CS3241
Computer Graphics

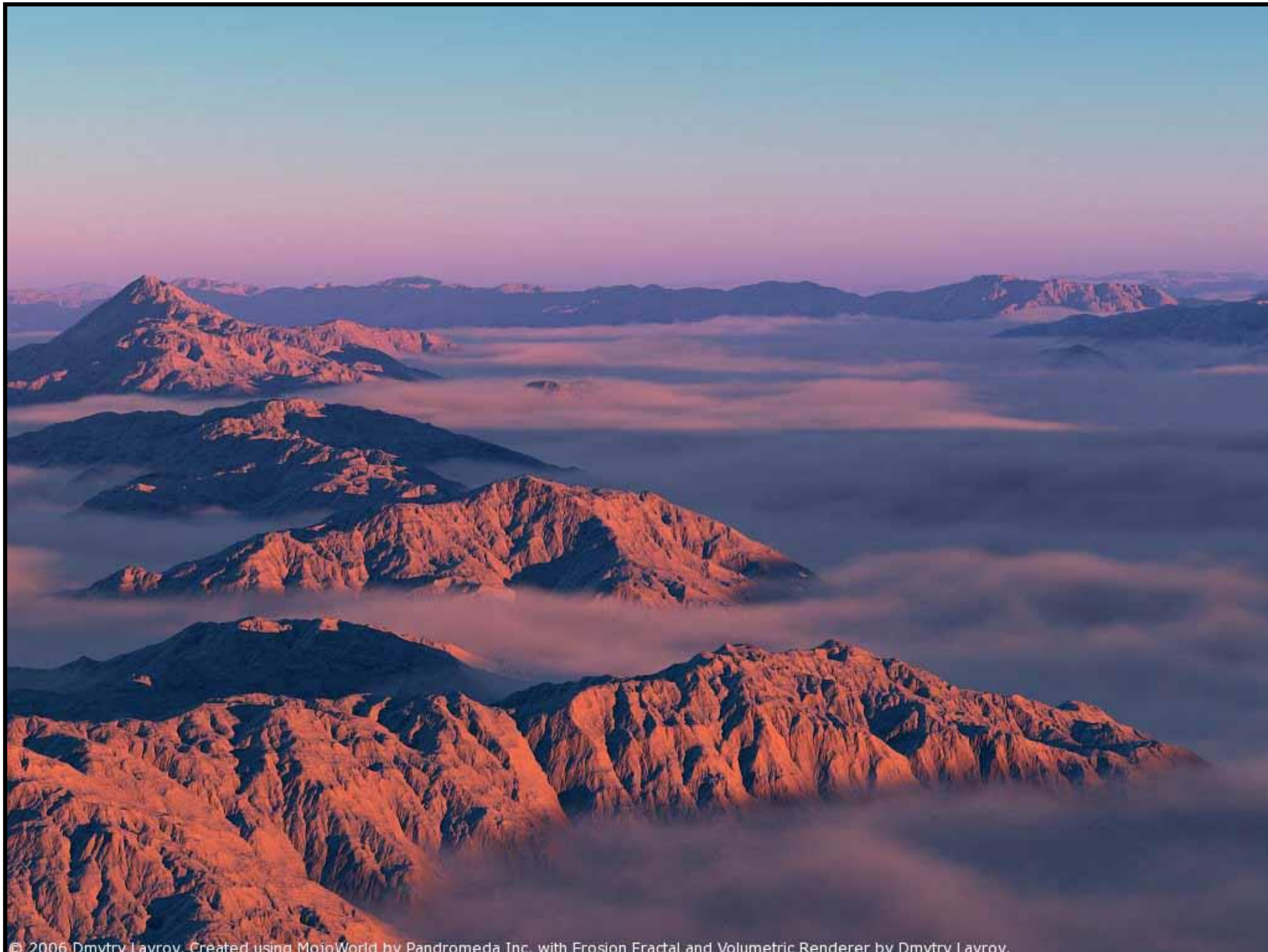
Semester 1, 2019/2020

Lecture 1

Introduction to Computer Graphics

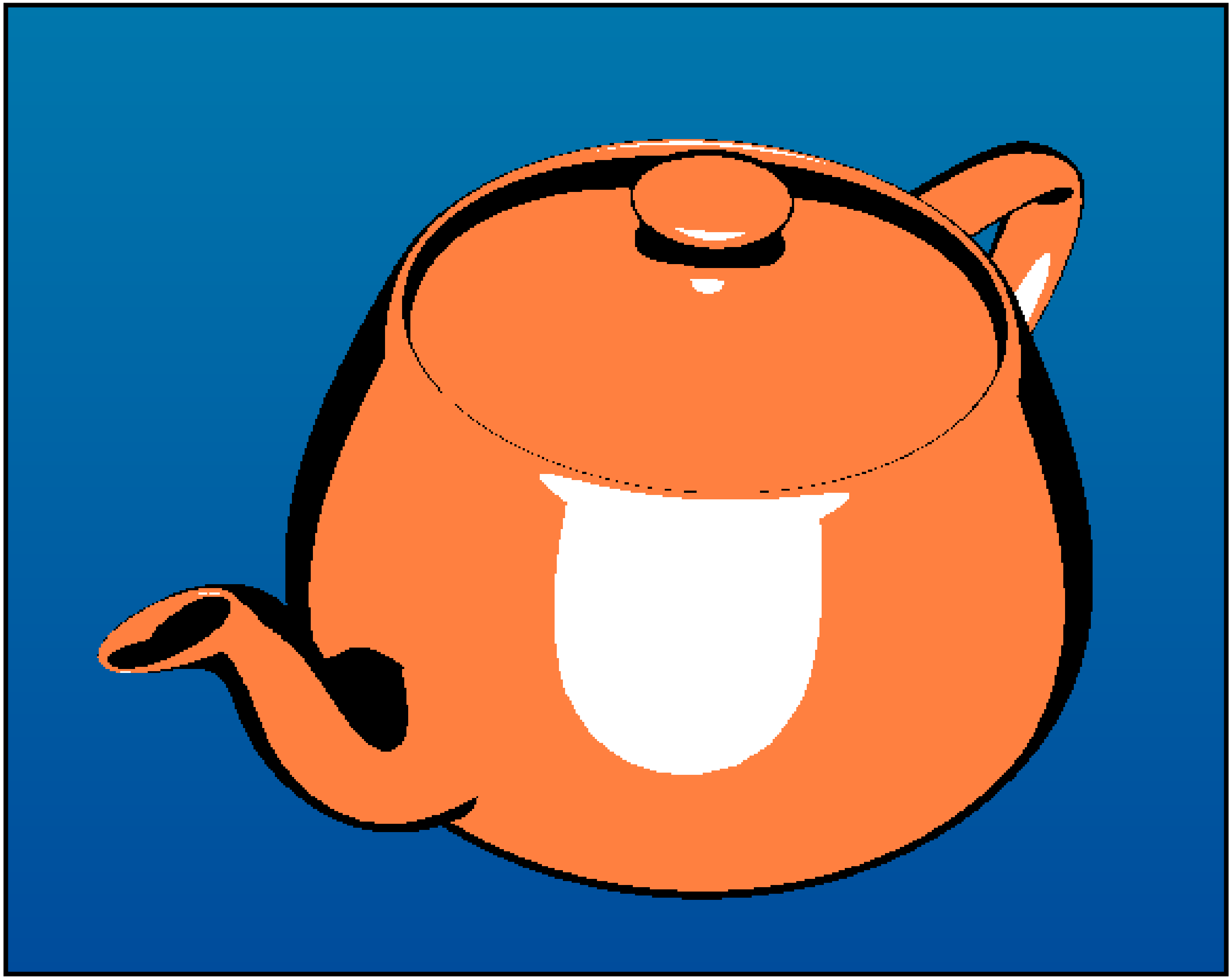
School *of* Computing
National University *of* Singapore


POVMAG


© 2006 Dmytry Lavrov. Created using MojoWorld by Pandromeda Inc. with Erosion Fractal and Volumetric Renderer by Dmytry Lavrov.

