CSE528 Computer Graphics: Theory, Algorithms, and Applications

Hong Qin
Department of Computer Science
Stony Brook University (SUNY at Stony Brook)
Stony Brook, New York 11794-2424
Tel: (631)632-8450; Fax: (631)632-8334
qin@cs.stonybrook.edu
http://www.cs.stonybrook.edu/~qin

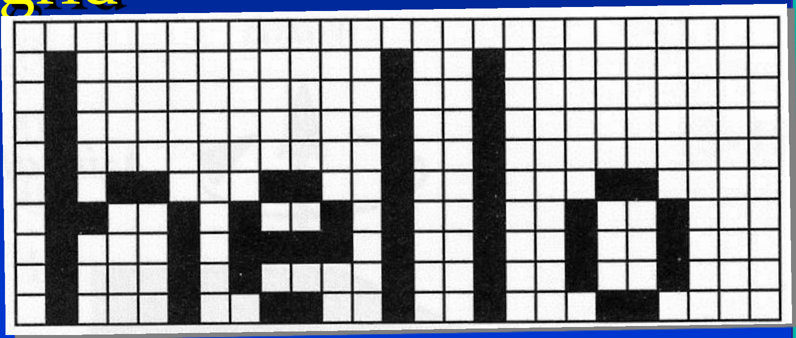
Graphics Hardware and Display Devices

Graphics Hardware

- Many graphics algorithms can be implemented efficiently and inexpensively in hardware
- Facilitates interactive graphics applications, including certain domains of scientific visualization
- Topics today:
  - Raster devices
  - Video controllers & raster-scan display processors
  - Important rasterization and rendering algorithms
  - Pixels and images

Raster Devices

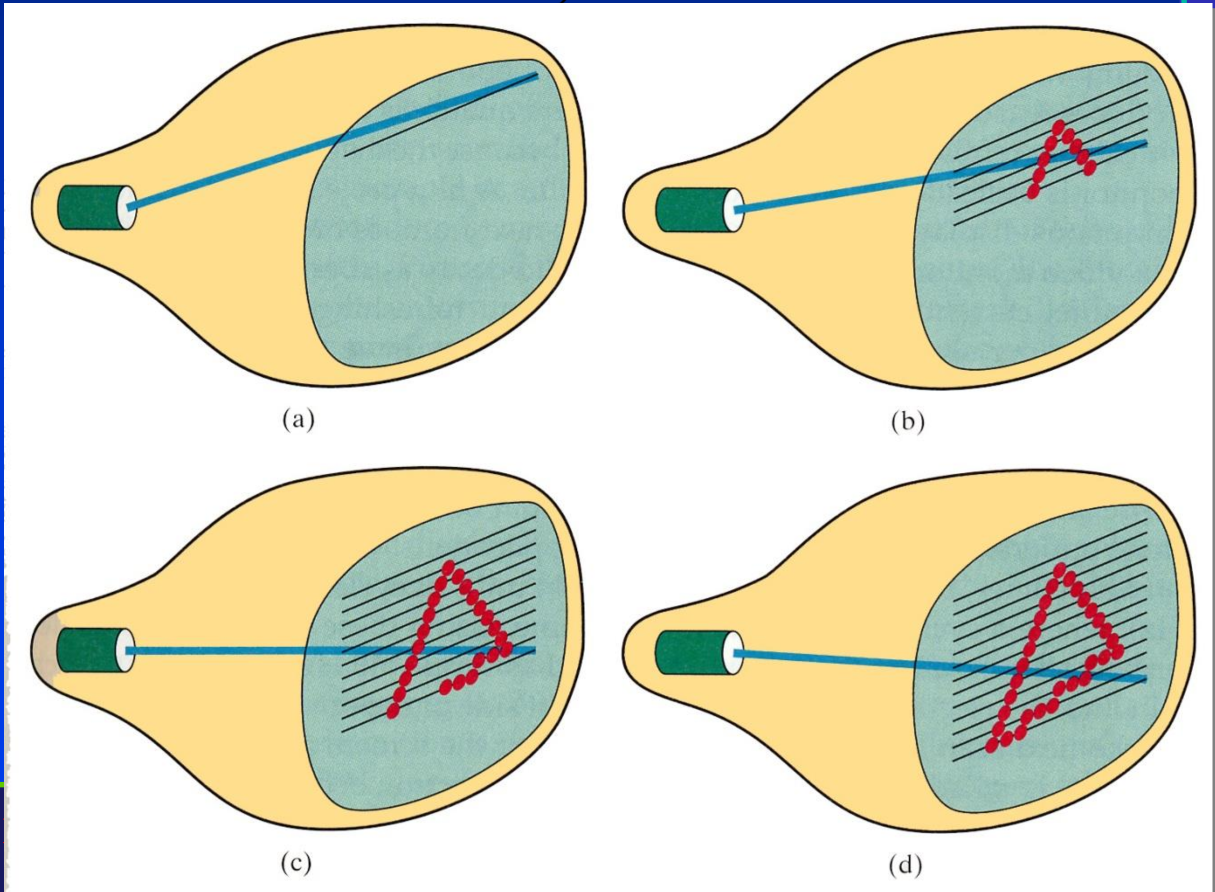
- Computer monitors (CRT, LCD, etc.), TVs
- These are *raster devices* because they display images on a *raster*, which is a regular n-D grid
- Each point on the grid is called a *pixel*, which stands for ________________



- Raster dimension given in pixels: 25 x 10 in this example
- In a monochrome display, each pixel is black or white
- In a color display, each pixel has an RGB triple

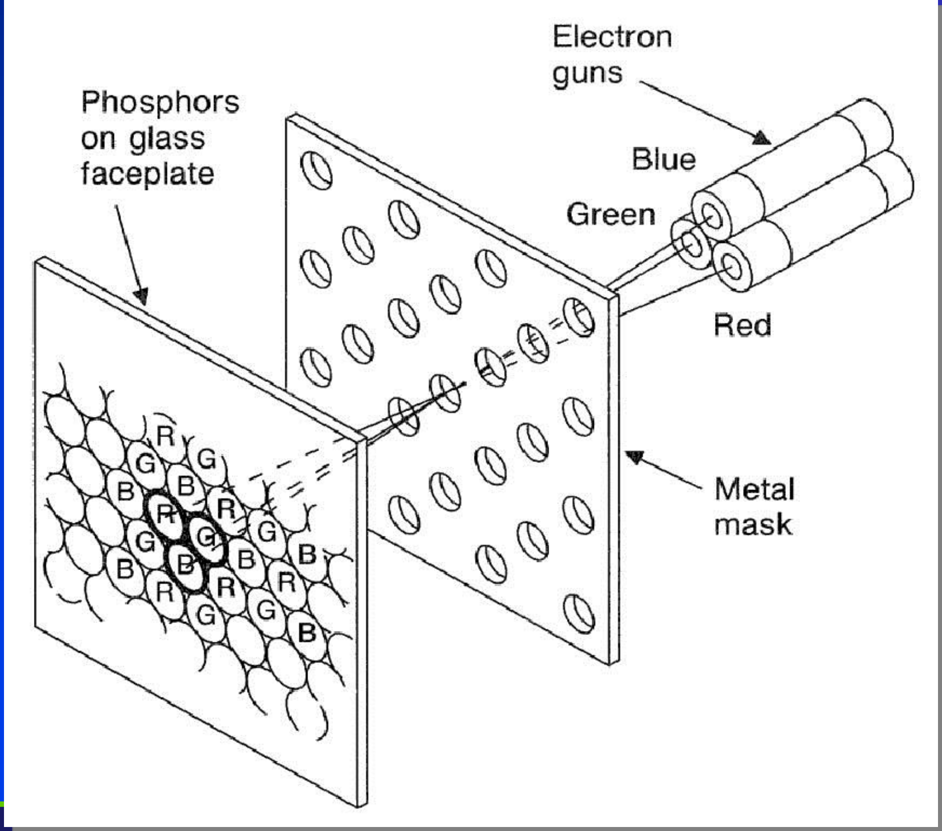
Raster Devices

- Also called *raster-scan* displays or systems
- Pixels are drawn in a strict order, called *raster-scan order*
- Cathode ray tube (CRT) shown here
- Monochrome display



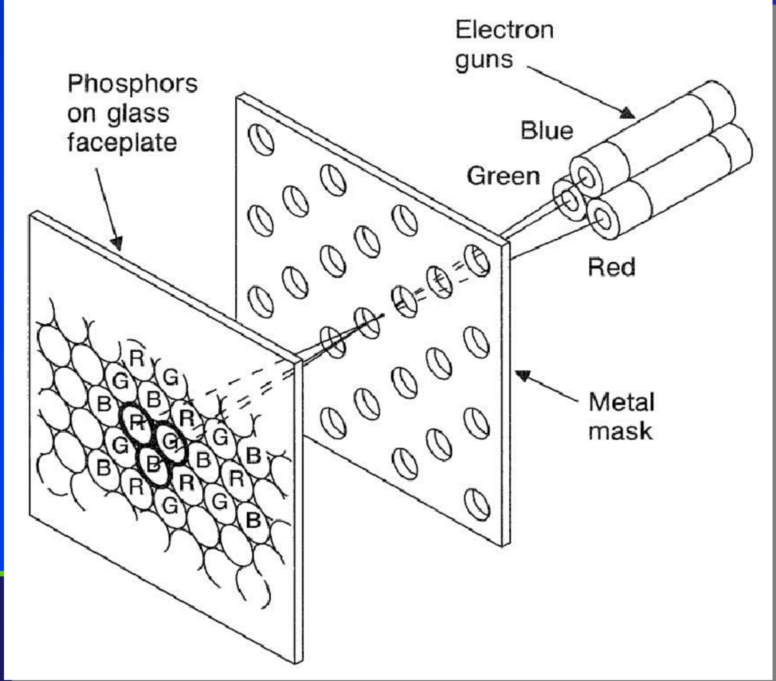
Color Display Technology – CRT

- Cathode ray tube - used in TVs and computer monitors (the large, clunky type)
- A color CRT has three electron guns: one for red, one for green, and one for blue
- The beams scan screen in horizontal scanlines
- Metal mask steers beams



