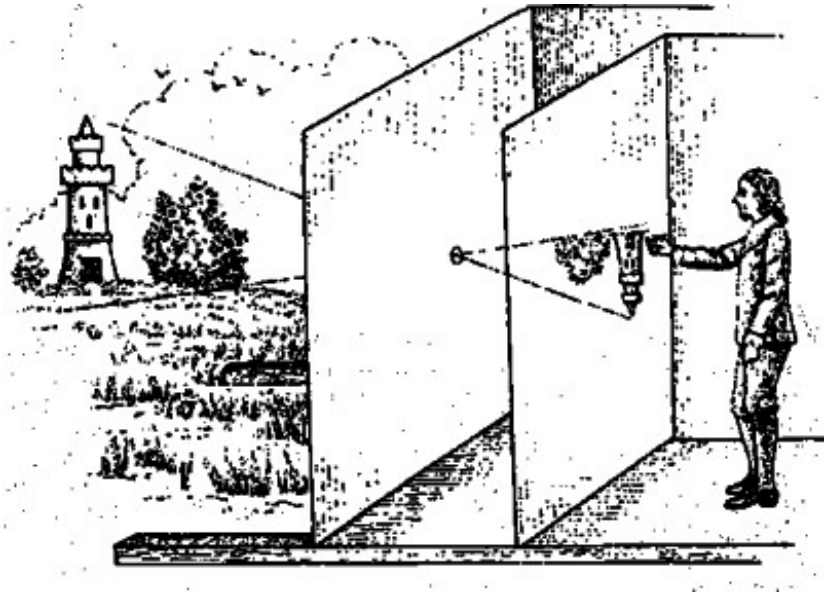
Computer Vision and Computer Graphics: Two sides of a coin

COS 116: Apr 22, 2008

Sanjeev Arora

Brief history of image-making



Camera obscura.

Known to chinese; 5th century BC

19th century: Replace hole with lens; sketchpaper with light-sensitive paper. “Camera”

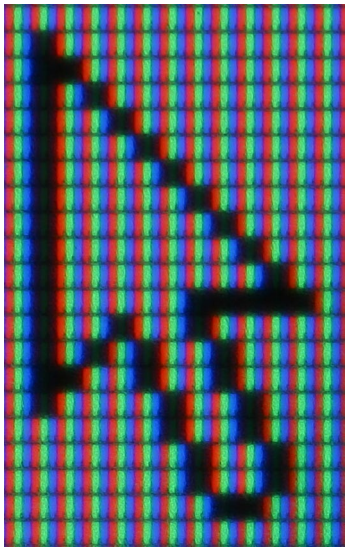
Late 20th century: Replace light-sensitive paper with electronic light sensor: “Digital camera.”

Theme 1: What is an image?

What is an image?

- Rectangular (2D) array of pixels

Continuous image



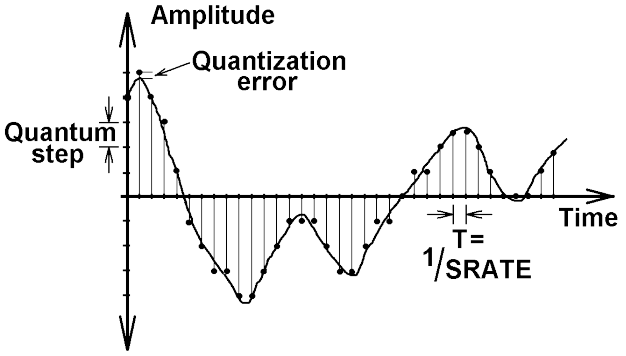
“Pixels”

Digital image

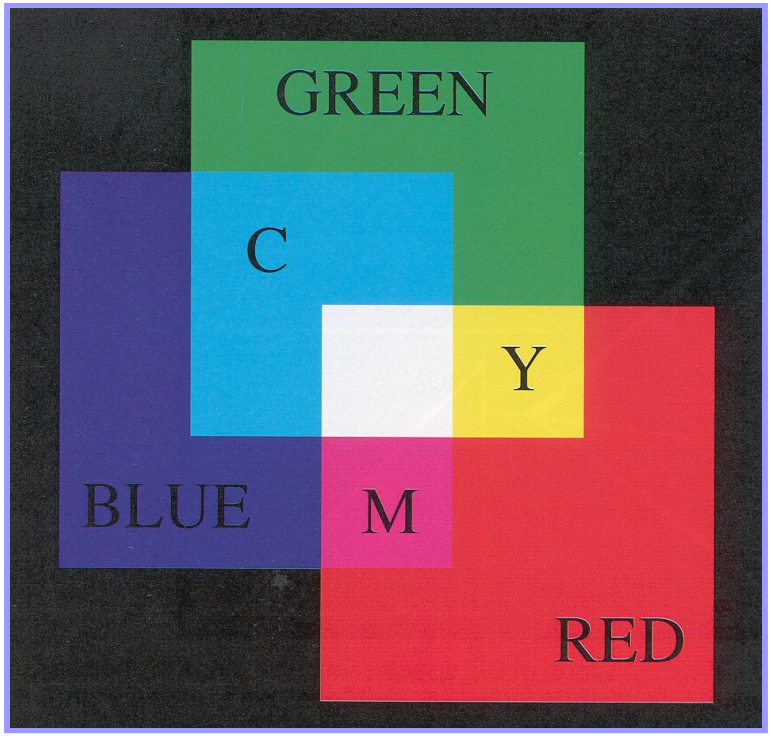
“Pixel” is a sample; need not be square

(Many choices for “rendering” the same information)

(Remember music lecture:



RGB Color Model



Colors are additive

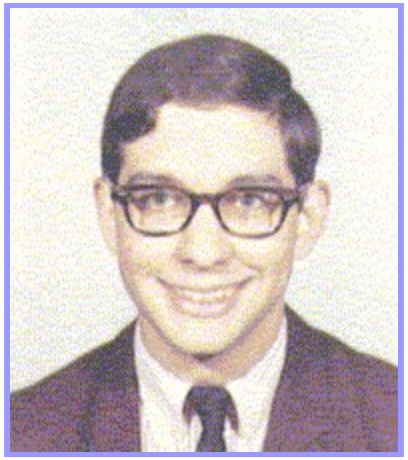
Plate II.3 from FvDFH

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	?

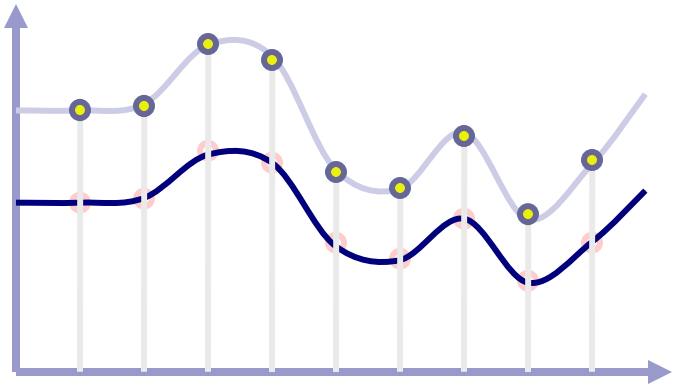
Adjusting Brightness

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

Original



Brighter

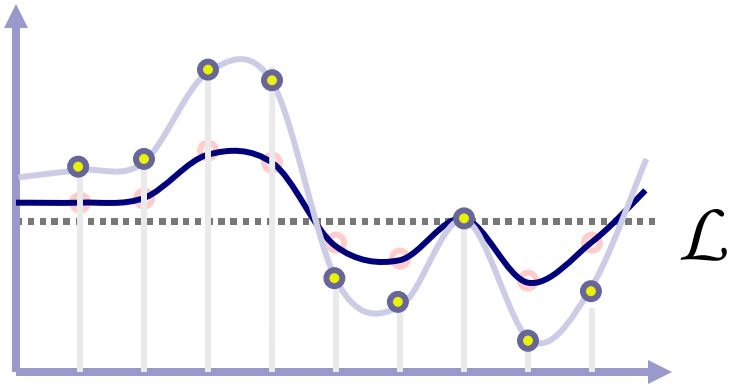


Adjusting Contrast

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

Original

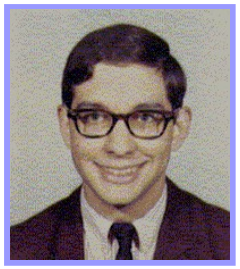
More Contrast



Scaling the image

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

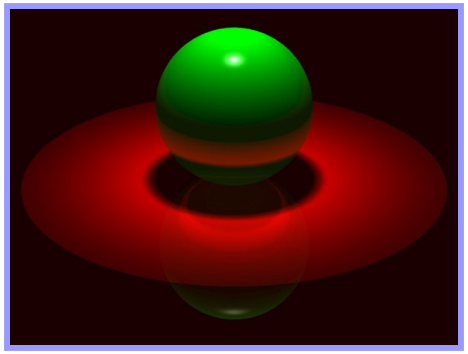
Original



1/4X resolution

4X resolution

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

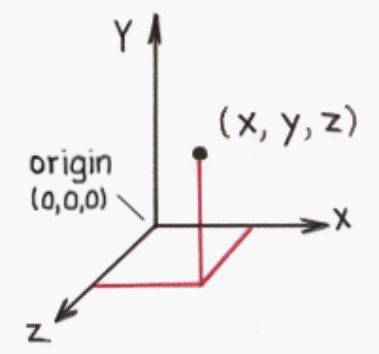


Computer Vision: Understanding the “content” of an image (usually by creating a “model” of the depicted scene)

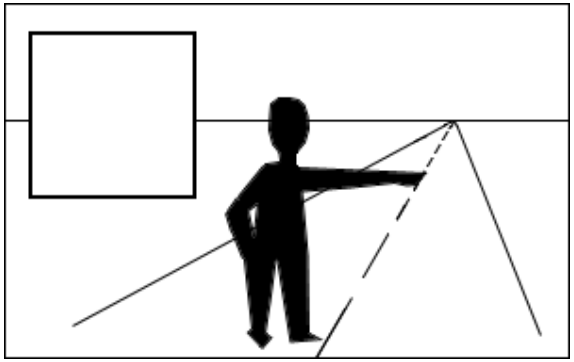
Computer graphics: Creating an image from scratch Using a computer model.