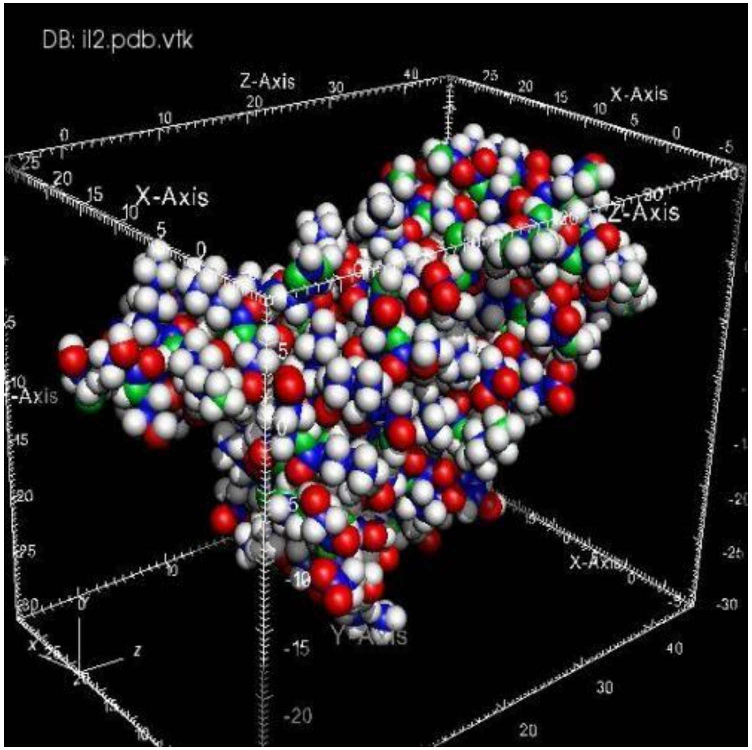
real | computer generated | real | computer generated

Non-Photorealistic Rendering

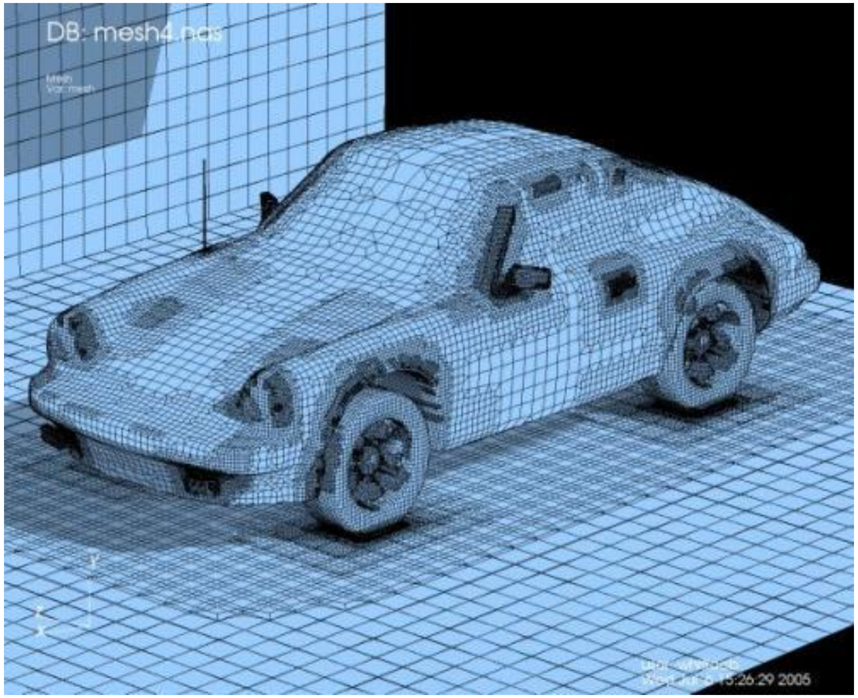


Real-Time Interactive Rendering

Scientific Visualization



3D modelling and design



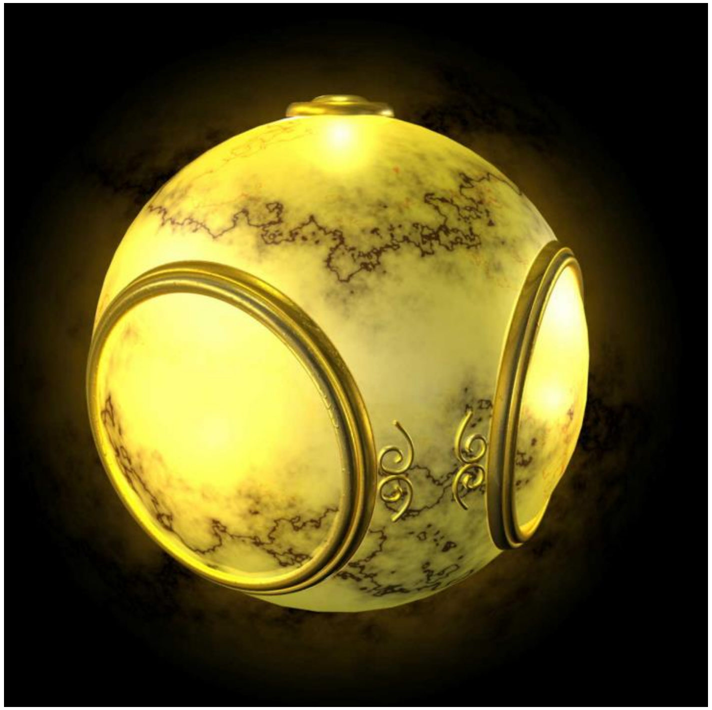
What is Computer Graphics?

What is Computer Graphics?

- Whatever areas that have appeared in the **SIGGRAPH** annual conferences
  - Special Interest Group on Computer GRAPHics and Interactive Techniques
  - https://en.wikipedia.org/wiki/SIGGRAPH
- **Computer graphics** deals with all aspects of creating images with a computer
  - Hardware
  - Software
  - Applications
- In this course, we focus on **3D** computer graphics

Example

- Where did this image come from?

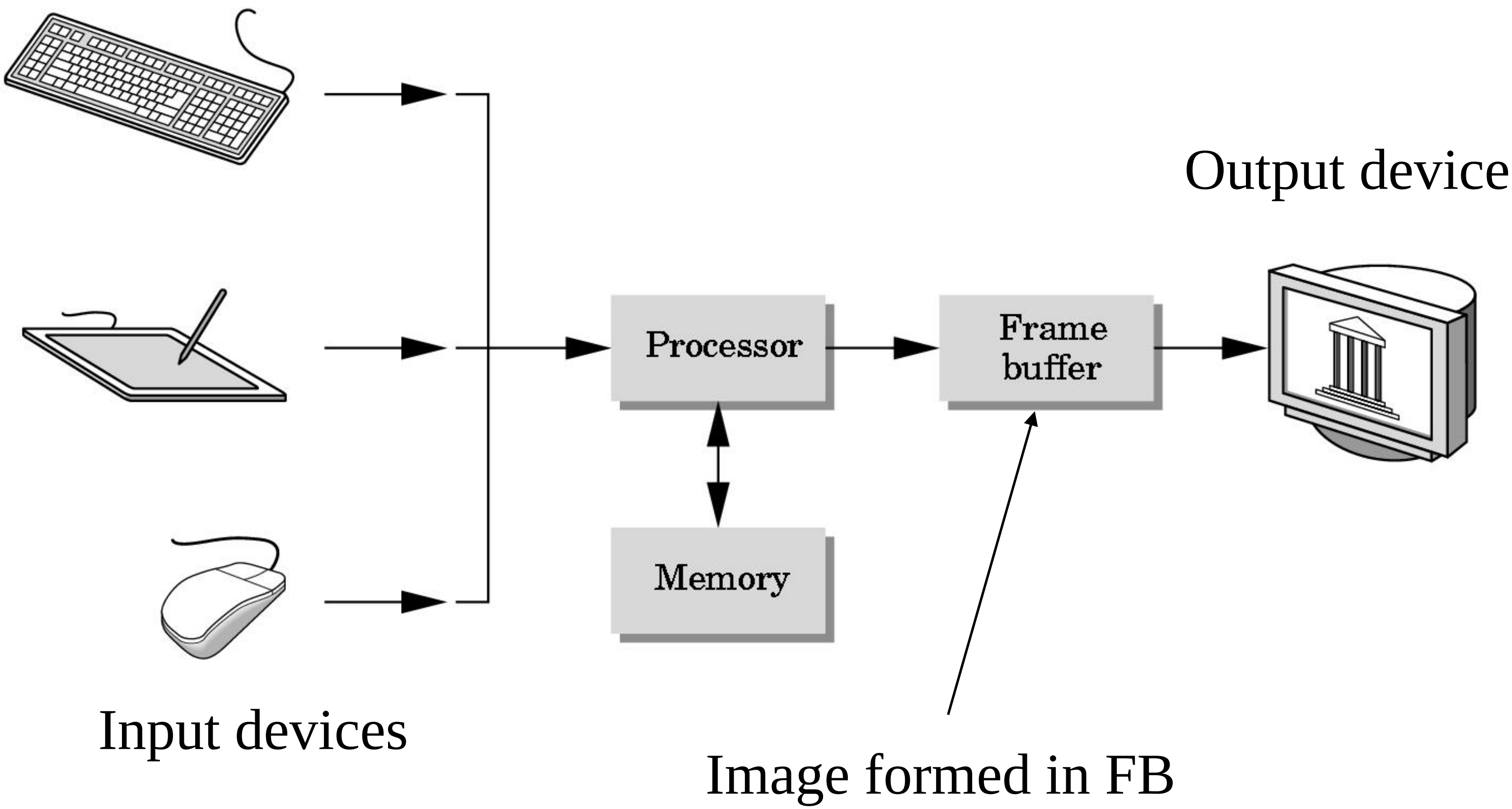


- What hardware/software did we need to produce it?

