Computer Graphics

Rendering Pipeline

Teacher: Dr. Zhuo SU (苏卓)

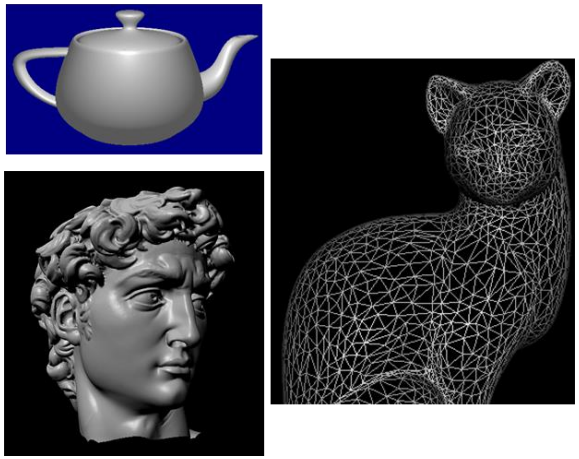
E-mail: suzhuo3@mail.sysu.edu.cn

School of Data and Computer Science

Outline

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

Modeling
(Creating 3D Geometry)



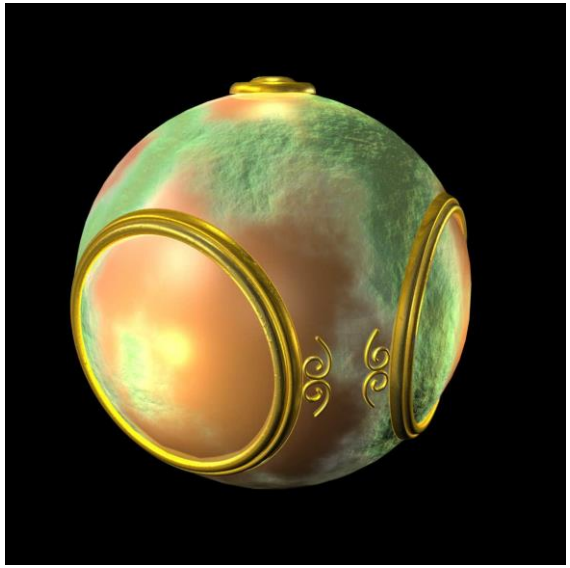
→

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

Example

- Where did these images come from?

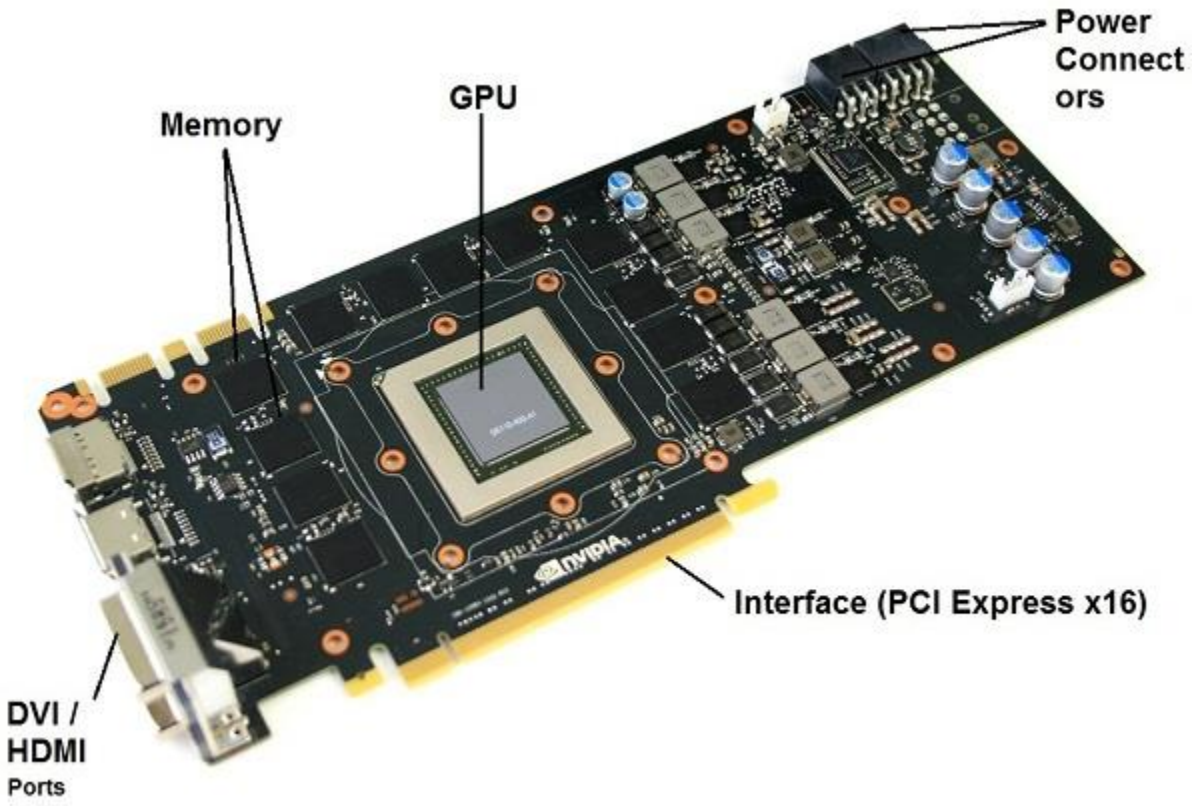




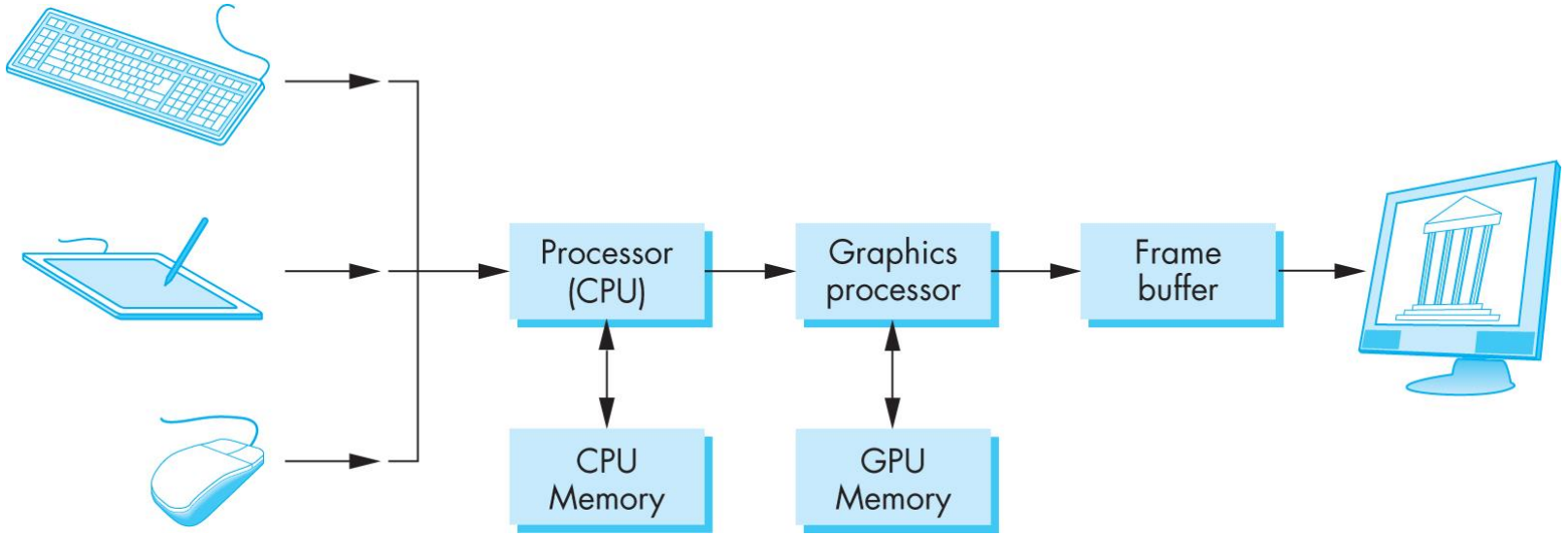
- What hardware/software did we need to produce it?
  - Software: Maya for modeling and rendering but Maya is built on top of OpenGL
  - Hardware: PC with graphics card for modeling and rendering

Software – Autodesk Maya

Hardware - display card



A computer graphics system



Input devices

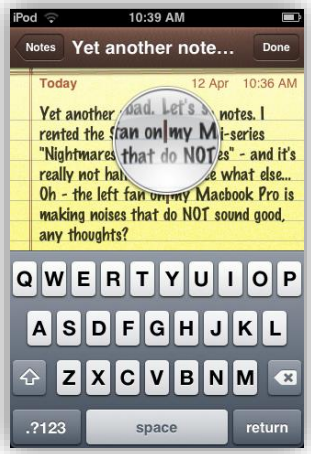
Image formed in frame buffer

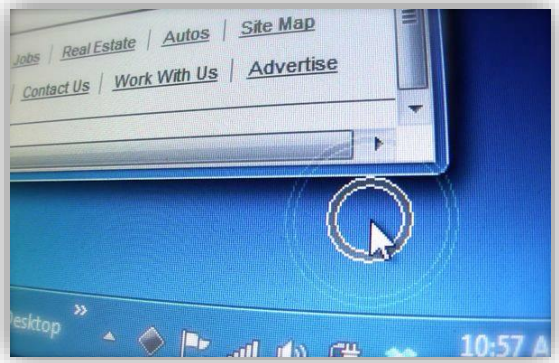
Output device

Input: Consumer devices

GRAPHIC|S

G|RAPHICS





Input: Image & video

Input: Image & video

The modern era of easy photography

Better photography in a simple way

$ 5000

$ 5

Mobile phone camera

Can these two ever be equally good at taking pictures?