Math used to understand/create images

1) Coordinate geometry (turns geometry into algebra)

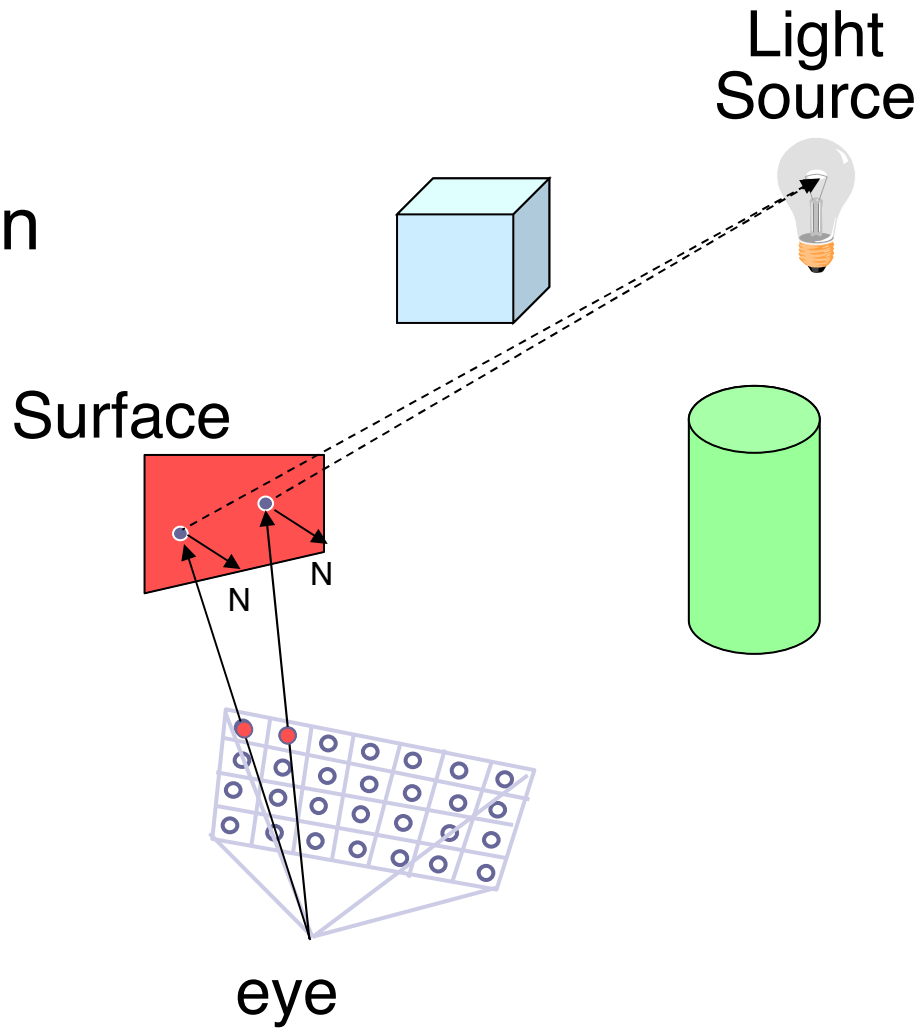


2) Laws of perspective



(Math needed..) Physics of light

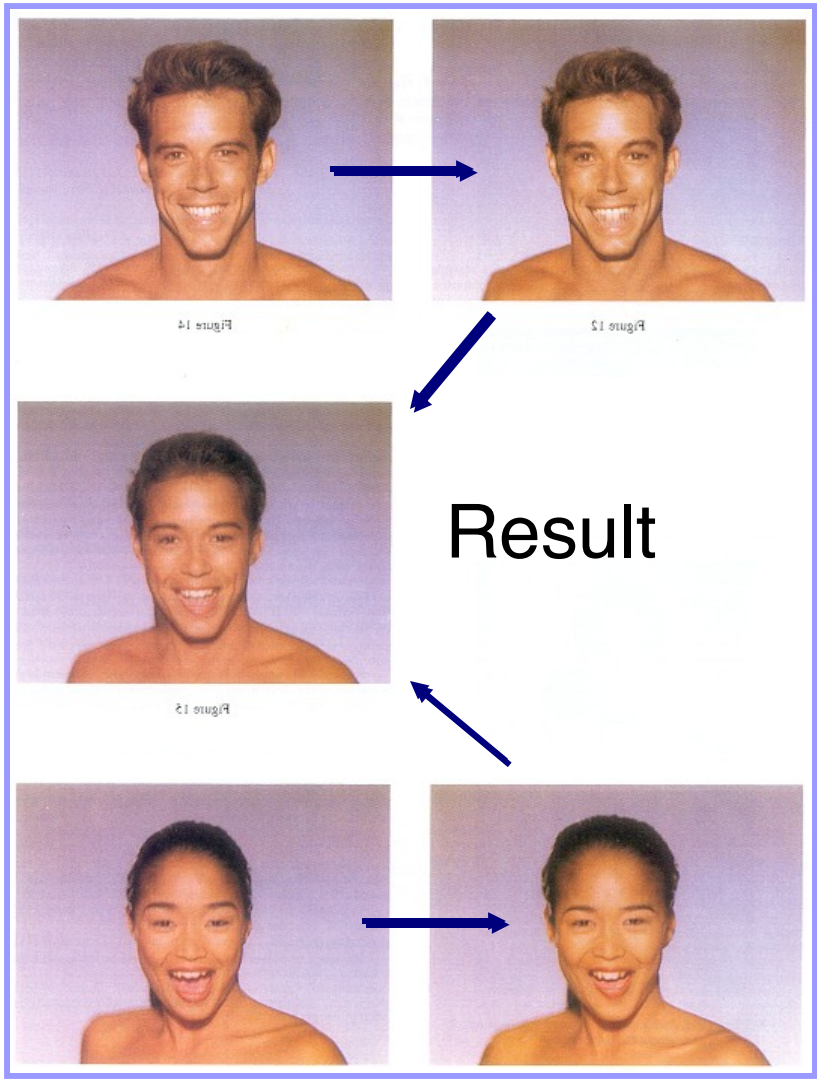
- Lighting parameters
  - Light source emission
  - Surface reflectance



Math needed in the design of algorithms

Example: Image Morphing

[Beier & Neeley]



Intro to computer vision.

What is depicted in this 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] <- 5 A[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.

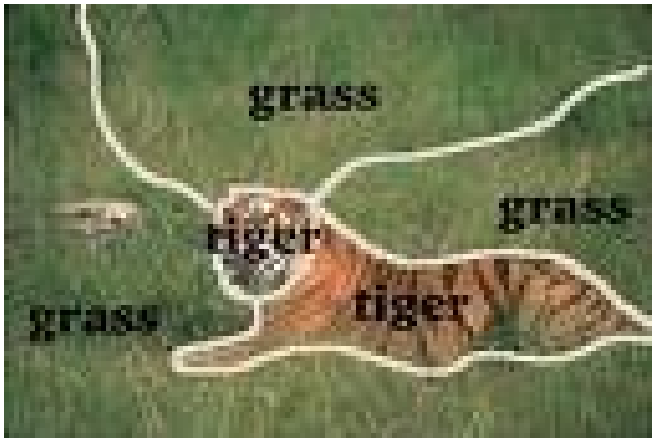
Edge info is still too “low level.”

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.