Preliminary Answer

- **Application**: The object is an artist's rendition of the sun for an animation to be shown in a domed environment (planetarium)
- **Software**: Maya for modeling and rendering but Maya is built on top of OpenGL
    - There is an interactive part for modeling and an non-interactive (slow) part for high-quality rendering
- **Hardware**: PC with graphics card for modeling and rendering

Basic Graphics System

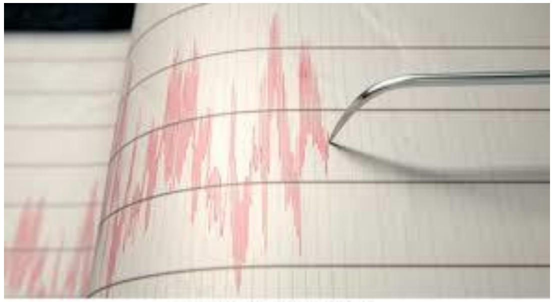


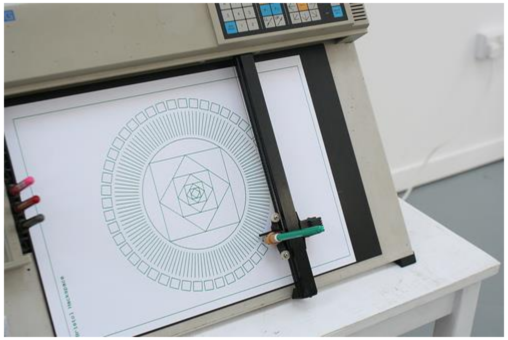
History of Computer Graphics

- When a Bit Became a Pixel: The History of Computer Graphics
  - https://www.youtube.com/watch?v=iw1o4ozvjEU
- A History of Computer Graphics
  - https://www.youtube.com/watch?v=Rk5dw4qcFUg
- A Brief History of Graphics (video games)
  - https://www.youtube.com/watch?v=QyjyWUrHsFc&t=2398s

Computer Graphics: 1950-1960

- Computer graphics goes back to the earliest days of computing
  - Strip charts
  - Pen plotters
  - Calligraphic CRT

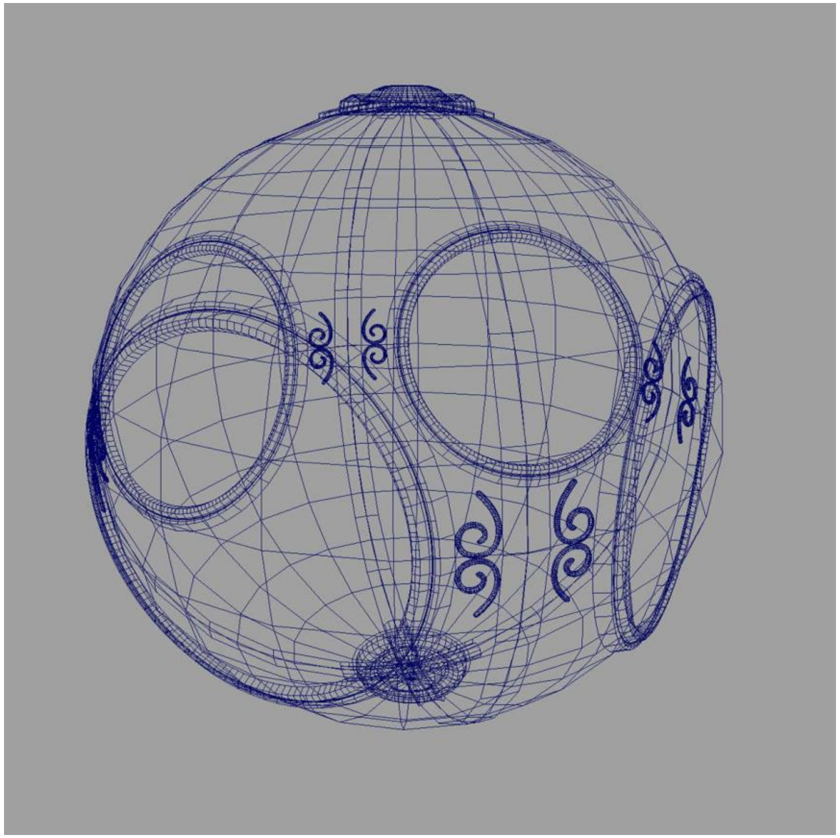




- Cost of refresh for CRT too high
  - Computers slow, expensive, unreliable

Computer Graphics: 1960-1970

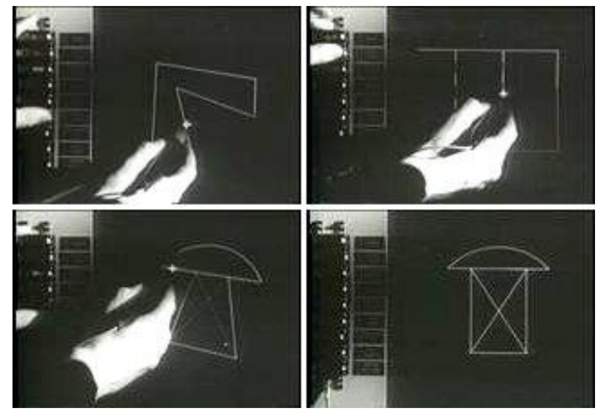
- *Wireframe* graphics
  - Draw only lines
- Sketchpad
- Display Processors



wireframe representation
of sun object

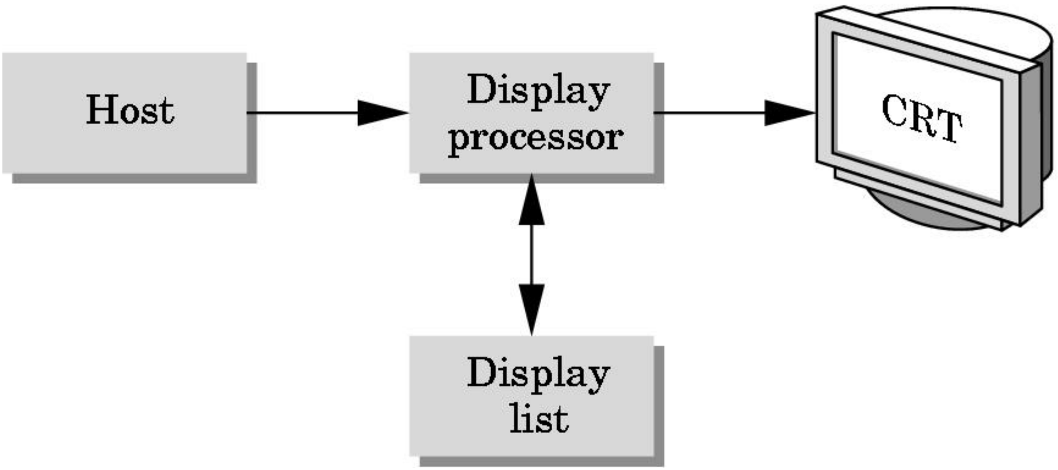
Sketchpad

- Ivan Sutherland's PhD thesis at MIT
  - Recognized the potential of *man-machine interaction*
  - Loop
    - Display something
    - User moves light pen
    - Computer generates new display
  - Sutherland also created many of the now common algorithms for computer graphics
- Ivan Sutherland is considered the Father of Computer Graphics



Display Processor

- Rather than have the host computer try to refresh display use a special purpose computer called a ***display processor*** (DPU)



- Graphics stored in display list (display file) on display processor
- Host *compiles* display list and sends to DPU