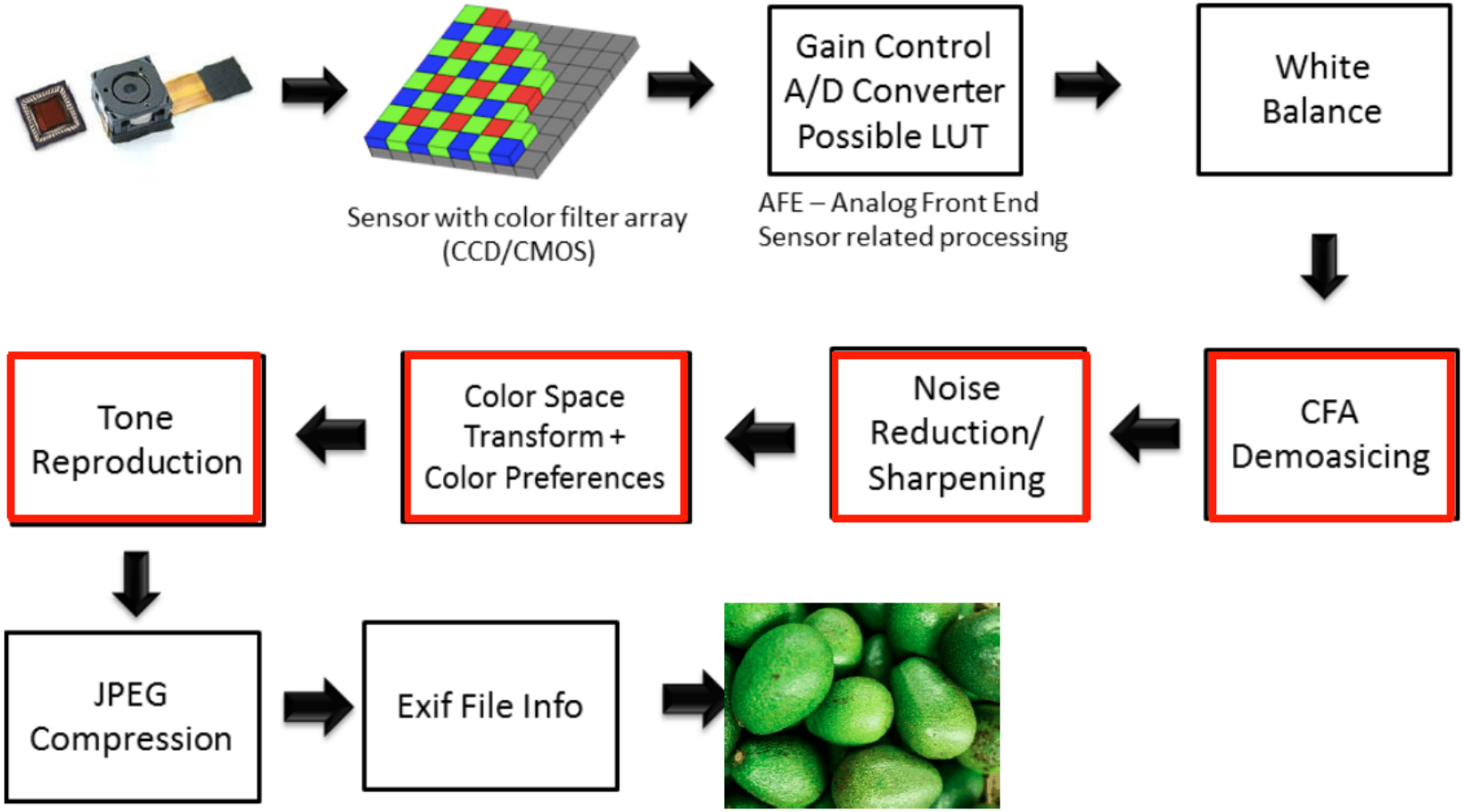
Better photography in a simple way

iPhone 3GS

Canon EOS 5D Mark II

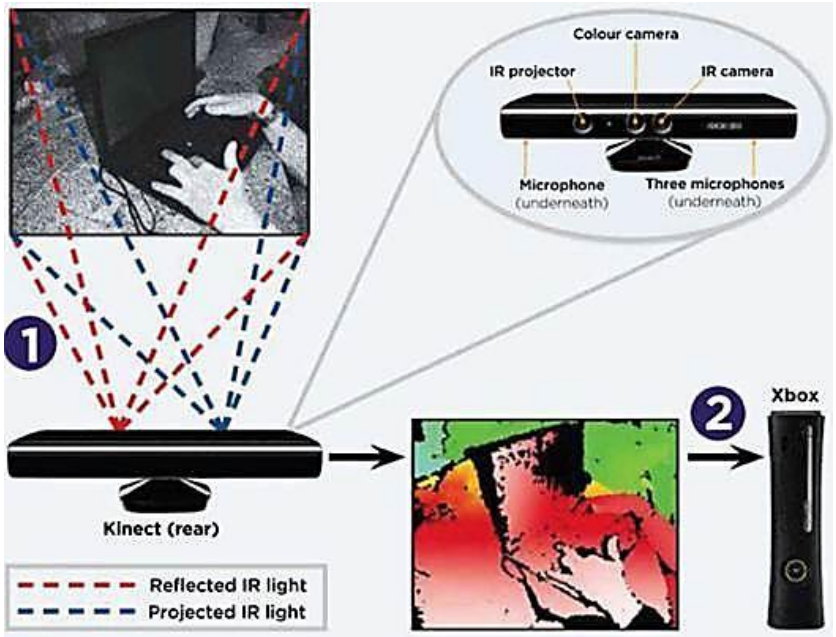
Better photography in a simple way

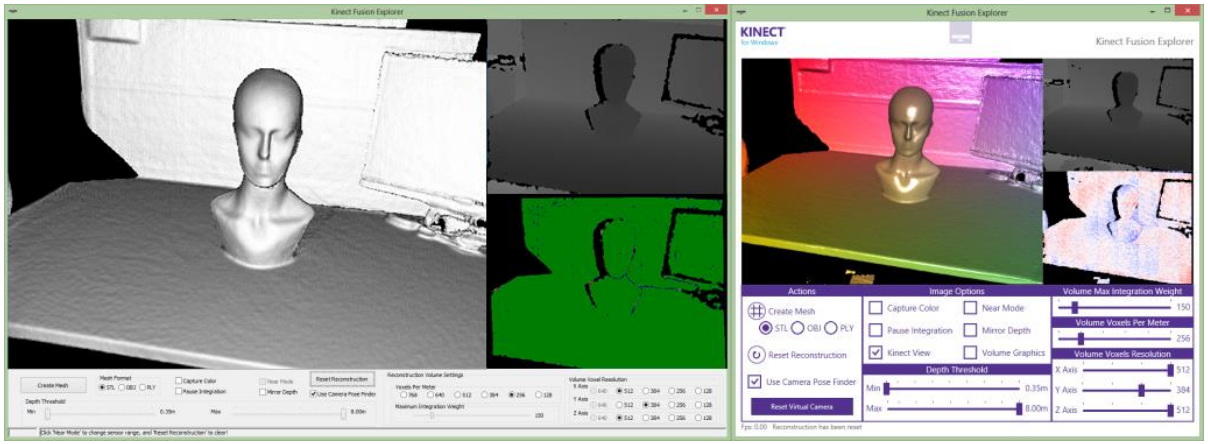
- A standard model for imaging pipeline



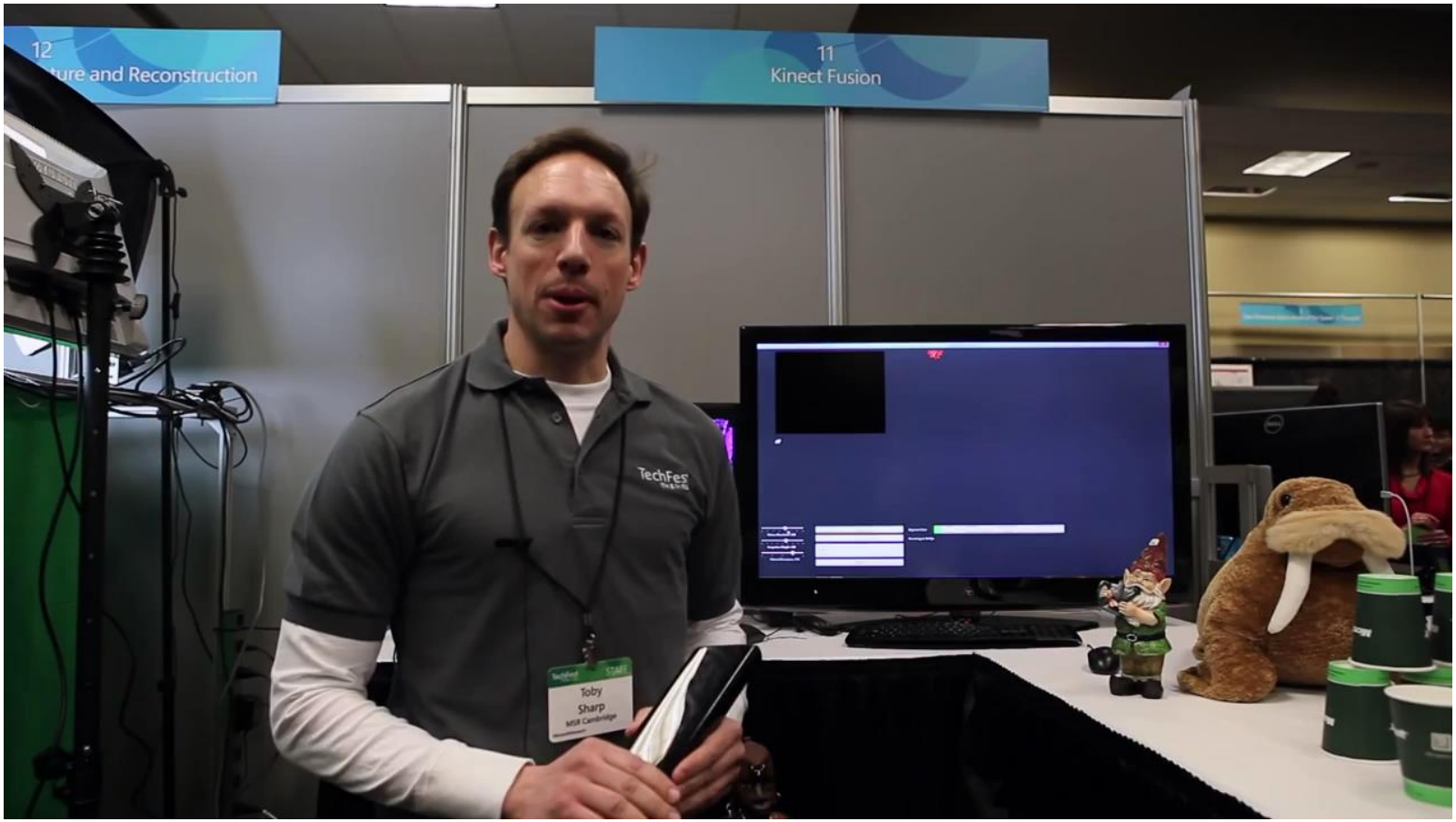
Input: Shape

Kinect Fusion

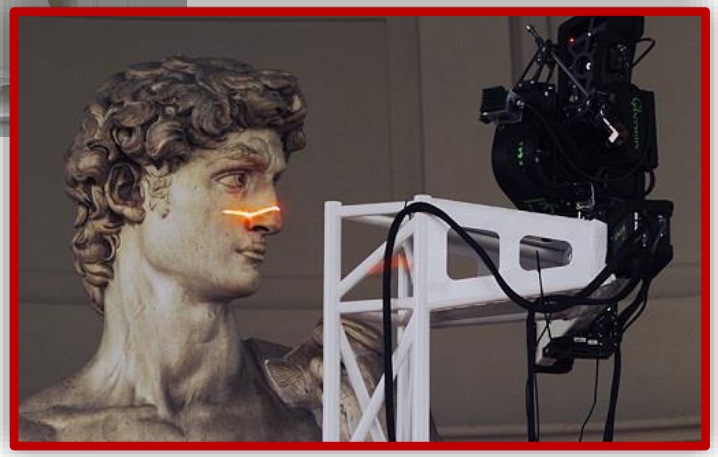


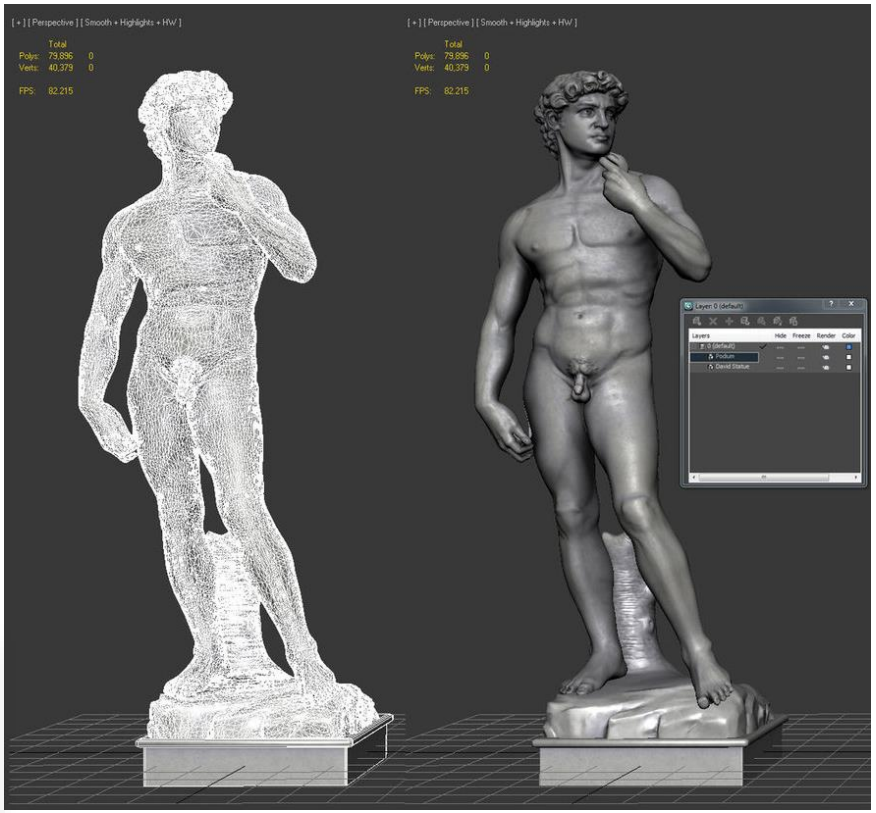