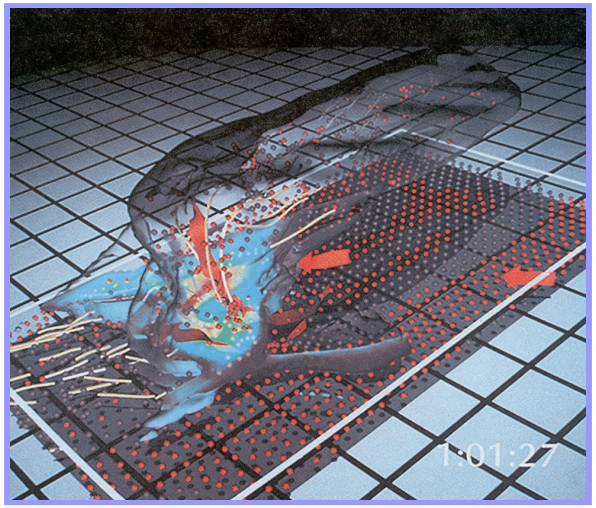
Aside

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

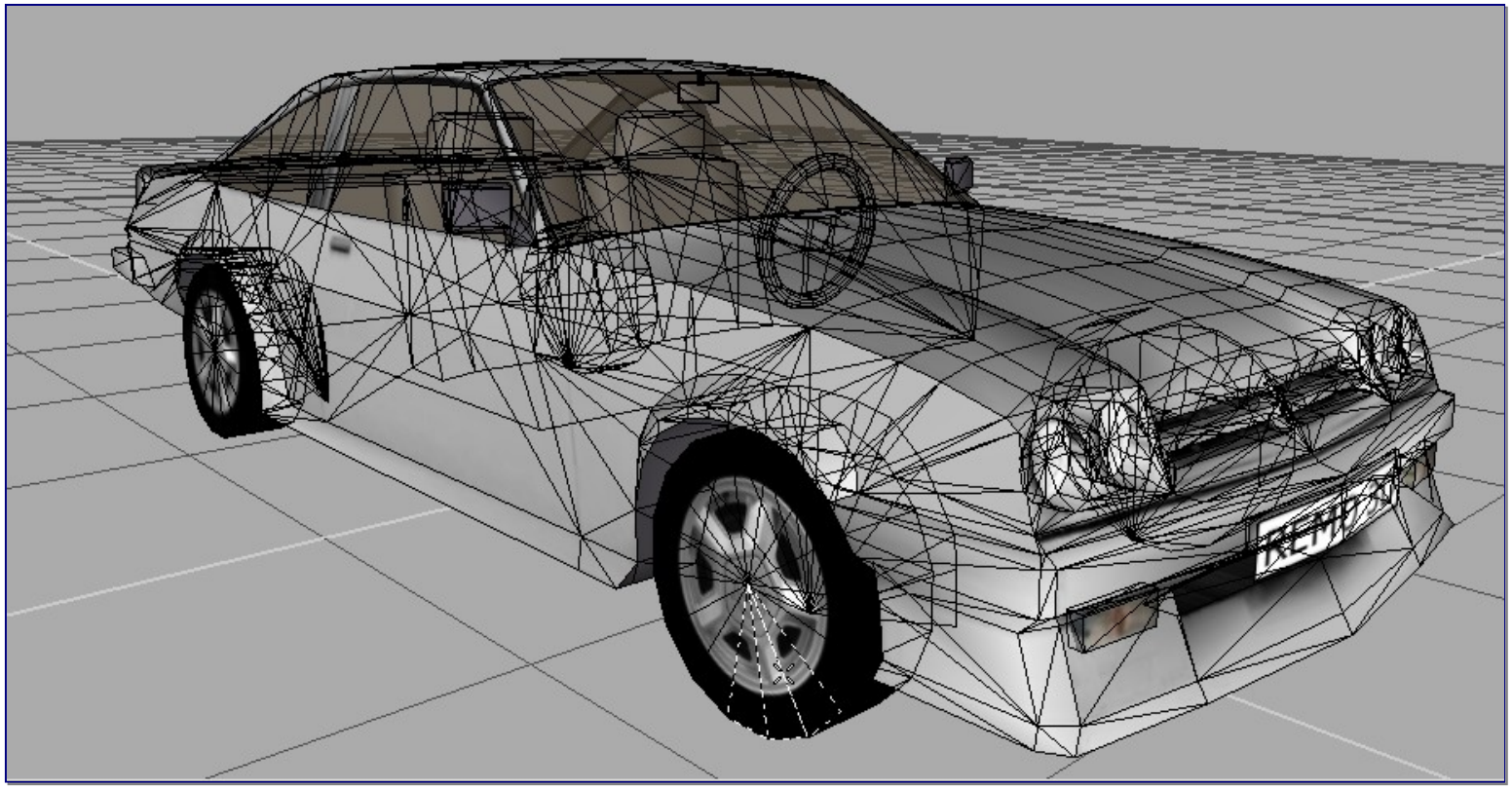


Inside a Thunderstorm
(Bob Wilhelmson, UIUC)

Boeing 777 Airplane

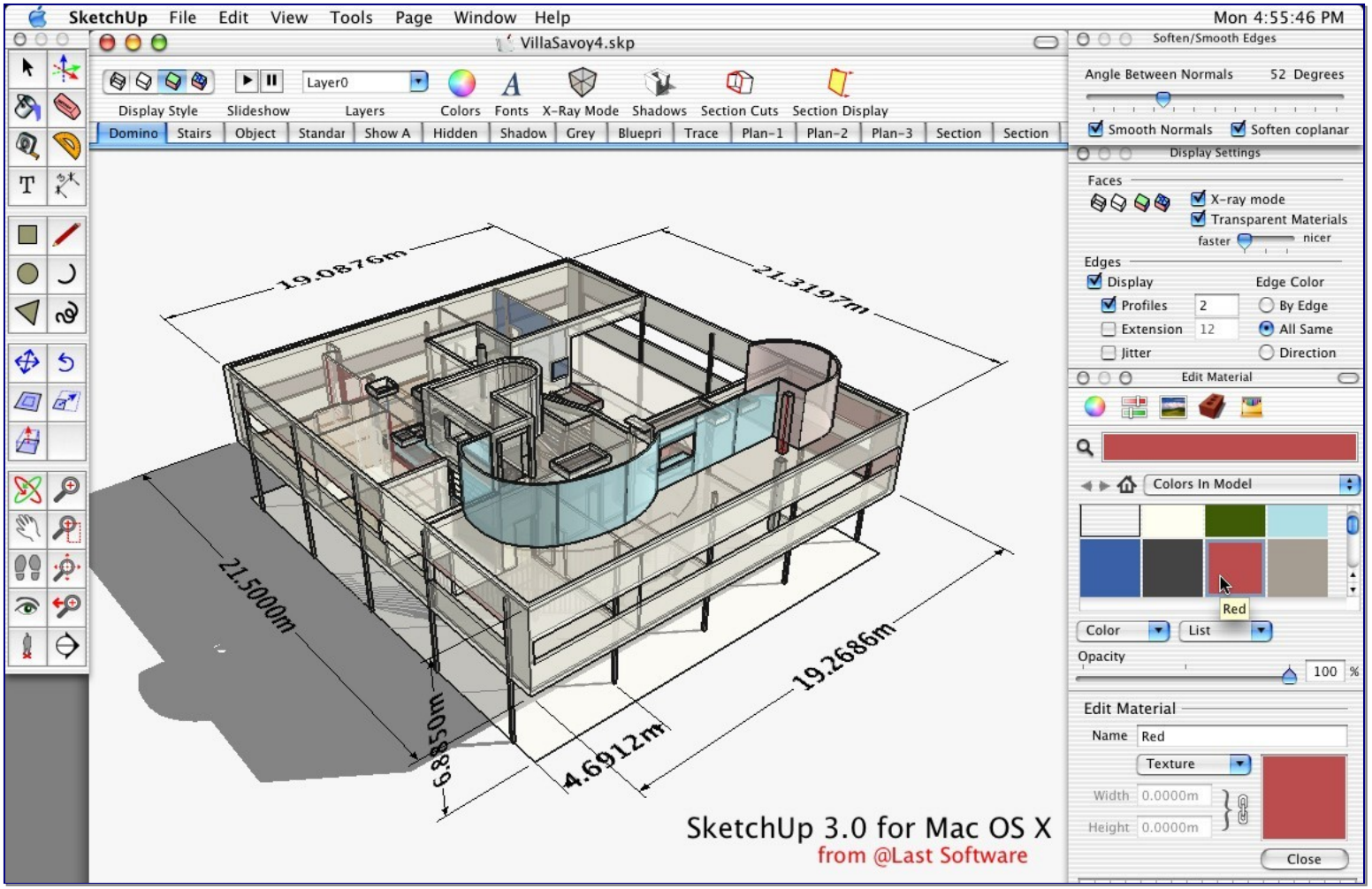
Step 1: Modeling

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



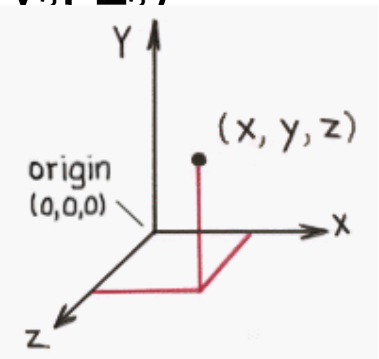
(Remo3D)

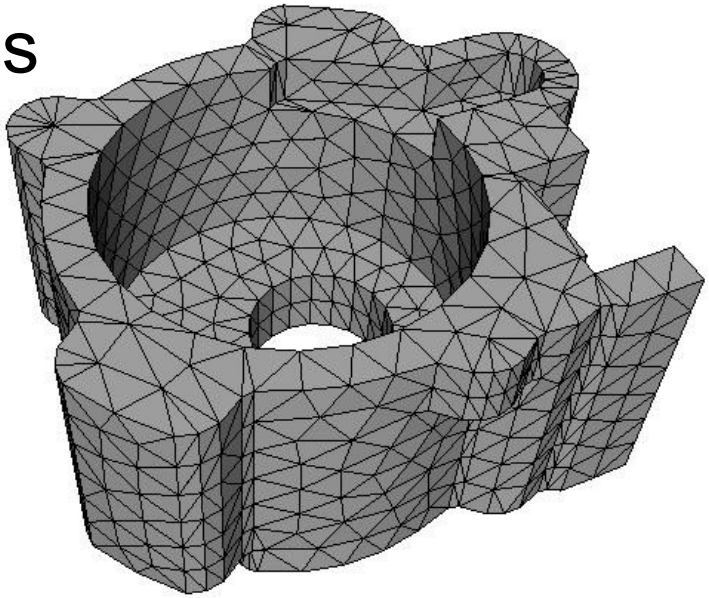
Modeling in SketchUp (demo)

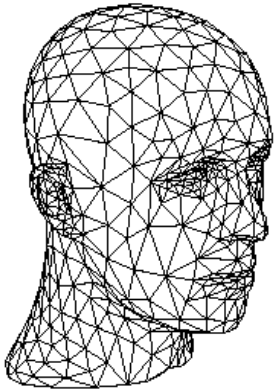


Example of “model”: wireframe

- 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)



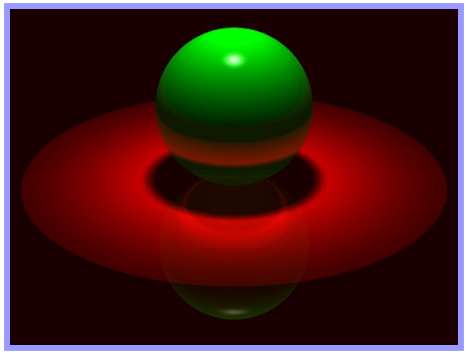




Usually would be augmented with info about texture, color etc.