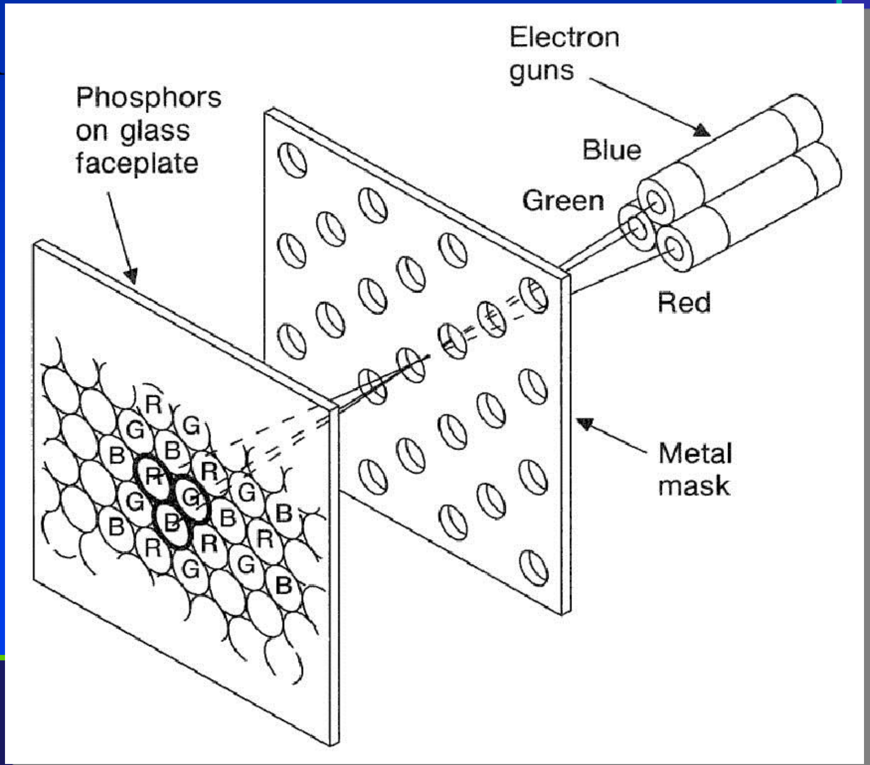
Color Display Technology – CRT

- Each screen pixel consists of a *phosphor* triple: one glowing red, one green, and one blue
- A phosphor is a circular spot of *phosphorescent* material that glows when electrons strike it
- Red phosphors glow red
- RGB triad together form a single pixel on screen



Color Display Technology – CRT

- Glowing phosphor triples blend together to form color encoded in RGB triple
- Amount of energy that electron guns deliver to each phosphor depends on RGB value of image pixel displayed there
- RGB values between 0 and 1 are mapped to voltages for the guns



Color Display Technology – CRT

- True or false: A color image in a CRT is generated by blending the three colored beams of light that are fired from the back of the monitor and blended on the front surface of the screen.

Color Display Technology – CRT

- The phosphors glow only for about 10-60 microseconds
- Image refreshed 30-60 times per second
- This rate is called the *refresh rate* and is given in Hz
- So if we redraw the image once every 1/60th of a second, but the image lasts only a few millionths of a second, what about the gap?
- 1/60th second is approximately 16667 microseconds
- (16667 - 10) microseconds = "long" delay between refreshes
- So why is there no visible flicker?

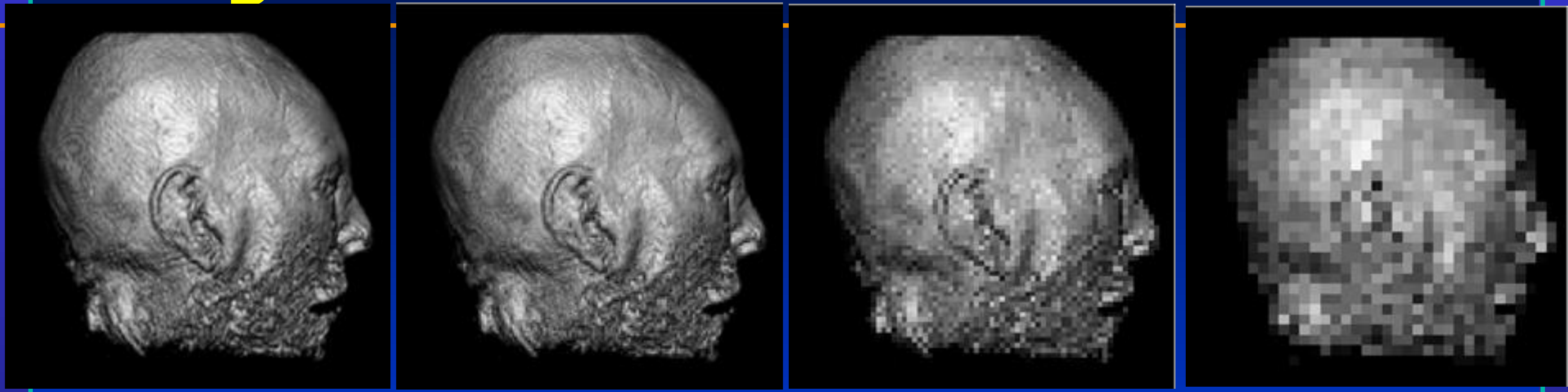
Raster Devices: Display Resolution

- The raster is not 100% perfect – points of light corresponding to pixels can overlap slightly
- Same is true of raster printing technologies, like laser and injket printers
- Pixels are more like circles than squares in reality
- Raster devices are also limited by resolution
  - Computer monitors 1600 x 1200 and higher
  - Laser printers 300 dpi, 600 dpi, 1200 dpi and higher
  - TV resolution? HDTV?

Raster Devices: Color Depth

- Horizontal lines of pixels are called *scanlines*
- TV: 640 HDTV: 720 or 1080
- Monochrome monitor has 1 bits per pixel (bpp)
- Grayscale has 8 bpp (usually)
- Color monitors most often have 24 bpp: 8 bits each for red, green and blue color channels
- How many different levels of gray can we represent with 8 bits per pixel?
- How many different colors can 24-bit color represent?

Image Resolution



res = 300^2 pixels   res = 150^2 pixels   res = 75^2 pixels   res = 37^2 pixels

- Image resolution very important in graphics rendering, why?
- When might we want to use a low resolution image?

How Many Bits Do We Need?

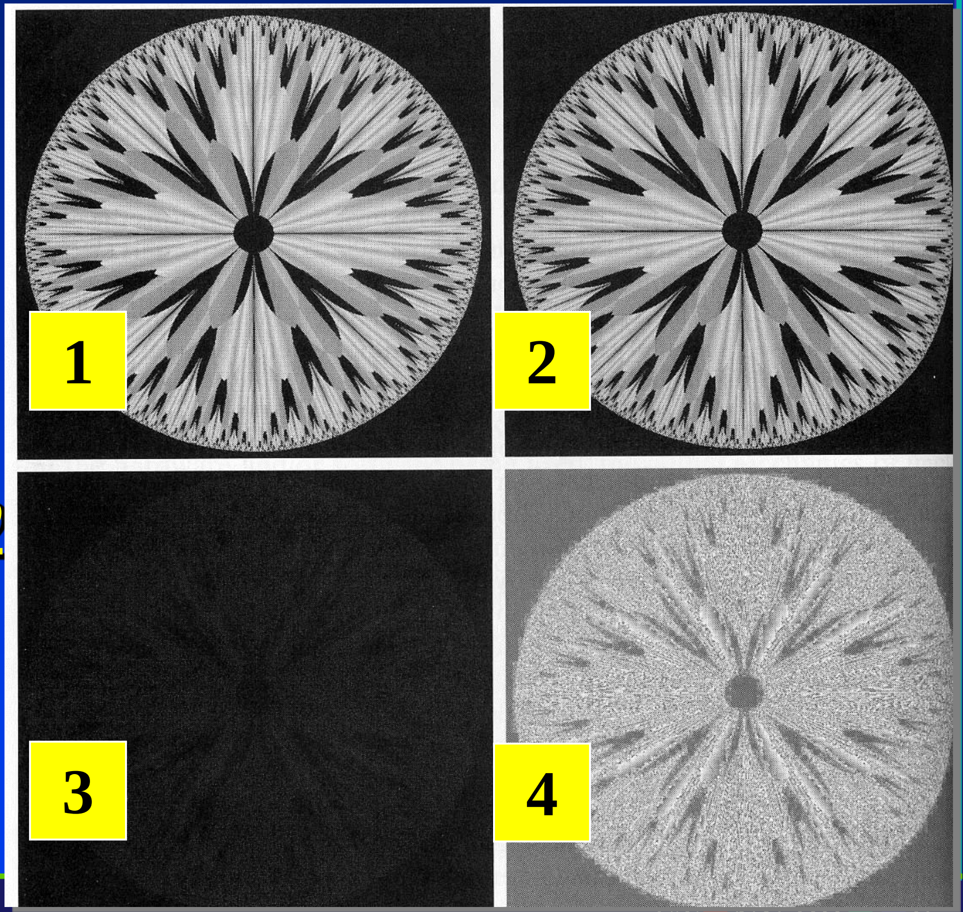
- Number of bits per pixel often called *bit depth*
- How many bits should we use in practice?

#1: 8-bit original image

#2: lower 4 bits dropped

#3: (image #1 - image #2

#4: image #3 enhanced

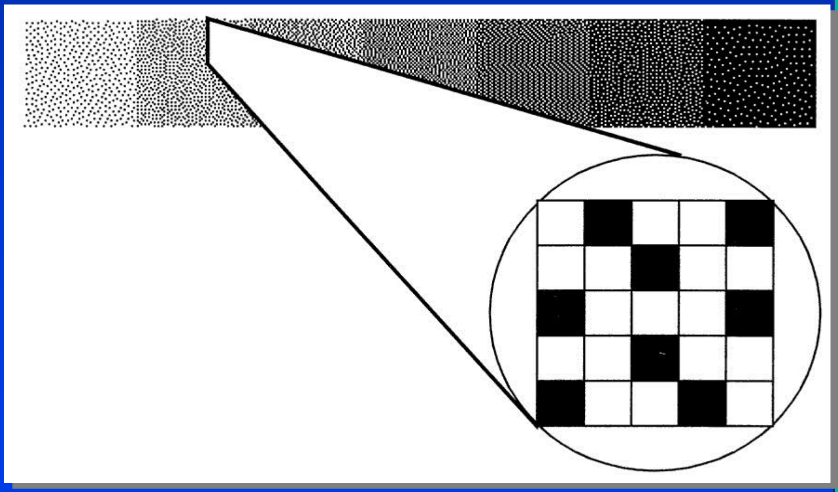


Bit Depth

- Suppose we want to display 256 gray levels, but we have only 1-bit color.
- What colors *can* we display?
- How do we accommodate grayscale images?
- How do we accommodate color images?
- Suppose we want to display 16.7 million colors on our color monitor, but we have only 8-bit color. What can we do?