Computer Graphics: 1970-1980

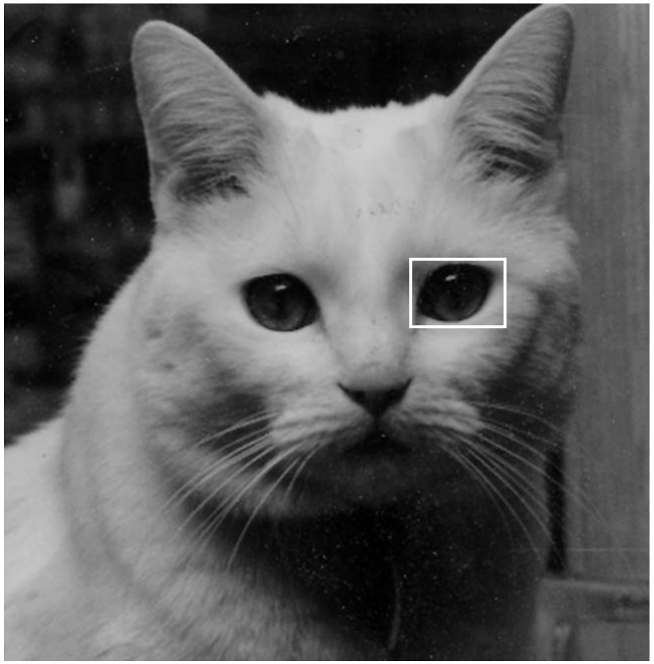
- Raster Graphics

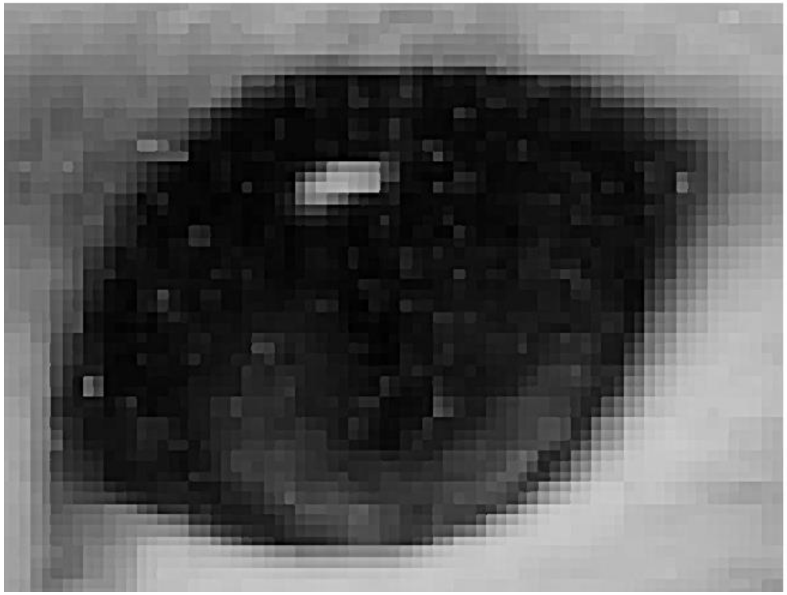
- Beginning of graphics standards
  - IFIPS
    - GKS: European effort
      - Becomes ISO 2D standard
    - Core: North American effort
      - 3D but fails to become ISO standard

- Workstations and PCs

Raster Graphics

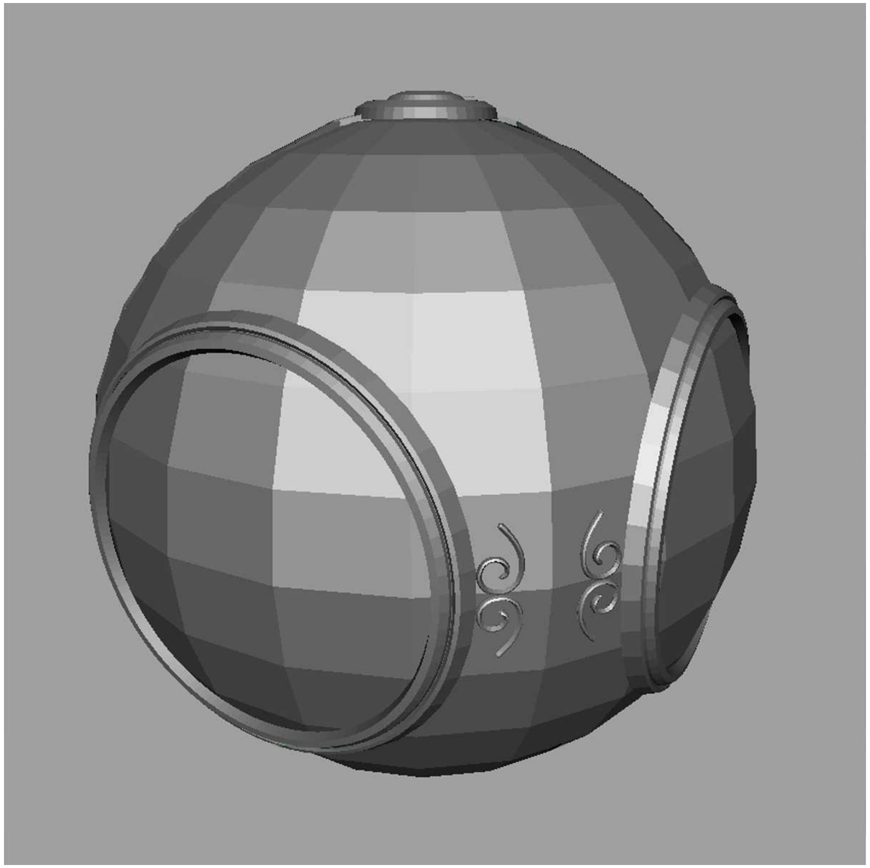
- Image produced as an array (the ***raster***) of picture elements (***pixels***) in the ***frame buffer***





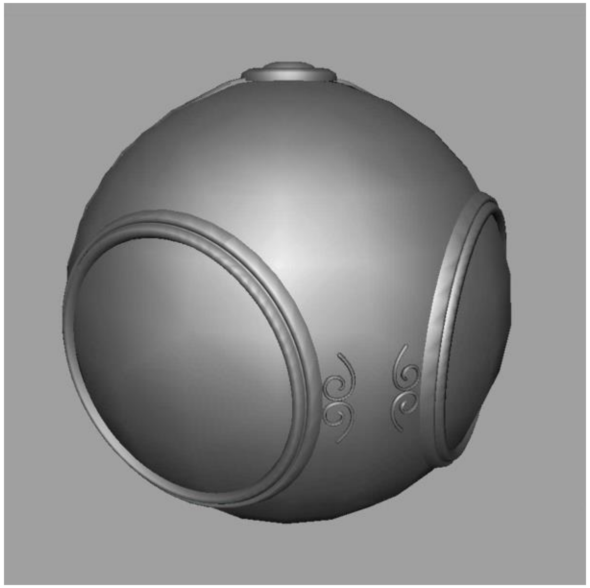
Raster Graphics

- Allows us to go from lines and wireframe images to ***filled polygons***



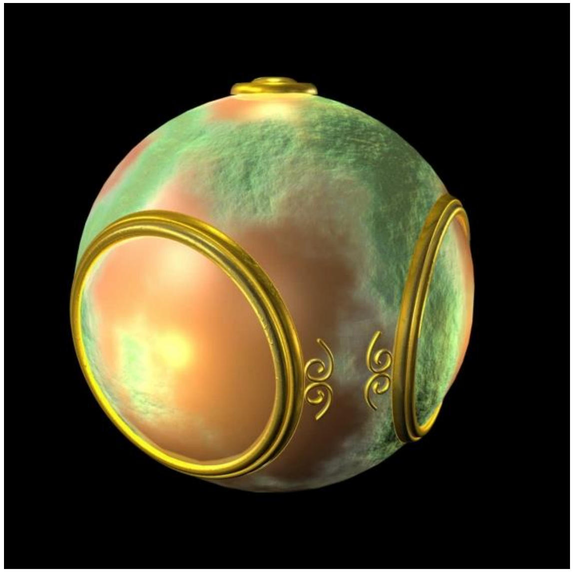
Computer Graphics: 1980-1990

- Realism comes to computer graphics



smooth shading

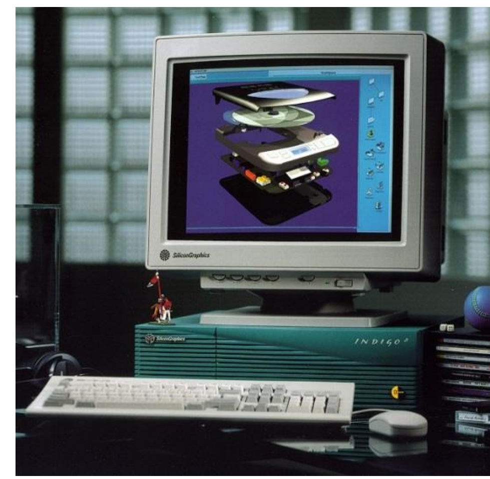
environment mapping



bump mapping

Computer Graphics: 1980-1990

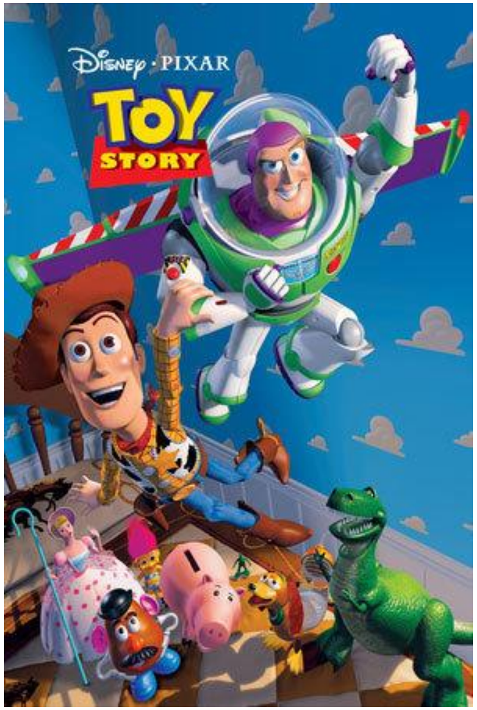
- Special purpose hardware
  - Silicon Graphics Geometry Engine
    - VLSI implementation of graphics pipeline