Step 2: Rendering

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



Rendering (contd)

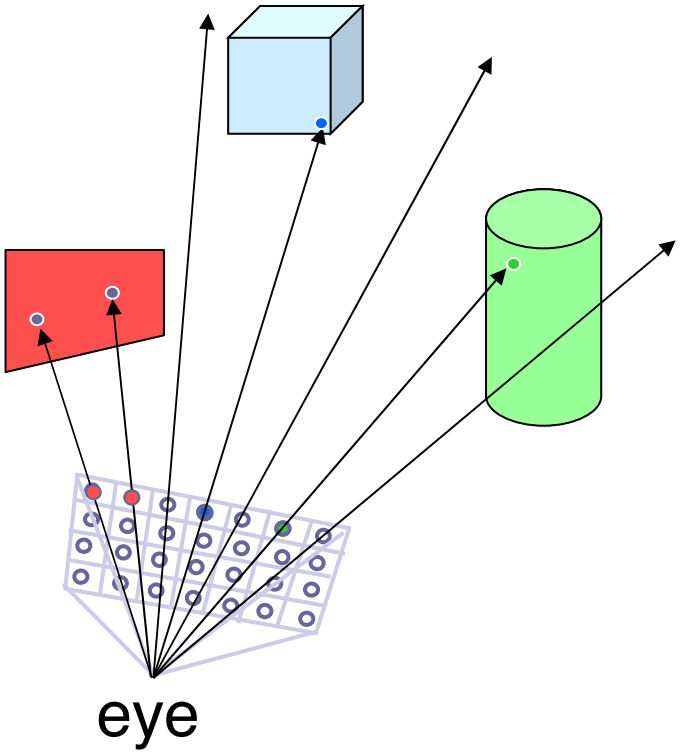
- Direct illumination
  - One bounce from light to eye
  - Implemented in graphics cards
  - OpenGL, DirectX, …
- Global illumination
  - Many bounces
  - Ray tracing

Direct Illumination
(Chi Zhang, CS 426, Fall99)

Ray Tracing
(Greg Larson)

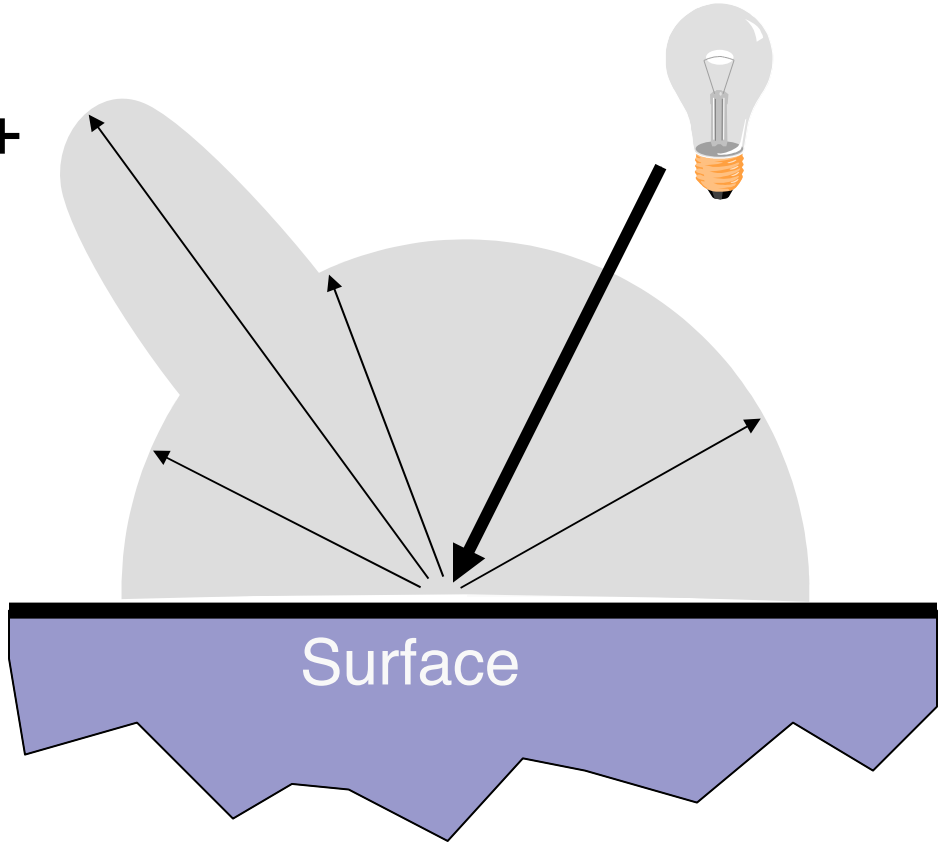
Ray Casting

- 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

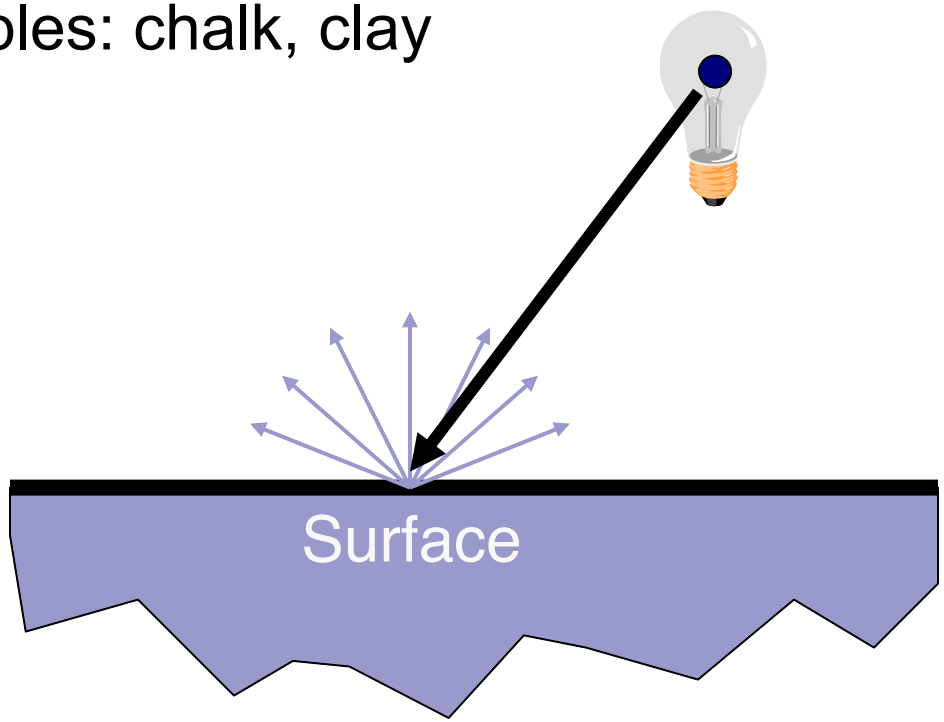
- Simple analytic model:
  - diffuse reflection +
  - specular reflection +
  - ambient lighting