Dithering

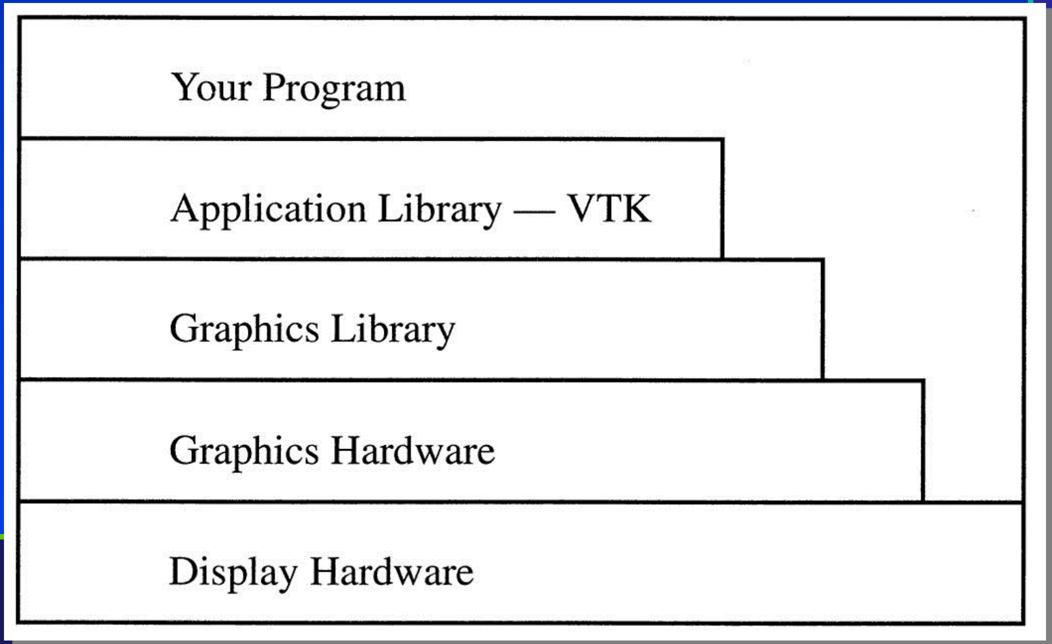
- *Dithering* is a way to use a mixture of colors to trick eye into seeing colors that cannot be actually represented by display device
- We can approximate gray by using a combination of black and white:
- The relative densities of black and white determine the "gray" value



- Also called *halftoning* (vb. *to halftone*)

Interfacing to Hardware

- A lot goes on “under the hood” in the graphics and display hardware
- Graphics hardware: converts geometry into pixels
- Display hardware: displays pixels
- Simplified hierarchy



Graphics Pipelines

- Graphics processes generally execute sequentially
- Typical ‘pipeline’ model
- There are two ‘graphics’ pipelines
  - The Geometry or 3D pipeline
  - The Imaging or 2D pipeline

Rasterization

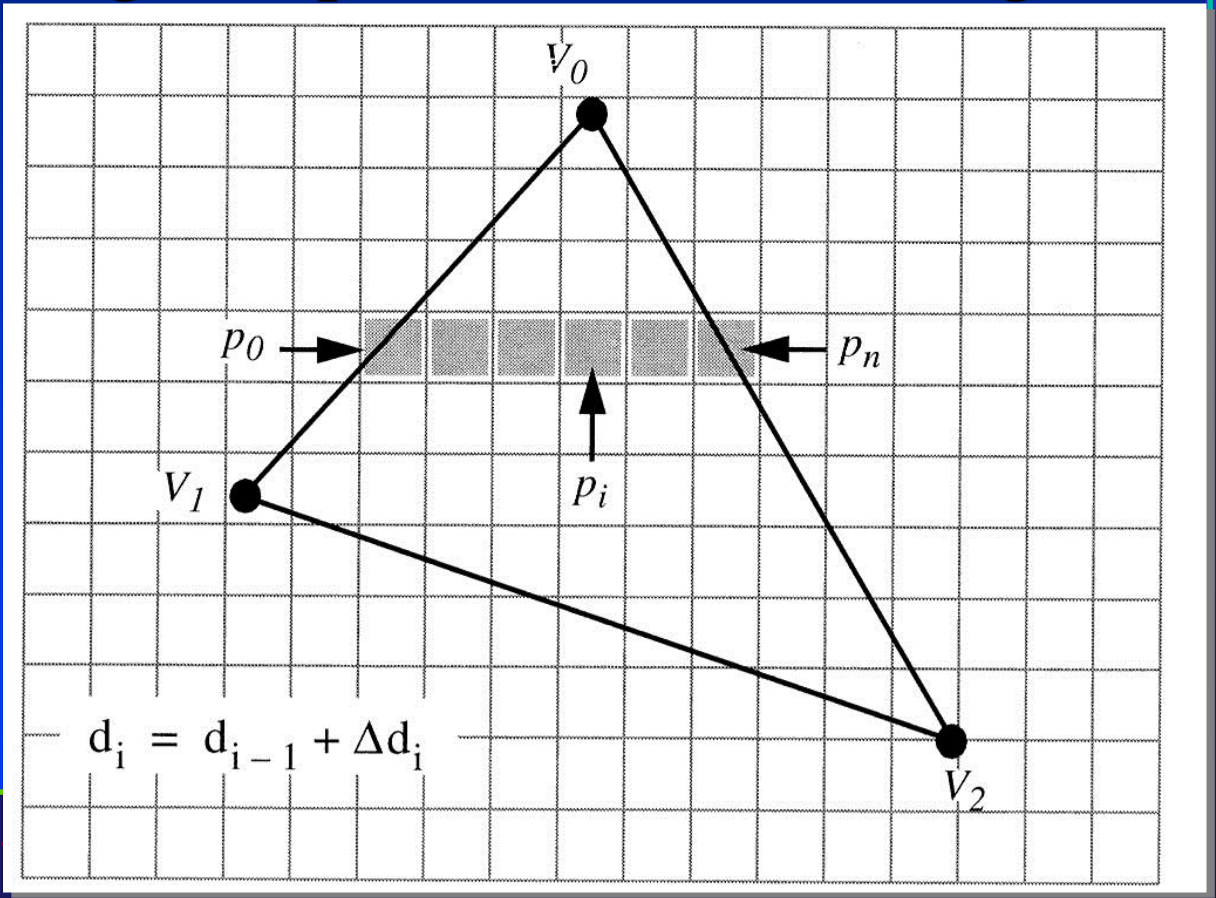
- We looked at raster devices and some different kinds of geometric objects we might wish to draw on the screen
- Process of converting geometry into pixels is called *rasterization* or *scan-conversion*
- Each triangle in our model is transformed (rotated, etc.) and projected by the transformation and projection matrices
- Next we *clip* each triangle to the image plane
- Each triangle is entirely inside, entirely outside, partially visible w.r.t the image plane

Rasterization

- We will take an *object-order approach*
- Question: In contrast, ray-tracing is *what*-order?
- We process each triangle one by one
- After we transform and clip it, we rasterize it – we figure how what pixels on screen we need to update to draw the triangle on screen

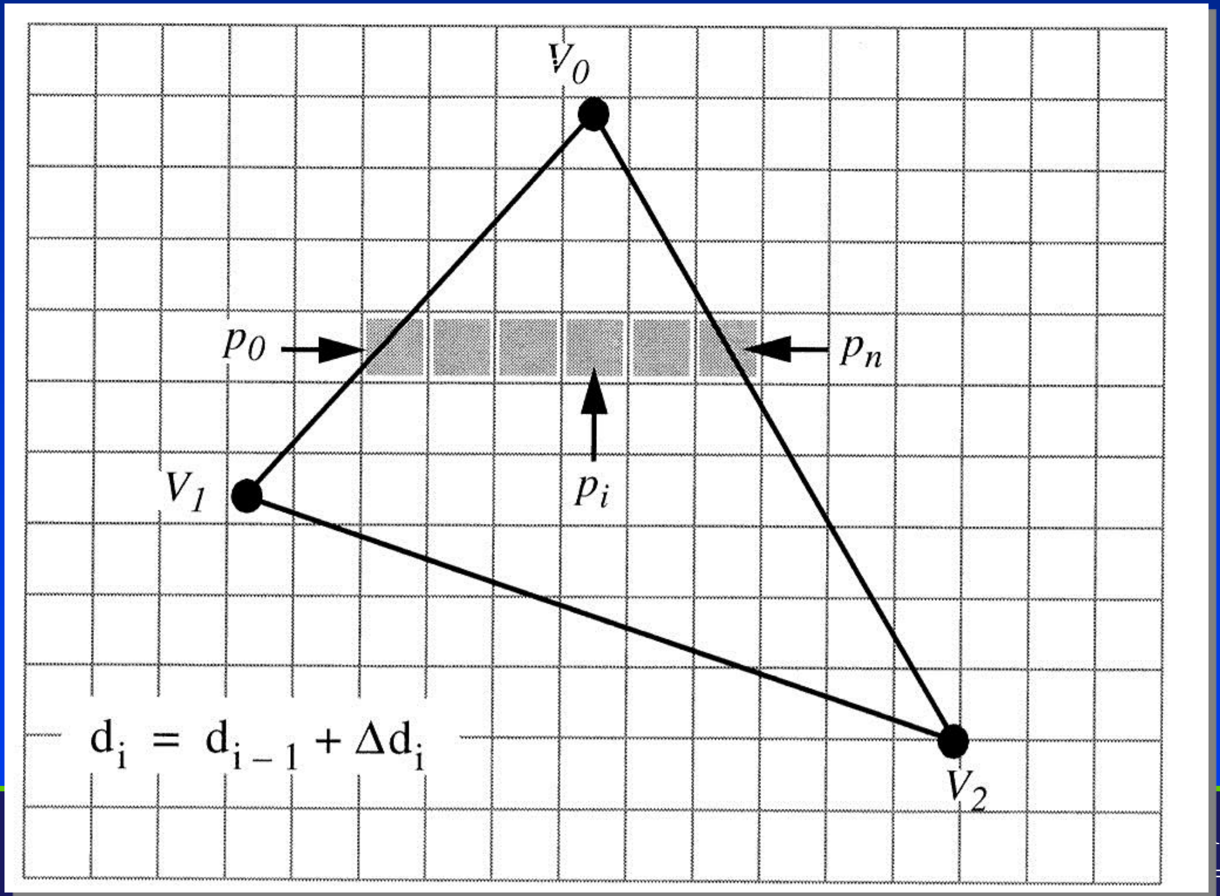
Rasterization

- We will process the triangle in *scan-line order*: left-to-right starting at top left corner, moving right and down



