Computer Graphics System

Teacher: A.prof. Chengying Gao(高成英)

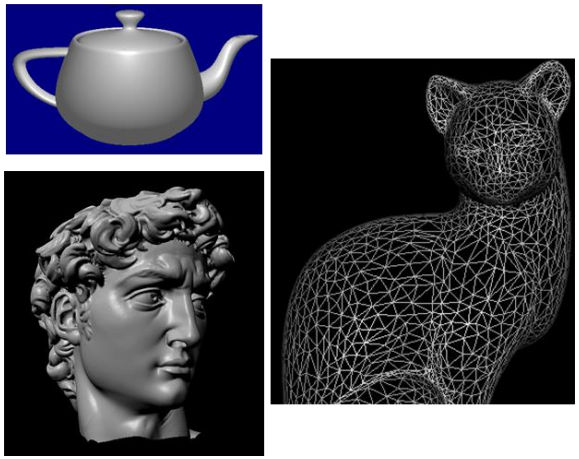
E-mail: mcsgcy@mail.sysu.edu.cn

School of Data and Computer Science

Outline

- **Computer Graphics System**
- Physical Imaging System
- Graphics Rendering Pipeline

Modeling
(Creating 3D Geometry)



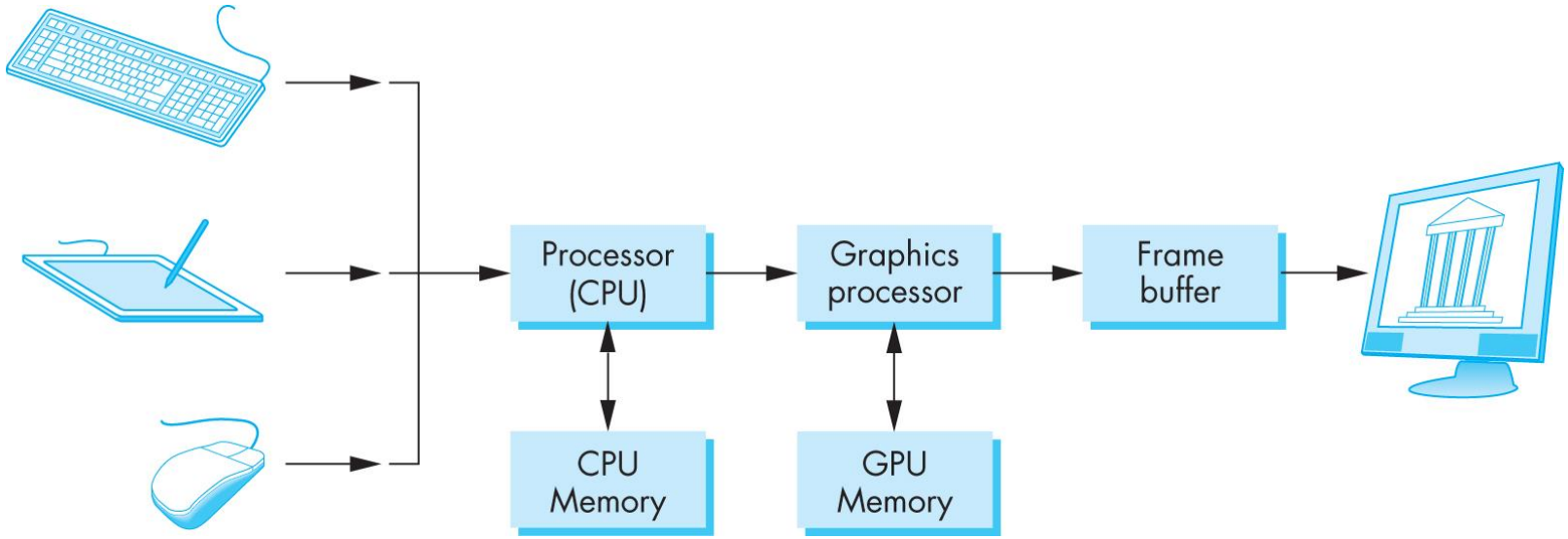
→

Rendering
(Creating, shading images from geometry, lighting, materials)

A computer graphics system

Current graphics systems consist of:

- An application, which talks to a…
- Graphics library (e.g., OpenGL or Direct3D), which talks to the…
- Graphics hardware ： Graphics hardware can do a lot of fancy work these days.

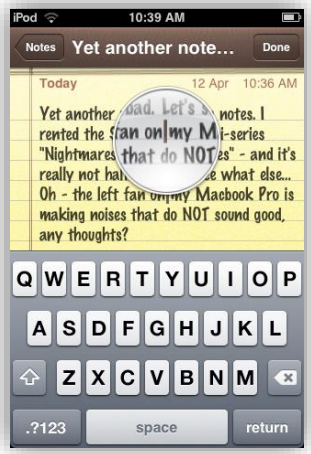


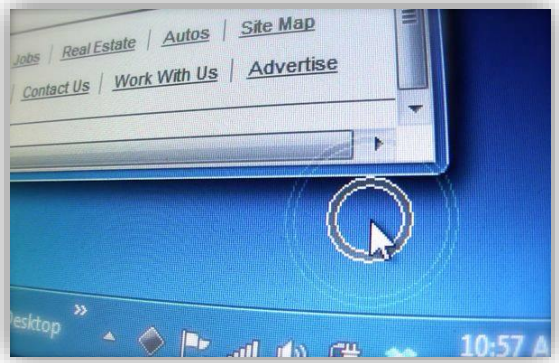
Input devices　　Image formed in frame buffer　　Output device

Input: Consumer devices

GRAPHICS

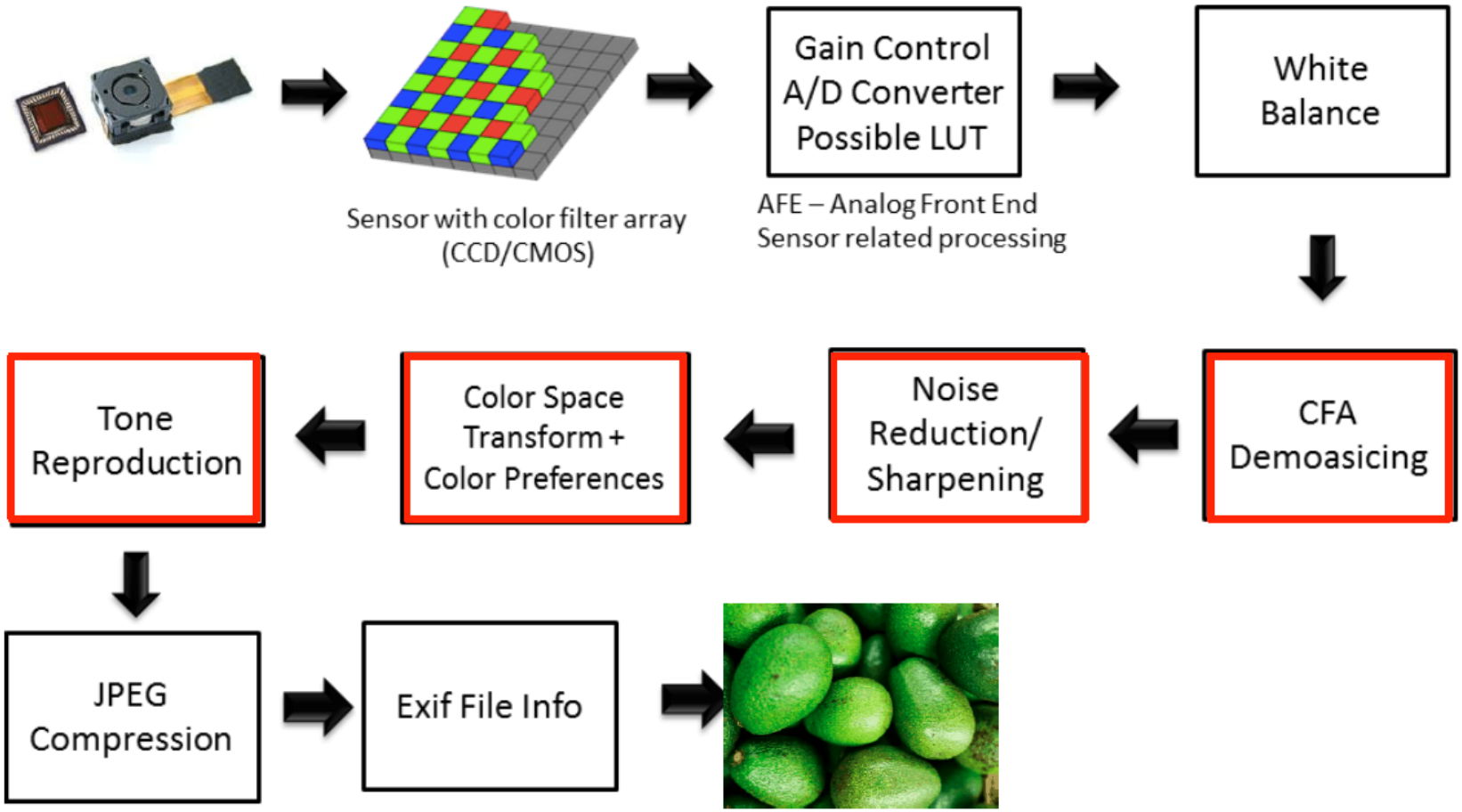
GRAPHICS





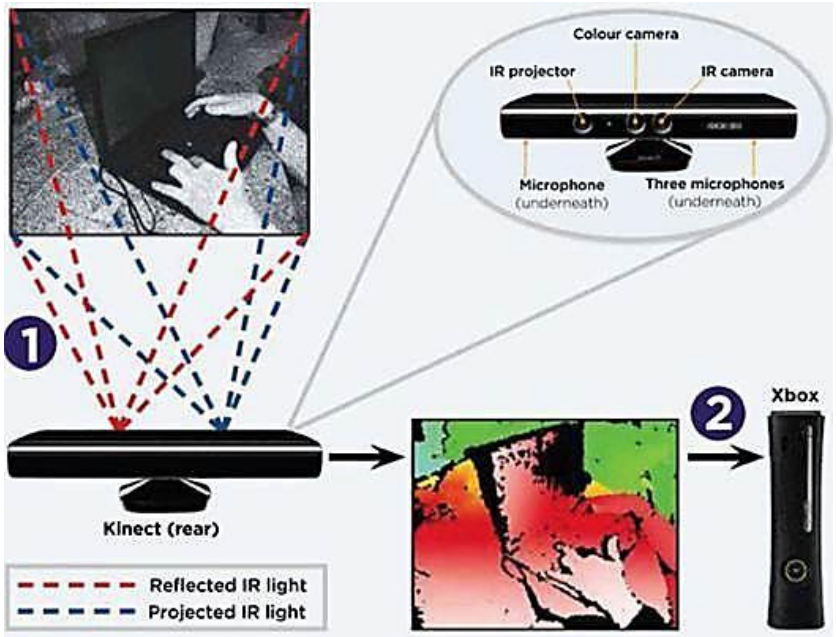
Input: Image & video

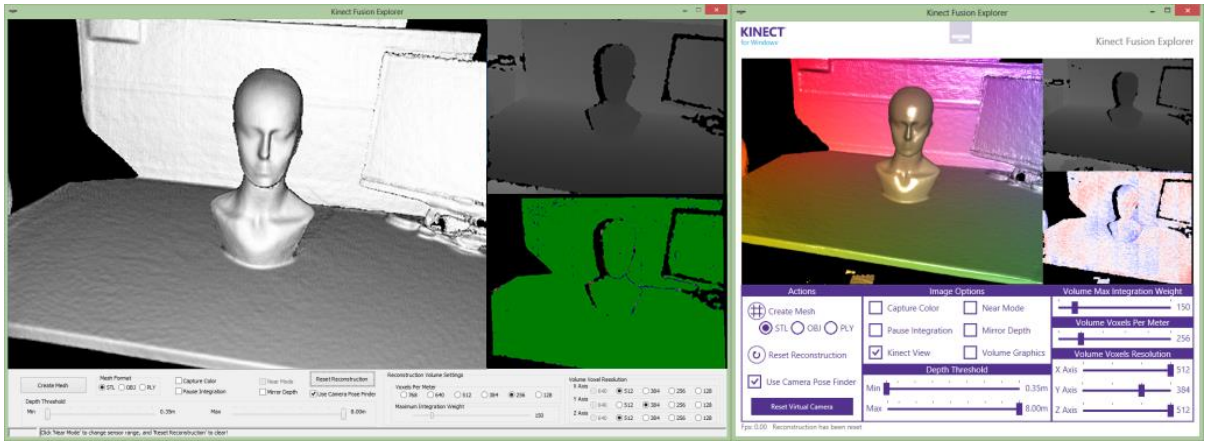
A standard model for imaging pipeline



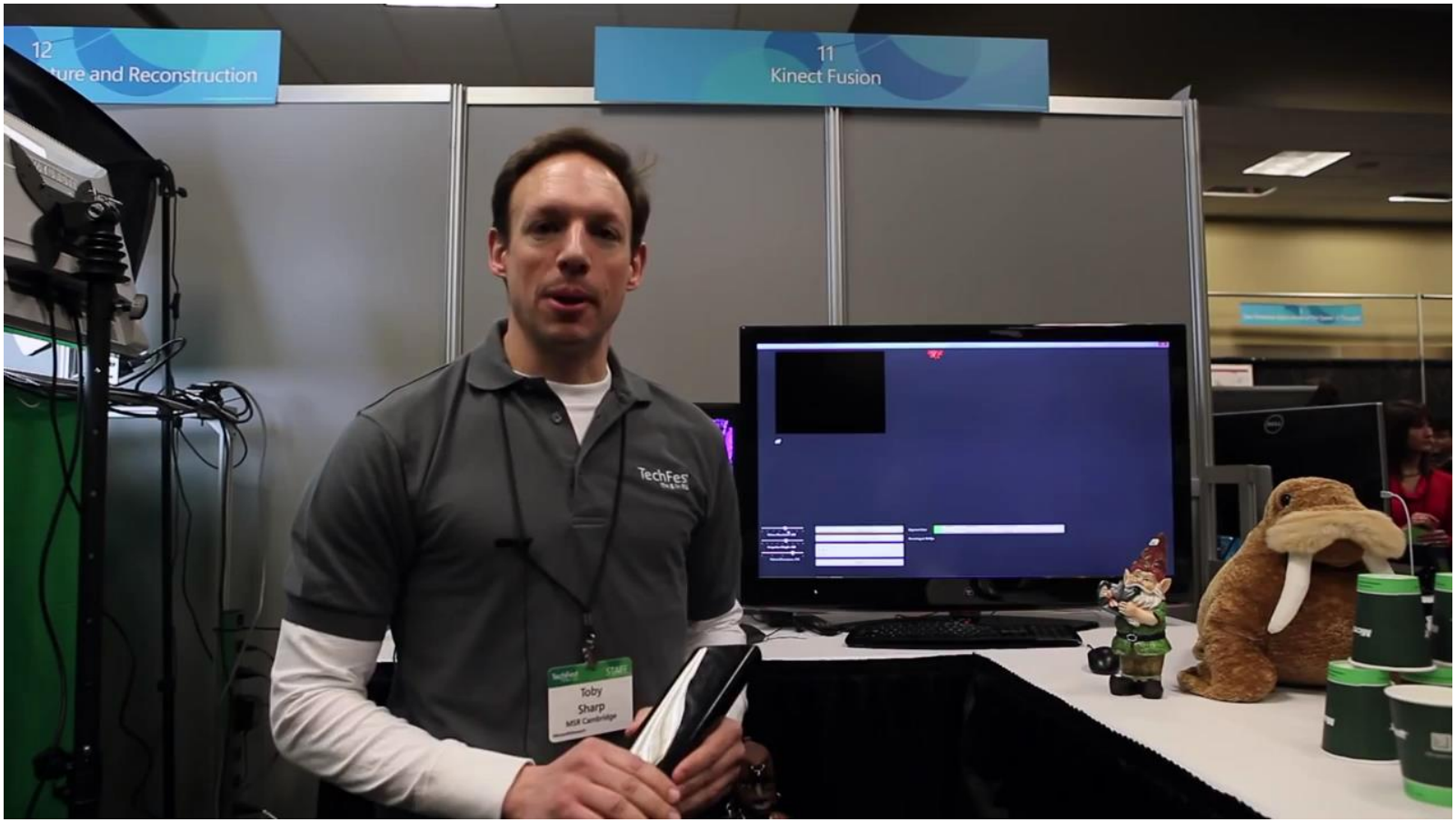
Input: Shape

Kinect

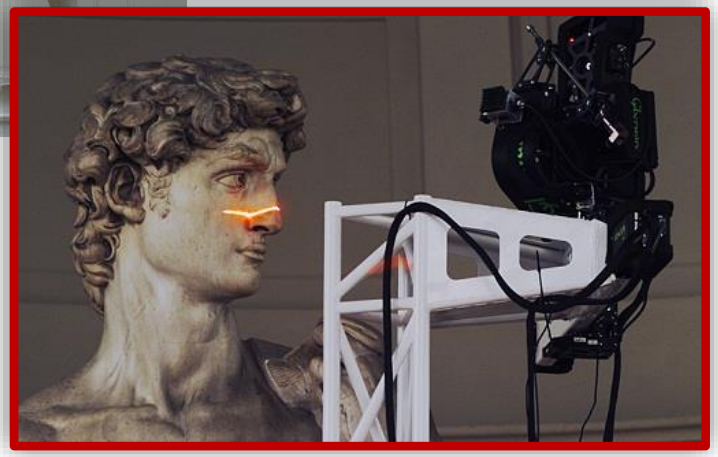

Colour camera
IR projector
IR camera
Microphone
(underneath)
Three microphones
(underneath)
1
2
Xbox
Kinect (rear)
Reflected IR light
Projected IR light