Based on model proposed by Phong

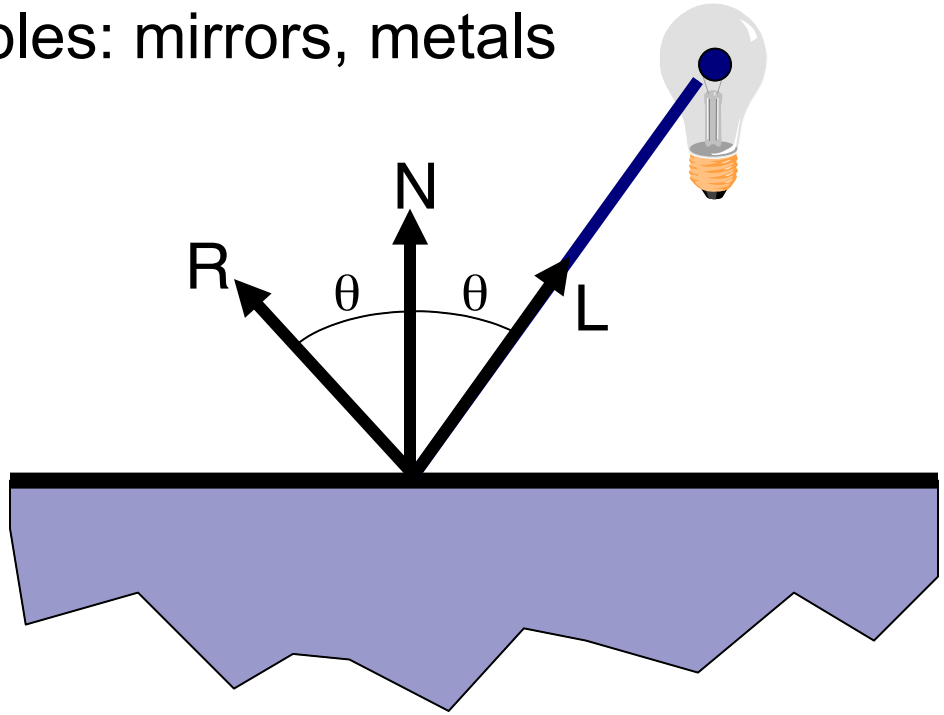
Diffuse Reflection

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



Specular Reflection

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



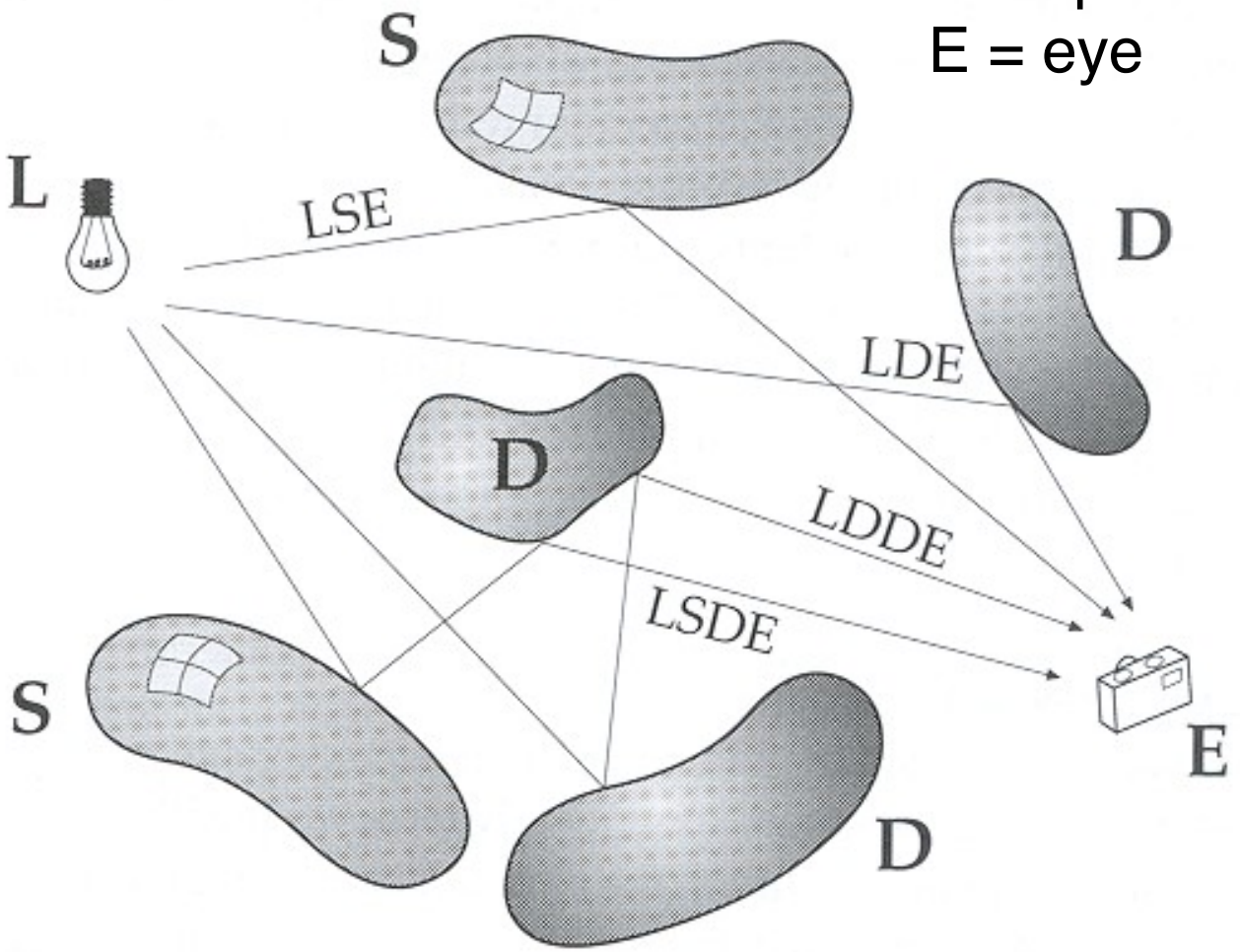
Ambient Lighting

- Represents reflection of all indirect illumination

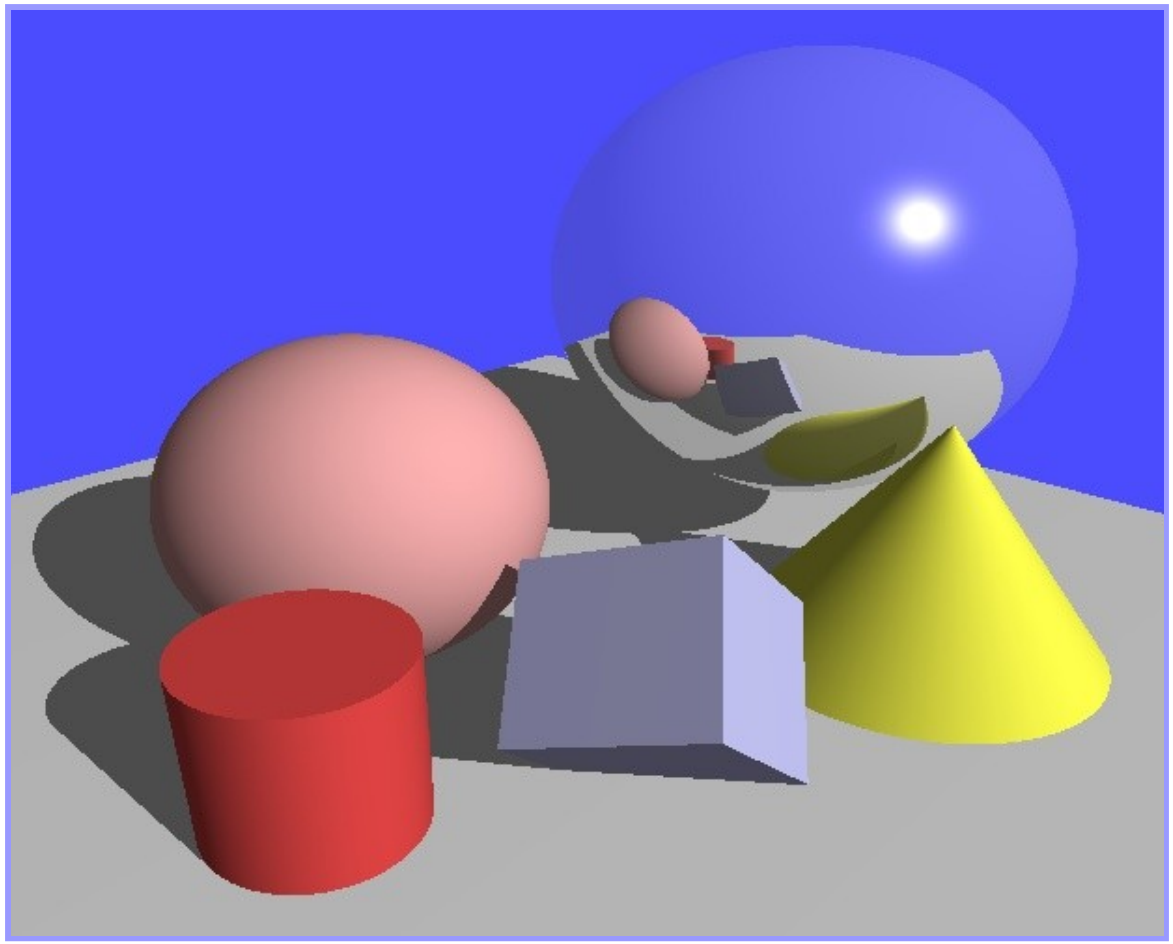
This is a total cheat (avoids complexity of global illumination)!

Path Types

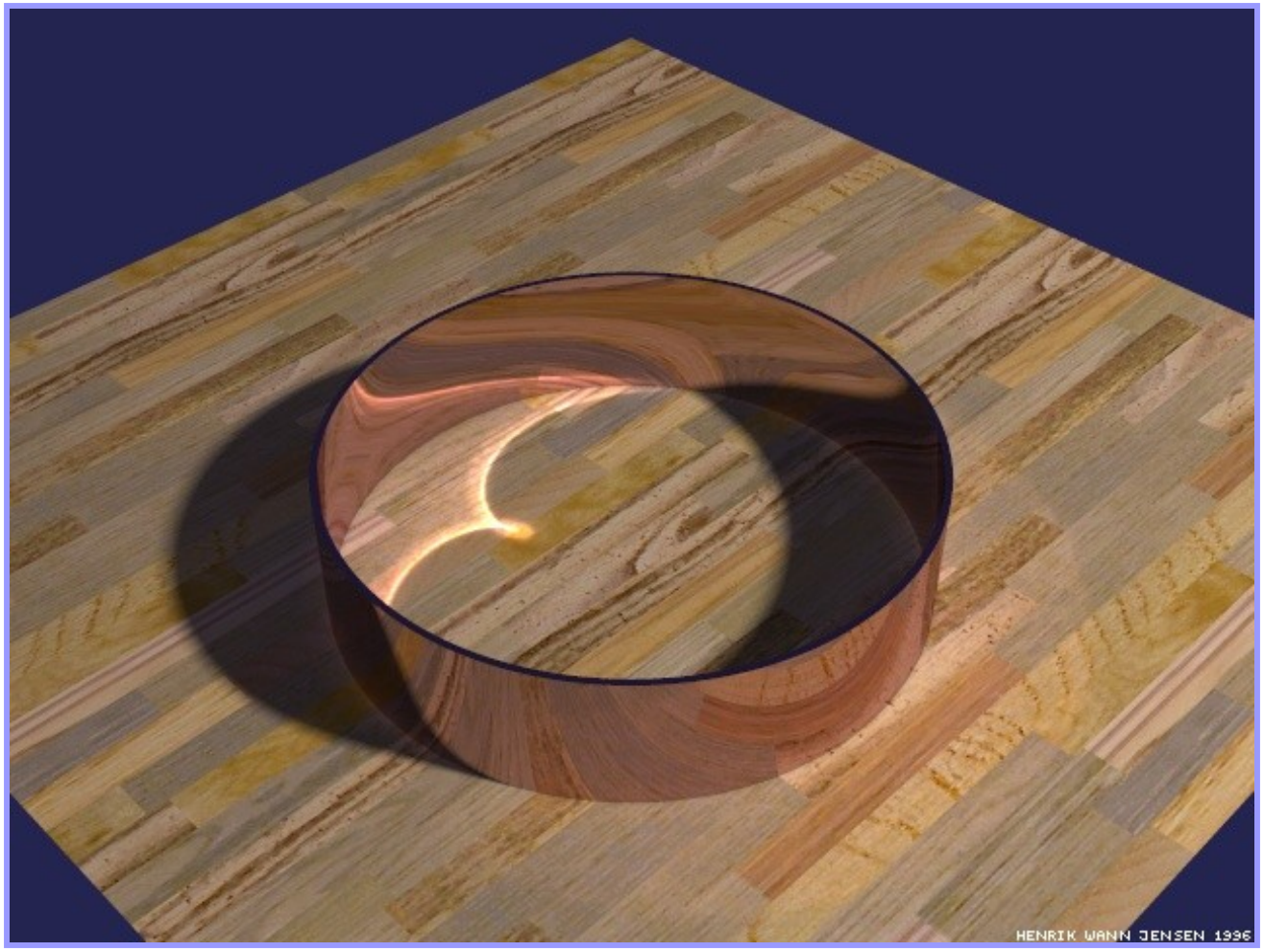
L = light
D = diffuse bounce
S = specular bounce
E = eye



Path Types?