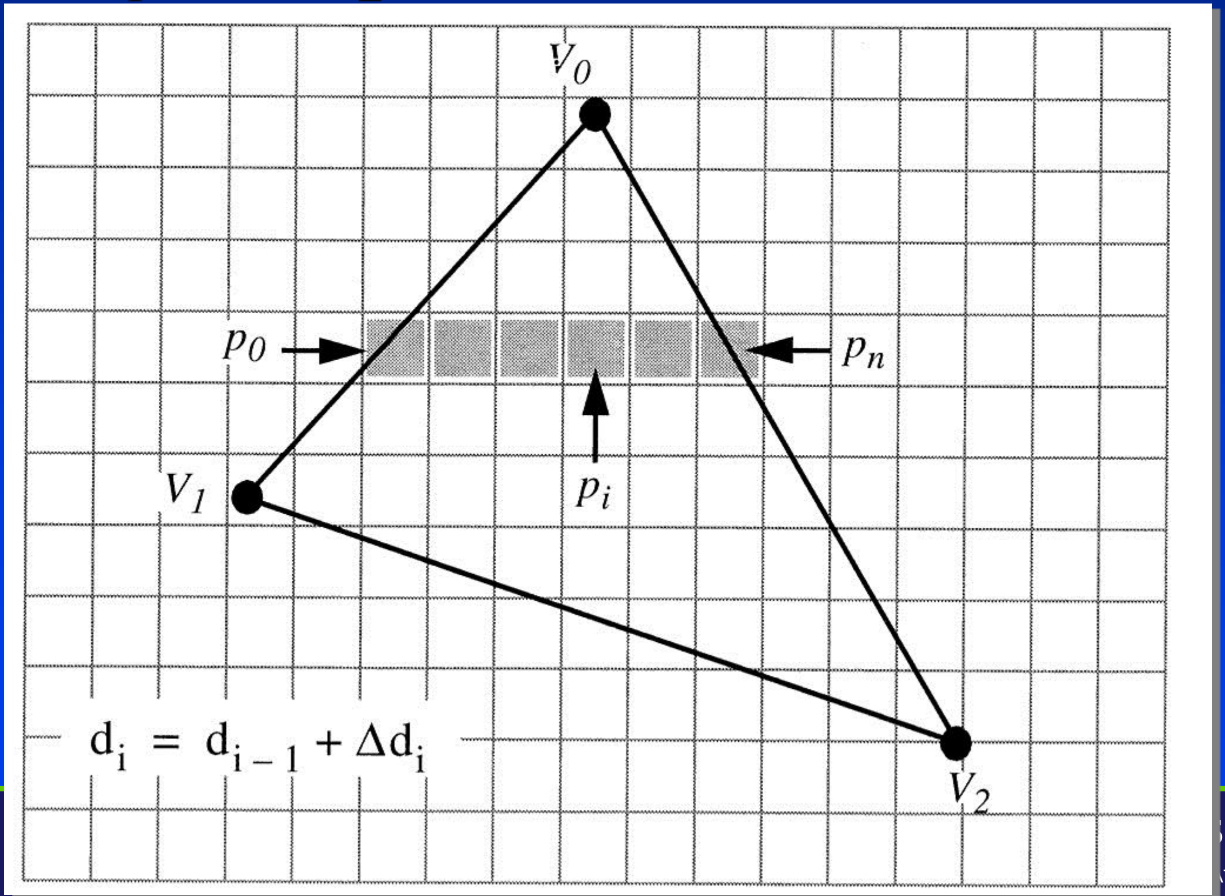
Rasterization

- We sort the vertices by their *y* values and find the vertex with the maximal *y* value; call this vertex v_0



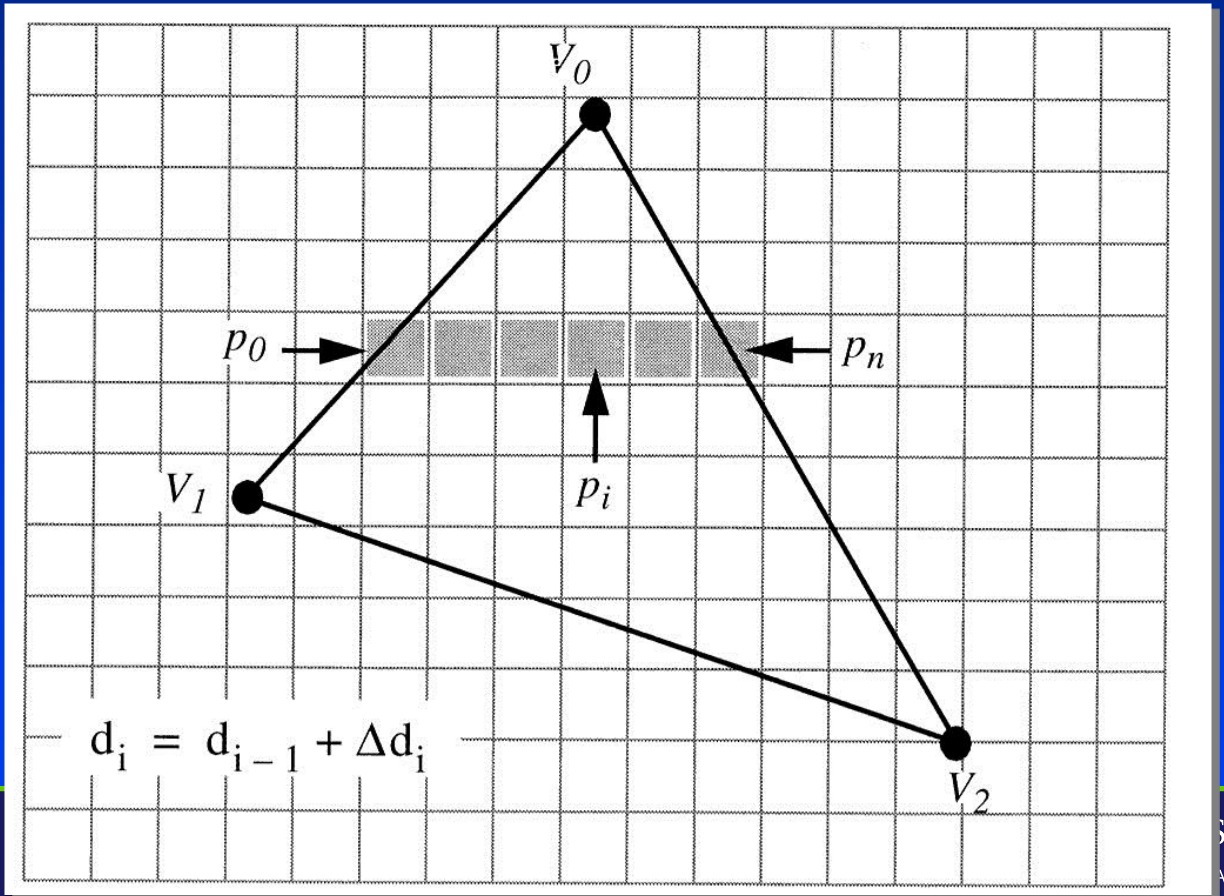
Rasterization

- This sorting allows us to identify the other two vertices, v_1 and v_2



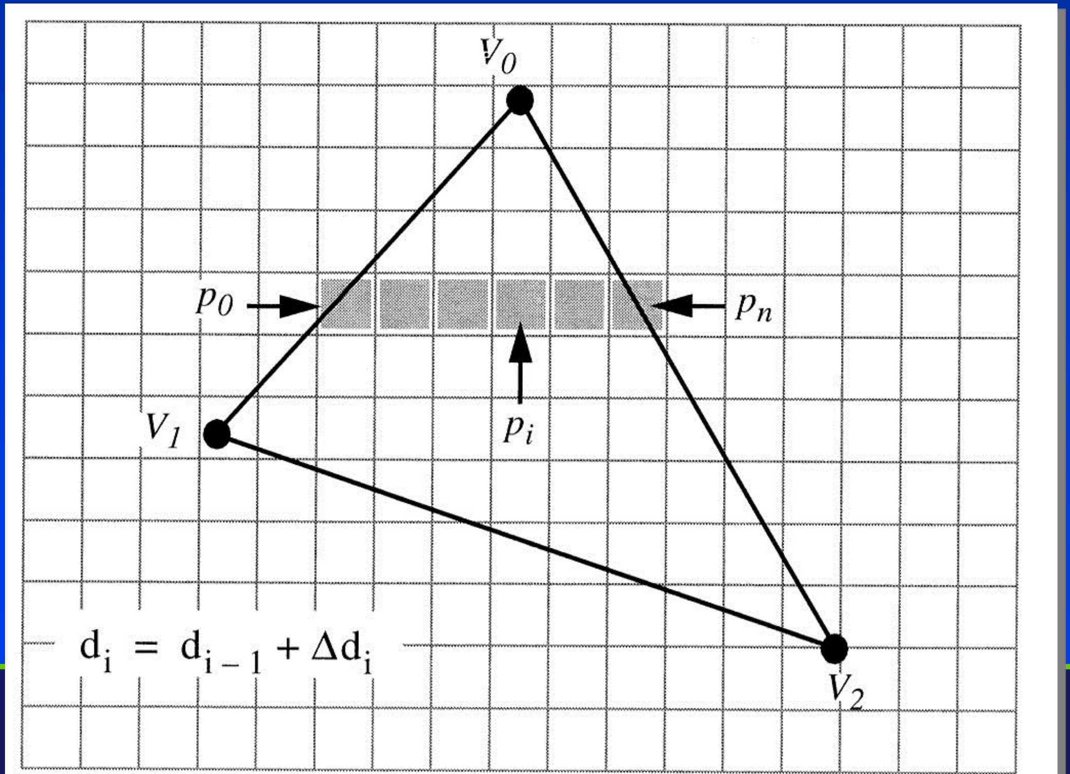
Rasterization

- Using the slopes of the edges we can compute each row of pixels to process, called a *span* of pixels



Rasterization

- Across each polygon we interpolate various data values d_i for each pixel
- Example: RGB to assign colors to vertices

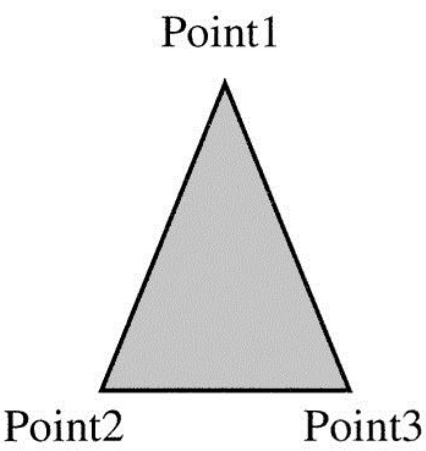


Rasterization

- But where do we get the RGB values?
- We will have to look at shading and illumination
- Now we will see how the theory is put into practice
- We will look at three ways of implementing the illumination equations:
  - Flat surface rendering
  - Gouraud surface rendering
  - Phong surface rendering

Graphics Primitives

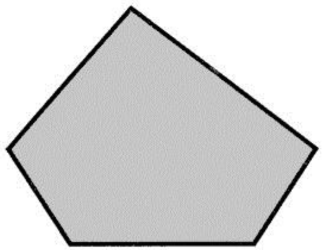
- Vertex: position, normal, color – how many values total?
- Polygon: series of connected vertices



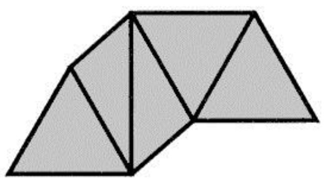

```
Point1
  position=(1,3,0)
  normal=  (0,0,1)
  color=   (.8,.8,.8)
Point2
  position=(0,0,0)
  normal=  (0,0,1)
  color=   (.8,.8,.8)
Point3
  position=(2,0,0)
  normal=  (0,0,1)
  color=   (.8,.8,.8)

Polygon1
  points= (1,2,3)
```

Graphics Primitives



Polygon — a set of edges, usually in a plane, that define a closed region. Triangles and rectangles are examples of polygons.