Shapes from Kinect Fusion



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

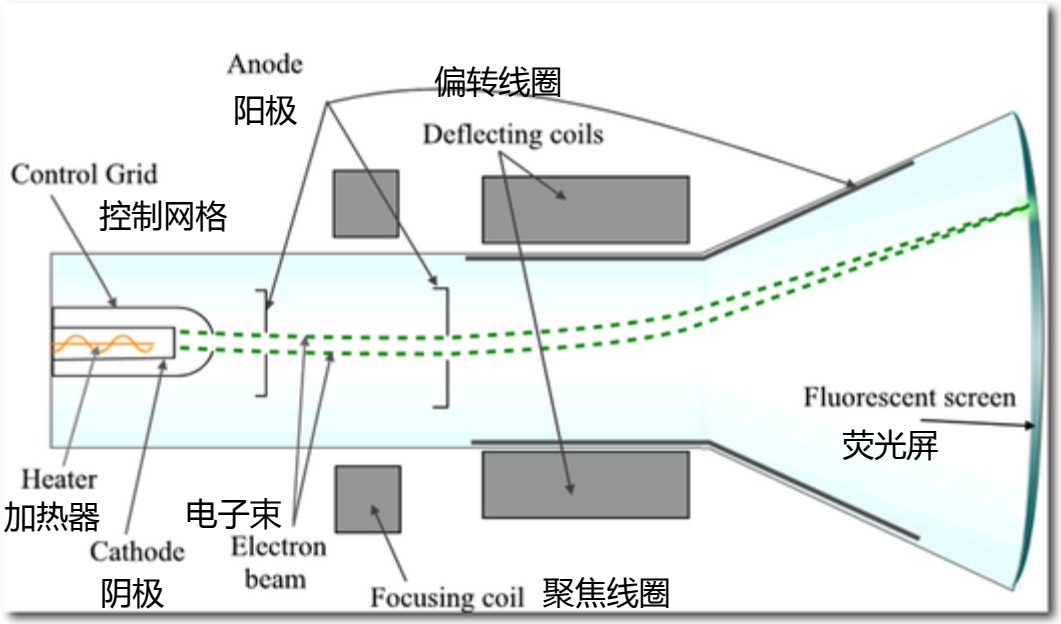
Desktop 3D scanner

Input: Specialized devices

What about the output device?

Display Devices

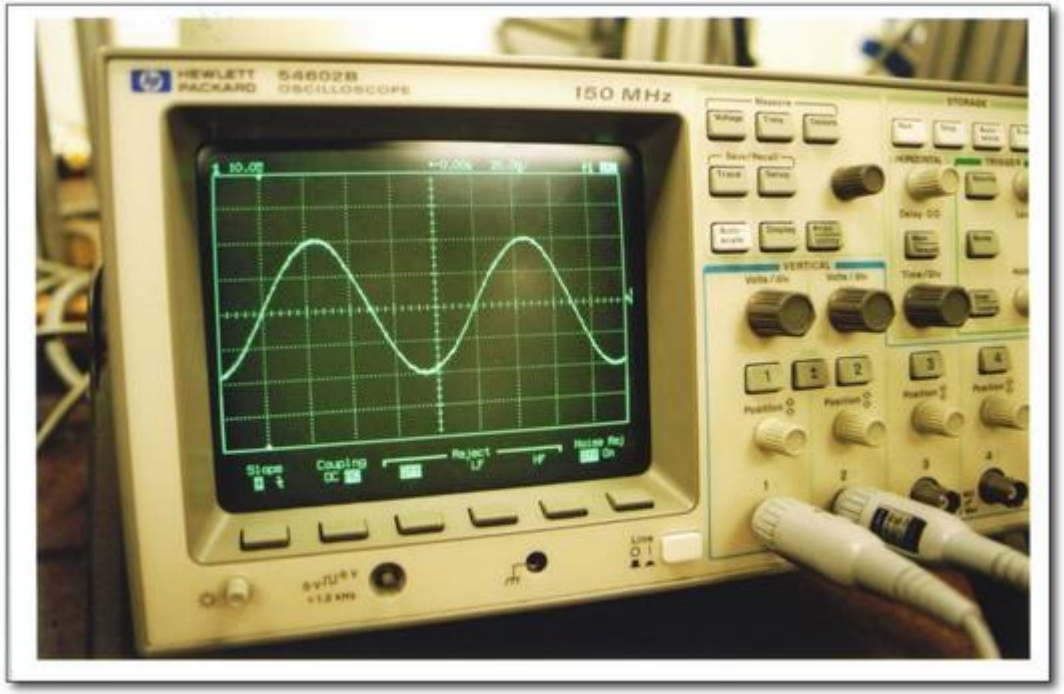
- Cathode Ray Tube (CRT)
  - CRT is a vacuum tube (电子管) containing one or more electron guns, and a phosphorescent screen (磷光屏) used to view images.