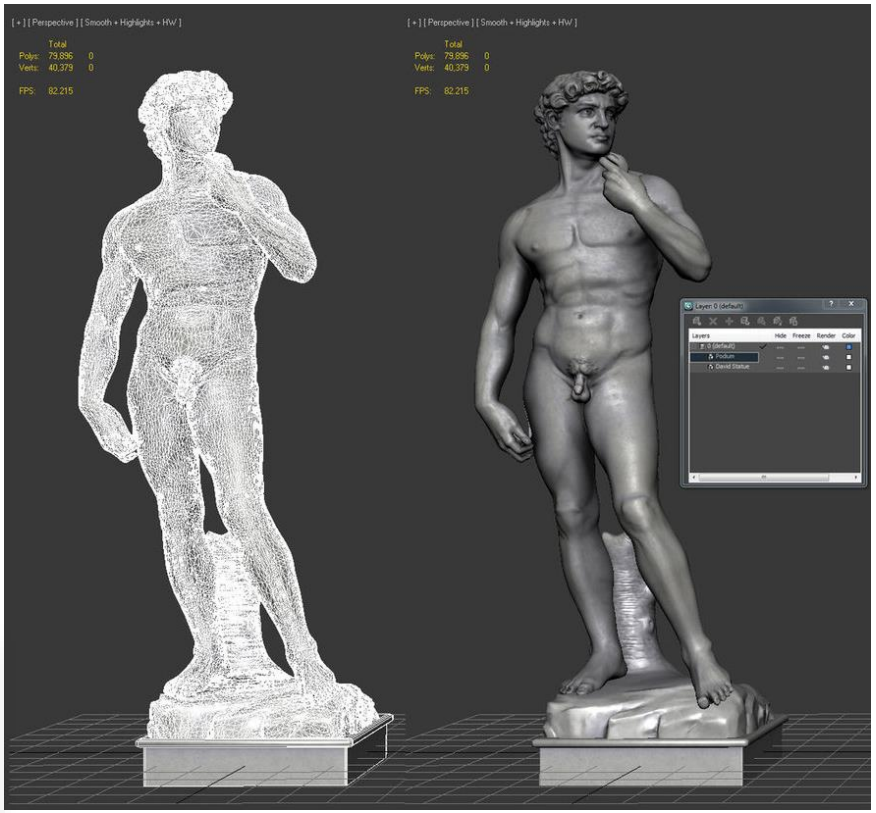

Kinect Fusion Explorer
KINECT
Kinect Fusion Explorer
Actions
Image Options
Volume Max Integration Weight
Create Mesh
Capture Color
Near Mode
Pause Integration
Mirror Depth
Volume Voxels Per Meter
Reset Reconstruction
Kinect View
Volume Graphics
Volume Voxels Resolution
Use Camera Pose Finder
Depth Threshold
Reset Virtual Camera
X Axis
Y Axis
Z Axis

Shapes from Kinect Fusion



3D Scanning




[+] [Perspective] [Smooth + Highlights + HW]
Total
Polys: 79,896 0
Verts: 40,379 0
FPS: 82.215
[+] [Perspective] [Smooth + Highlights + HW]
Total
Polys: 79,896 0
Verts: 40,379 0
FPS: 82.215

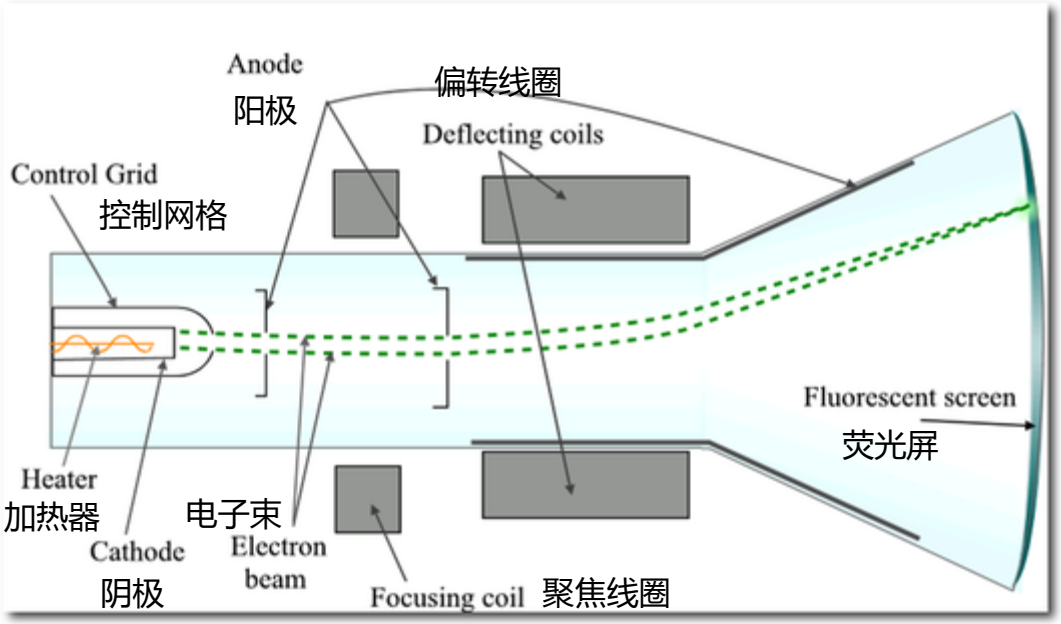
Desktop 3D scanner

Input: Specialized devices

What about the output device?

Display Devices

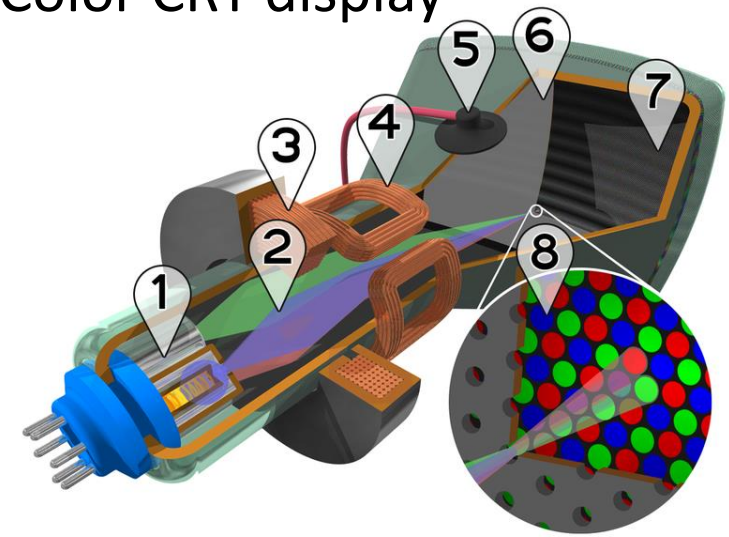
- Cathode Ray Tube (CRT)
  - CRT is a vacuum tube (电子管) containing one or more electron guns, and a phosphorescent screen (磷光屏) used to view images.

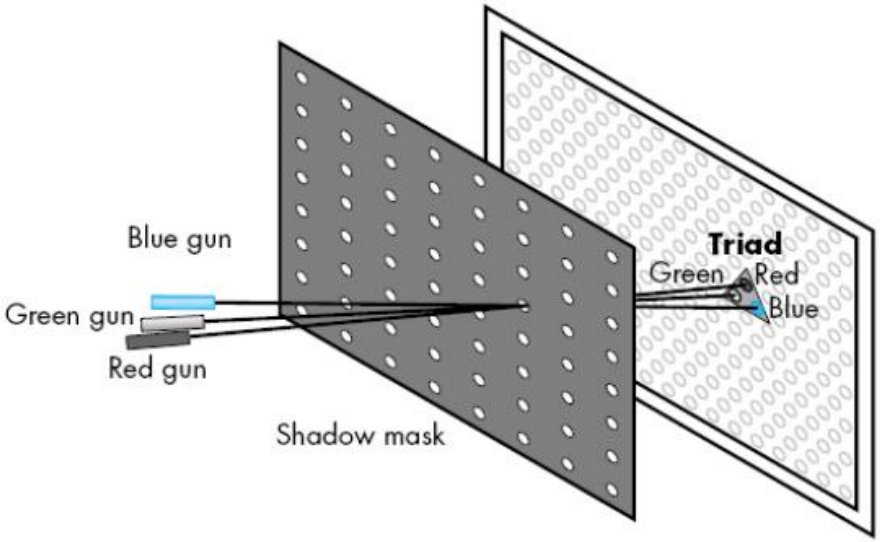


Karl Ferdinand Braun, 1897
Image courtesy of Wikipedia

Display Devices

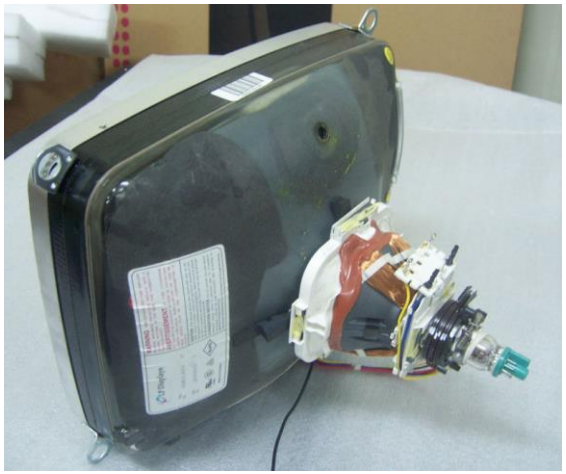
- Color CRT display





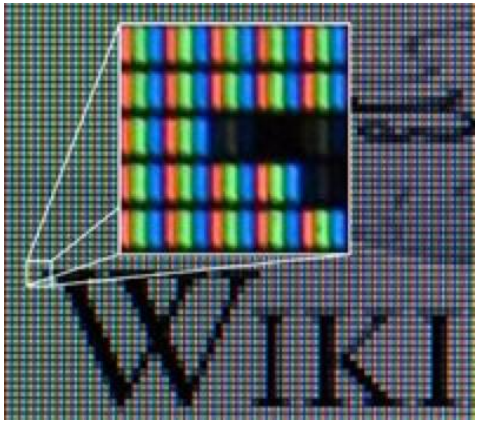
Cutaway rendering of a color CRT:

1. Three electron guns (for red, green, and blue phosphor dots)
2. Electron beams
3. Focusing coils
4. Deflection coils
5. Anode connection
6. Mask for separating beams for red, green, and blue part of displayed image
7. Phosphor layer with red, green, and blue zones
8. Close-up of the phosphor-coated inner side of the screen



Display Devices

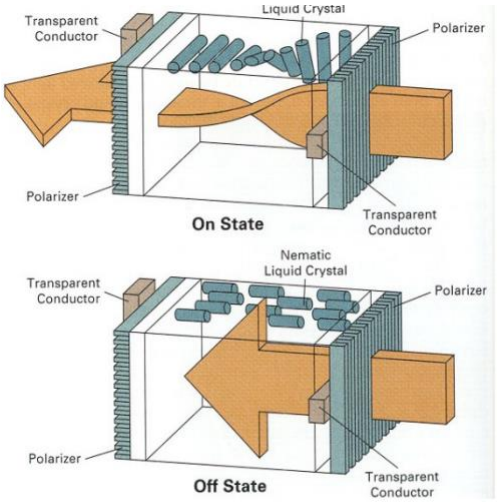
- **Flat-panel monitors**
  - LED (light-emitting diodes): 发光二级管
  - LCD (liquid-crystal displays): 液晶显示器
  - Plasma panels: 等离子显示器
- **Advantages:**
  - low consumption
  - small radiation
  - flicker free
  - No geometric distortion

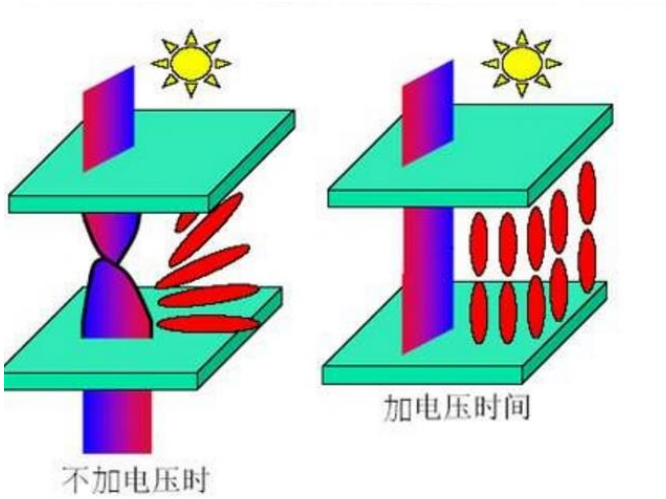


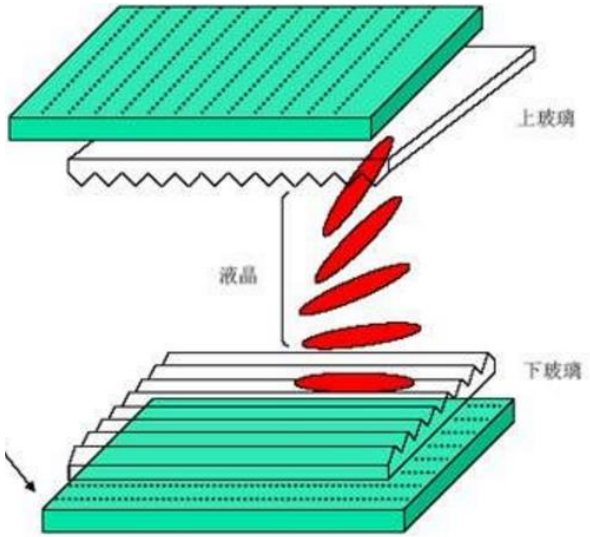
Close-up of pixels on an LCD display

Liquid-crystal displays

- Laptops typically use **liquid crystal displays**
  - Light enters a vertical polarizer
  - Nematic crystal twists light based on applied voltage(more voltage, less twisting)
  - Light passes through horizontal polarizer



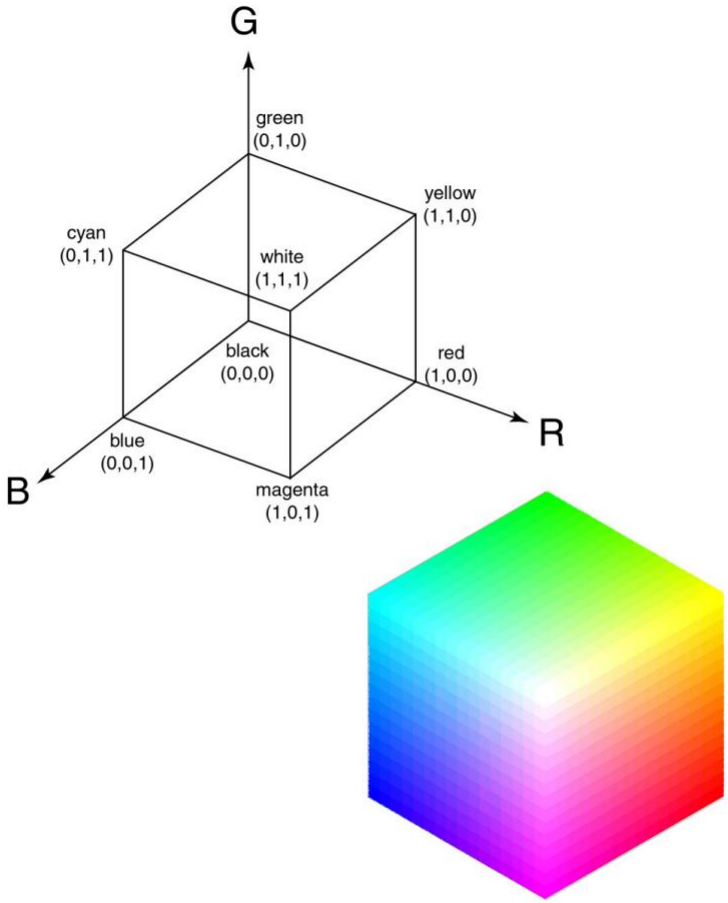




[Hearn and Baker, 2004]

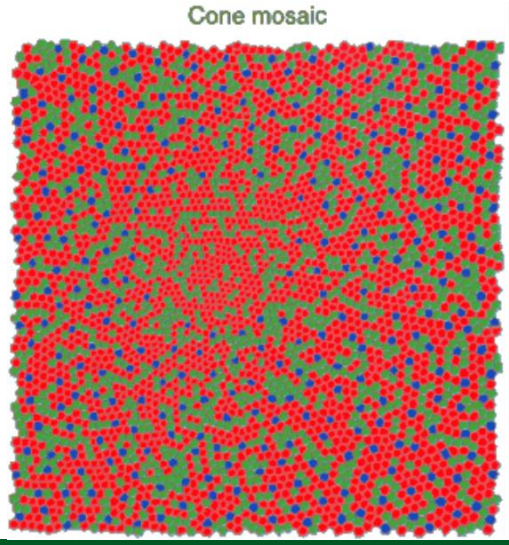
Adding color mixing

- All colors on a display are produced using combinations of R,G, and B.