Ray Tracing

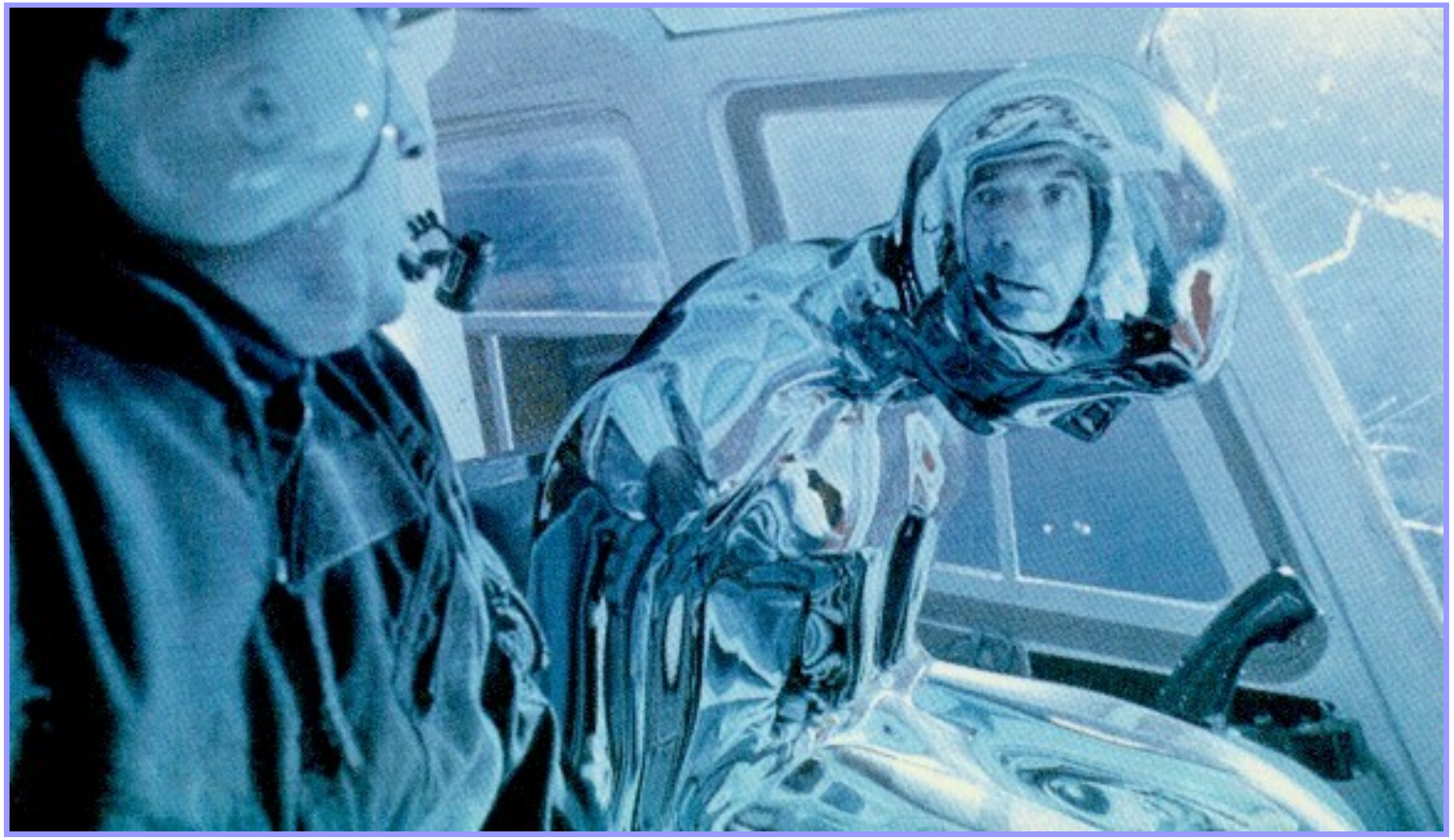


Henrik Wann Jensen

Ray Tracing

RenderPark

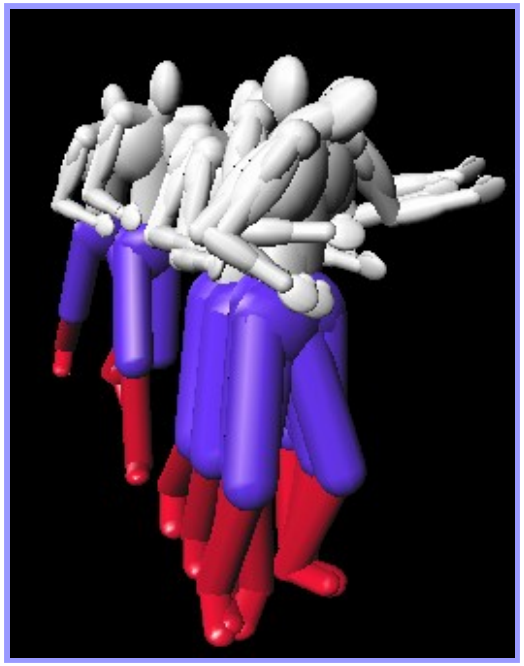
Ray Tracing



Terminator 2

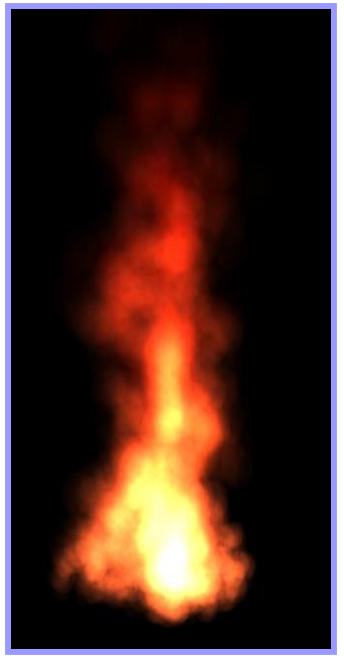
Step 3: Animation

- Keyframe animation
  - Articulated figures
- Simulation
  - Particle systems



Animation

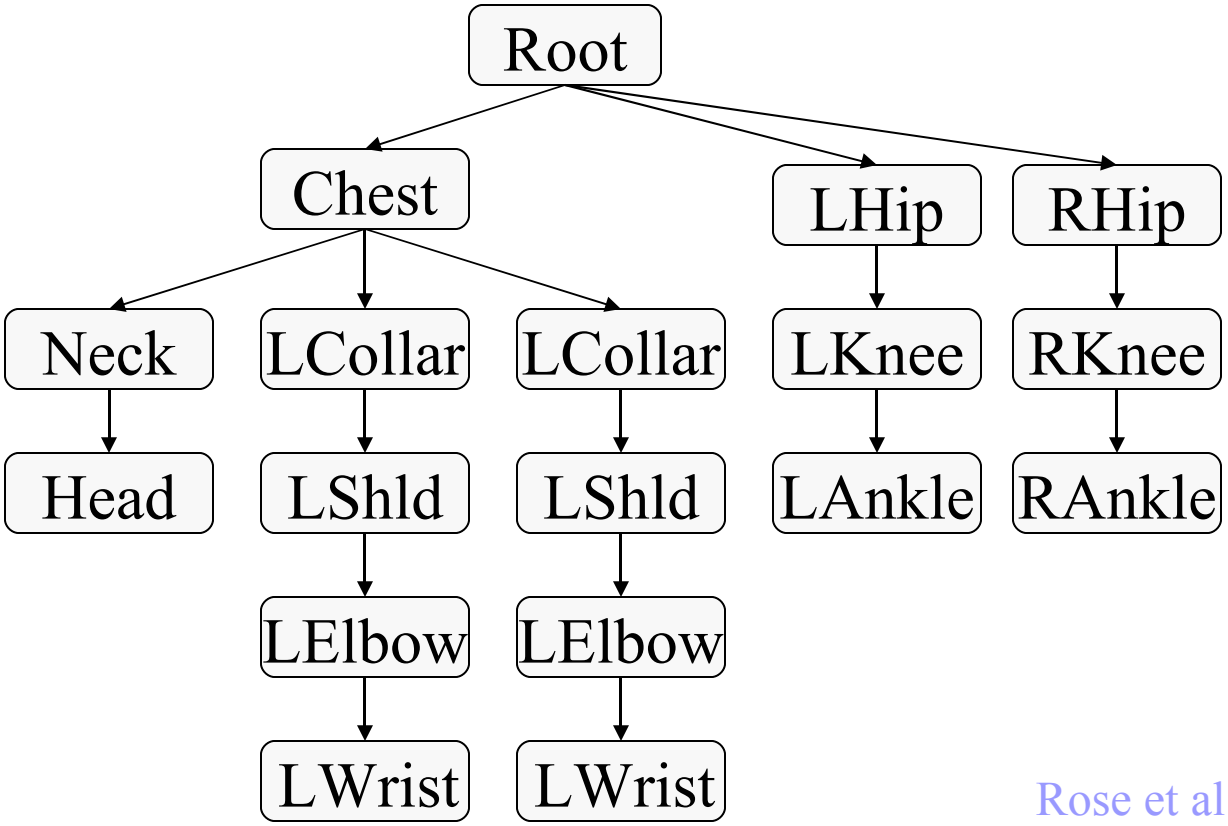
(Jon Beyer, CS426, Spring04)