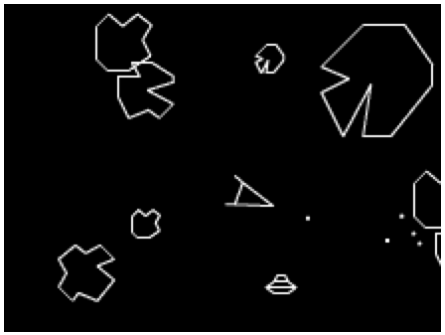
Karl Ferdinand Braun, 1897
Image courtesy of Wikipedia

Display Devices

- Vector Displays



HP Oscilloscope

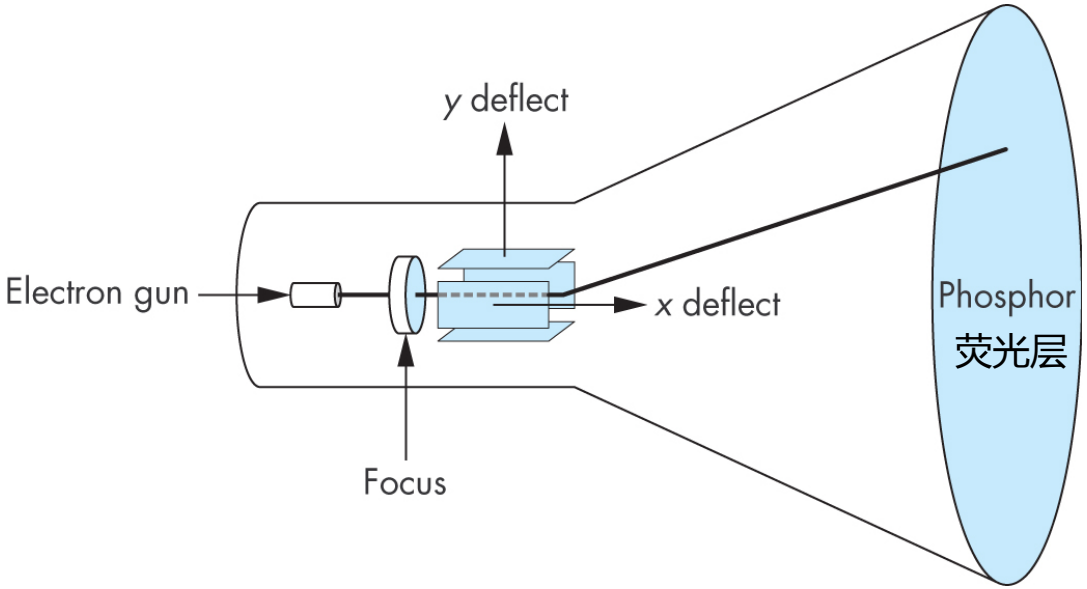


Asteroids, 1979 (行星游戏)

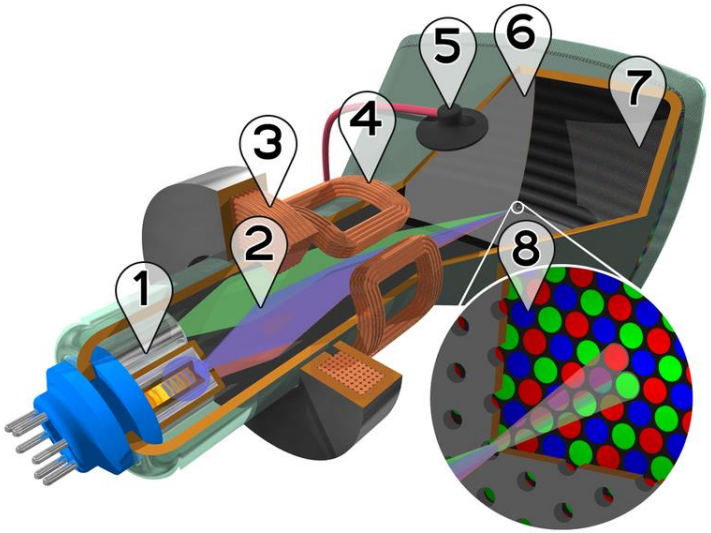
Star Wars, 1983 (星球大战)

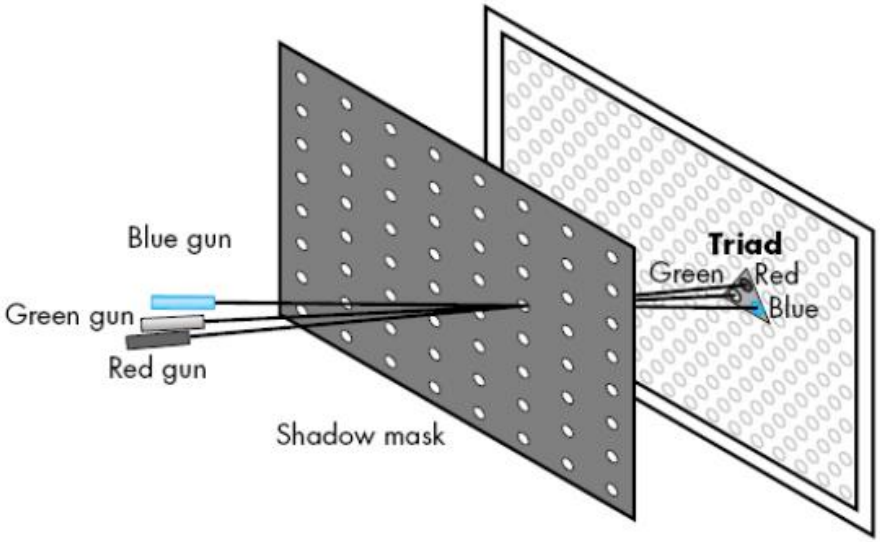
Display Devices

- Raster Display Devices (光栅显示器)
  - The refresh type raster scan display : Get the pixels from the frame buffer individually and corresponding location on the screen display.
  - Refresh Rate: 刷新率
  - Interlaced scan, Non-Interlaced or Progressive scan: 逐行扫描和隔行扫描



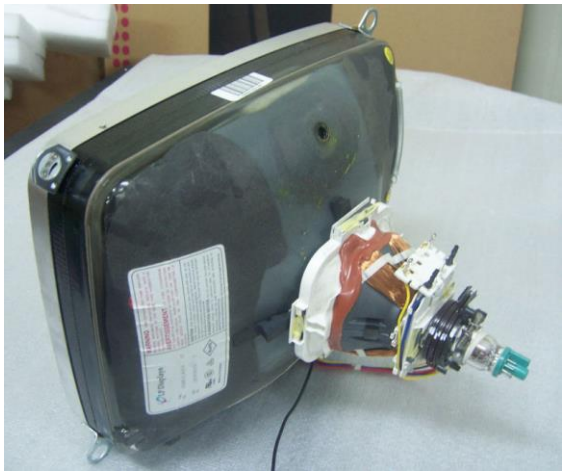
Color CRT display



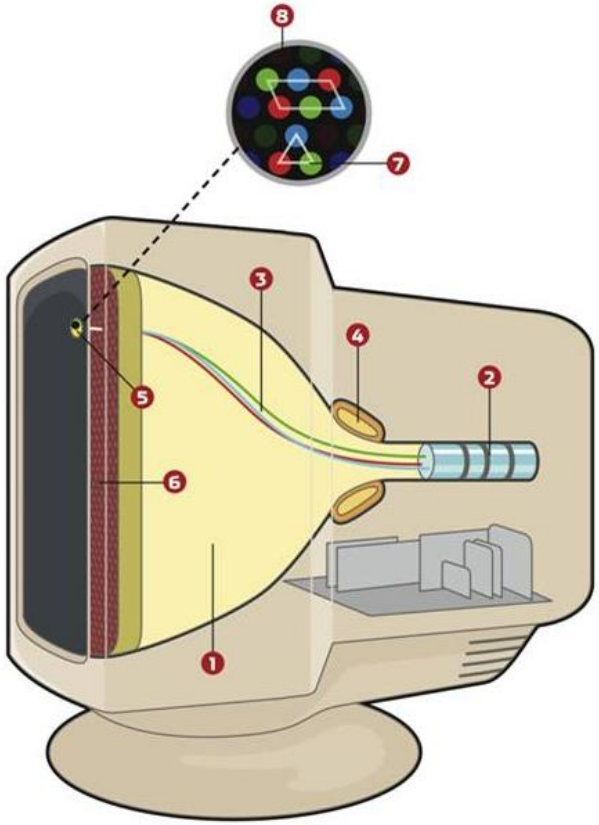


Cutaway rendering of a color CRT:

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

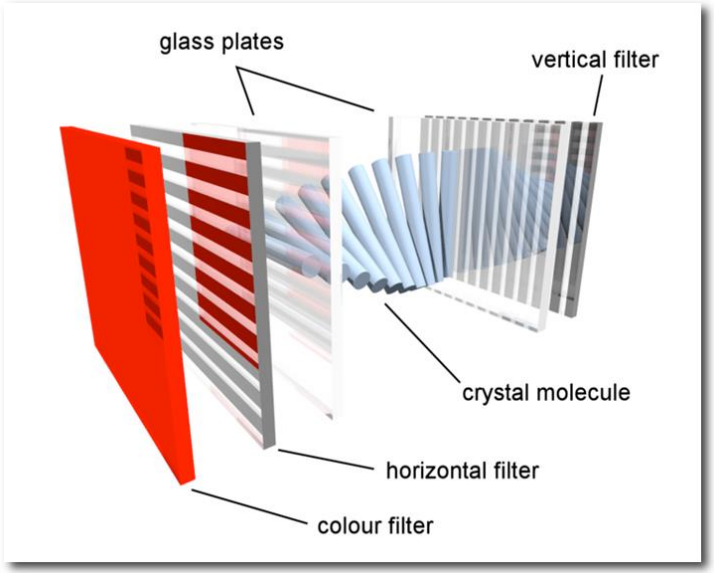


Color CRT display

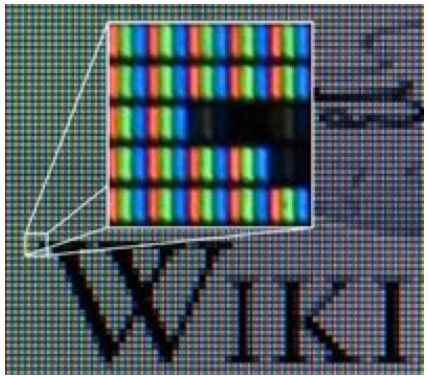


Display Devices

- **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 distorti

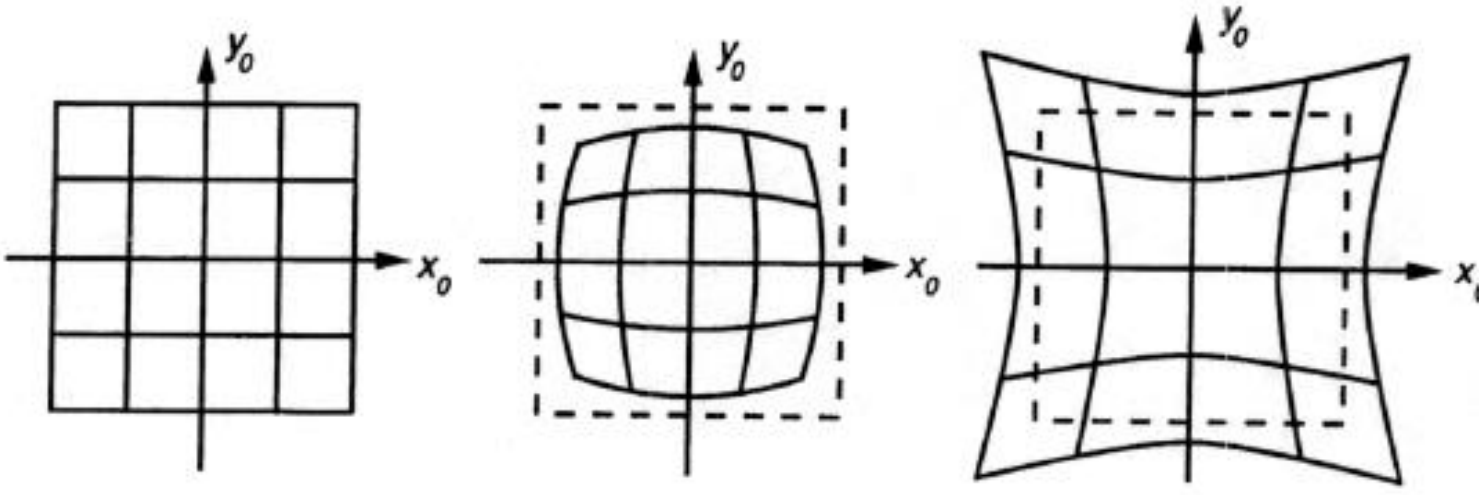


A single subpixel of an LCD display



Close-up of pixels on an LCD display

Geometric distortion in display monitor



No Distortion

无失真

Barrel Distortion

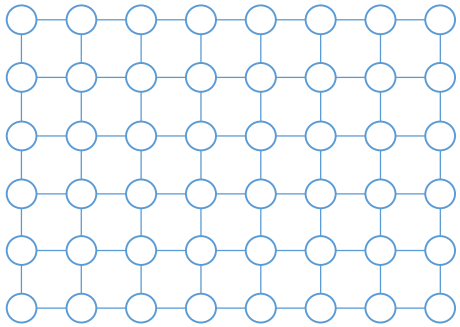
桶形失真

Pincushion Distortion

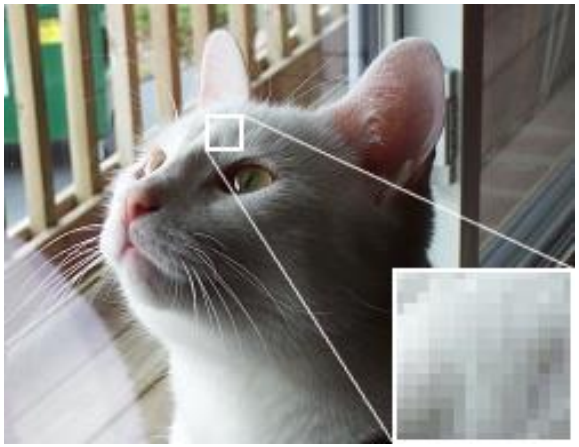
枕形失真

What does “image” mean for us?

- A 2D domain with samples at regular points (almost always a rectilinear grid)
  - Can have multiple values sampled per point
  - Meaning of samples depend on the application (red, green, blue, opacity, depth, etc.)

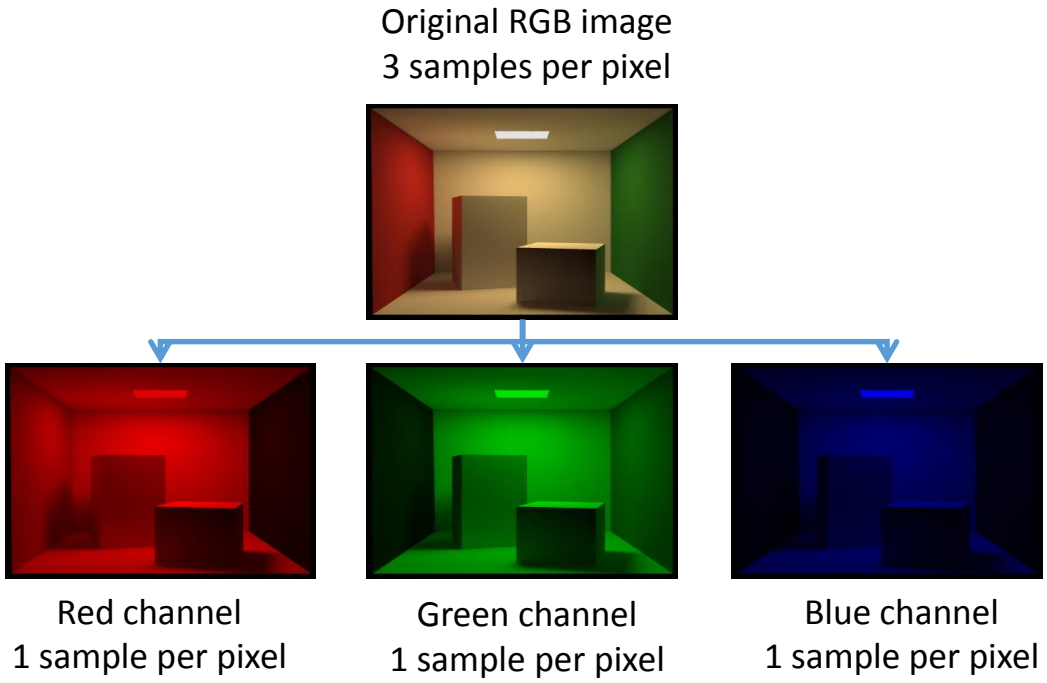


- Units also depend on the application
  - e.g., a computed int or float to be mapped to voltage needed for display of a pixel on a screen
  - e.g., as a physical measurement of incoming light (e.g., a camera pixel sensor)



What is a channel?

- A channel is all the samples of a particular type
- RGB is a common format for image channels
  - Easy to implement in h/w
  - 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



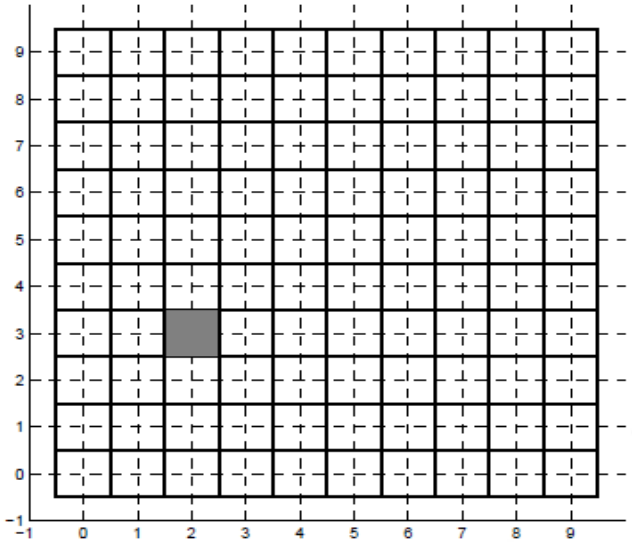
The alpha channel

- 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 $\alpha = 0.8$

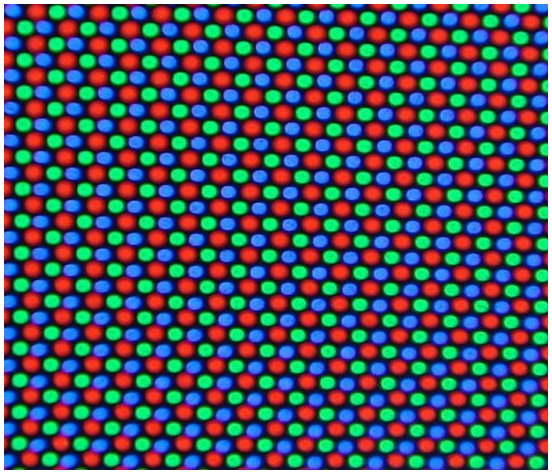
Modeling an image

- 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

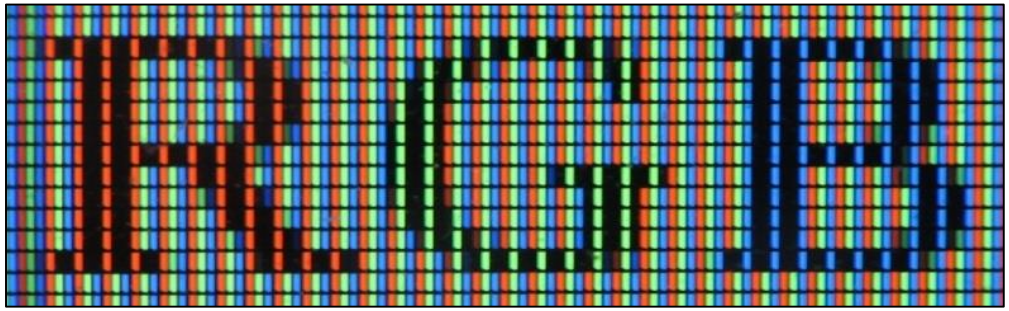


Pixels

- Pixels are point samples, not “squares” or “dots”
- Point samples reconstructed for display (often using multiple subpixels for primary colors)