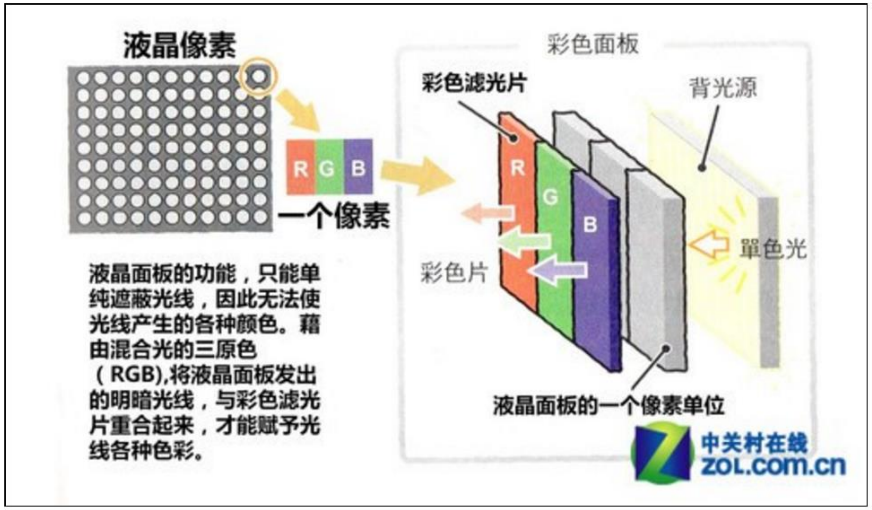
RGB Color Space

- RGB color space is commonly used color space in Graphics
  - Color is represented as RGB triple(r, g, b);
  - People perform almost all operations on each channel separately
  - Usually, each color value is a float value range from 0 to 1, or 0-255 if we use 8 bit integers.
- Color are represented as combinations of three primaries: red, green, blue)
  - C = rR +gG + bB
  - The reason why we pick red, green, and blue?
    - Essentially because of the structure of our visual system
    - Our visual system is sensitive to those three colors



LCD Color

- Color is obtained using color filters:

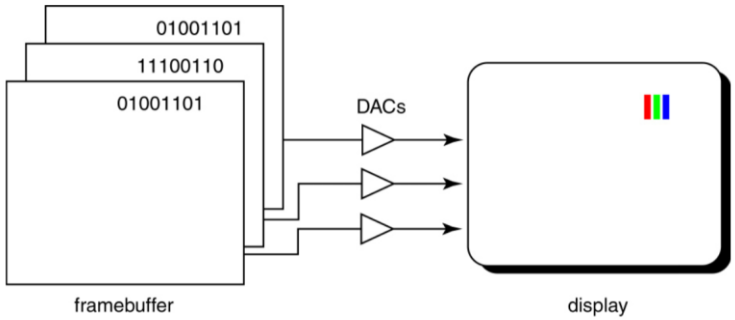


彩色液晶面板的基本原理

- Pixel is one region on the display corresponding to one color sample of an image being shown.
- Our eyes average the closely spaced RGB colors spatially to created the impression of a composite color at each pixel.

RGB Framebuffer

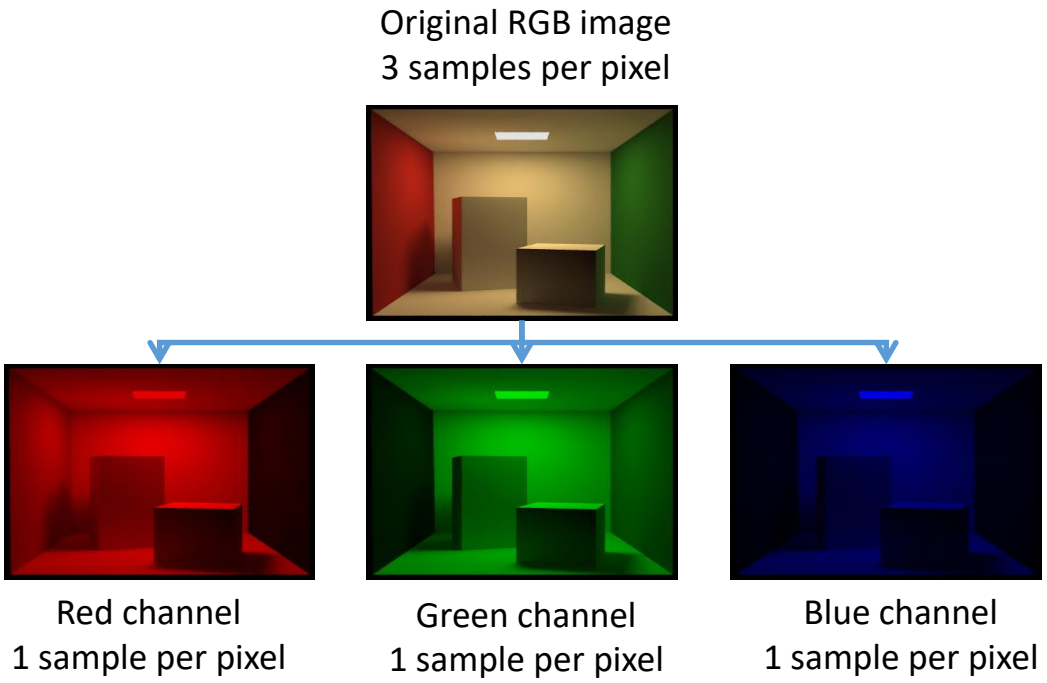
- The brightness of each LCD element is controlled by a dedicated memory array called a framebuffer(a buffer that stores the contents of an image pixel by pixel).
- Each element of the framebuffer is associated with a single pixel on the screen.



- Typicall, a display allows 256 voltage settings for each of R,G, and B.
- We sometimes call each R,G, and B component a **channel** (so the "red channel" of an image is only the R component per pixel.)

What is a channel?

- RGB is a common format for image channels
  - Corresponds approximately to human visual system anatomy (specialized “R, G, and B” cones)
  - Samples represent the intensity of the light at a point for a given wavelength (red, green, or blue)
- The R channel of an image is an image containing just the red samples.
- Gray images is one single channel.



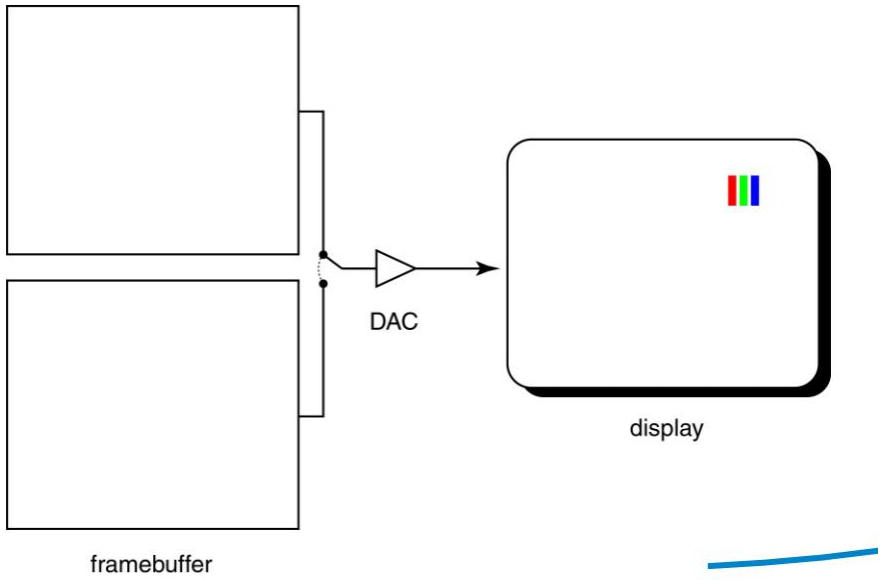
The alpha channel

- In addition to the R, G, and B channels of an image, add a fourth channel called α (transparency/opacity/translucency)
- Alpha varies between 0 and 1
  - Value of 1 represents a completely opaque pixel, one you cannot see through
  - Value of 0 is a completely transparent pixel
  - Value between 0< α < 1 determines translucency
- Useful for blending images
  - Images with higher alpha values are less transparent
  - Linear interpolation (αX + (1- α)Y) or full Porter-Duff compositing algebra

The orange box is drawn on top of the purple box using $\alpha = 0.8$

Double-buffering

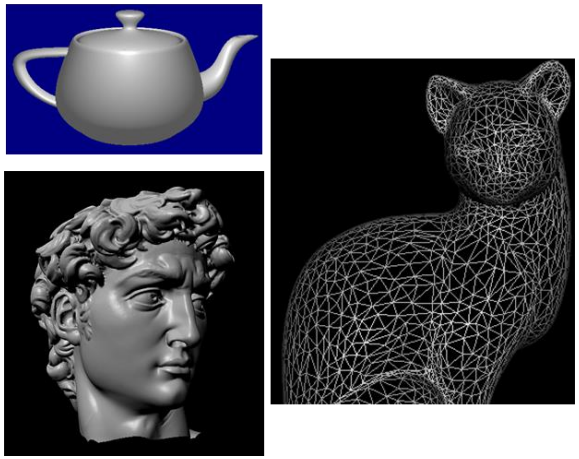
- What happens when you write to the framebuffer while it is being displayed on the monitor?
  - **Double-buffering** provides a solution.



Outline

- Computer Graphics System
- **Physical Imaging System**
- Graphics Rendering Pipeline

Modeling
(Creating 3D Geometry)

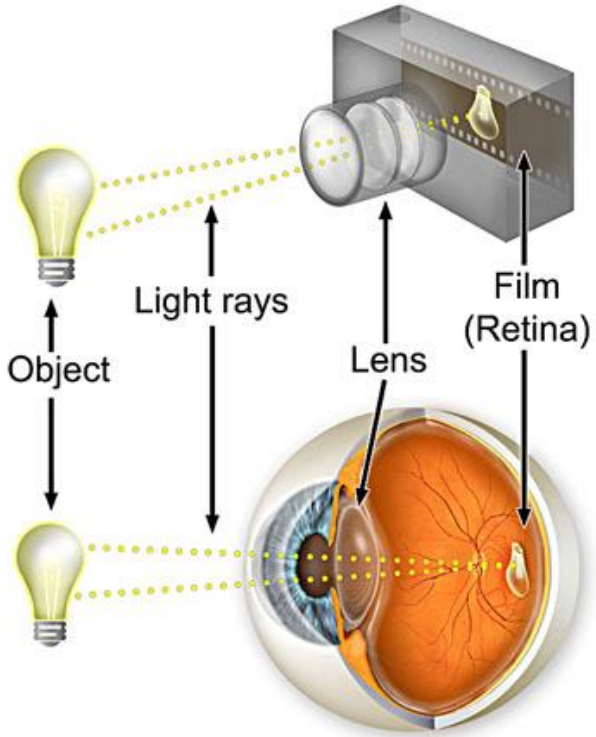


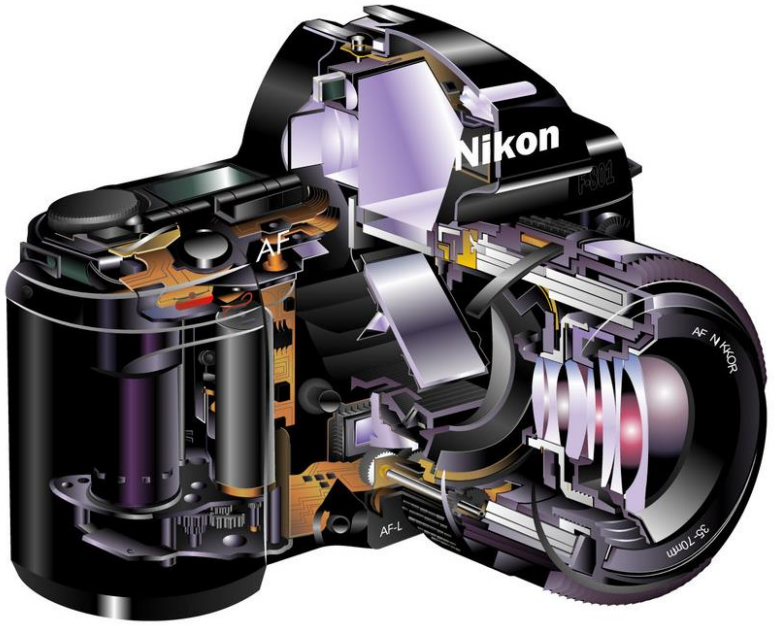
→

Rendering
(Creating, shading images from geometry, lighting, materials)

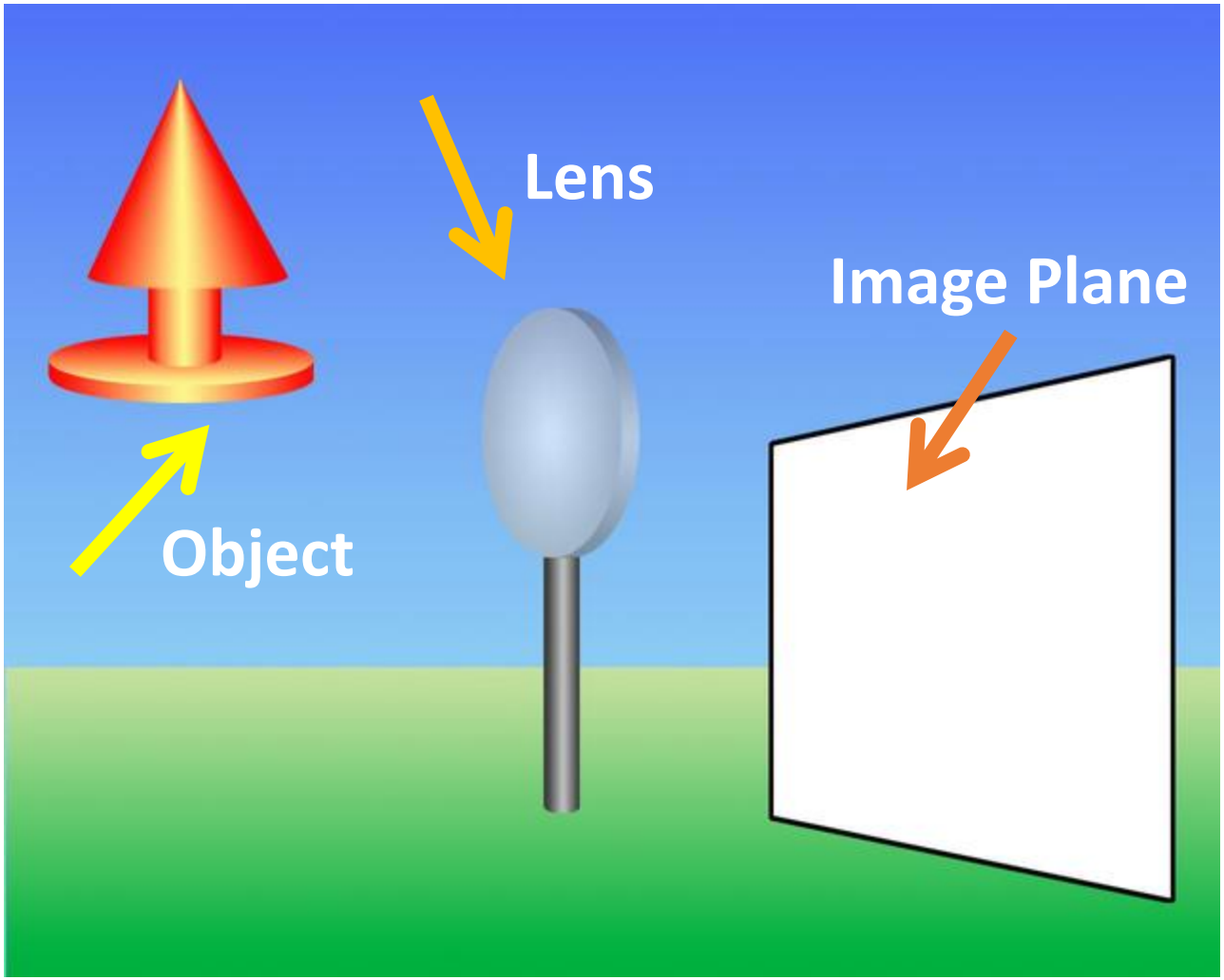
Physical Imaging System

- Eye(biology)
- Camera: film (chemistry), digital (physical + digital)