Triangle Strip — a series of triangles where each triangle shares its edges with its neighbors.

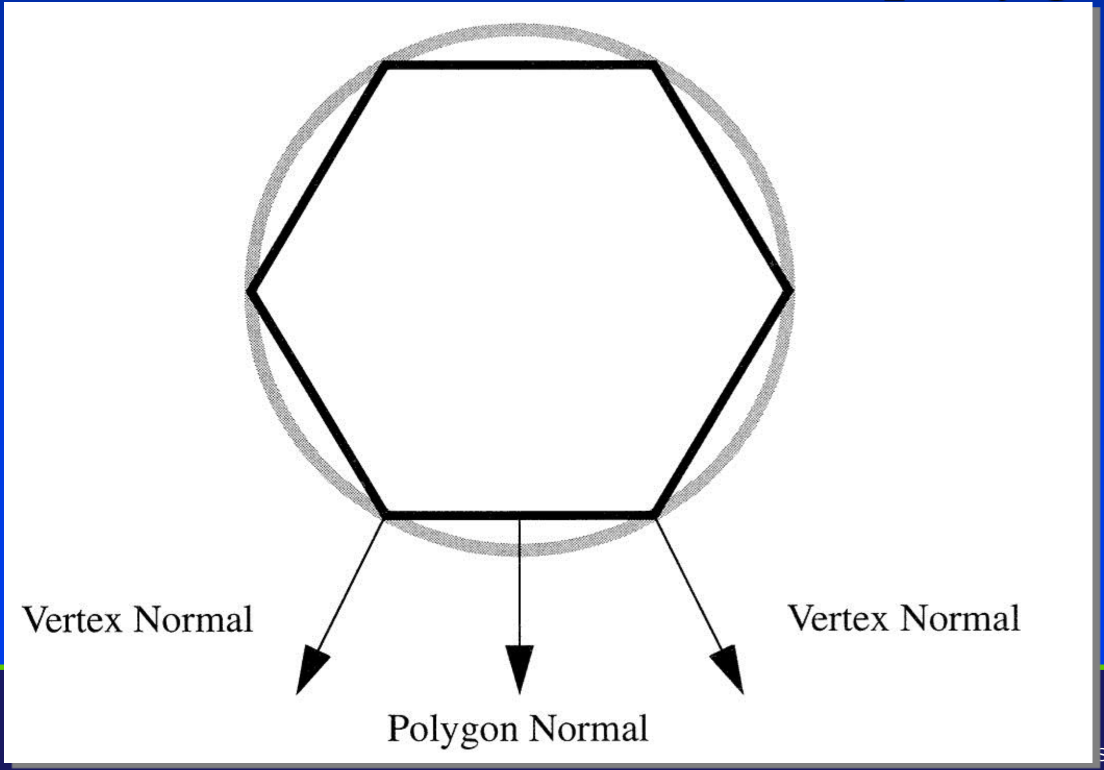
Line — connects two points.

Polyline — a series of connected lines.

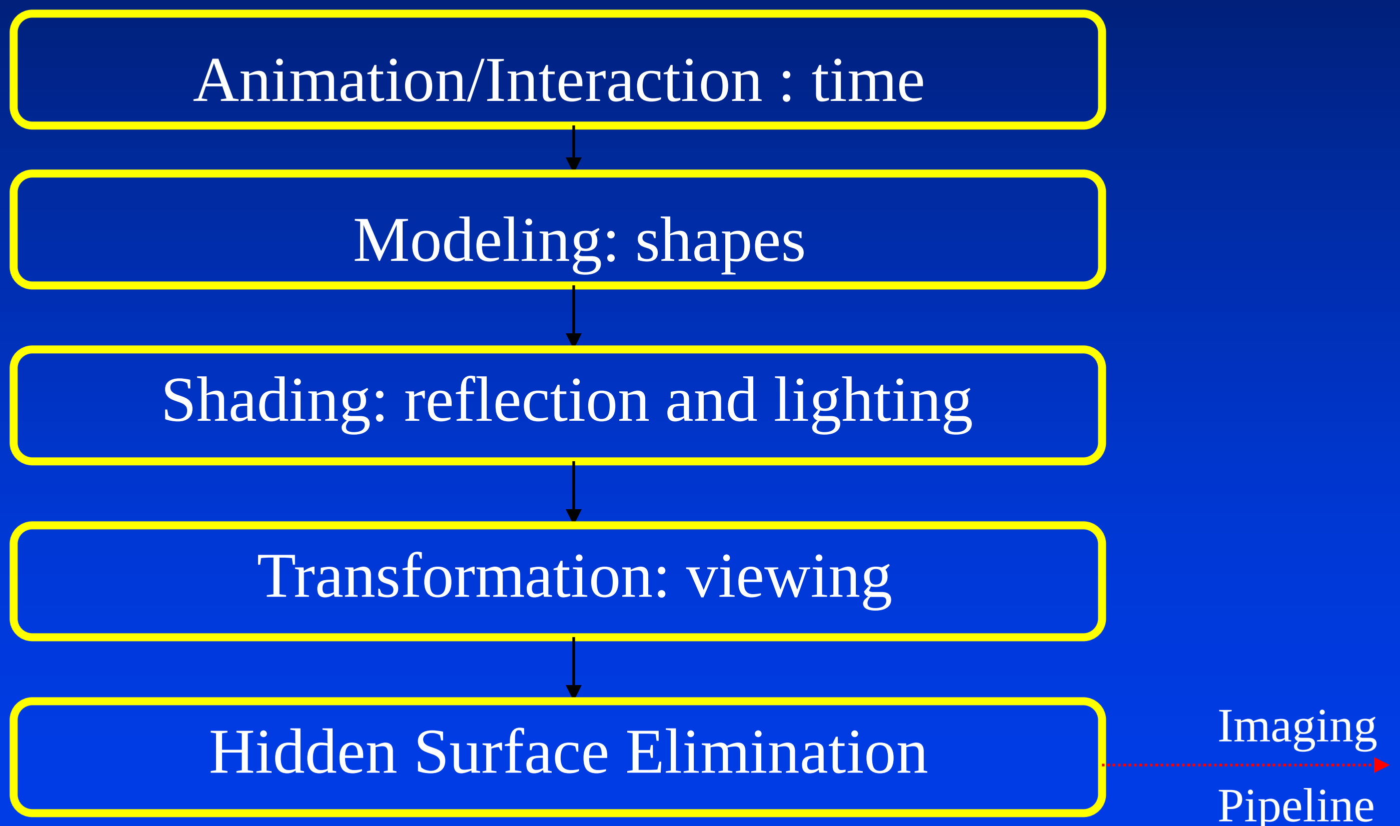
Point — a 3D position in space.

Graphics Primitives

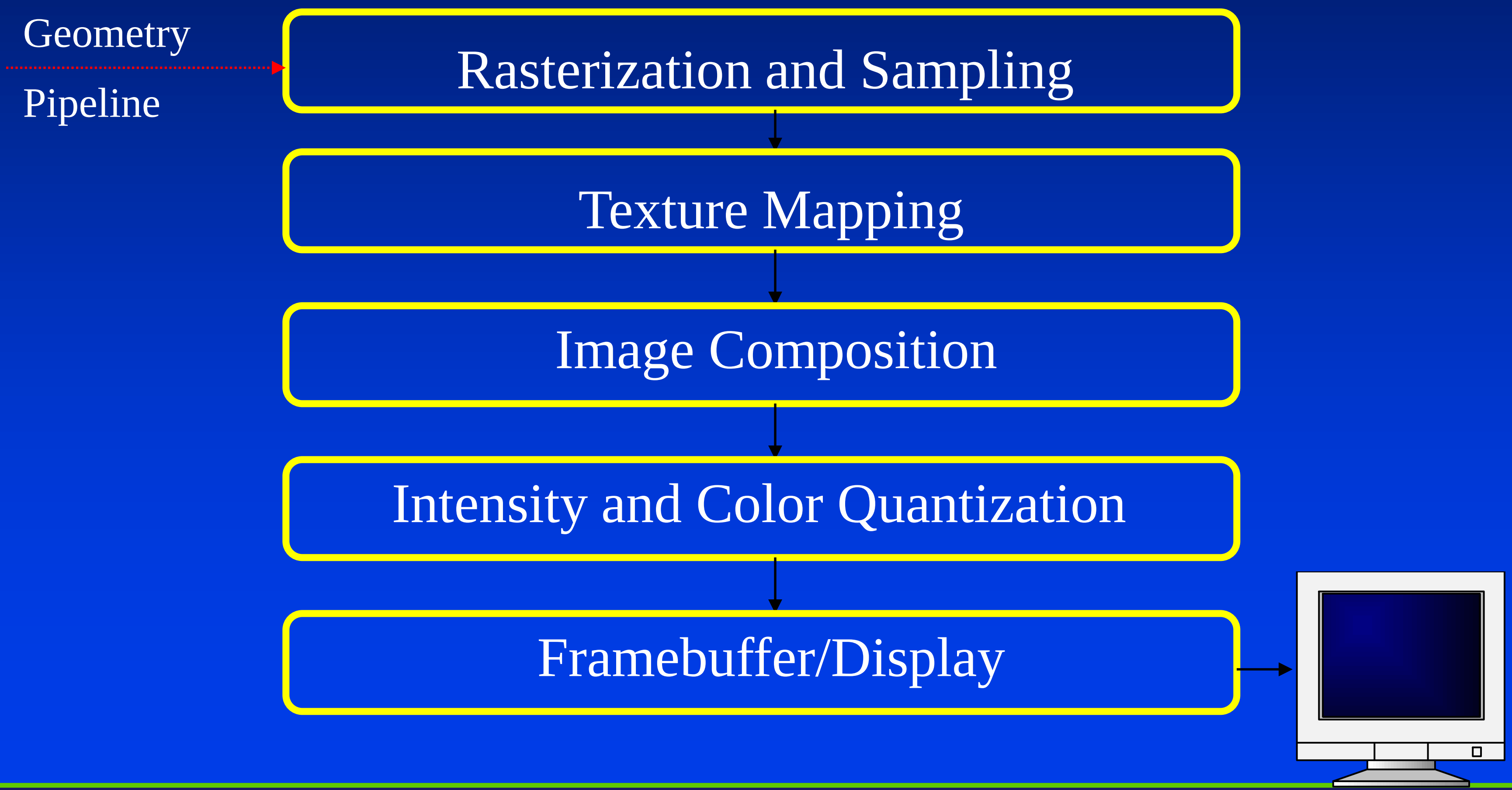
- Normal vectors: why for vertices?
- If our polygonal object came from curved surface, vertex normals will not be same as polygonal normals