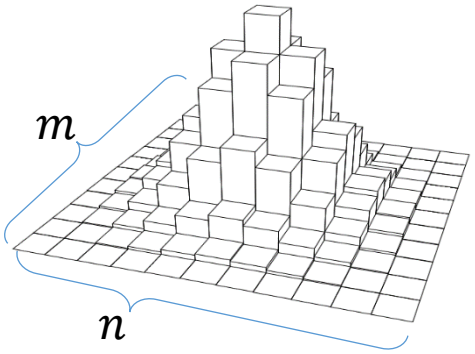
Close-up of a CRT screen

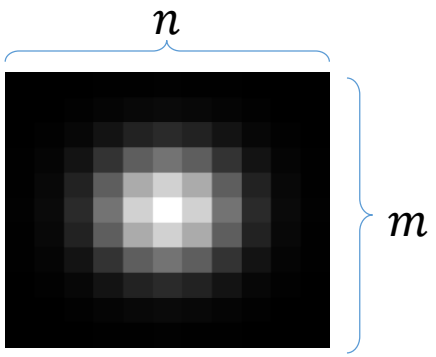


Close-up of an LCD screen

Discrete Images vs. Continuous Images

- Two kinds of images
  - Discrete
  - Continuous
- Discrete image
  - Function from $\mathbb{Z}^2$ to $\mathbb{R}$
  - How images are stored in memory
  - The kind of images we generally deal with as computer scientists

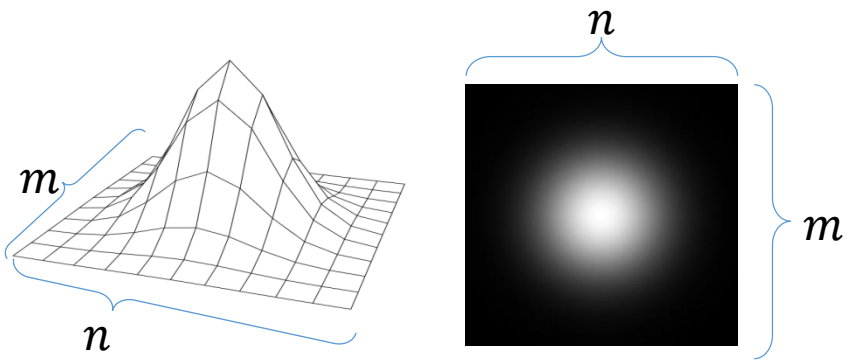




Discrete image $u(i,j)$

Discrete Images vs. Continuous Images

- Continuous image
  - Function from $\mathbb{R}^2$ to $\mathbb{R}$
  - Images in the real world
  - "Continuous" refers to the domain, not the values (discontinuities could still exist)
- Example: Gaussian distribution
  - i_0 and j_0 are the center of the Gaussian
  - $u: \mathbb{Z}^2 \to \mathbb{R}, u(i,j) = e^{-(i-i_0)^2-(j-j_0)^2}$
  - $v: \mathbb{R}^2 \to \mathbb{R}, v(i,j) = e^{-(i-i_0)^2-(j-j_0)^2}$
  - $i_0 = (n-1)/2$ and $j_0 = (k-1)/2$ (n odd)
  - Here $n = 11$ and $m = 11$

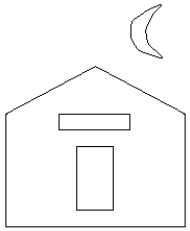


Continuous image $v(i,j)$

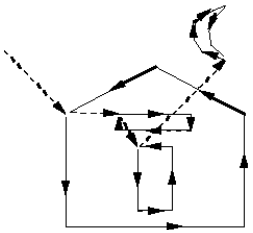
Graphics Display Hardware

Vector (calligraphic, stroke, random-scan)

- Driven by display commands
  - (move (x, y), char("A") , line(x, y)…)
- Survives as "scalable vector graphics"



Ideal Drawing



Vector Drawing

Raster (TV, bitmap, pixmap) used in displays and laser printers

- Driven by array of pixels (no semantics, lowest form of representation)
- Note "jaggies" (aliasing errors) due to discrete sampling of continuous primitives

Outline

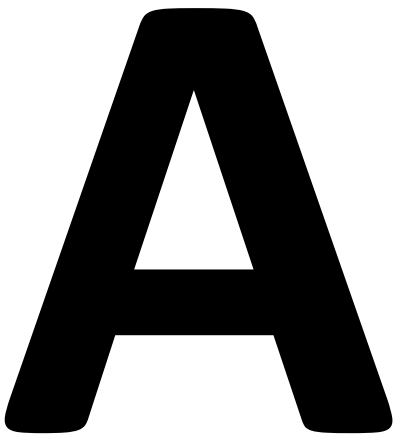
Filled

Graphics Display Hardware

Vector (calligraphic, stroke, random-scan)

- Driven by display commands
  - (move (x, y), char("A") , line(x, y)…)
- Survives as "scalable vector graphics"

A



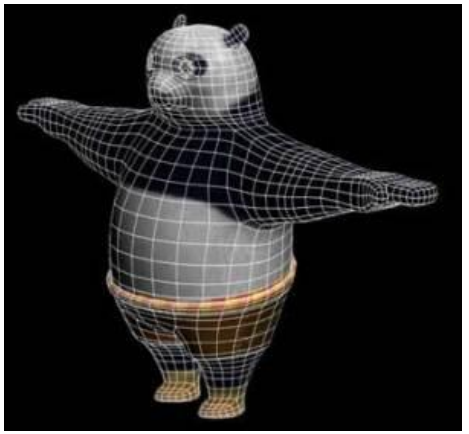
Raster (TV, bitmap, pixmap) used in displays and laser printers

- Driven by array of pixels (no semantics, lowest form of representation)
- Note "jaggies" (aliasing errors) due to discrete sampling of continuous primitives

A

Application Distinctions: Two Basic Paradigms

Sample-based graphics vs Geometry-based graphics

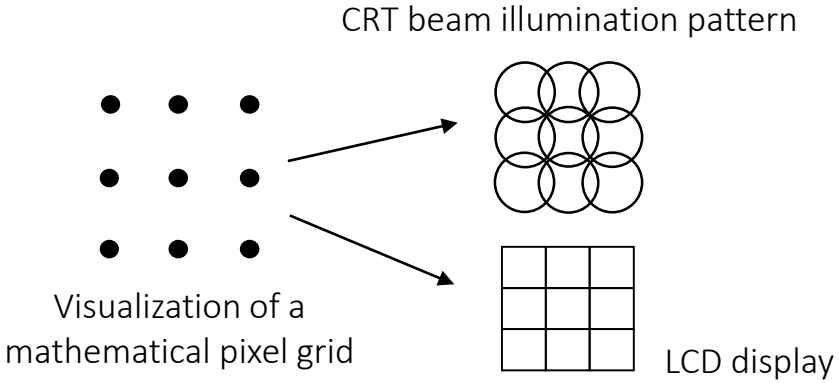


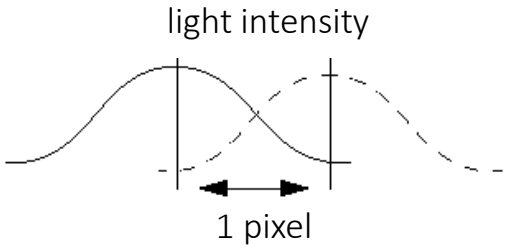
Sample-based graphics

- **Sample-based graphics:** Discrete samples are used to describe visual information
  - pixels can be created by digitizing images, using a sample-based "painting" program, etc.
  - often some aspect of the physical world is sampled for visualization, e.g., temperature across the US
  - example programs: Adobe Photoshop™, GIMP™ , Adobe AfterEffects™

Sample-based graphics

- Pixels are point locations with associated sample values, usually of light intensities/colors, transparency, and other control information
- When we sample an image, we sample the point location along the continuous signal and we cannot treat the pixels as little circles or squares, though they may be displayed as such





Can't visually resolve adjacent pixels on CRT

Sample-based graphics

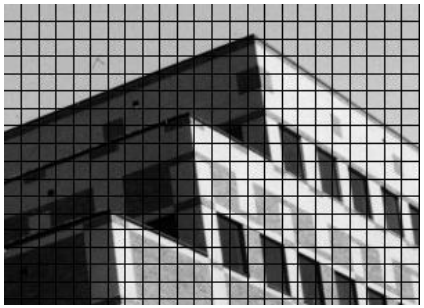
- Samples created directly in paint-type program, or by sampling of continuous (analog) visual materials (light intensity/color measured at regular intervals) with many devices including:
  - flatbed and drum scanners
  - digital still and motion (video) cameras
- Sample values can also be input numerically (e.g., with numbers from computed dataset)
- Once an image is defined as pixel-array, it can be manipulated
  - **Image editing**: changes made by **user**, such as cutting and pasting sections, brush-type tools, and processing selected areas
  - **Image processing**: algorithmic operations that are performed on image (or pre-selected portion of image) without user intervention. Blurring, sharpening, edge-detection, color balancing, rotating, warping. These are front-end processes to **Computer Vision**.