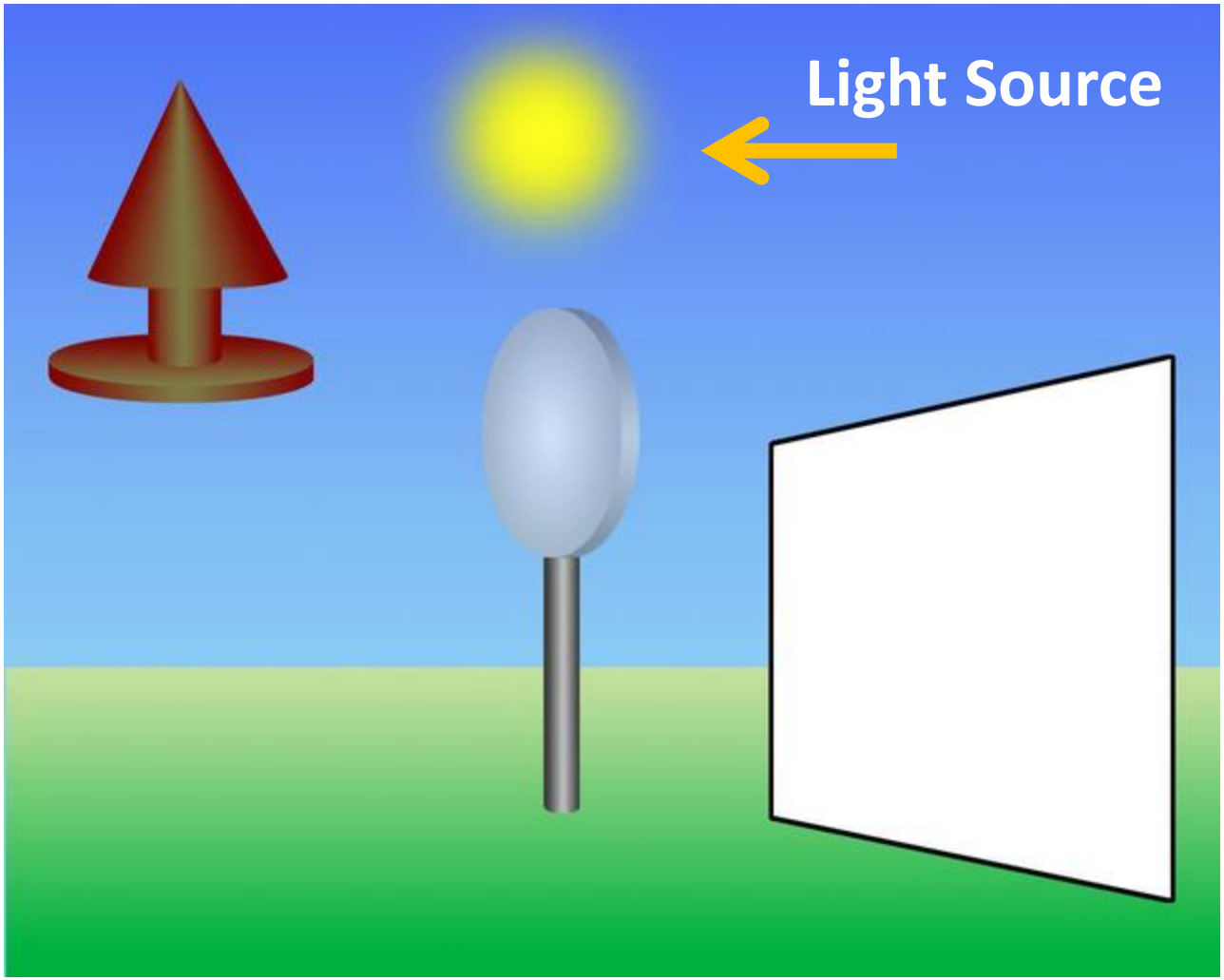




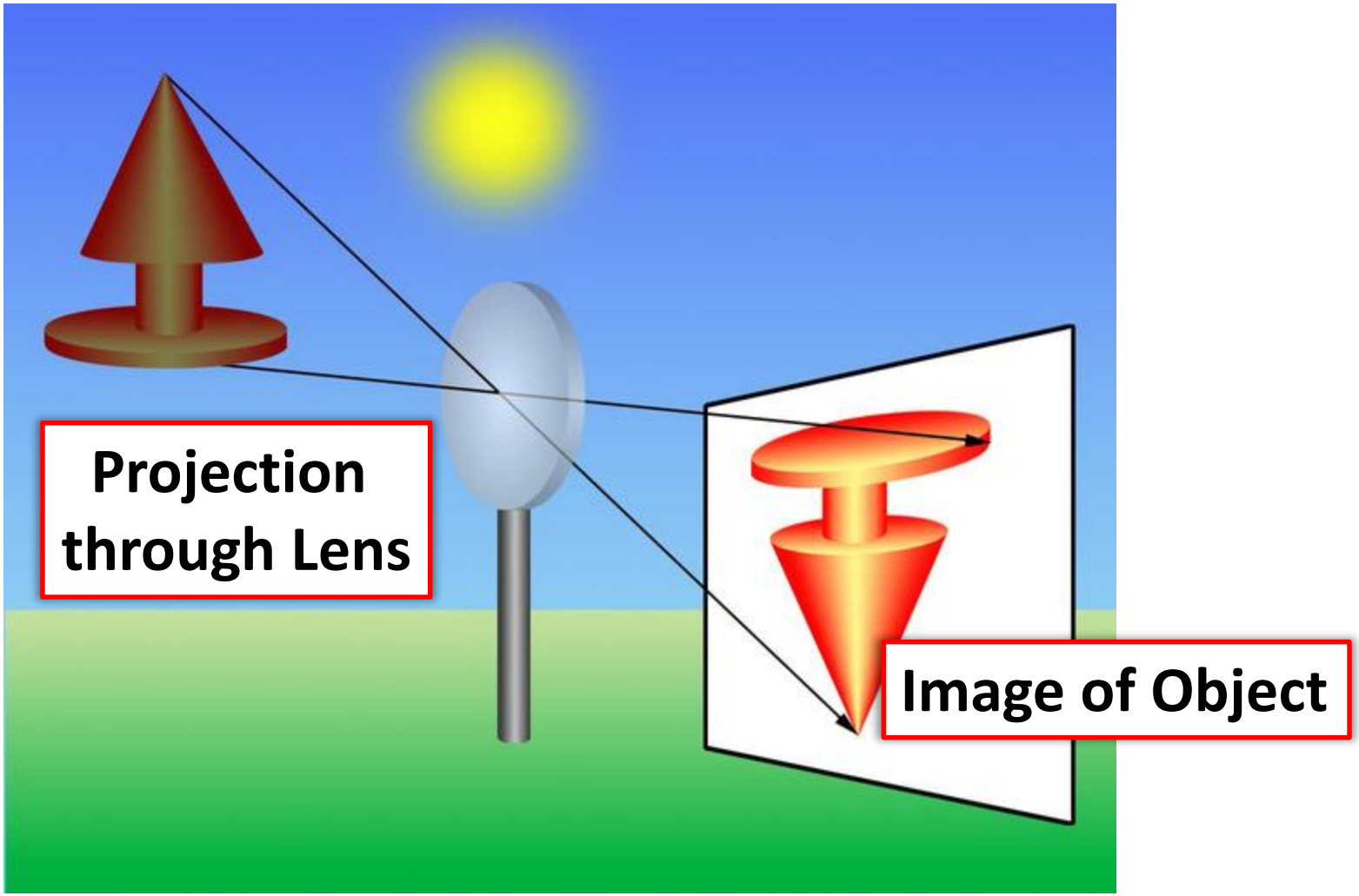
Imaging Principle



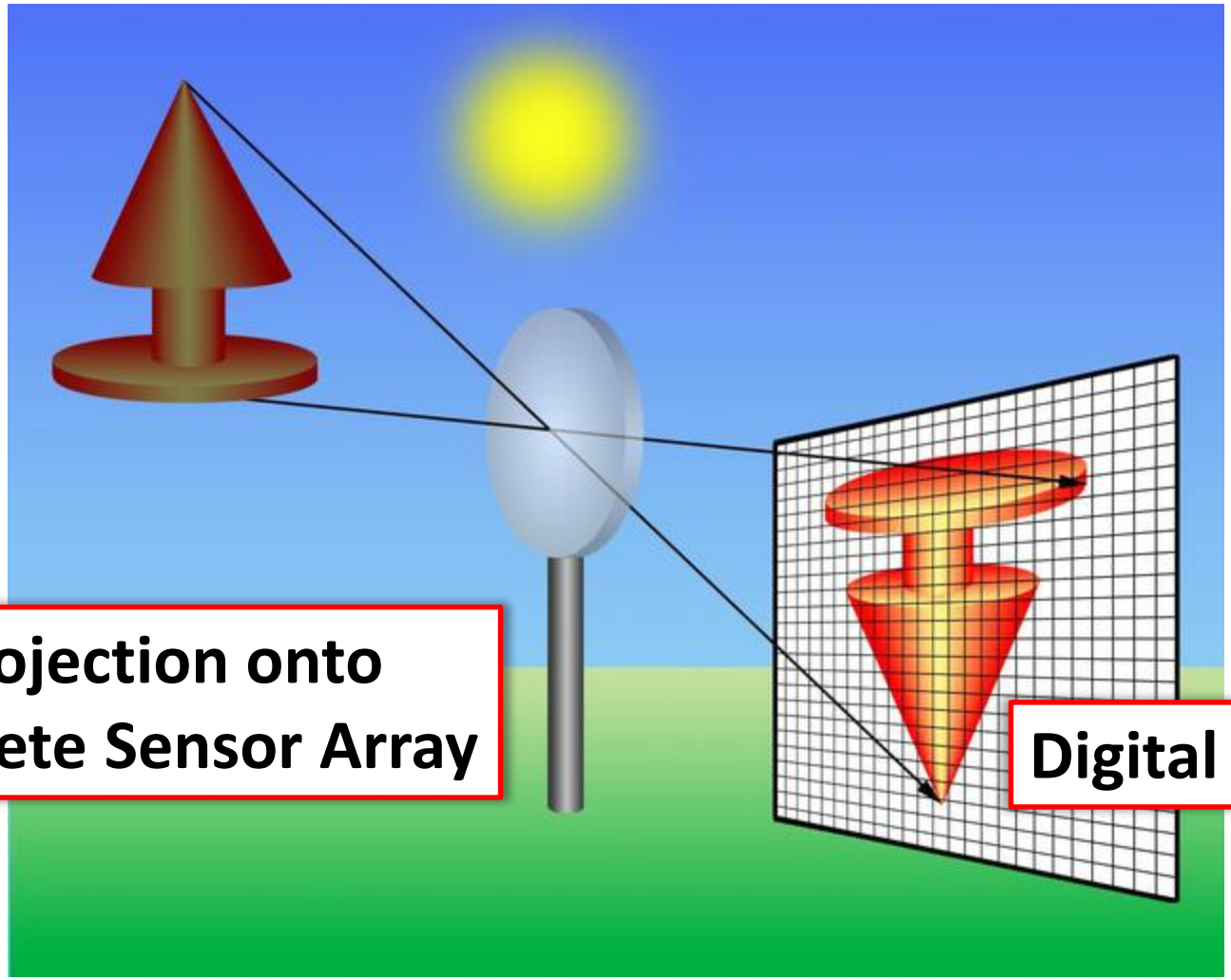
Imaging Principle



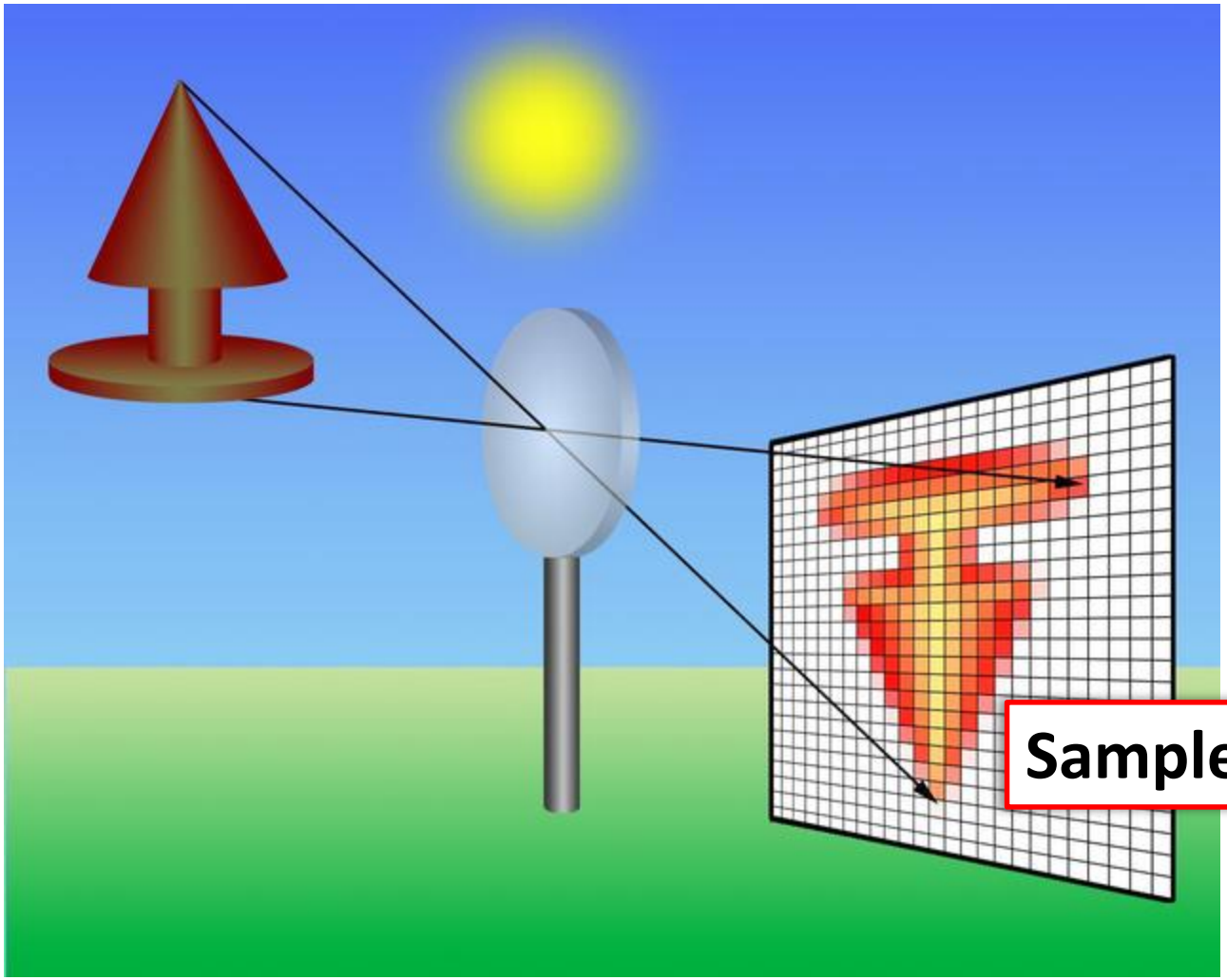
Imaging Principle



Imaging Principle



Imaging Principle



What is an image?

- We can think of an image as a function f, from R^2 to R:
  - f(x, y) gives the intensity of a channel at position(x, y)
  - Realistically, we expect the image only to be defined over a rectangle, with a finite range:
    - $f: [a, b] \times [c, d] \rightarrow [0,1]$

- A color image is just three functions pasted together. We can write this as a "vector-valued" function:

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

What is a digital image?

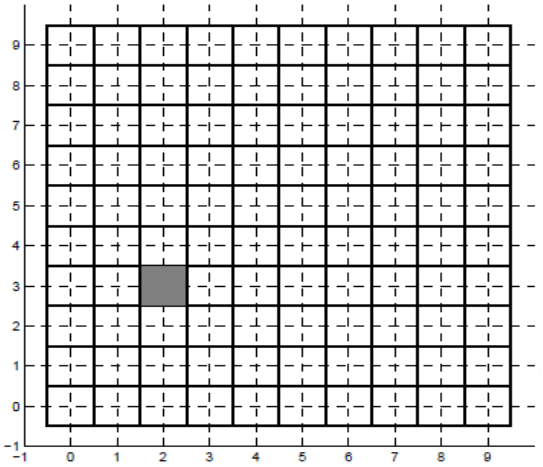
- In computer graphics, we usually operate on **digital**(**discrete**) images:
  - **Sample** the space on a regular grid
  - **Quantize** each sample(round to nearest integer)

- If our samples are Δ apart, we can write this as:

$$f[i,j] = \text{Quantize}\{ f(i\Delta, j\Delta) \}$$

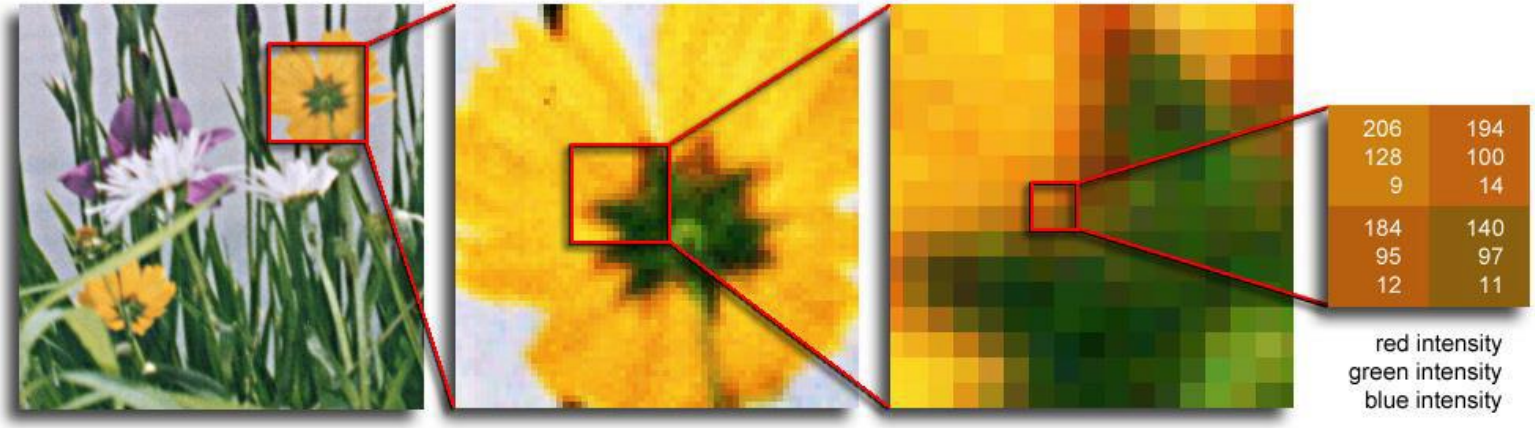
Modeling an image

- Model a one-channel $\mathrm{m} \times \mathrm{n}$ image as the function $u(i, j)$
  - Maps pairs of integers (pixel coordinates) to real numbers
  - i and j are integers such that $0 \leq i < \mathrm{m}$ and $0 \leq j < \mathrm{n}$
- Associate each pixel value $u(i, j)$ to small area around display location with coordinates (i, j)
- A pixel here looks like a square centered over the sample point, but it's just a scalar value and the actual geometry of its screen appearance varies by device
  - Roughly circular spot on CRT
  - Rectangular on LCD panel