- Industry-based standards
  - PHIGS
  - RenderMan
- Networked graphics: X Window System
- Human-Computer Interface (HCI)

Computer Graphics: 1990-2000

- OpenGL API
- Completely computer-generated feature-length movies (e.g. Toy Story) are successful
- New hardware capabilities
  - Texture mapping
  - Blending
  - Accumulation, stencil buffers



Computer Graphics: 2000-

- Photorealism
- Affordable graphics cards for PCs dominate market
  - NVIDIA, AMD/ATI
- Game console boxes and game players determine direction of market
- Computer graphics routine in movie industry
- Programmable pipelines

State of the Art Examples

- Unreal Engine 4
  - https://www.youtube.com/watch?v=zKu1Y-LIfNQ

- Ray Tracing in Minecraft
  - https://www.youtube.com/watch?v=xg4-4XXZiLY

- NVIDIA Real-Time Ray Tracing
  - https://www.youtube.com/watch?v=u8tDgvvGWSE
  - https://www.youtube.com/watch?v=AV279wThmVU

Image Formation

Objectives

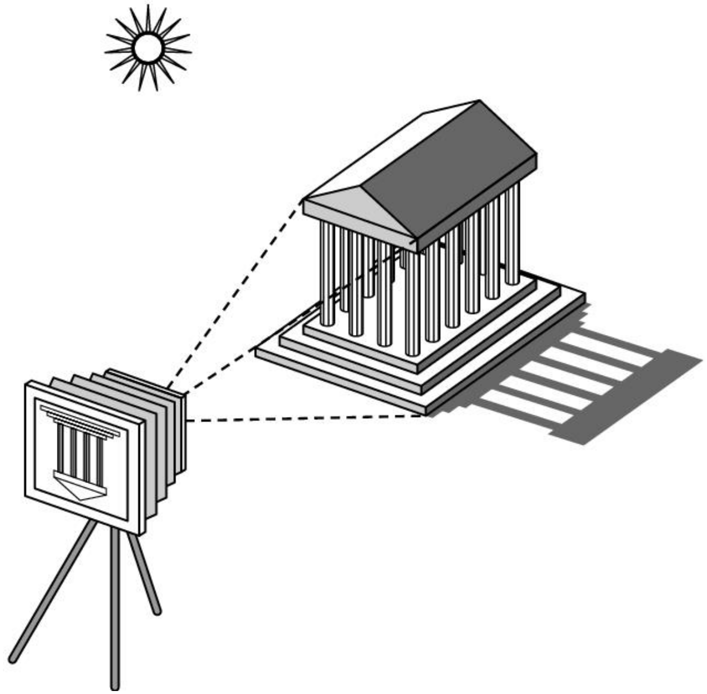
- Fundamental imaging notions
- Physical basis for image formation
  - Light
  - Color
  - Perception
- Synthetic camera model
- Other models

Image Formation

- In computer graphics, we form images which are generally two dimensional using a process analogous to how images are formed by physical imaging systems
  - Cameras
  - Microscopes
  - Telescopes
  - Human visual system

Elements of Image Formation

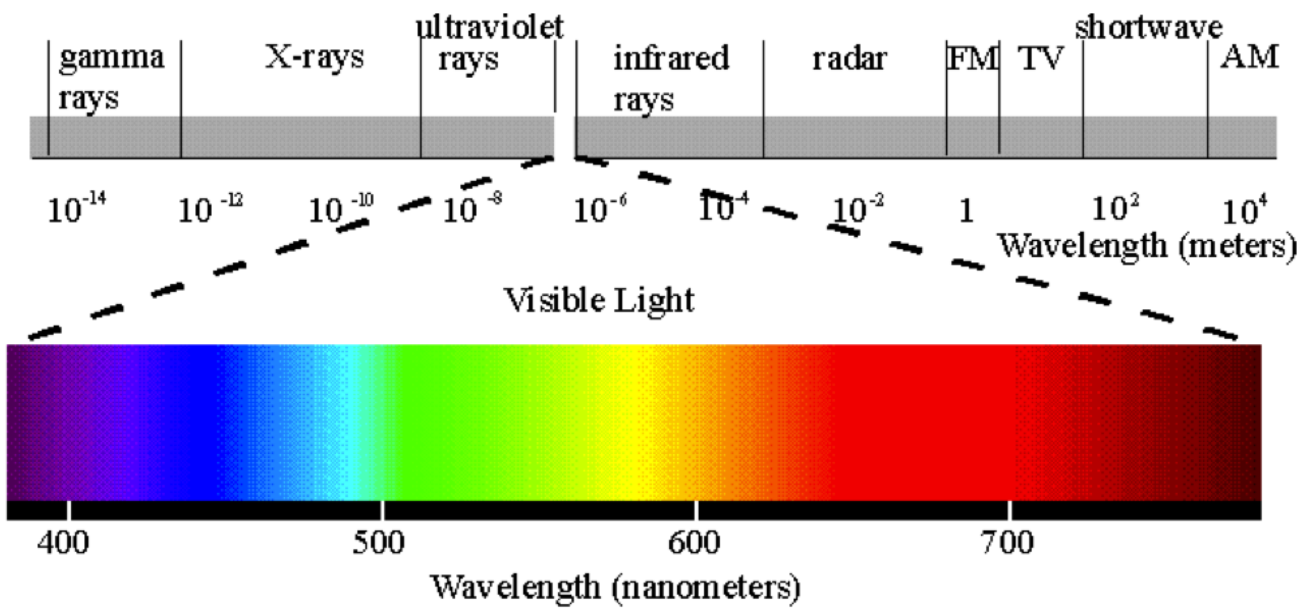
- Objects
- Viewer
- Light source(s)
- Materials
  - Attributes that govern how light interacts with the materials in the scene



- Note the independence of the objects, the viewer, and the light source(s)

Light

- *Light* is the part of the electromagnetic spectrum that causes a reaction in our visual systems
- Generally these are wavelengths in the range of about 350-750 nm (nanometers)
- Long wavelengths appear as reds and short wavelengths as blues

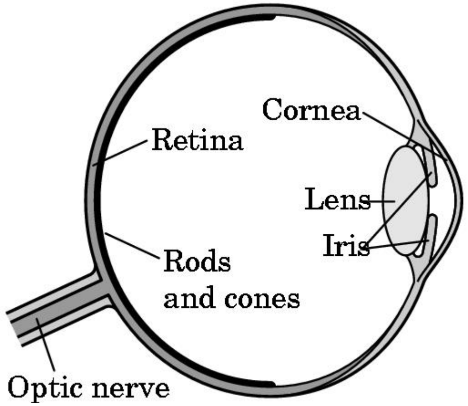


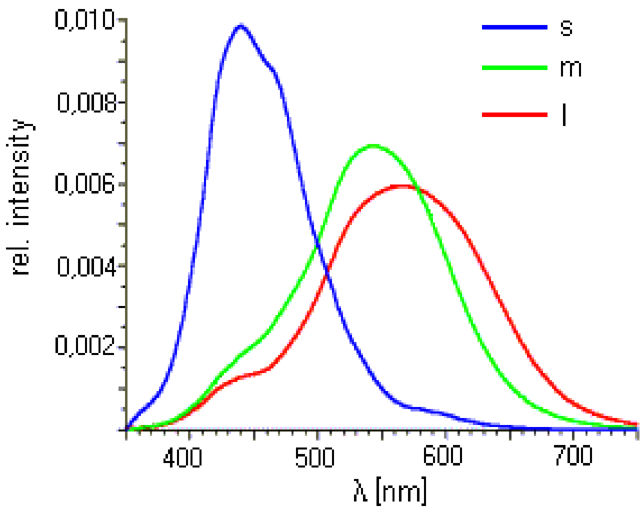
Luminance and Color Images

- Luminance Image
  - Monochromatic
  - Values are gray levels
  - Analogous to working with black and white film or television
- Color Image
  - Has perceptional attributes of hue, saturation, and lightness
  - Do we have to match every frequency in visible spectrum? No! Why?