Sampling an Image

- Let's do some sampling of the building image

- A color value is measured at every grid point and used to color corresponding grid square

0 = white, 5 = gray, 10 = black



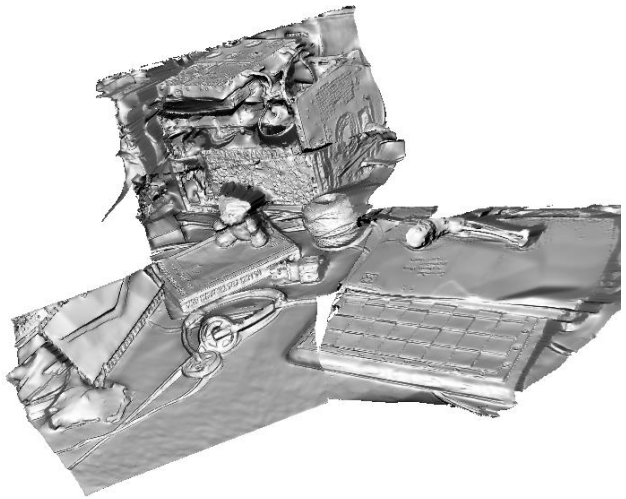
- Crude sampling and image reconstruction method creates blocky image

What's the Advantage?

- Once image is defined in terms of colors at (x, y) locations on grid, can change image easily by altering location or color values
- E.g., if we reverse our mapping above and make 10 = white and 0 = black, the image would look like this:
- Pixel information from one image can be copied and pasted into another, replacing or combining with previously stored pixels

What's the Disadvantage?

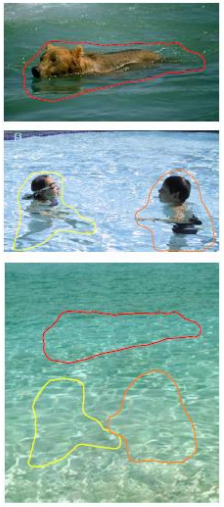
- WYSIAYG (What You See Is All You Get): No additional information
  - no depth information
  - can't examine scene from different point of view
  - at most can play with the individual pixels or groups of pixels to change colors, enhance contrast, find edges, etc.
  - But increasingly great success in image-based rendering to fake 3D scenes and arbitrary camera positions. New images constructed by interpolation, composition, warping and other operations.
  - For a computational and cognitive science perspective, Take Thomas Serre's Computational Vision.



Question?

- How to recover the high-definition image from a low-resolution sample?

- How to paste some objects into a new image?



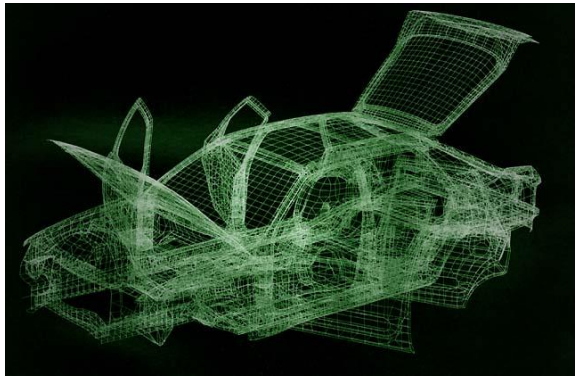
sources/destinations

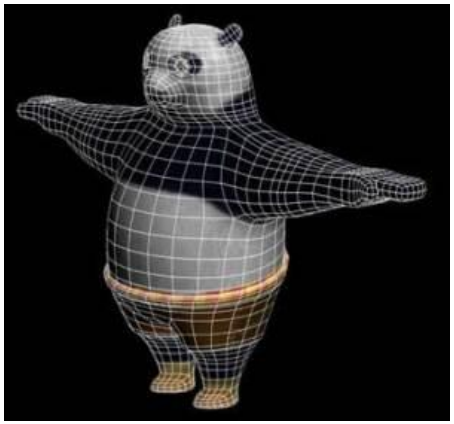
cloning

seamless cloning

Geometry-Based Graphics

- **Geometry-based graphics** (also called scalable vector graphics or object-oriented graphics): geometrical model is created, along with various appearance attributes, and is then sampled for visualization (rendering, a.k.a image synthesis)
  - often some aspect of physical world is visually simulated, or “synthesized”
  - examples of 2D apps: Adobe Illustrator™, Adobe Freehand™, Corel CorelDRAW™
  - examples of 3D apps: Autodesk’s AutoCAD™, Autodesk’s (formerly Alias|Wavefront’s) Maya™, Autodesk’s 3D Studio Max™





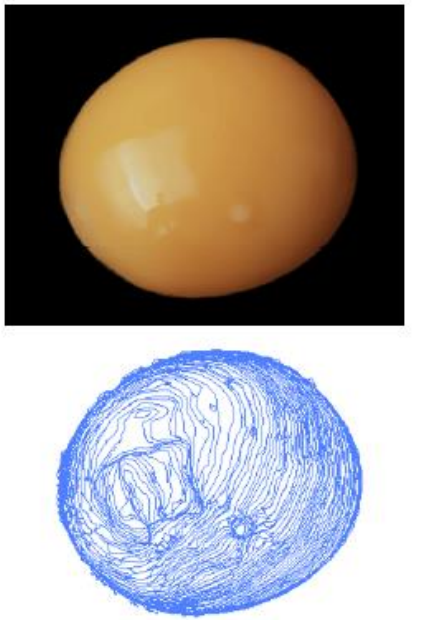
Geometry-Based Graphics

- Geometry-based graphics applications
  - Store mathematical descriptions, or “models,” of geometric elements (lines, polygons, polyhedrons, polygonal meshes…) and associated attributes (e.g., color, material properties).
  - Geometric elements are primitive shapes, primitives for short.
  - Images are created via sampling of geometry for viewing, but not stored as part of model.
  - Users cannot usually work directly with individual pixels in geometry-based programs; as user manipulates geometric elements, program resamples and redisplays elements
- Increasingly rendering combines geometry- and sample-based graphics, both as performance hack and to increase quality of final product.
  - geometric characters on painted or scanned scene images

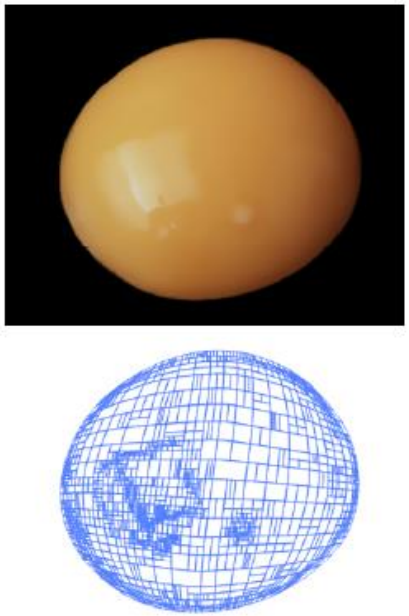
Image Vectorization

- From Sample-based to Geometry-based Graphics

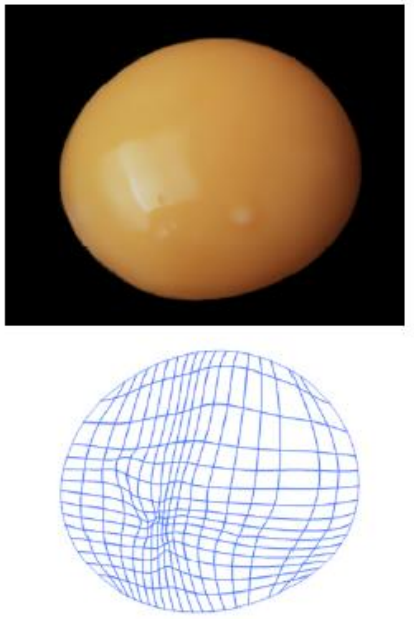
(a) input raster image



(b) Adobe Live Trace result



(c) subdivision mesh

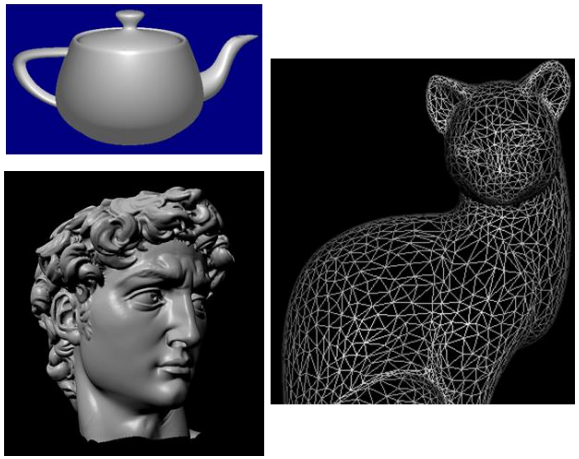


(d) optimized gradient mesh

Outline

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

Modeling
(Creating 3D Geometry)