Digital Image

Color Image



Grey Image

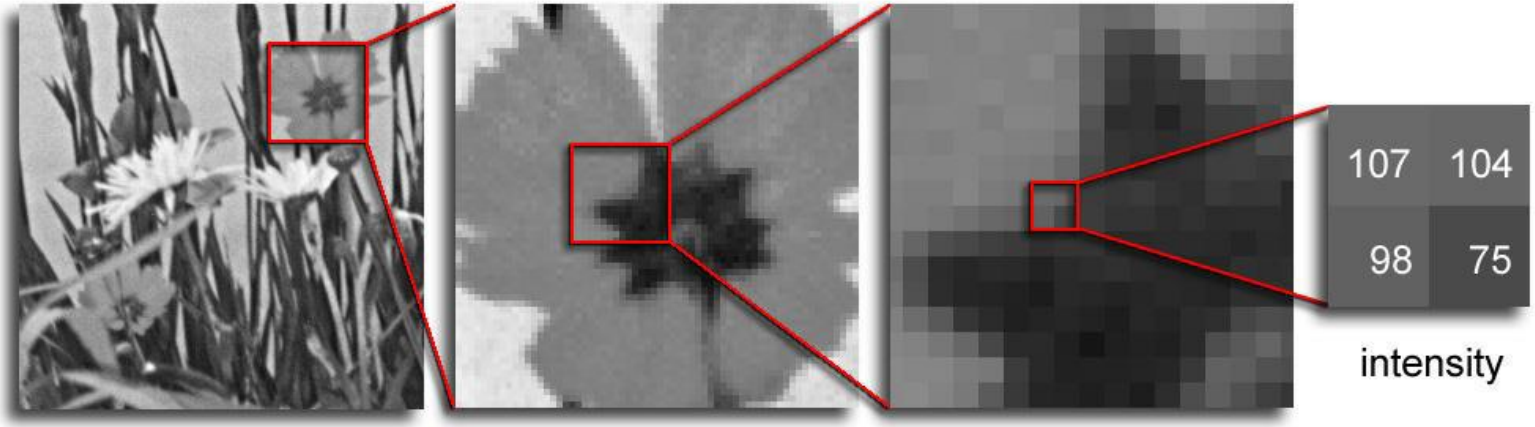
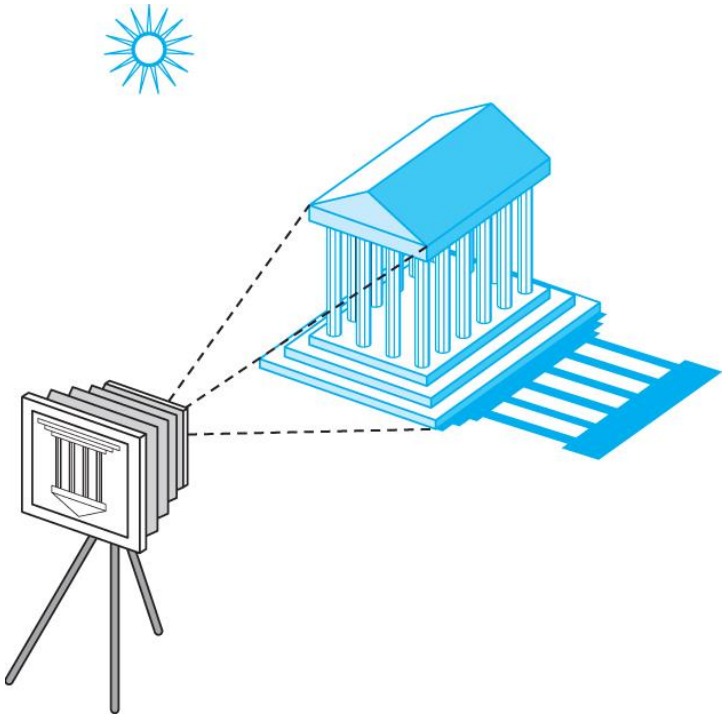


Image Synthetic Element

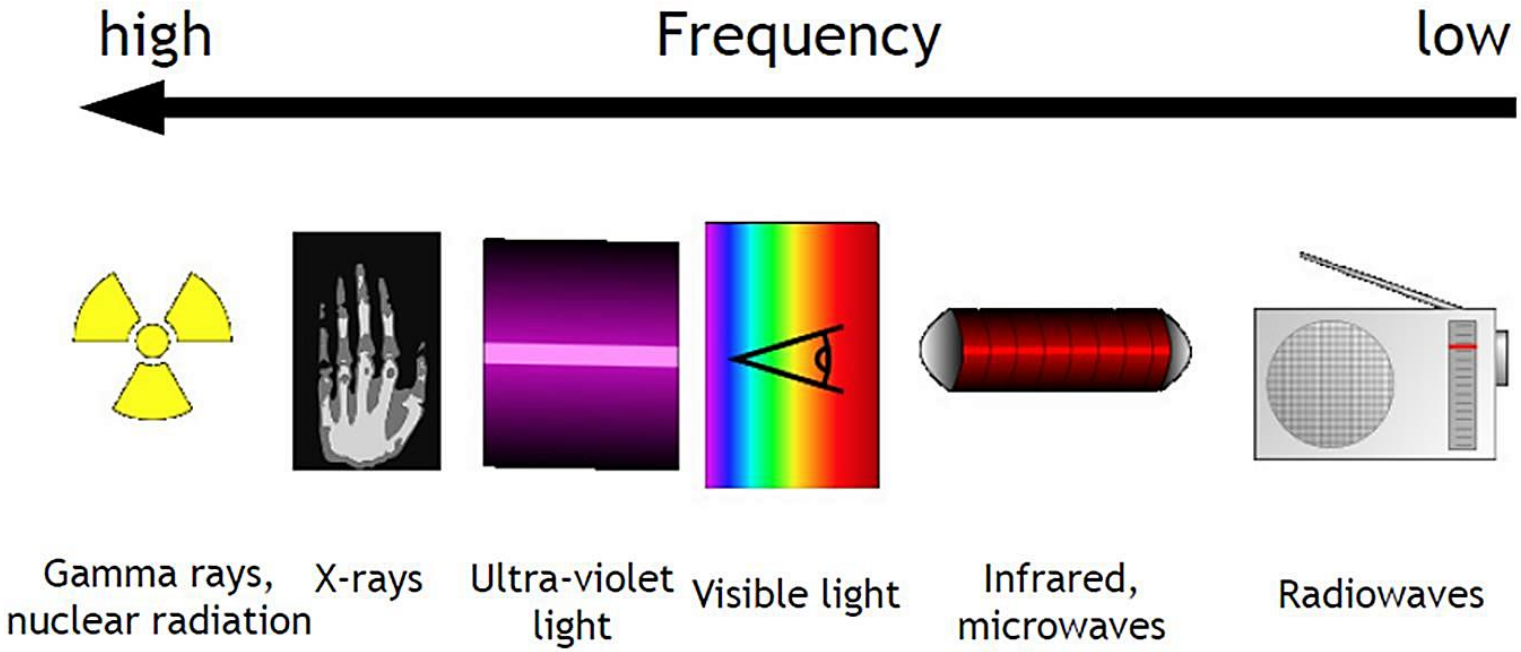
- Object
- Observer
- Light



- Properties of Light, material of object: To determine the effect of light on the object

Light

- Light is a part of electromagnetic wave, it is a wave.
- It' visible range: between 350nm and 780nm
  - Different frequencies



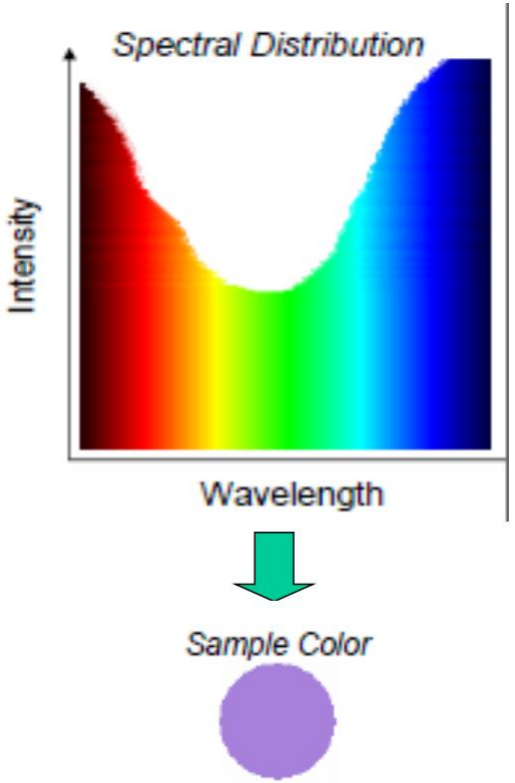
Color Perception

- What is color?
  - Colors are the sensations that arise from light energy of different wavelengths
  - Electromagnetic waves with different wavelengths are corresponding to different colors

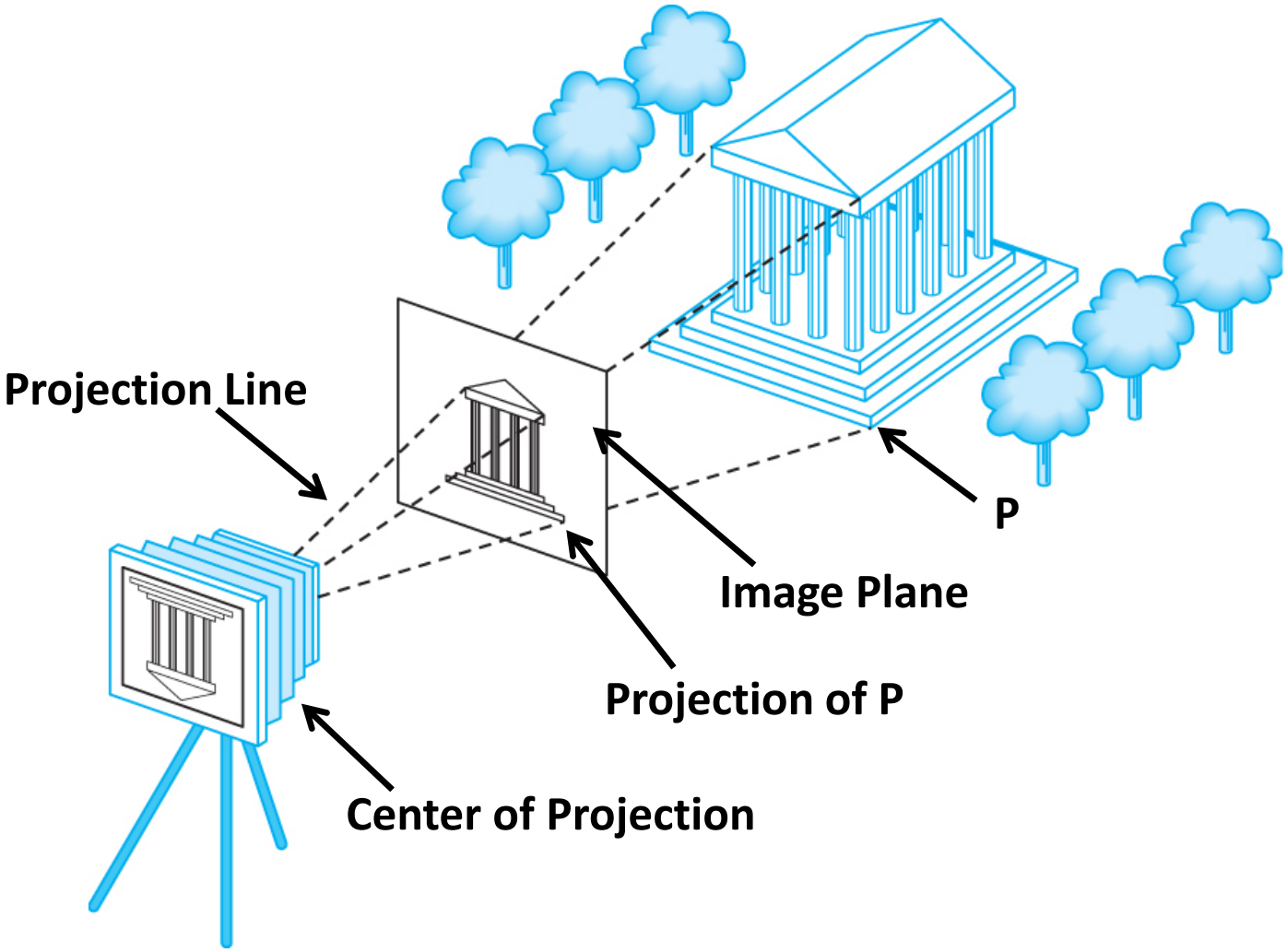
Spectral color	Wavelength range	Frequency
Red	ard 625—740 nm	ard 480—405 MHz
Orange	ard 590—625 nm	ard 510—480 MHz
Yellow	ard 565—570 nm	ard 530—510 MHz
Green	ard 500—565 nm	ard 600—530 MHz
Cyan	ard 485—500 nm	ard 620—600 MHz
Blue	ard 440—485 nm	ard 680—620 MHz
Violet	ard 380—440 nm	ard 790—680 MHz

Spectral Distribution of Light

- "Light" is a mixture of many wavelengths, each with some intensity
  - E.g., White Light means light include all wavelengths with the same intensity.
- Spectral distribution: intensity as a function of wavelength over the entire spectrum
- So, colors can be represented as such distribution function



Synthetic camera model



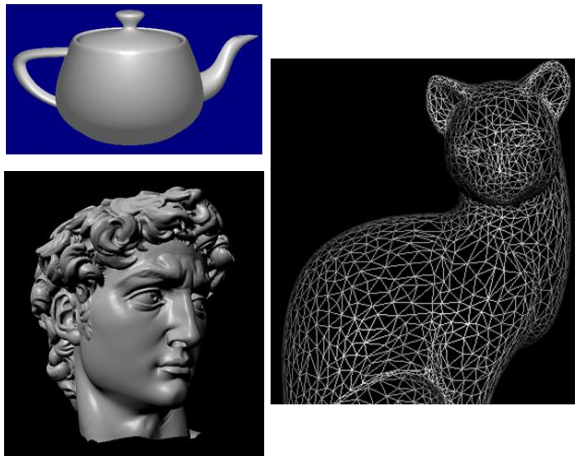
Advantages of Synthetic camera model

- Object (物体), Observer (观察者) and light (光源) is complete independent
- Easy to implement by API
  - Set the properties of light, camera and material.
  - To determine the results of image by API.
- Quick hardware implement is supported :**OpenGL**, Direct 3D etc. is based on this model.

Outline

- Computer Graphics System
- Physical Imaging System
- **Graphics Rendering Pipeline**

Modeling
(Creating 3D Geometry)

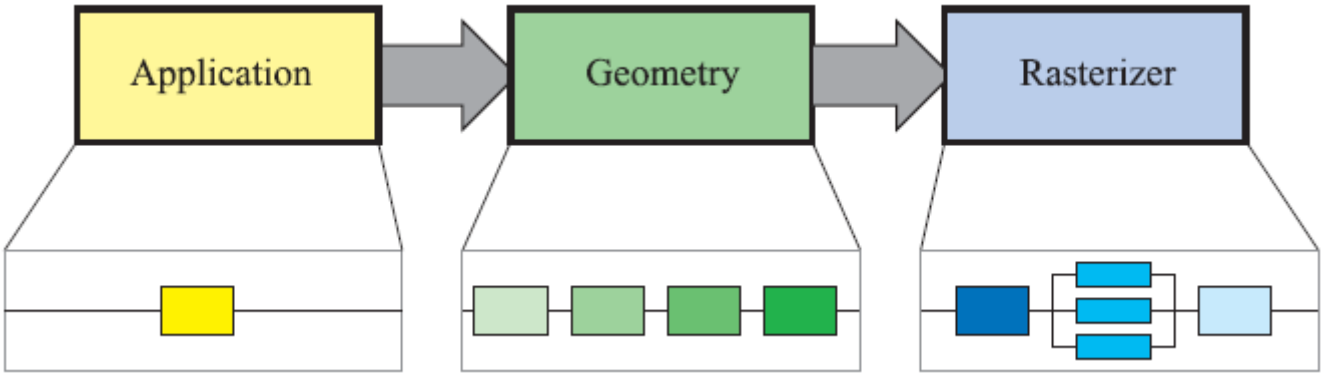


→

Rendering
(Creating, shading images from geometry, lighting, materials)

Graphics Rendering Pipeline

- What is the rendering process (Graphic Pipeline)?
  - the sequence of **steps** used to create a 2D raster representation of a 3D scene.