Geometry Pipeline

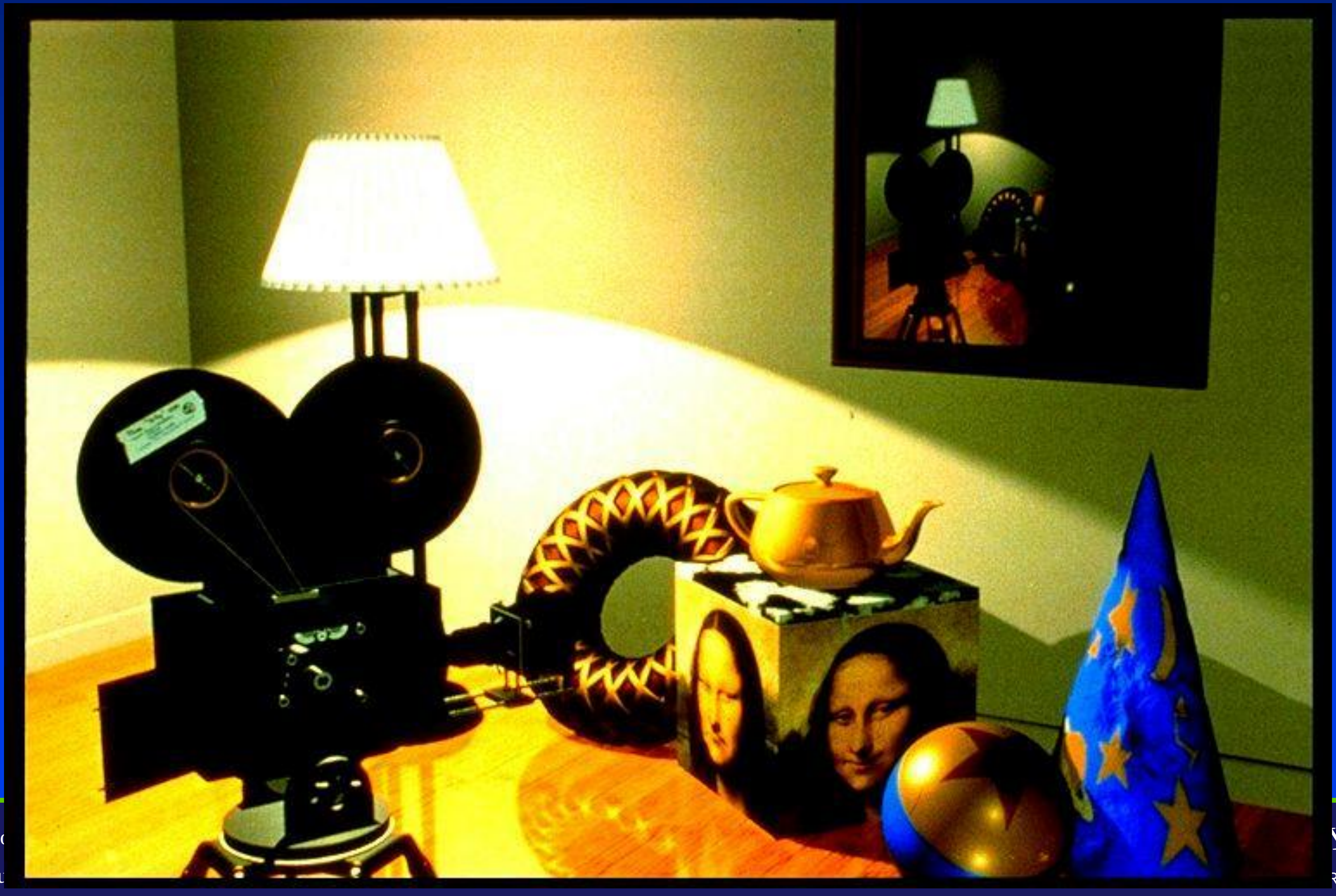


Imaging Pipeline

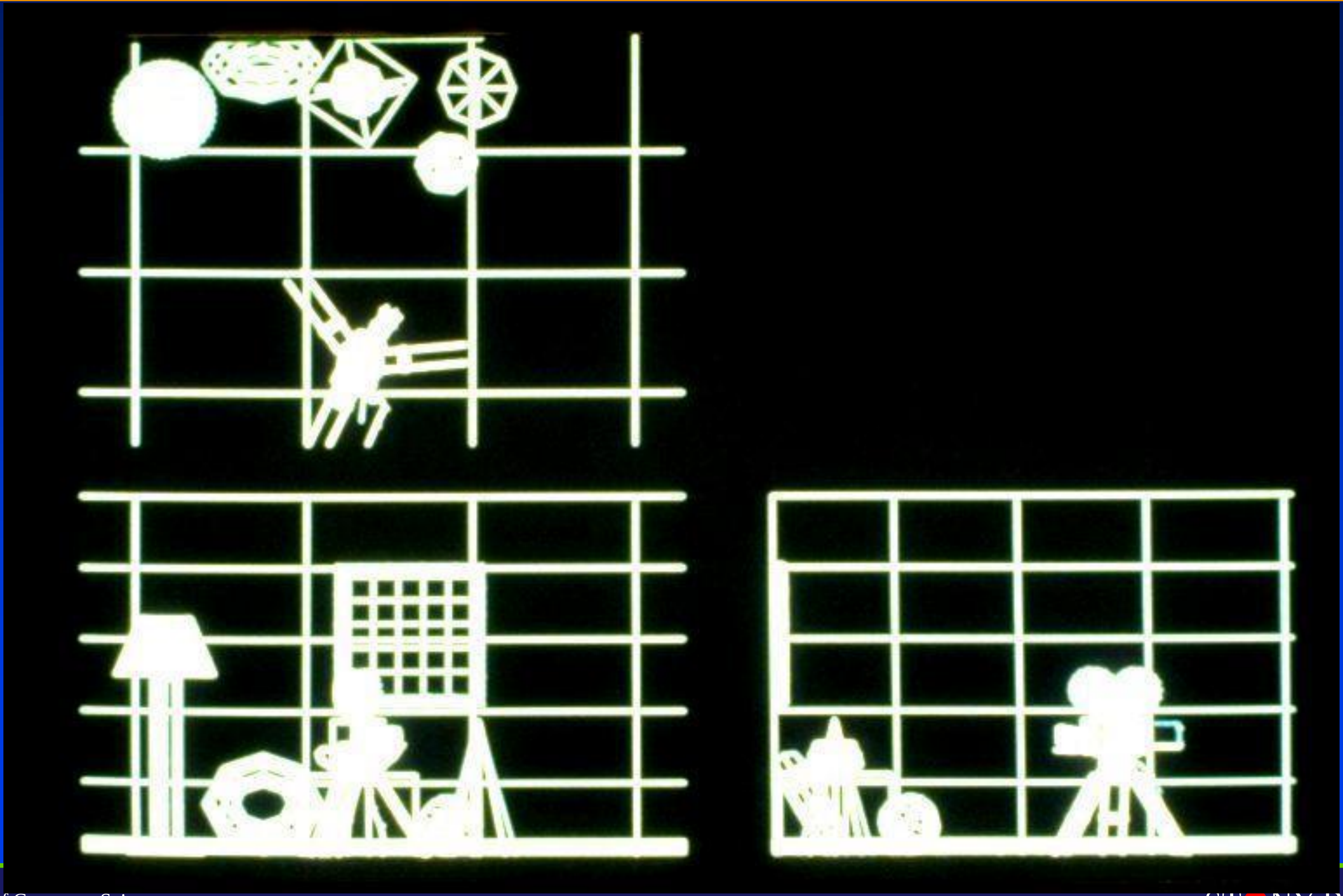


An Example through the Pipeline…

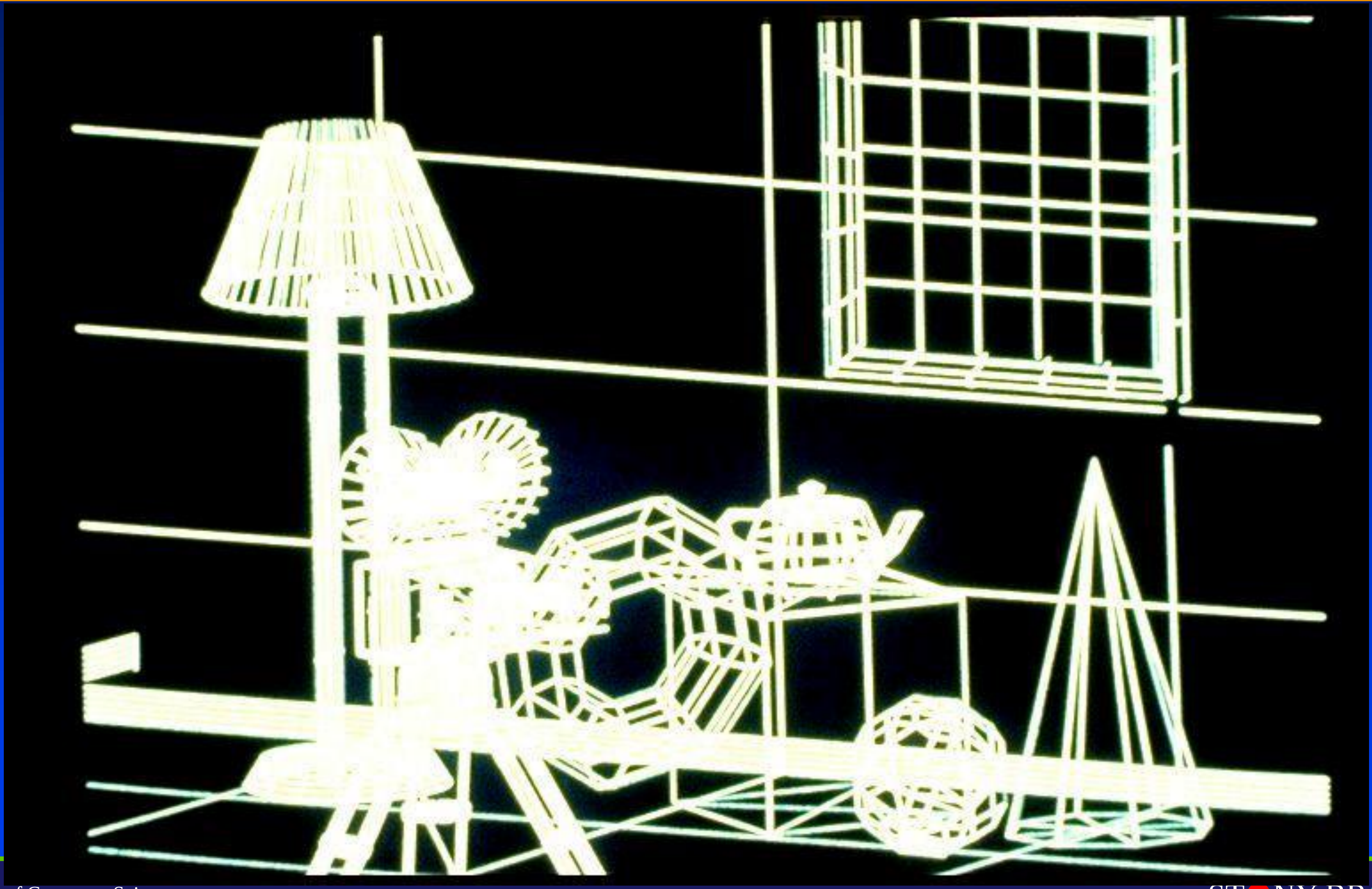
The scene we are trying to represent:



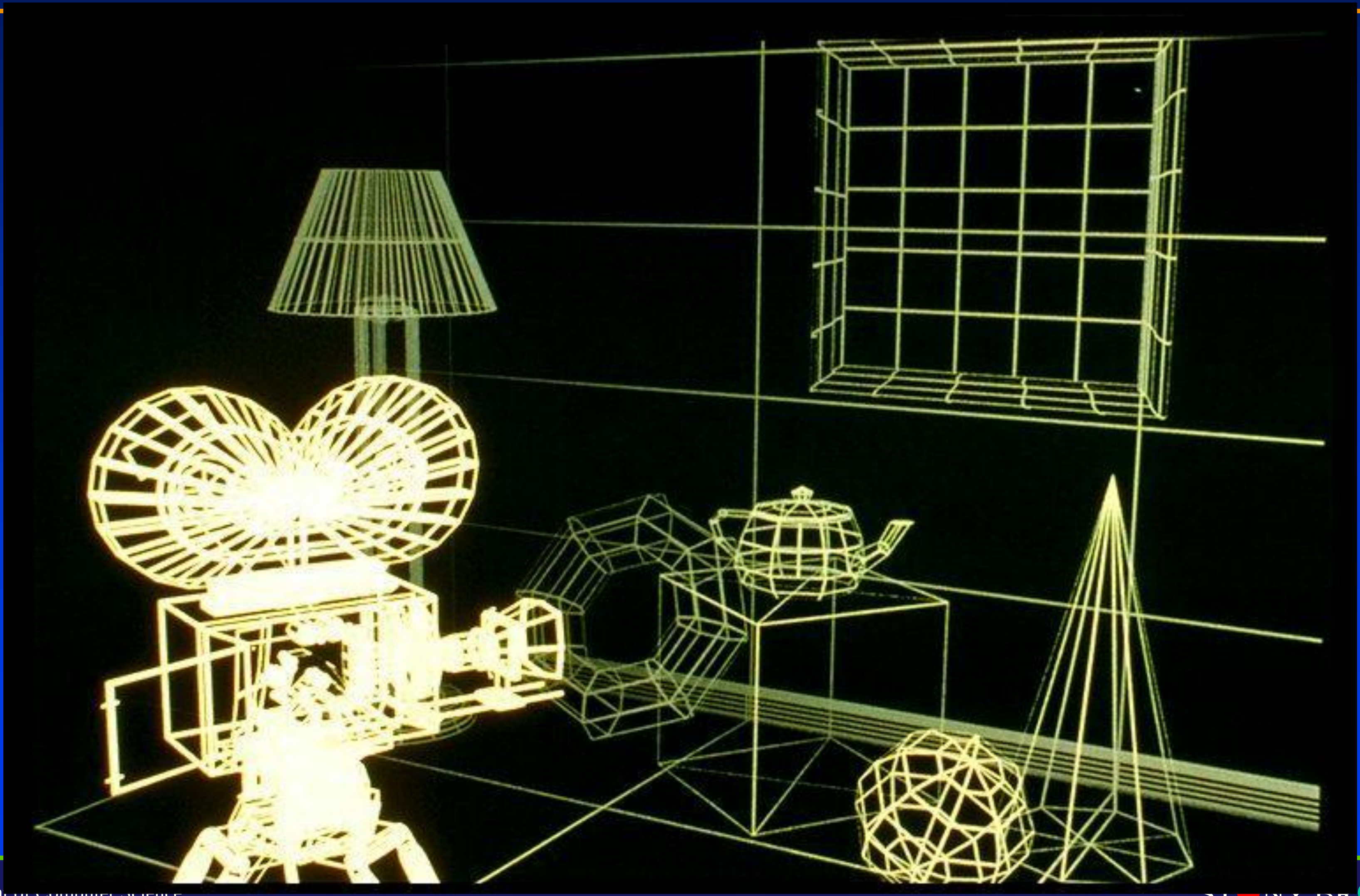
Wireframe Model – Orthographic Views



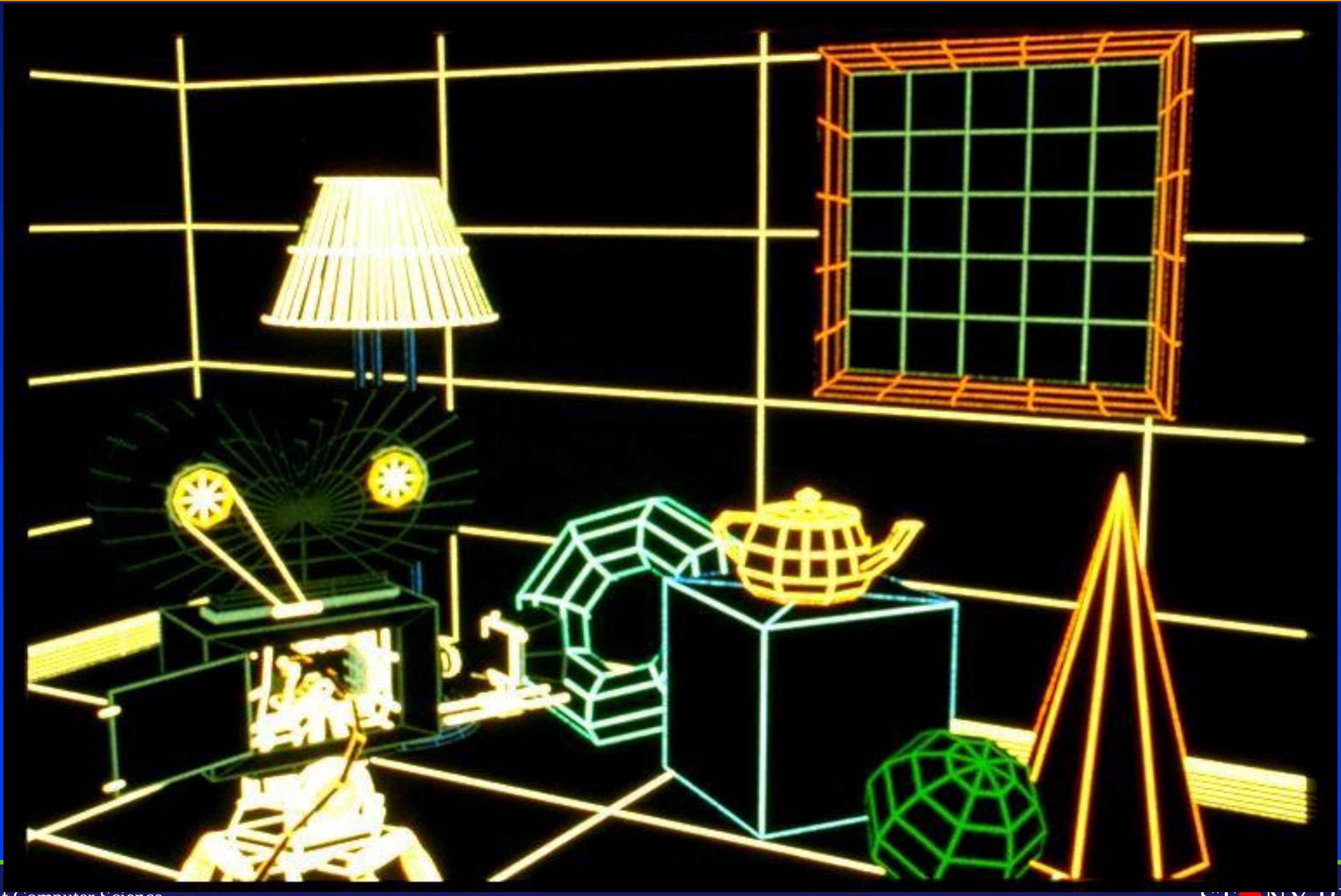
Perspective View



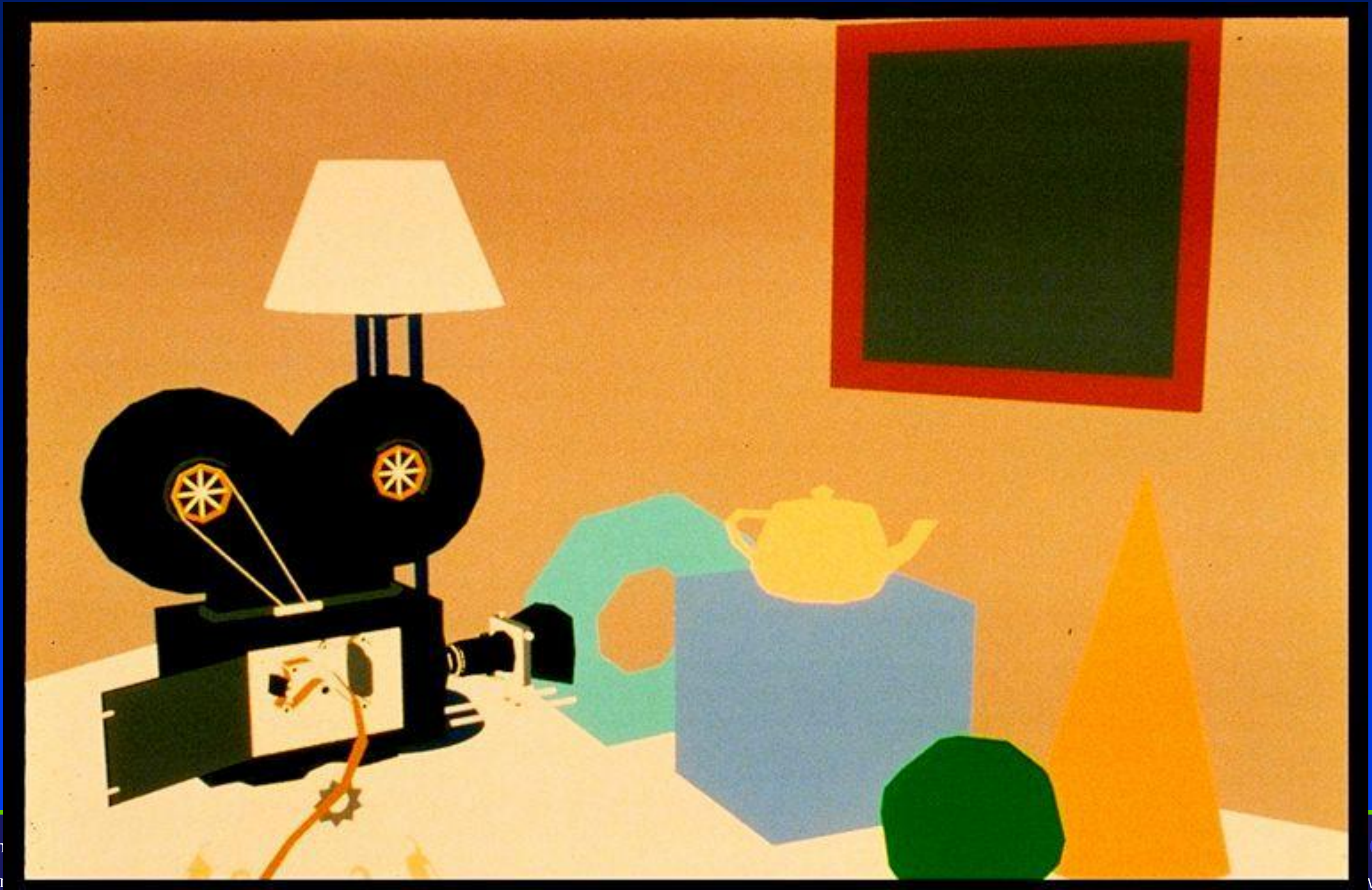
Depth Cue



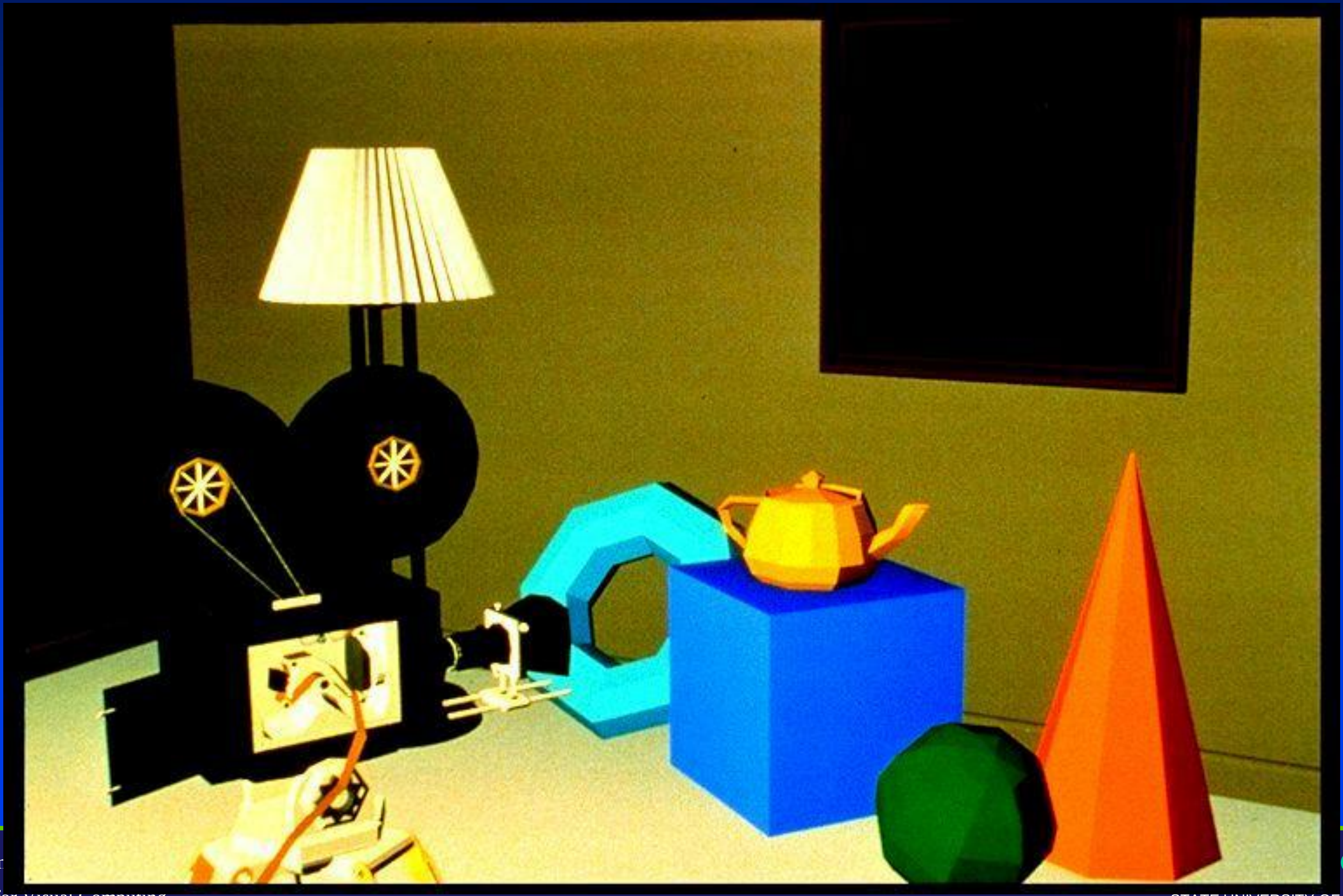
Hidden Line Removal – Add Color