Three-Color Theory

- Human visual system has two types of sensors
  - *Rods*: monochromatic, night vision
  - *Cones*
    - Color sensitive
    - Three types of cones
    - Only three values (the ***tristimulus*** values) are sent to the brain
- Need only match these three values
  - Need only three *primary* colors
    - Red (R), Green (G), Blue (B)



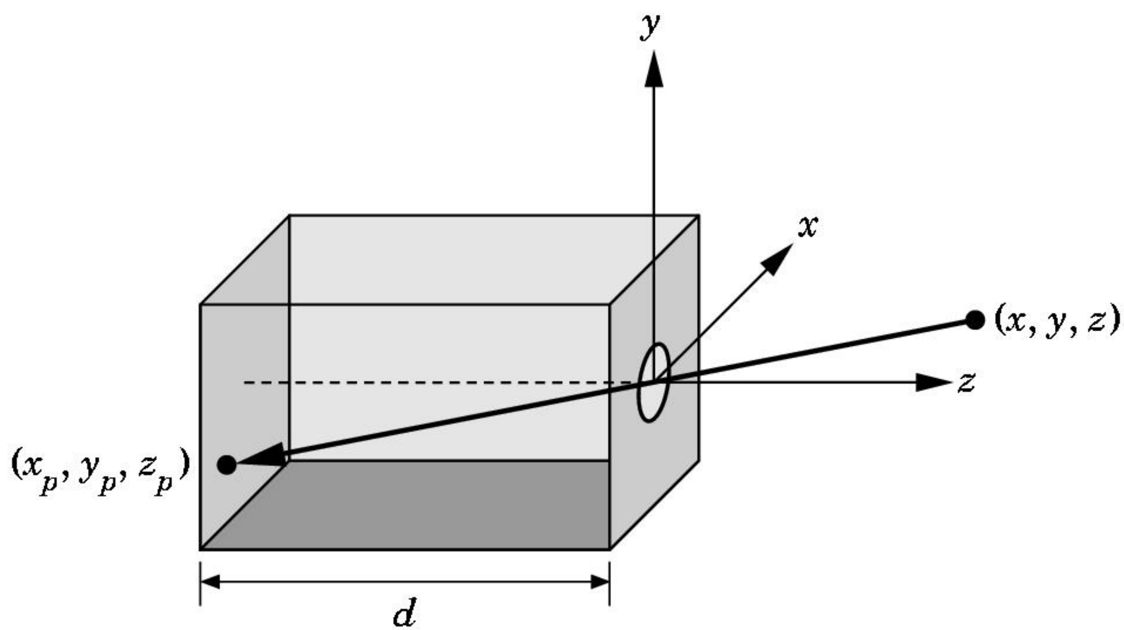


Additive and Subtractive Color

- Additive color
  - Form a color by adding amounts of three primaries
    - CRTs, projection systems, positive film
  - Primaries are Red (R), Green (G), Blue (B)

- Subtractive color
  - Form a color by filtering white light with cyan (C), Magenta (M), and Yellow (Y) filters
    - Light-material interactions
    - Printing
    - Negative film

Pinhole Camera

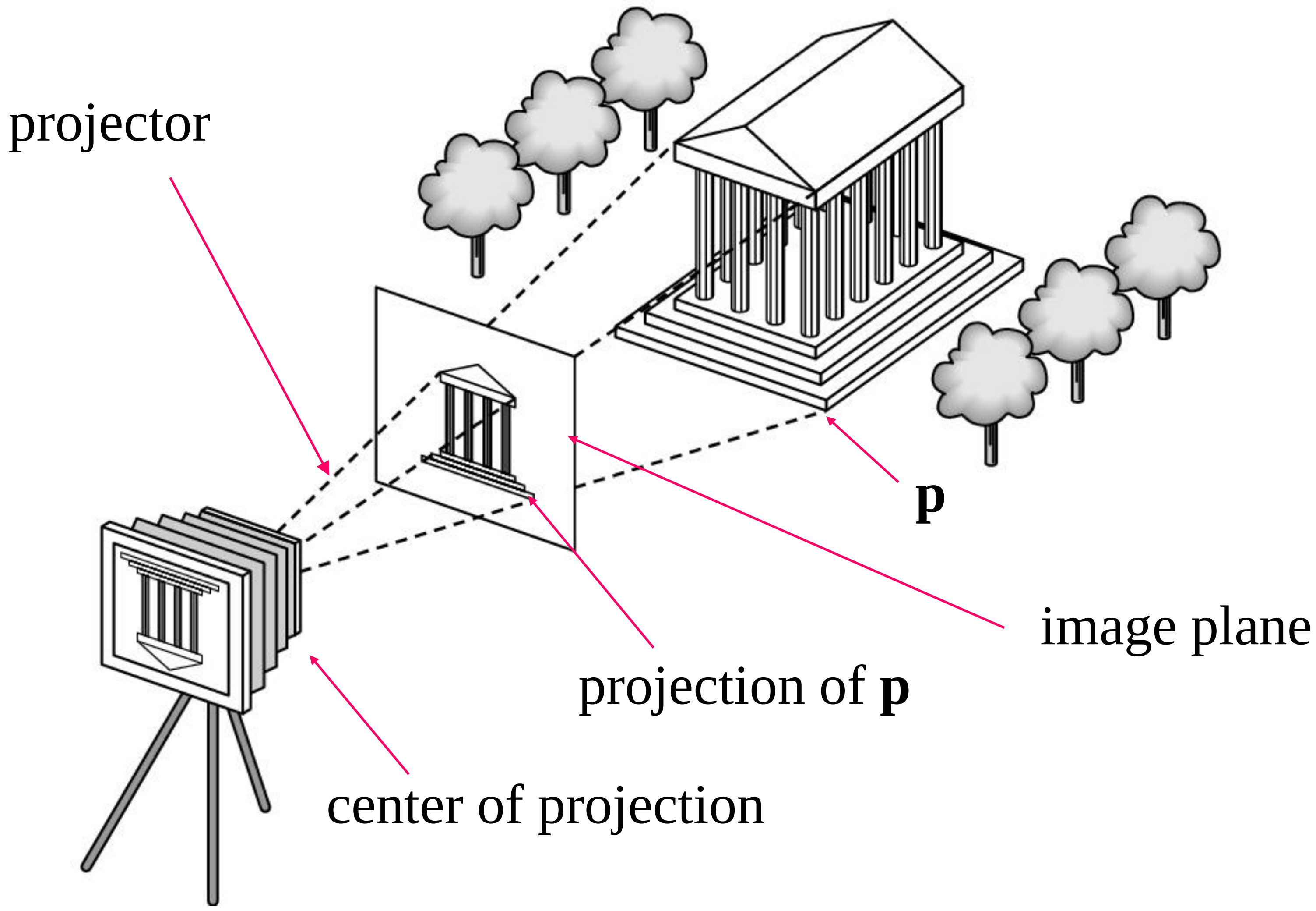


Use trigonometry to find projection of point at (x, y, z)

$$x_p = -dx/z \qquad y_p = -dy/z \qquad z_p = -d$$

These are equations of simple perspective

Synthetic Camera Model

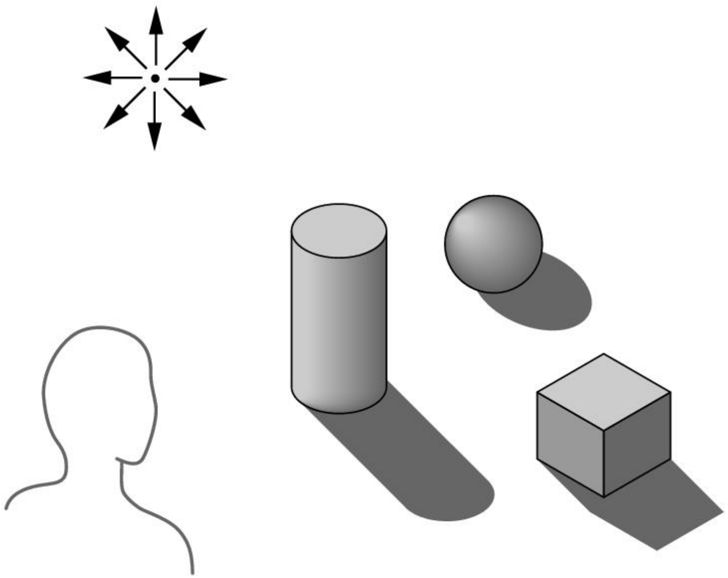


Advantages

- Separation of objects, viewer, light sources
- Two-dimensional graphics is a special case of three-dimensional graphics
- Leads to simple software API
  - Specify objects, lights, camera, attributes
  - Let implementation determine image
- Leads to fast hardware implementation

Global vs Local Lighting

- Cannot compute color or shade of each object independently
  - Some objects are blocked from light
  - Light can reflect from object to object
  - Some objects might be translucent