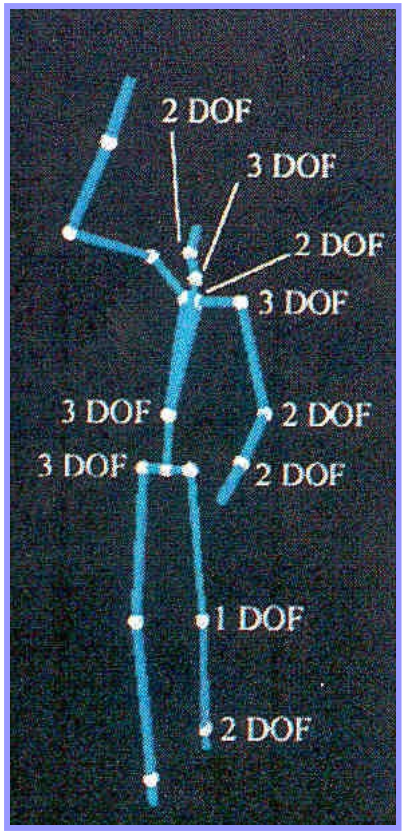


Simulation

Articulated Figures

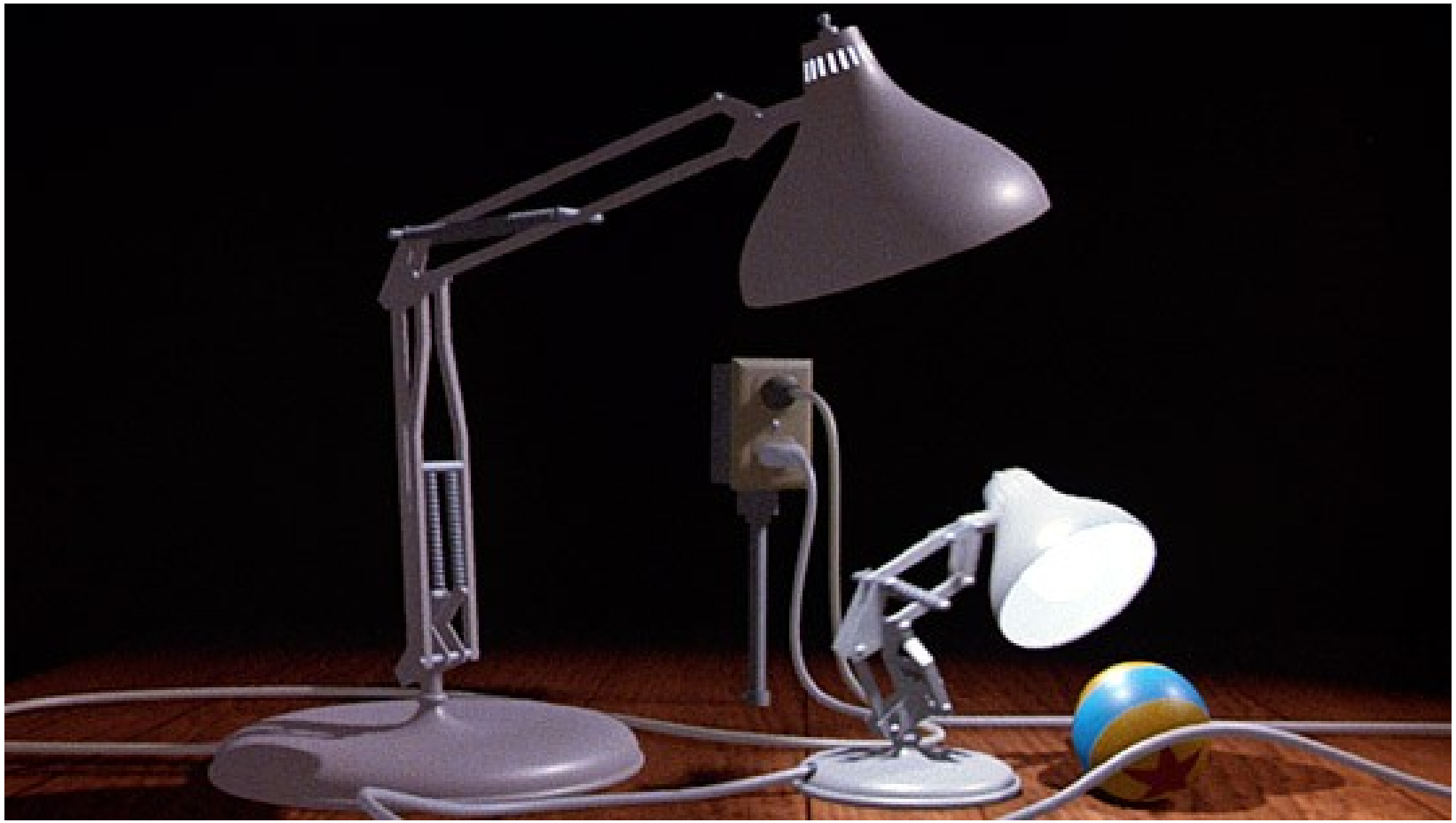
- Well-suited for humanoid characters





Rose et al. `96

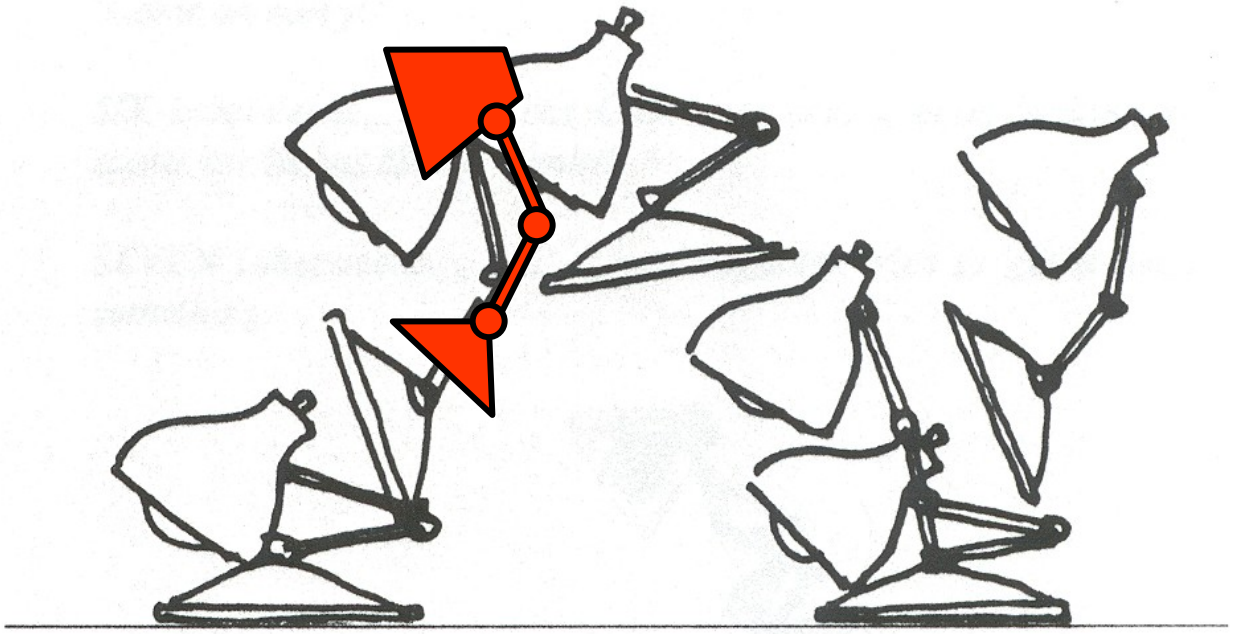
Keyframe Animation: Luxo Jr.