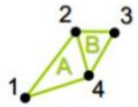
Rendering Pipeline Stages

Graphics Rendering Pipeline

渲染管线做的事情就是让计算机完成图形功能，而图形功能其实就是做了下面图里这件事情

有了光源、房子和相机，然后计算相机上面拍出的照片到底是什么样子的。这个任务比较复杂，然后大家就把它分成了几个步骤来完成

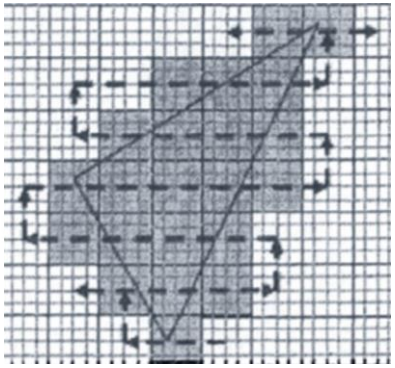
Graphics Rendering Pipeline



Vertex Array
(x1, y1, z1)
(x2, y2, z2)
…

Index Array
V1, V2, V4 – represents Triangle A
V4, V2, V3 – represents Triangle B

Vertices are only stored once.
Triangles point to their vertices.

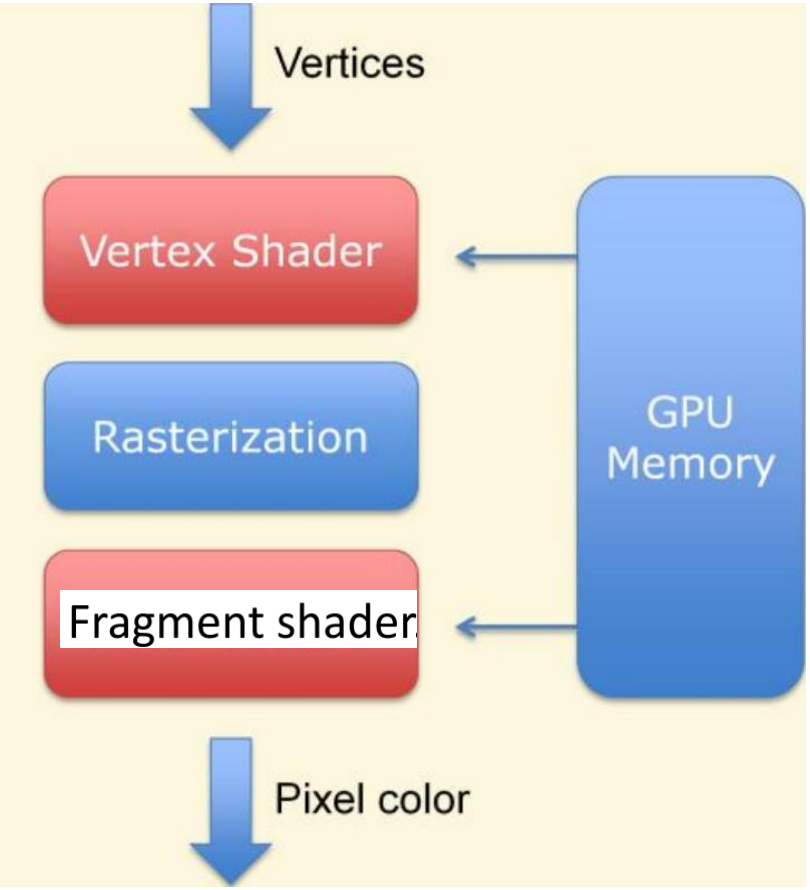


Fixed-function Pipeline


```
GLAPI void APIENTRY glLightModelf (GLenum pname, GLfloat param);
GLAPI void APIENTRY glLightModelfv (GLenum pname, const GLfloat *params);
GLAPI void APIENTRY glLightModelx (GLenum pname, GLfixed param);
GLAPI void APIENTRY glLightModelxv (GLenum pname, const GLfixed *params);
GLAPI void APIENTRY glLightf (GLenum light, GLenum pname, GLfloat param);
GLAPI void APIENTRY glLightfv (GLenum light, GLenum pname, const GLfloat *params);
GLAPI void APIENTRY glLightx (GLenum light, GLenum pname, GLfixed param);
GLAPI void APIENTRY glLightxv (GLenum light, GLenum pname, const GLfixed *params);
GLAPI void APIENTRY glLineWidth (GLfloat width);
GLAPI void APIENTRY glLineWidthx (GLfixed width);
GLAPI void APIENTRY glLoadIdentity (void);
GLAPI void APIENTRY glLoadMatrixf (const GLfloat *m);
GLAPI void APIENTRY glLoadMatrixx (const GLfixed *m);
GLAPI void APIENTRY glLogicOp (GLenum opcode);
GLAPI void APIENTRY glMaterialf (GLenum face, GLenum pname, GLfloat param);
GLAPI void APIENTRY glMaterialfv (GLenum face, GLenum pname, const GLfloat *params);
GLAPI void APIENTRY glMaterialx (GLenum face, GLenum pname, GLfixed param);
GLAPI void APIENTRY glMaterialxv (GLenum face, GLenum pname, const GLfixed *params);
GLAPI void APIENTRY glMatrixMode (GLenum mode);
GLAPI void APIENTRY glMultMatrixf (const GLfloat *m);
GLAPI void APIENTRY glMultMatrixx (const GLfixed *m);
```

Programmable pipeline

- Vertex Transformation stage replaced by Vertex Shader
- Pixel Color Determination replaced by Fragment shader
- Access to GPU-memory



Programmable pipeline

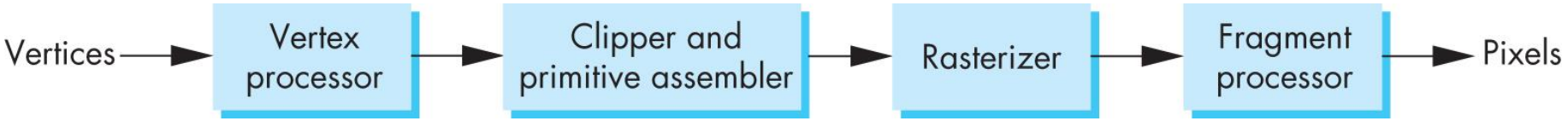
The programmable pipeline replaces some stages with fully programmable shader stages to give more control.

The vertex transformation stage is replaced by the vertex shader, and the pixel color determination stage is replaced by the Pixel Shader. Note that the Pixel Shader is sometimes called the Fragment shader. Fragment shader is the term used in OpenGL, whereas Pixel Shader is used by DirectX.

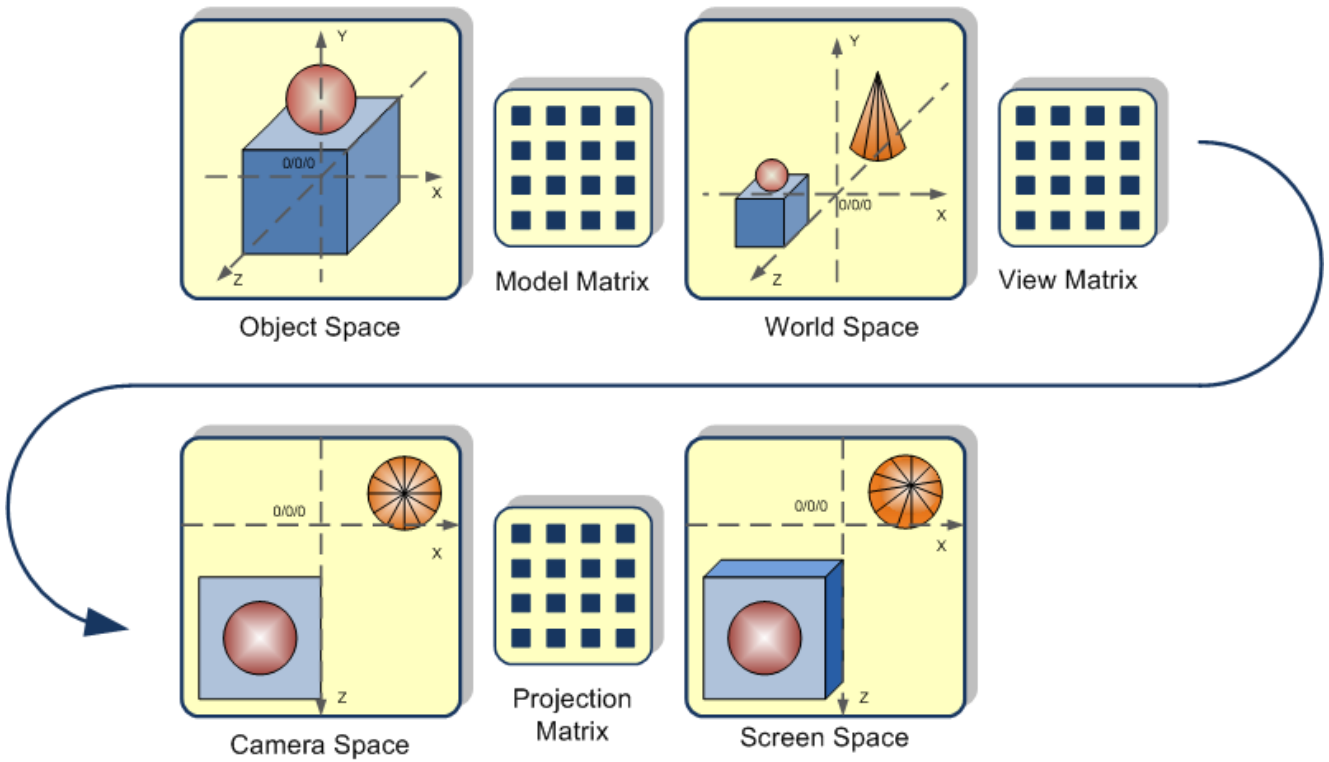
There is one disadvantage: all the things the replaced stages already did, aren't done anymore! We have to program them manually in the shaders.
The shaders have access to the GPU memory, so they can also access matrices and lighting information. But you can also pass different things to your shaders, for example other settings to use in your shader.

Vertex Processing

- Much of the work in the pipeline is in converting object representations from one coordinate system to another
  - Object coordinates
  - World coordinates
  - Camera (eye) coordinates
  - Screen coordinates
- Every change of coordinates is equivalent to a matrix transformation



Vertex Processing



1. Vertices of the Object to draw are in **Object space** (as modelled in your 3D Modeller)

2. … get transformed into World space by multiplying it with the **Model Matrix**

3. Vertices are now in **World space** (used to position the all the objects in your scene)

4. … get transformed into Camera space by multiplying it with the **View Matrix**

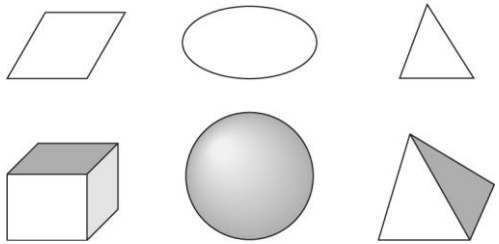
5. Vertices are now in **View Space** – think of it as if you were looking at the scene throught "the camera"

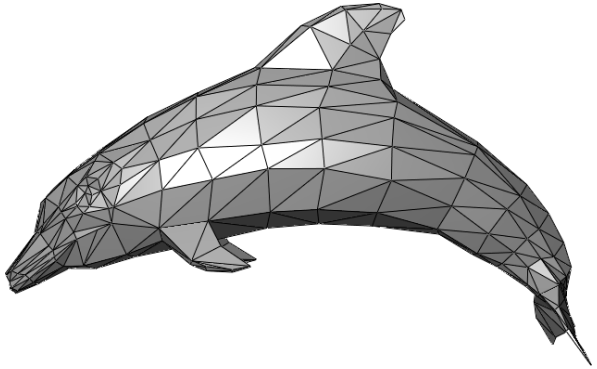
6. … get transformed into Screen space by multiplying it with the **Projection Matrix**

7. Vertex is now in **Screen Space** – This is actually what you see on your Display.

Geometric Primitives

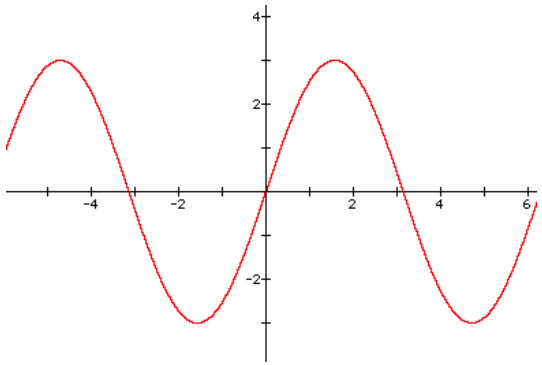
- Different philosophies:
  - Collections of complex shapes
    - Spheres, cones, cylinders, tori, …
  - One simple type of geometric primitive
    - Triangles or triangle meshes
  - Small set of complex primitives with adjustable parameters
    - E.g. "all polynomials of degree 2"
    - Splines, NURBS (details in CPSC 424)
    - Implicits



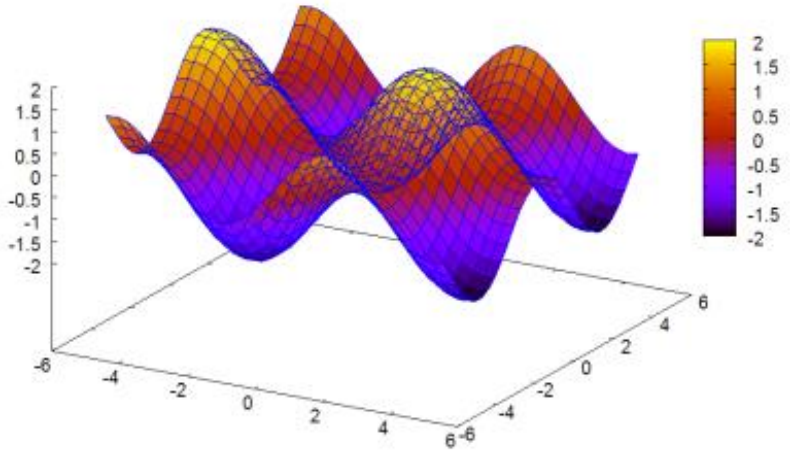


Geometric Primitives

- Explicit Functions
- Curves:
  - y is a function of x: **y:=sin(x)**
  - Only works in 2D



- Surfaces:
  - z is a function of x and y: **z:=sin(x)+cos(y)**
  - Cannot define arbitrary shapes in 3D



Geometric Primitives

- Parametric Functions
- Curves:
  - 2D: x and y are functions of a parameter value t
  - 3D: x, y, and z are functions of a parameter value t

$$C(t) := \begin{pmatrix} \cos(t) \\ \sin(t) \\ t \end{pmatrix}$$

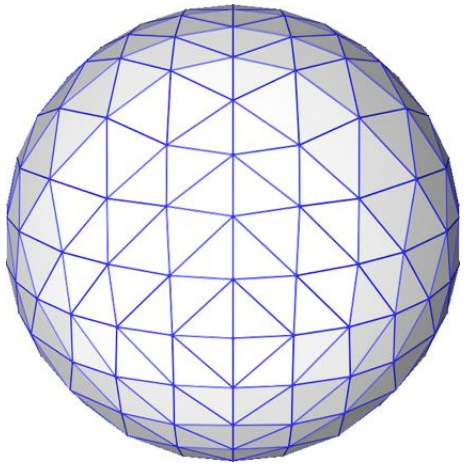
- Surfaces:
  - Surface S is defined as a function of parameter values s, t
  - Names of parameters can be different to match intuition

$$S(\phi, \theta) := \begin{pmatrix} \cos(\phi)\cos(\theta) \\ \sin(\phi)\cos(\theta) \\ \sin(\theta) \end{pmatrix}$$

Geometric Primitives

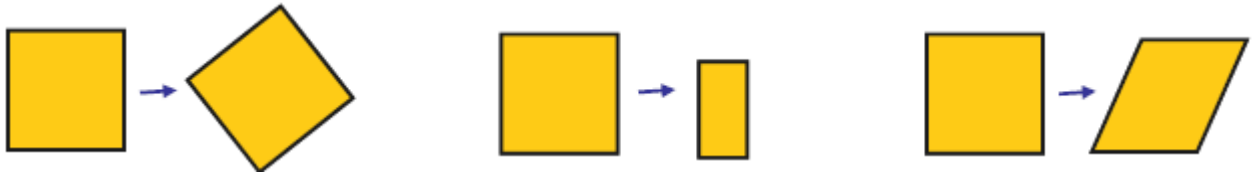
- Implicit Functions
- Surface:
  - Surface defined by zero set (roots) of function
  - E.g.