Models and Architectures

Objectives

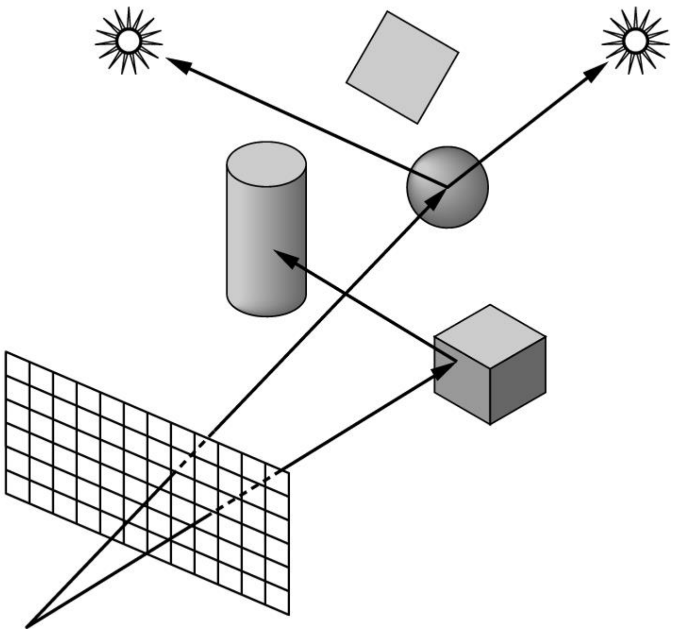
- Learn the basic design of a graphics system
- Introduce pipeline architecture
  - Very suitable for raster graphics rendering
- Examine software components for an interactive graphics system

Image Formation Revisited

- Can we mimic the synthetic camera model to design graphics hardware software?
- Application Programmer Interface (API)
  - Need only specify
    - Objects
    - Materials
    - Viewer
    - Lights
- But how is the API implemented?

Physical Approaches

- **Ray tracing**: follow rays of light from center of projection until they either are absorbed by objects or go off to infinity
  - Can handle global effects
    - Multiple reflections
    - Translucent objects
  - Slow
  - Must have whole database available at all times

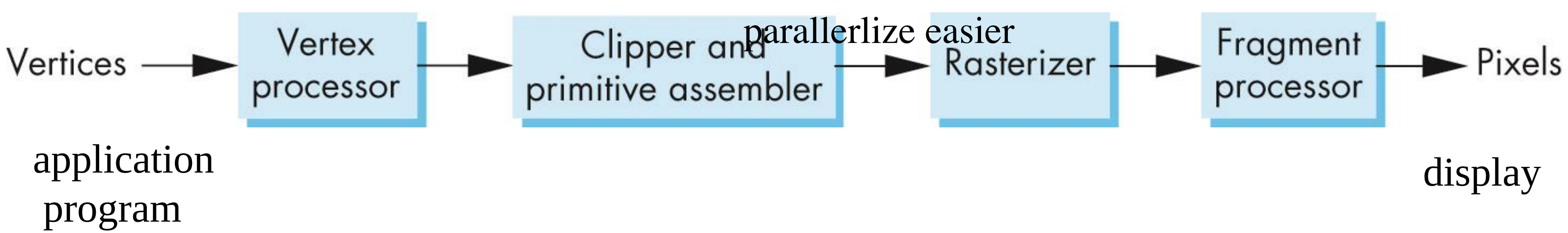


- **Radiosity**: Energy based approach
  - Very slow
  - Not general

Practical Approach for Real-Time Rendering

- Process objects/primitives one at a time in the order they are generated by the application
  - Can consider only local lighting
  - Object-oriented rendering

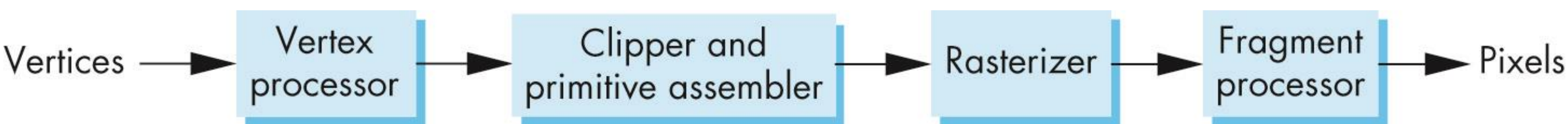
- Pipeline architecture

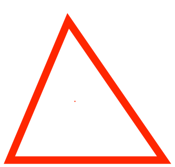


- All steps can be implemented in graphics hardware

Vertex Processing

- Much of the work in the pipeline is in converting object representations from one coordinate system to another
  - Object coordinates
  - Camera (eye) coordinates
  - Screen coordinates
- Every change of coordinates is equivalent to a matrix transformation
- Vertex processor also computes vertex colors (lighting)





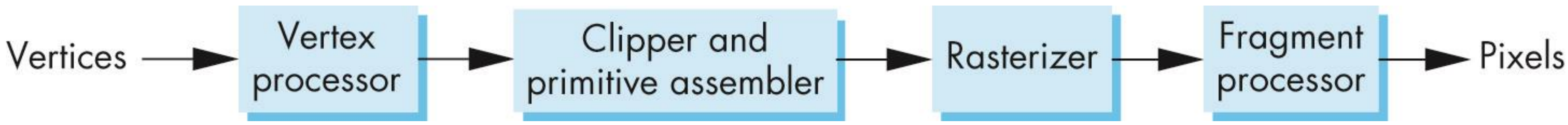
clip: part of the shape out of the screen ? primitive assembler: let the computer know it is dealing with a triangle for example

rasterizer: turn on the pixel with certain color inside the vertex? (the shape form by vertex

texture mapping? in fragment processor, deal with each pixel

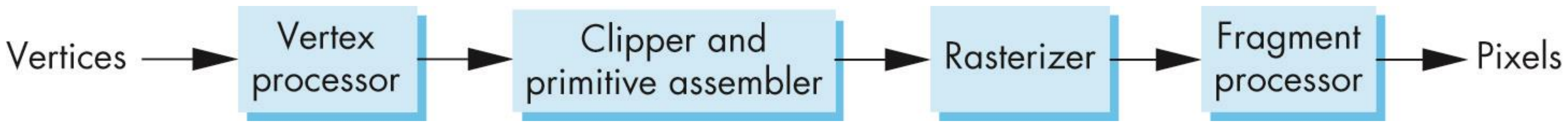
Projection

- *Projection* is the process that combines the 3D viewer with the 3D objects to produce the 2D image
  - **Perspective projections**: all projectors meet at the center of projection
  - **Parallel projection**: projectors are parallel, center of projection is replaced by a direction of projection



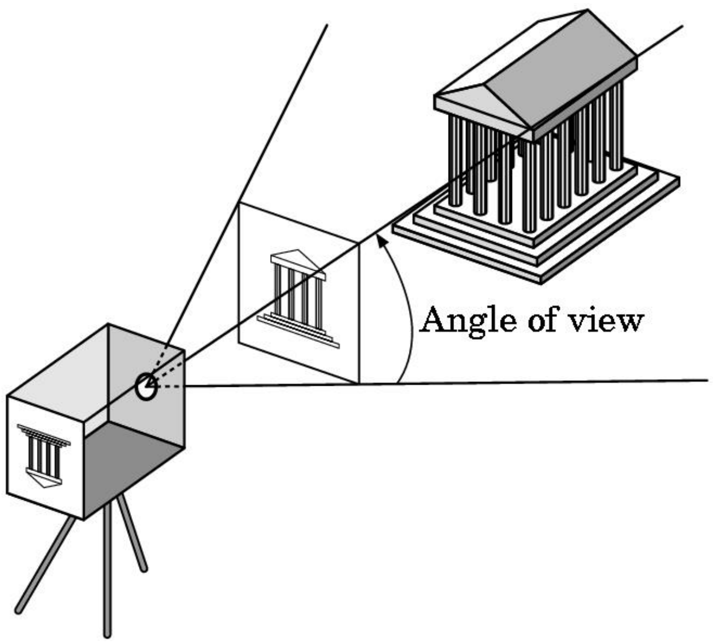
Primitive Assembly

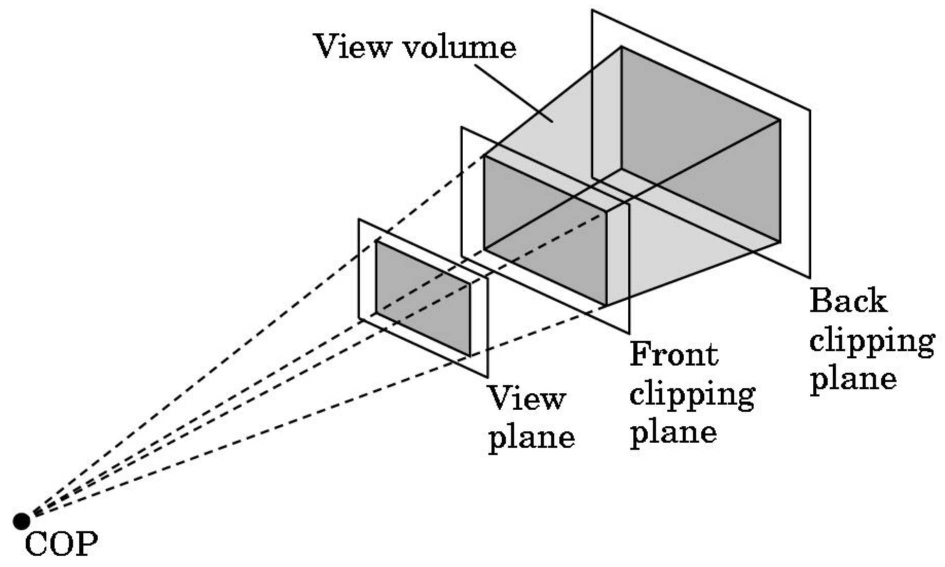
- Vertices must be collected into geometric objects/primitives before clipping and rasterization can take place
  - Line segments
  - Polygons
  - Curves and surfaces