Constant Shading - Ambient



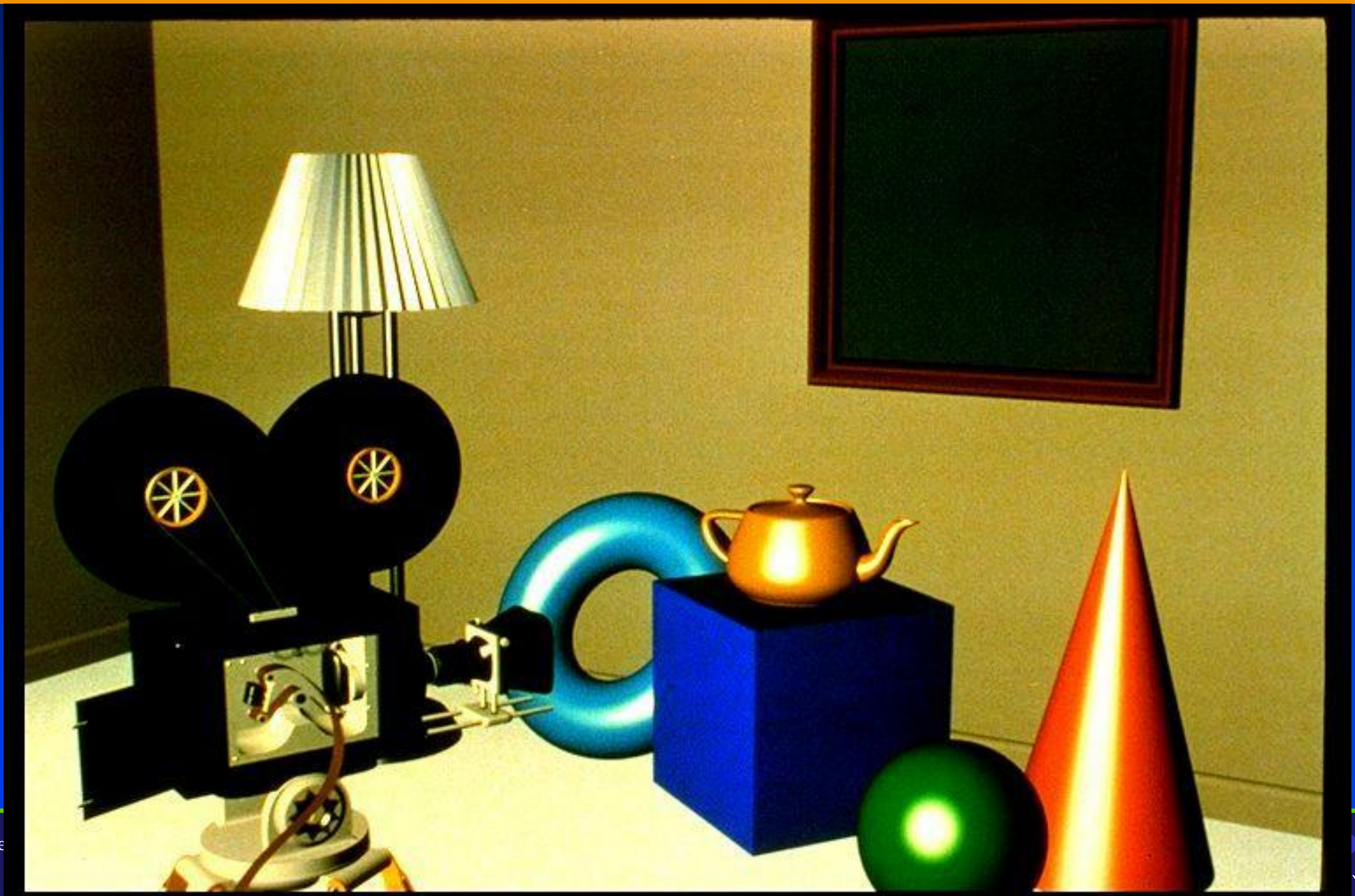
Faceted Shading - Flat



Gouraud Shading, No Specular Highlights

Specular Highlights

Phong Shading



Texture Mapping

Texture Mapping

Reflections, Shadows & Bump mapping