$$S(x, y, z) : x^2 + y^2 + z^2 - 1 = 0$$

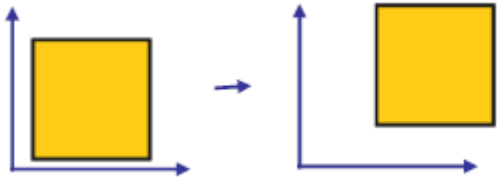


Model/View Transformation

- Types of transformations:
  - Rotations, scaling, shearing

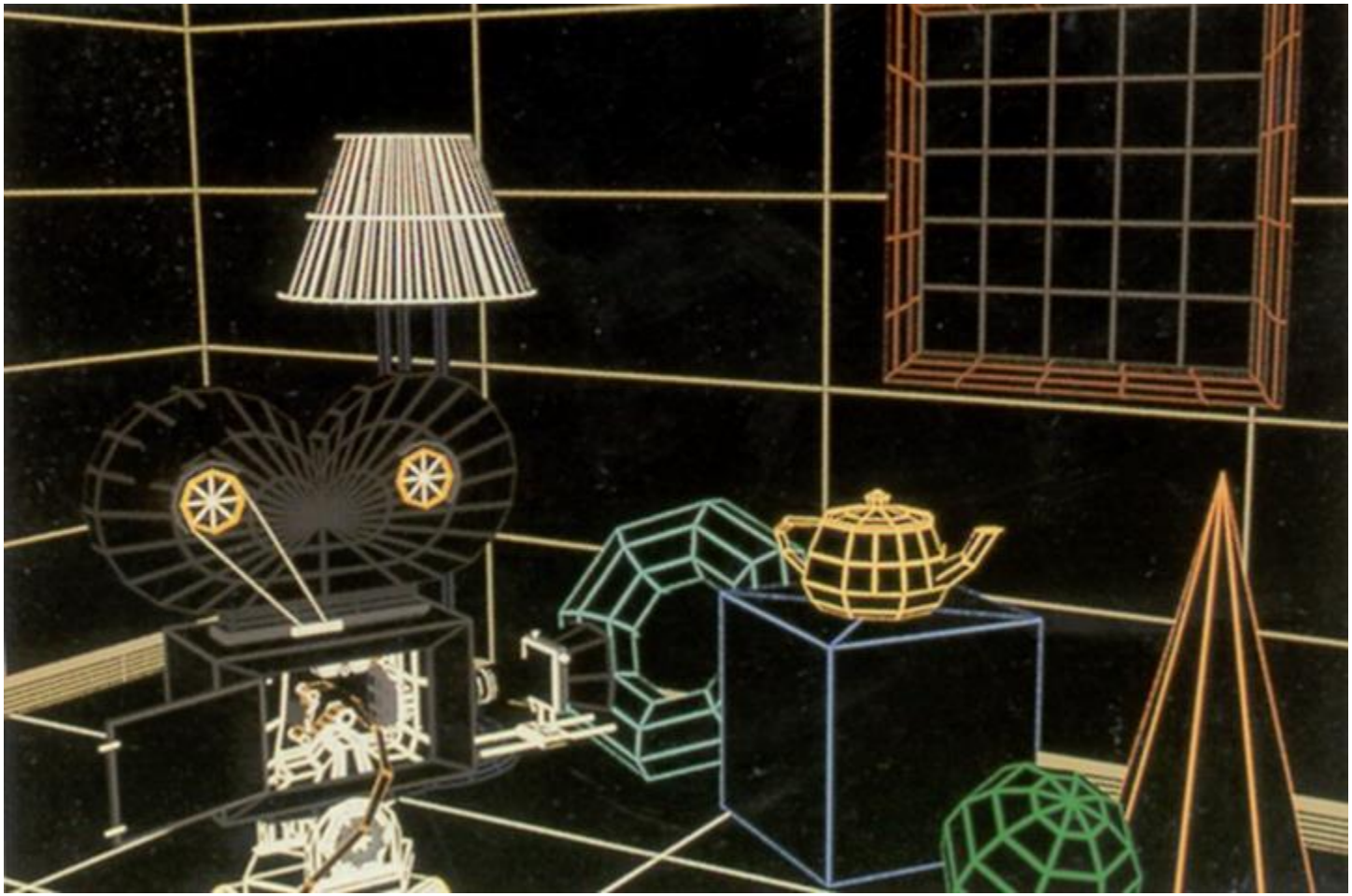


  - Translations



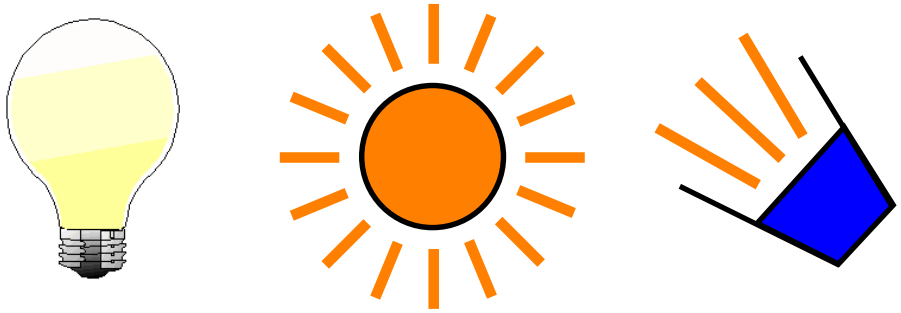
  - Other transformations (not handled by rendering pipeline)
    - Freeform deformation

Example - Modeling and Viewing Transformations



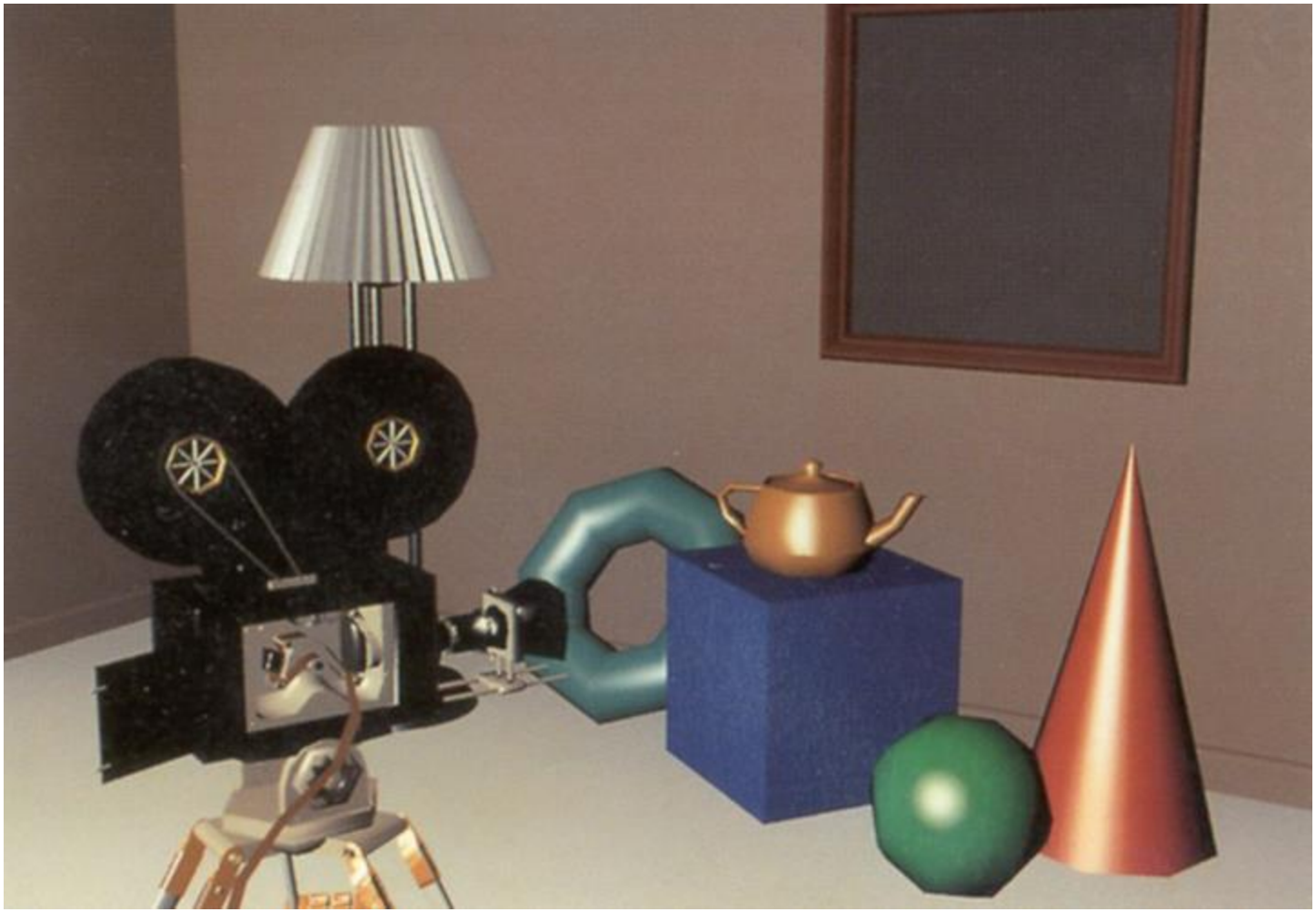
Lighting

- Compute brightness based on property of material and light position(s)
- Computation is performed per-vertex
- There are several kinds of lights (or light sources)
  - Light bulbs
  - The sun
  - Spot Lights
  - Ceiling Lights



- These are different because they emit photons (光子) differently

Example - Lighting



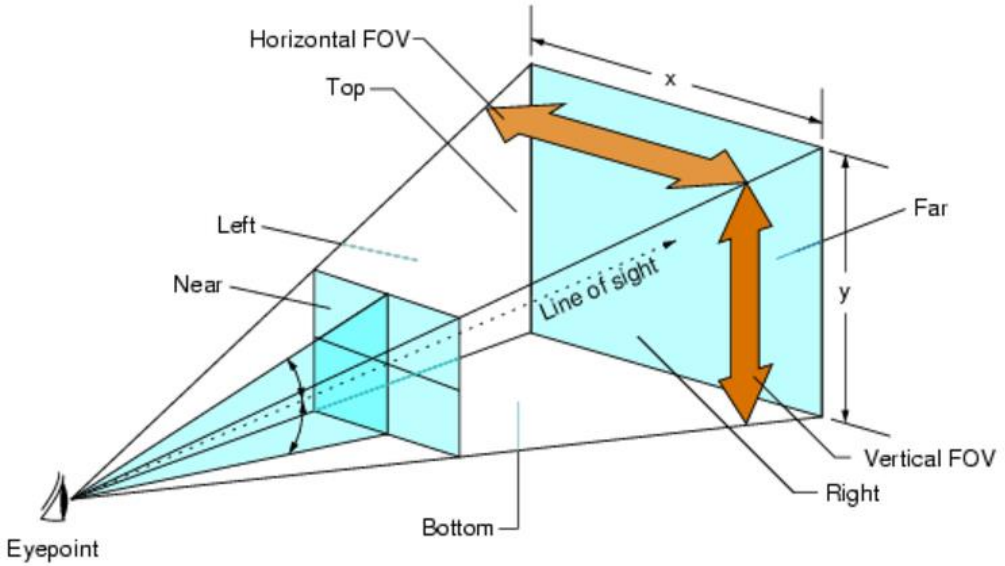
Shading

- Problem: How do we determine the color of a piece of geometry?
- In the real world, color depends on the object's surface color and the color of the light.
- It is the same way in computer graphics.
- "Shading" is the process by which color is assigned to geometry.

Example - Complex Lighting and Shading

3D to 2D (Projection)

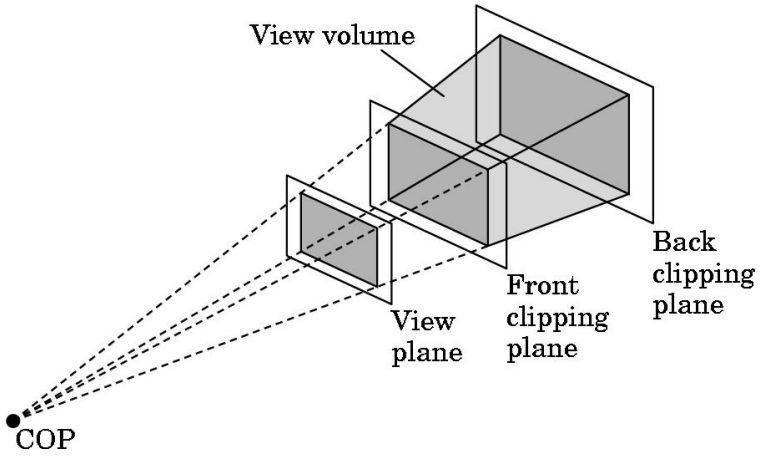
- Problem: The display is, virtually always, only 2D
  - Need to transform the 3D model into 2D
- We do this with a virtual camera
- Represented mathematically by a 3x4 projection (or P) matrix

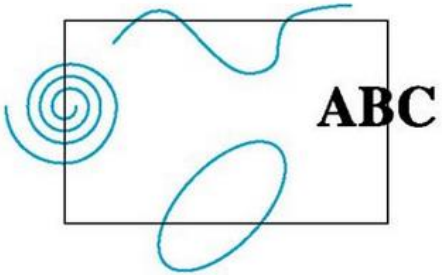


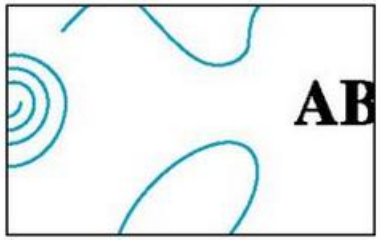
$$\text{Aspect Ratio} = \frac{y}{x} = \frac{\tan(\text{vertical FOV}/2)}{\tan(\text{horizontal FOV}/2)}$$

Clipping

- Problem: The camera doesn't see the whole scene
  - In particular, the camera might only see parts of objects
- Solution: Find objects that cross the edge of the viewing volume, and "clip" them
  - Clip: Cut a polygon into multiple parts, such that each is entirely inside or outside the display area

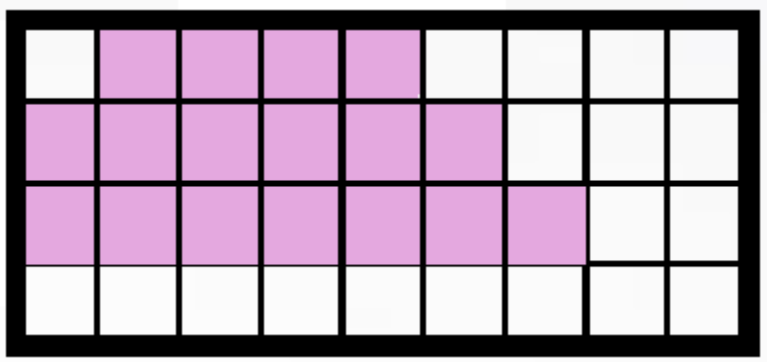






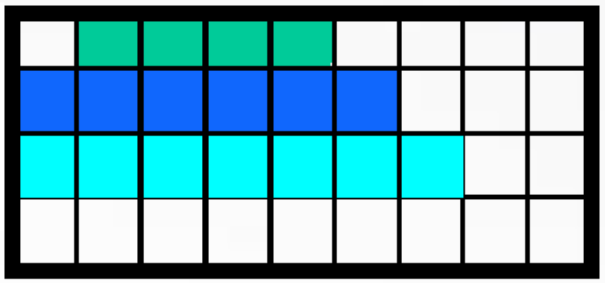
Scan conversion

- Convert continuous 2D geometry (lines, polygons etc.) to discrete
- Raster display – Conversion from two-dimensional vertices in screen space – each with a z-value (depth value), and various shading information associated with each vertex – into pixels on the screen (On GPU)

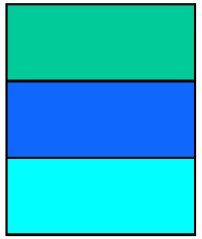


Texture mapping

- “Gluing (粘合) images onto geometry”
- Color of every fragment is altered by looking up a new color value from an image.