Clipping

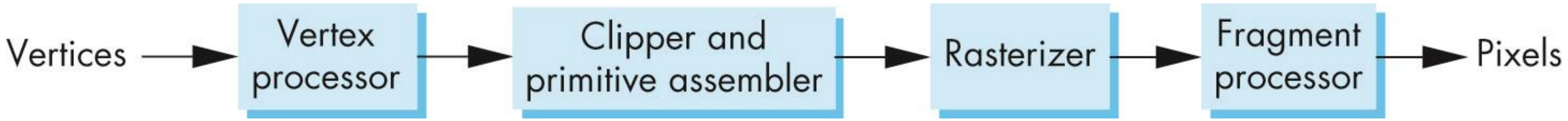
- Just as a real camera cannot "see" the whole world, the virtual camera can only see part of the world or object space
  - Objects that are not within this volume are said to be *clipped* out of the scene





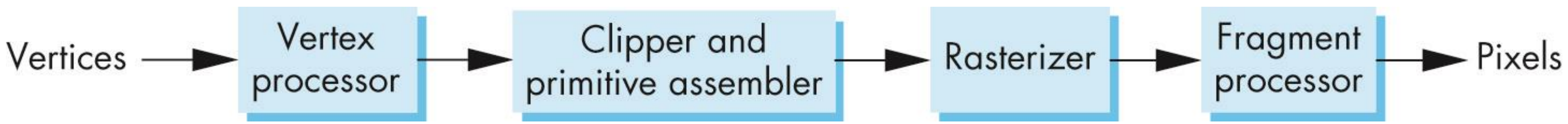
Rasterization (Scan Conversion)

- If an object/primitive is not clipped out, the appropriate pixels in the frame buffer must be assigned colors
- Rasterizer produces a set of fragments for each object
- *Fragments* are “potential pixels”
  - Have a location in frame bufffer
  - Color and depth attributes
- Vertex attributes are interpolated over objects by the rasterizer



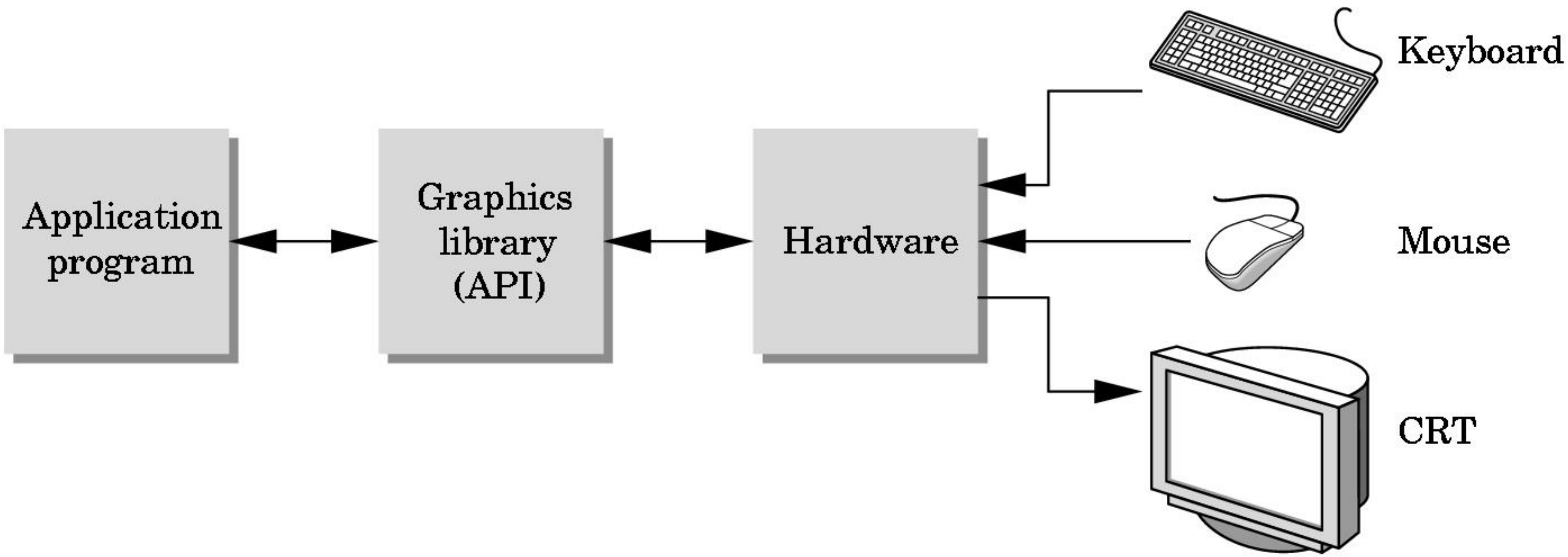
Fragment Processing

- Fragments are processed to determine the color of the corresponding pixel in the frame buffer
- Colors can be determined by ***texture mapping*** or interpolation of vertex colors
- Fragments may be blocked by other fragments closer to the camera
  - Hidden-surface removal



The Programmer's Interface

- Programmer sees the graphics system through a software interface: the Application Programmer Interface (API)



API Contents

- Functions that specify what we need to form an image
  - Objects
  - Viewer
  - Light Source(s)
  - Materials
- Other information
  - Input from devices such as mouse and keyboard
  - Capabilities of system

Object Specifications

- Most APIs support a limited set of primitives including
  - Points (0D object)
  - Line segments (1D objects)
  - Polygons (2D objects)
  - Some curves and surfaces
    - Quadrics
    - Parametric polynomials

- All are defined through locations in space or ***vertices***

Example (using OpenGL)

type of object

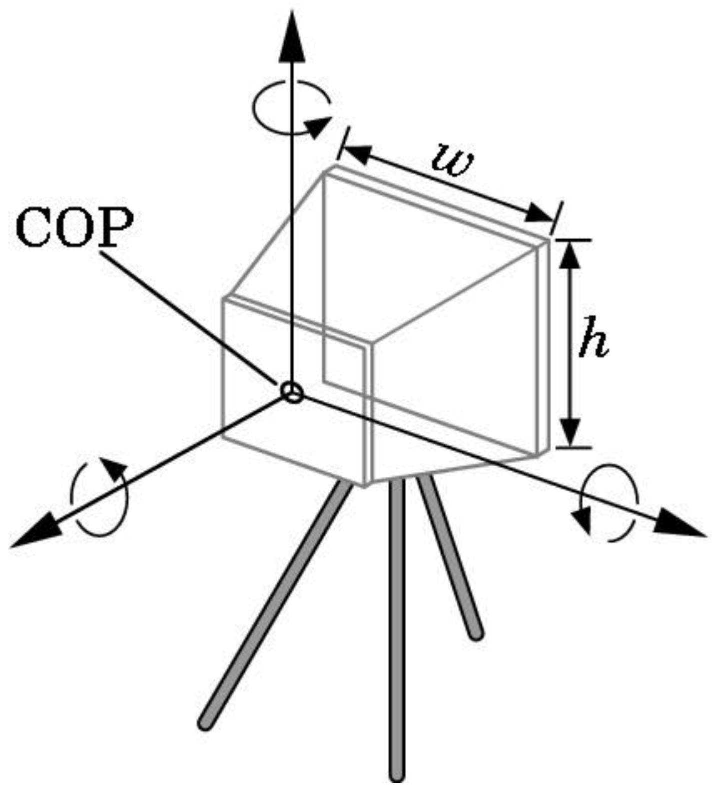
location of vertex

```
glBegin(GL_POLYGON)
  glVertex3f(0.0, 0.0, 0.0);
  glVertex3f(0.0, 1.0, 0.0);
  glVertex3f(0.0, 0.0, 1.0);
glEnd( );
```

end of object definition

Camera Specification

- Six degrees of freedom
  - Position of center of lens
  - Orientation
- Lens
- Film size
- Orientation of film plane



Lights and Materials

- Types of lights
  - Point sources vs distributed sources
  - Spot lights
  - Near and far sources
  - Color properties

- Material properties
  - Absorption: color properties
  - Scattering
    - Diffuse
    - Specular

End of Lecture 1