Pixar

Keyframe Animation

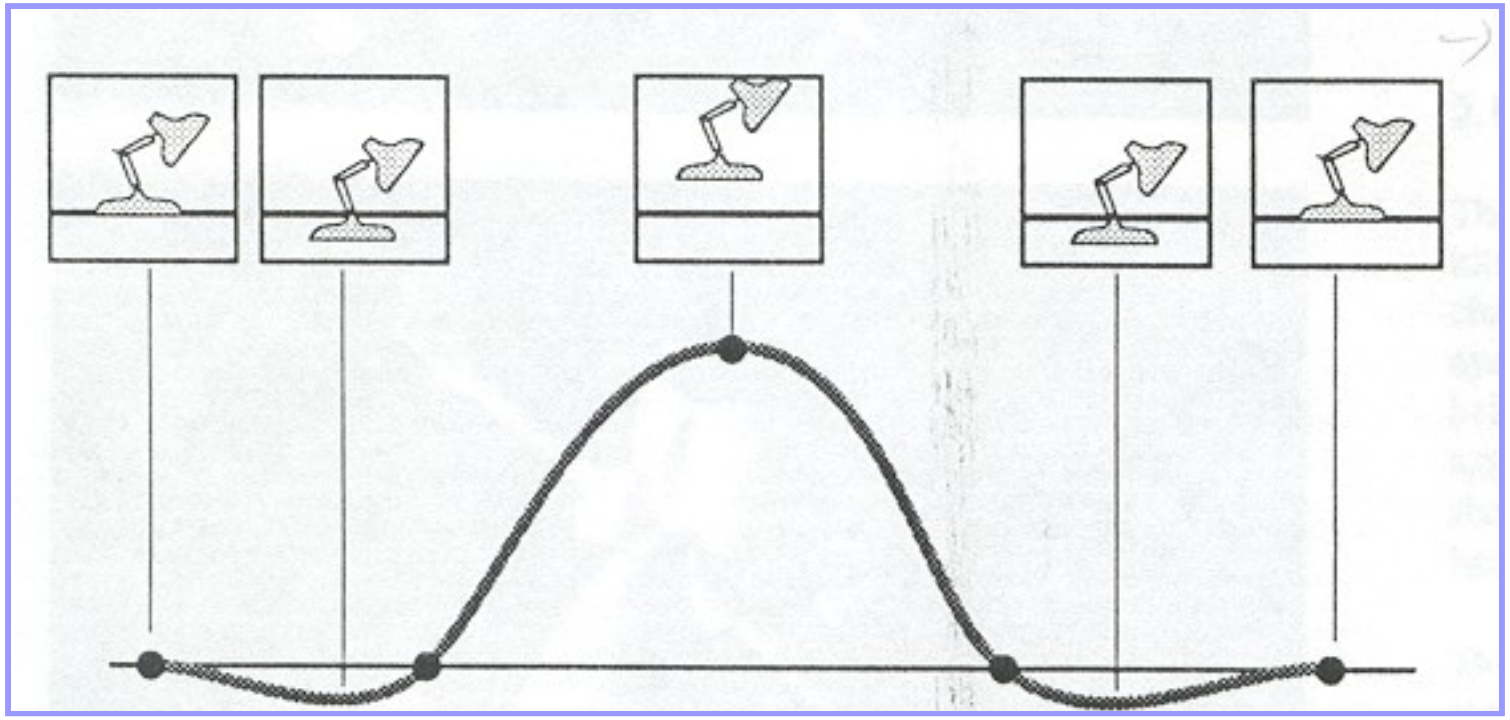
- Define character poses at specific times: “keyframes”
- “In between” poses found by interpolation



Lasseter `87

Keyframe Animation

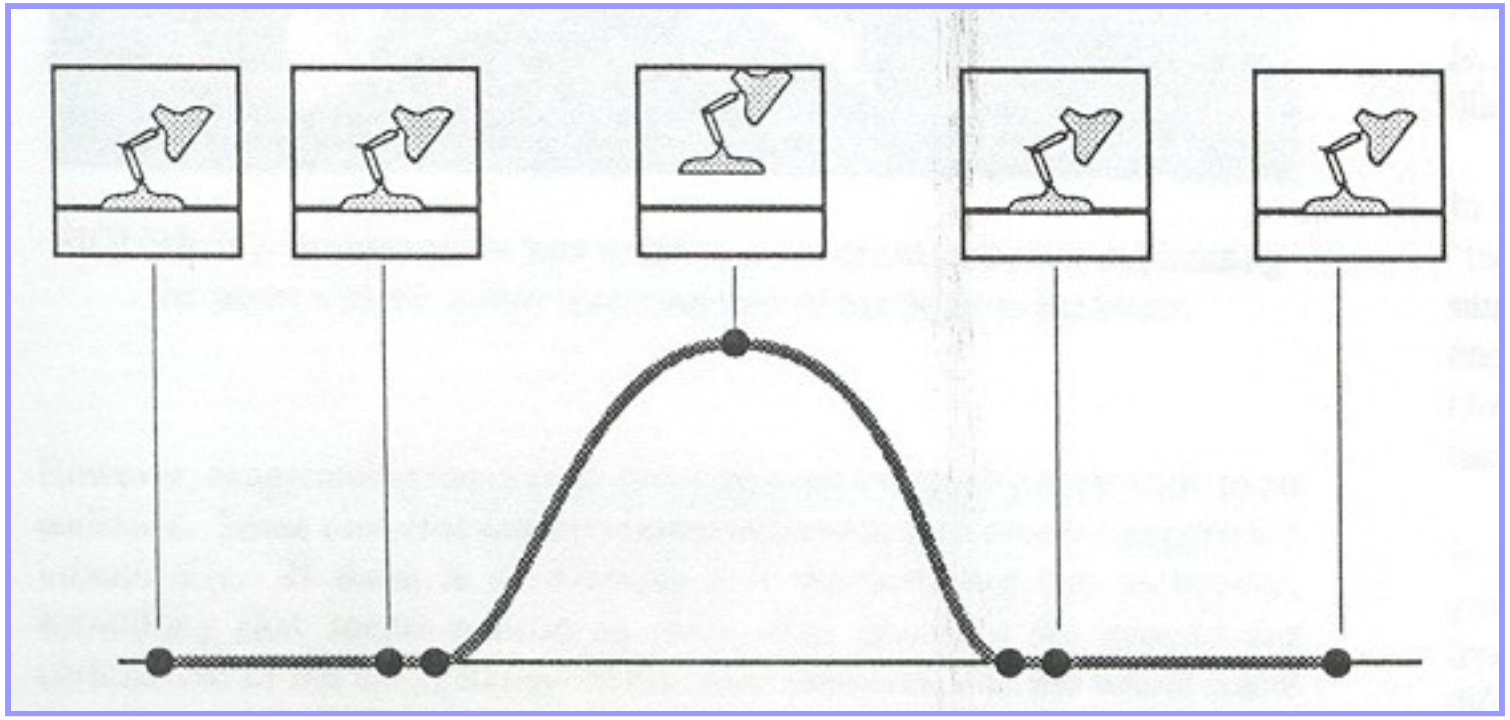
- Inbetweening: may not be plausible



Lasseter `87

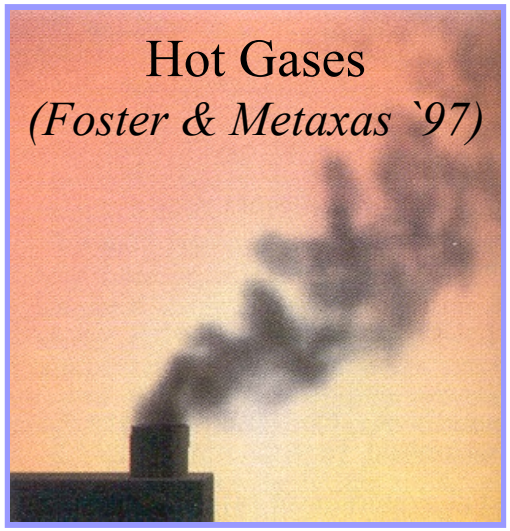
Keyframe Animation

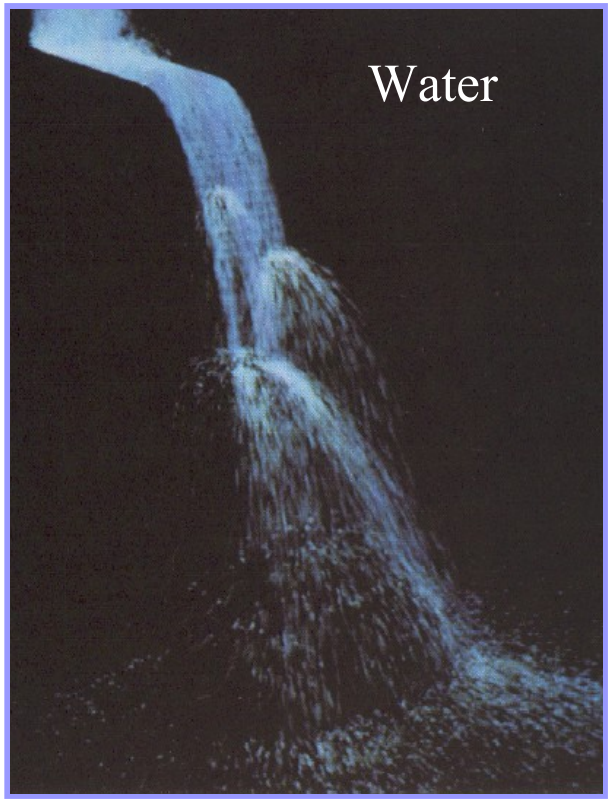
- Solution: add more keyframes

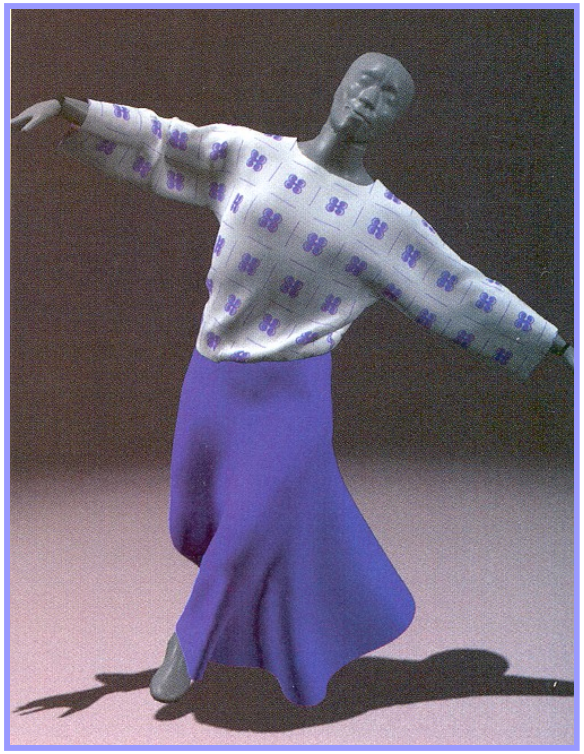


Lasseter `87

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



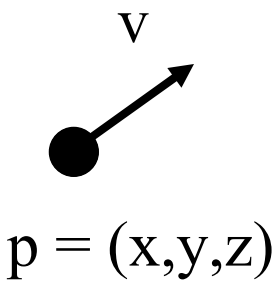




Cloth
(Baraff & Witkin `98)

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