Image

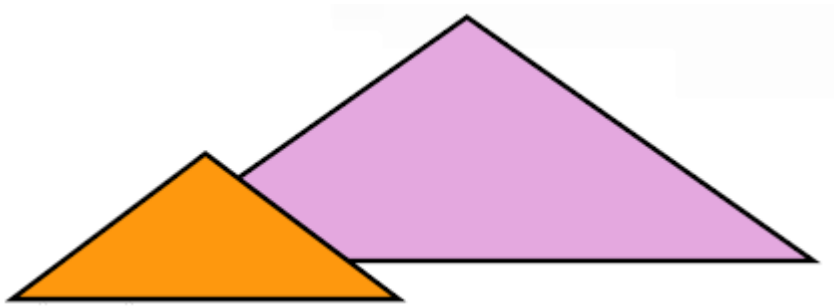


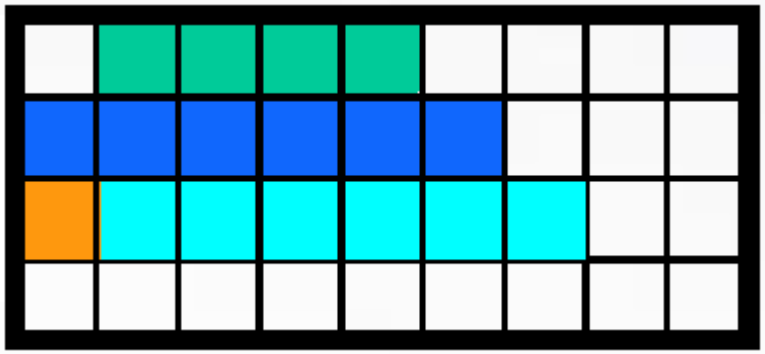
Skin Image

Example – Texture Mapping

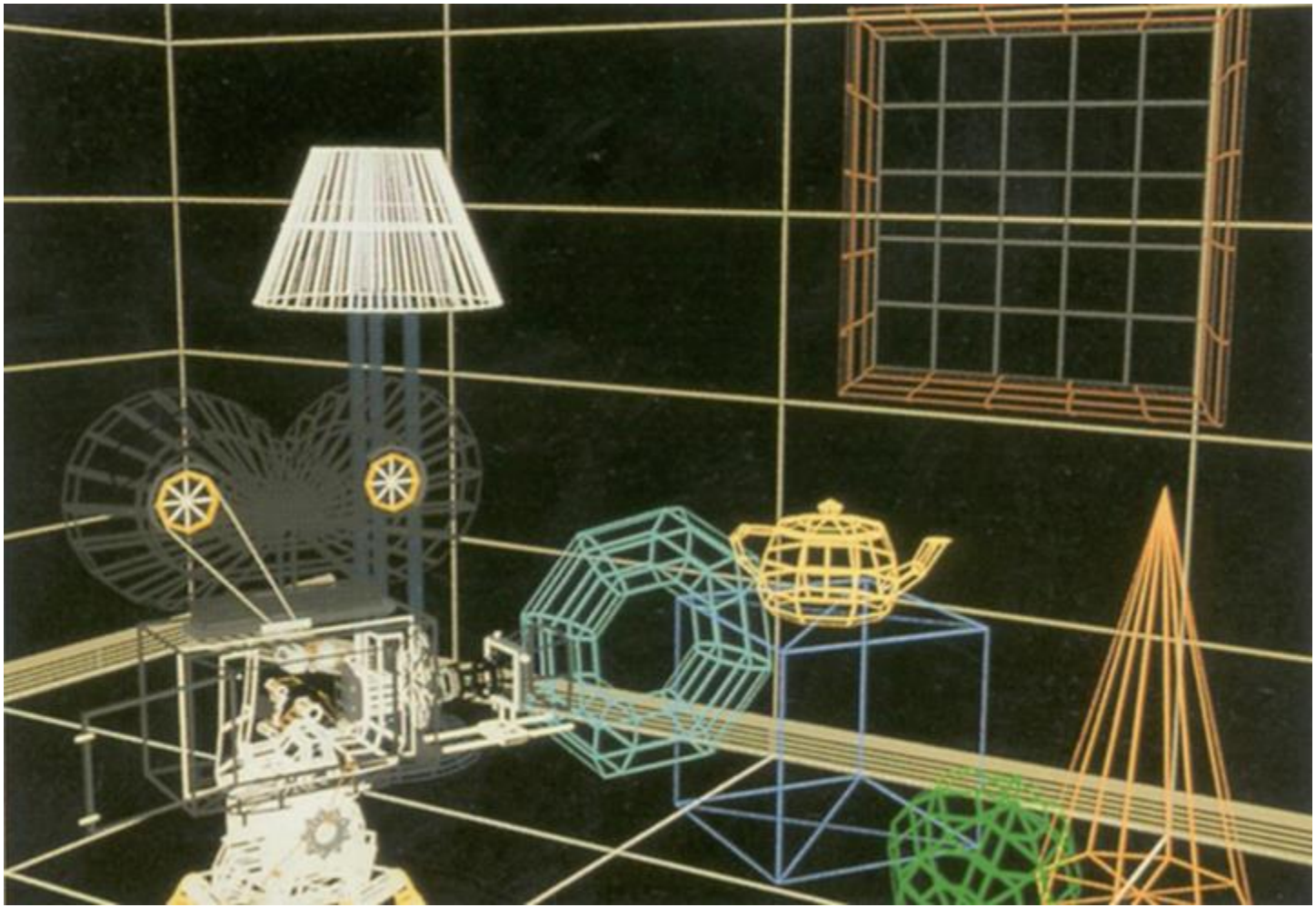
Depth Test

- Remove parts of geometry hidden behind other geometric objects
- Perform on every individual fragment

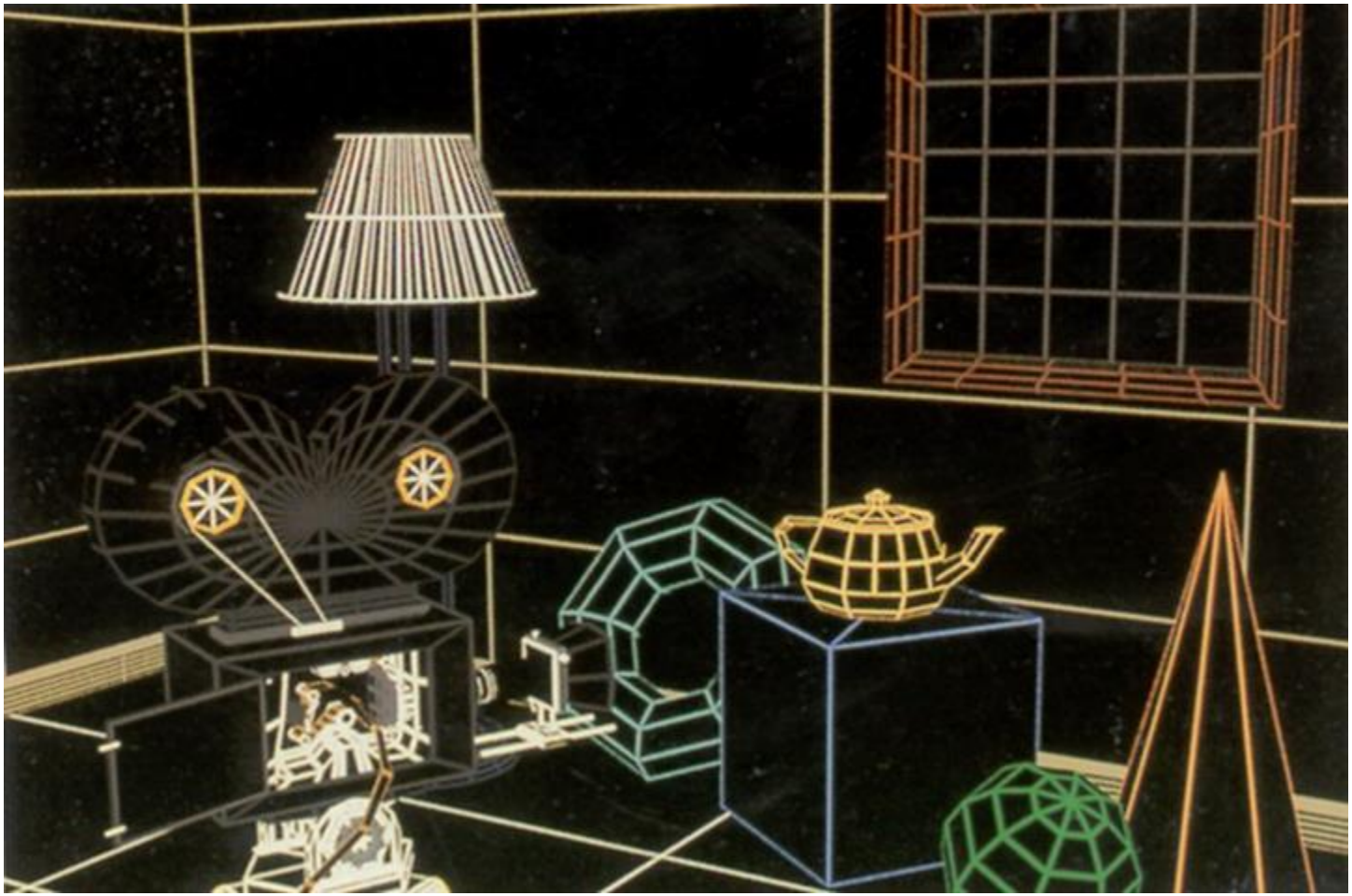




Example - Without Hidden Line Removal

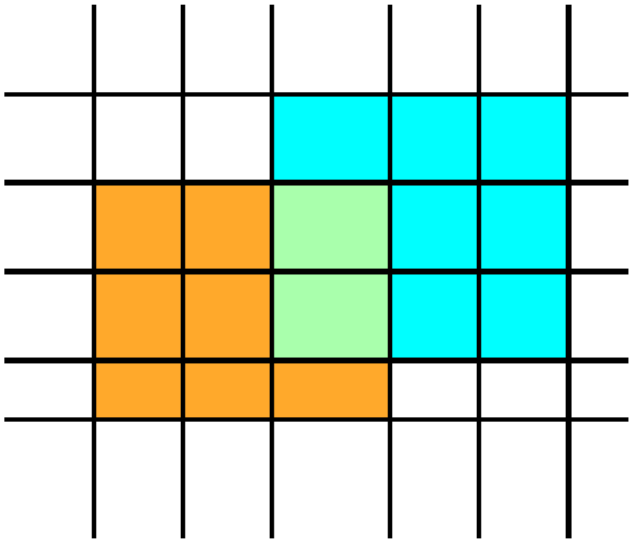


Example - Hidden Line Removal



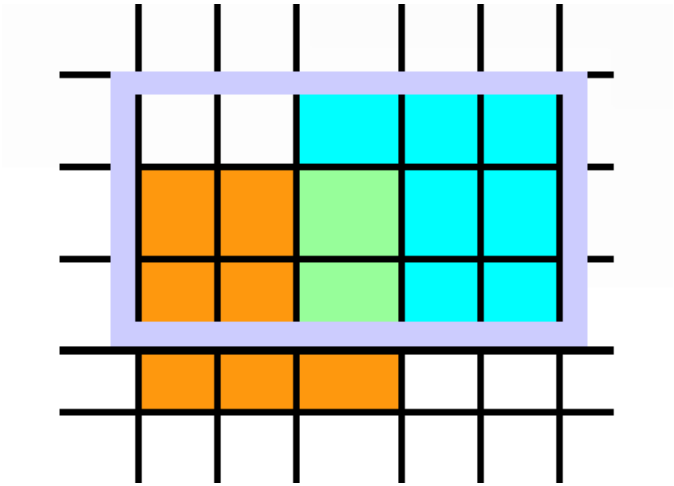
Blending

- final image: write fragments to pixels
- draw from farthest to nearest
  - no blending – replace previous color
  - blending: combine new & old values with arithmetic operations



Frame-buffer

- video memory(图象存储器) on graphics board that holds image
- double-buffering: two separate buffers
  - draw into one while displaying other, then swap to avoid flicker



255 255 255	255 255 255	0 255 255	0 255 255	0 255 255
255 155 0	255 155 0	155 255 155	0 255 255	0 255 255
255 155 0	255 155 0	155 255 155	0 255 255	0 255 255

Modeling vs. Rendering

- **Modeling**
  - Create models
  - Apply materials to models
  - Place models around scene
  - Place lights in scene
  - Place the camera

- **Rendering**

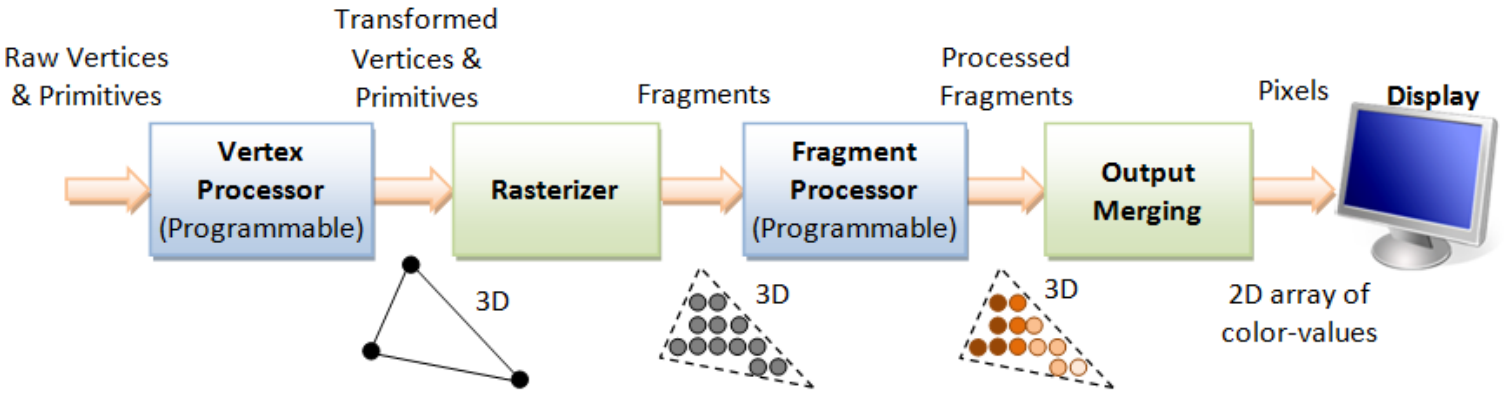
  Take “picture” with camera

- Both can be done with commercial software:

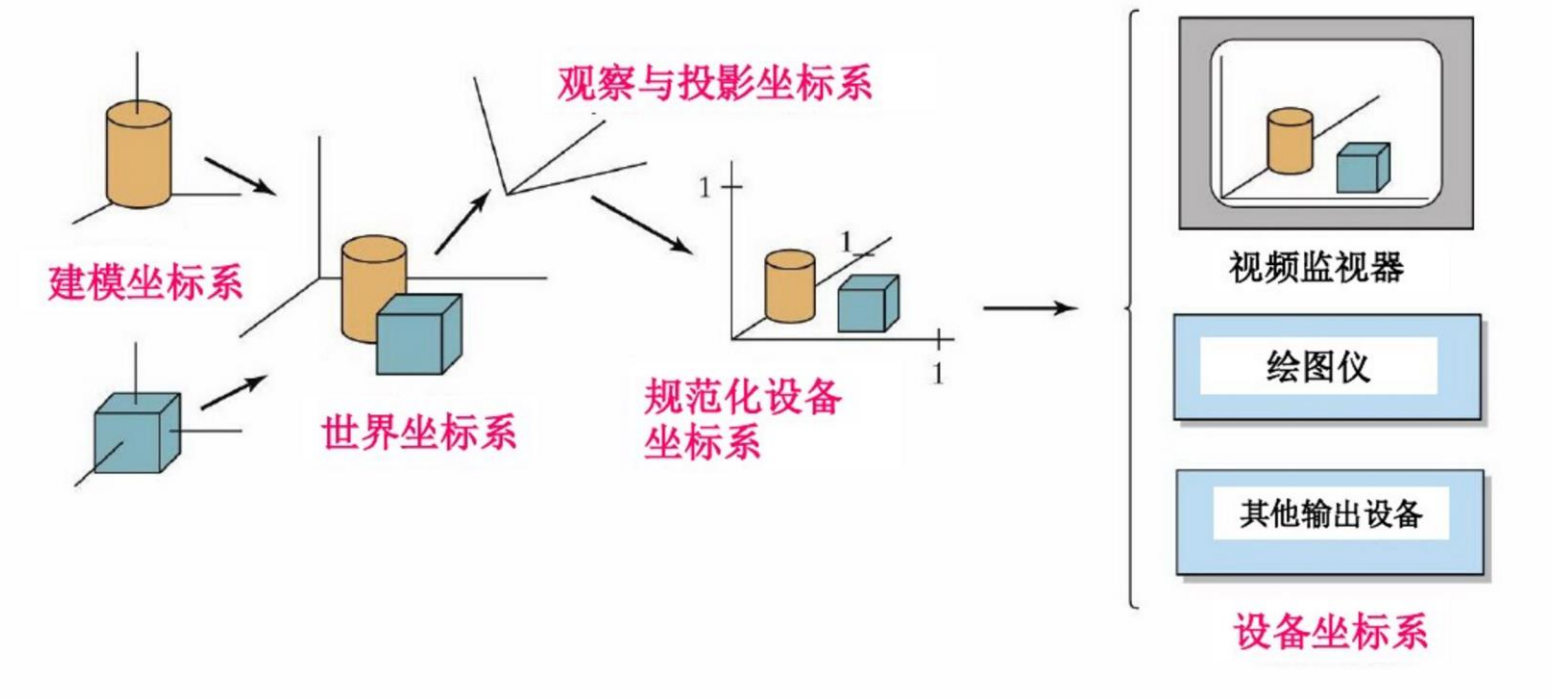
  Autodesk Maya™ ,3D Studio Max™, Blender™, etc.

Summary

- The 3D graphics rendering pipeline consists of the following main stages:
  - **Vertex Processing**: Process and transform individual vertices.
  - **Rasterization**: Convert each primitive (connected vertices) into a set of fragments. A fragment can be treated as a pixel in 3D spaces, which is aligned with the pixel grid, with attributes such as position, color, normal and texture.
  - **Fragment Processing**: Process individual fragments.
  - **Output Merging**: Combine the fragments of all primitives (in 3D space) into 2D color-pixel for the display.



3D Graphics Rendering Pipeline: Output of one stage is fed as input of the next stage. A vertex has attributes such as (x, y, z) position, color (RGB or RGBA), vertex-normal (n_x, n_y, n_z), and texture. A primitive is made up of one or more vertices. The rasterizer raster-scans each primitive to produce a set of grid-aligned fragments, by interpolating the vertices.


观察与投影坐标系
建模坐标系
世界坐标系
规范化设备
坐标系
1
1
1
视频监视器
绘图仪
其他输出设备
设备坐标系