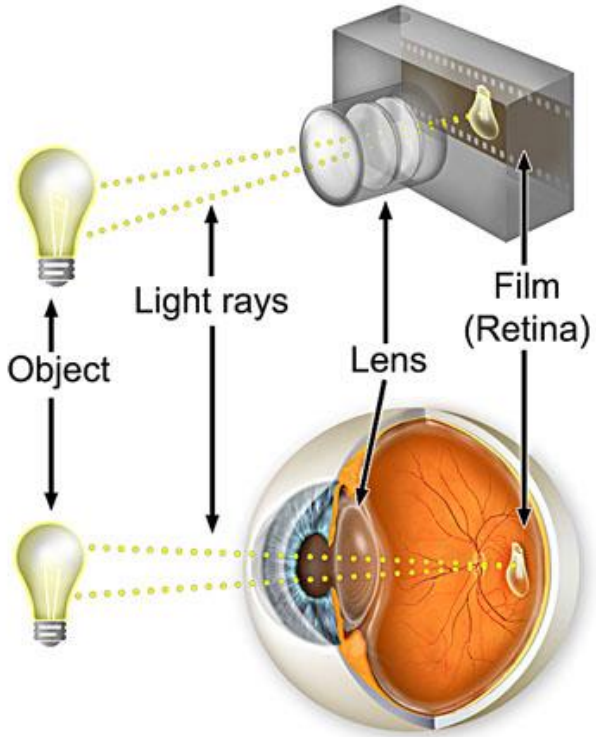


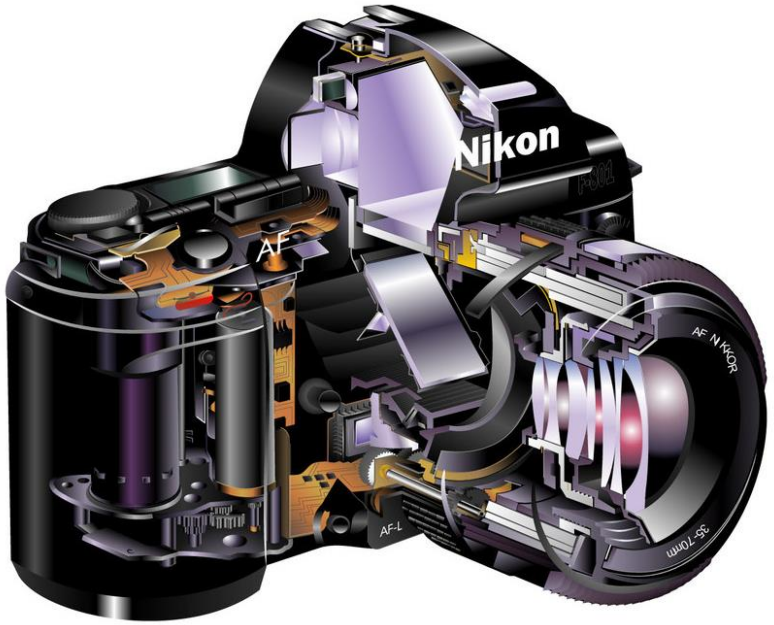
→

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

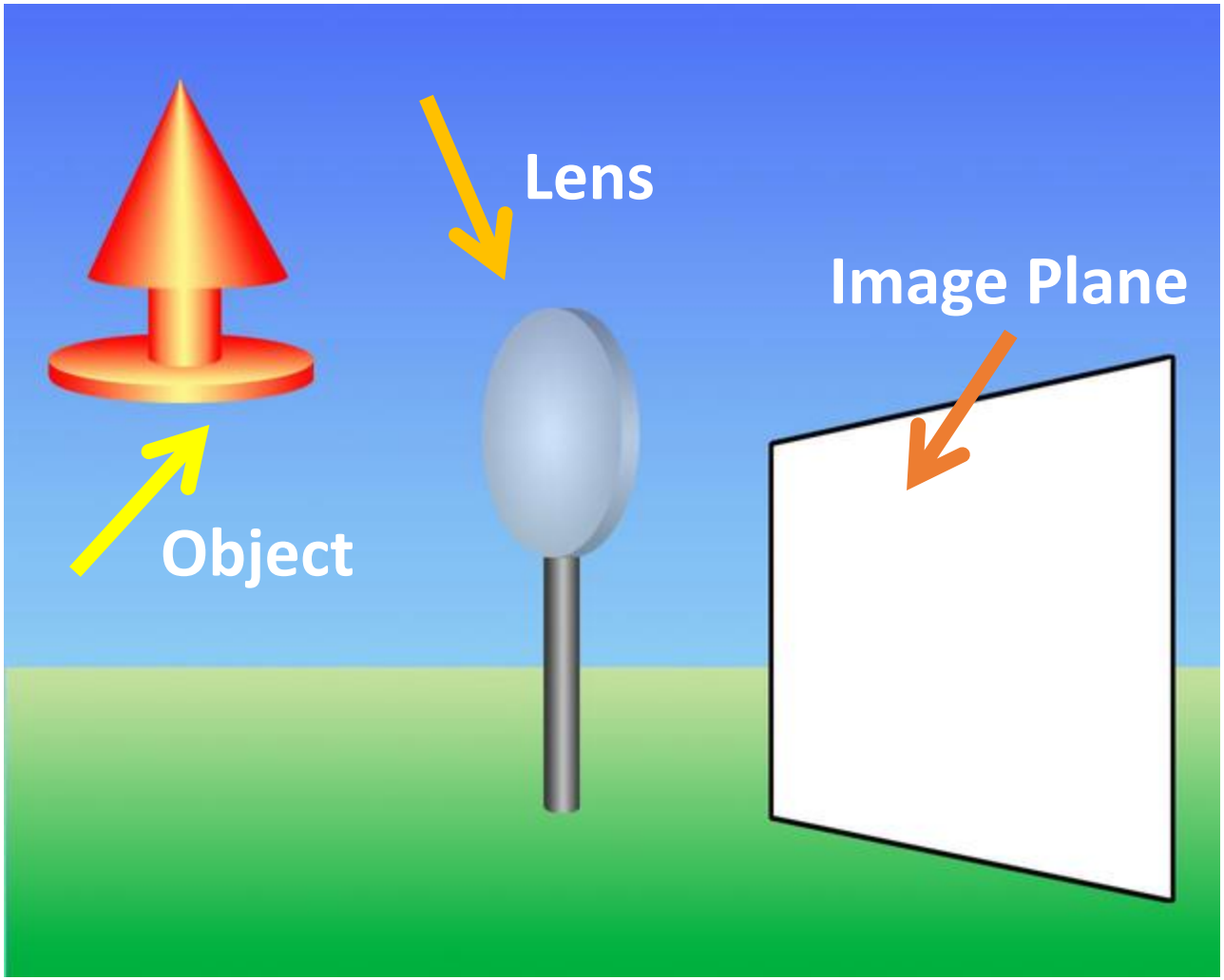
Physical Imaging System

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

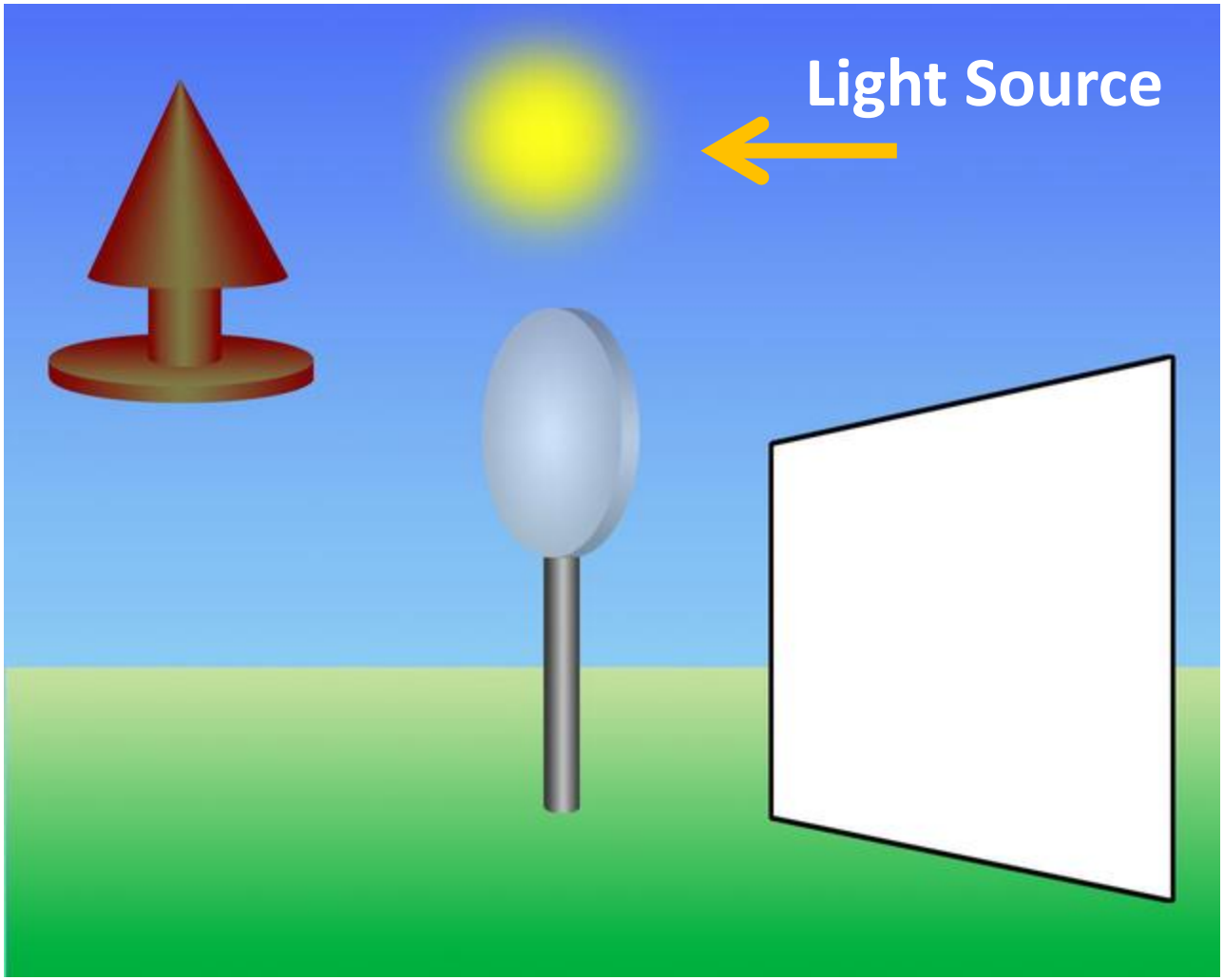




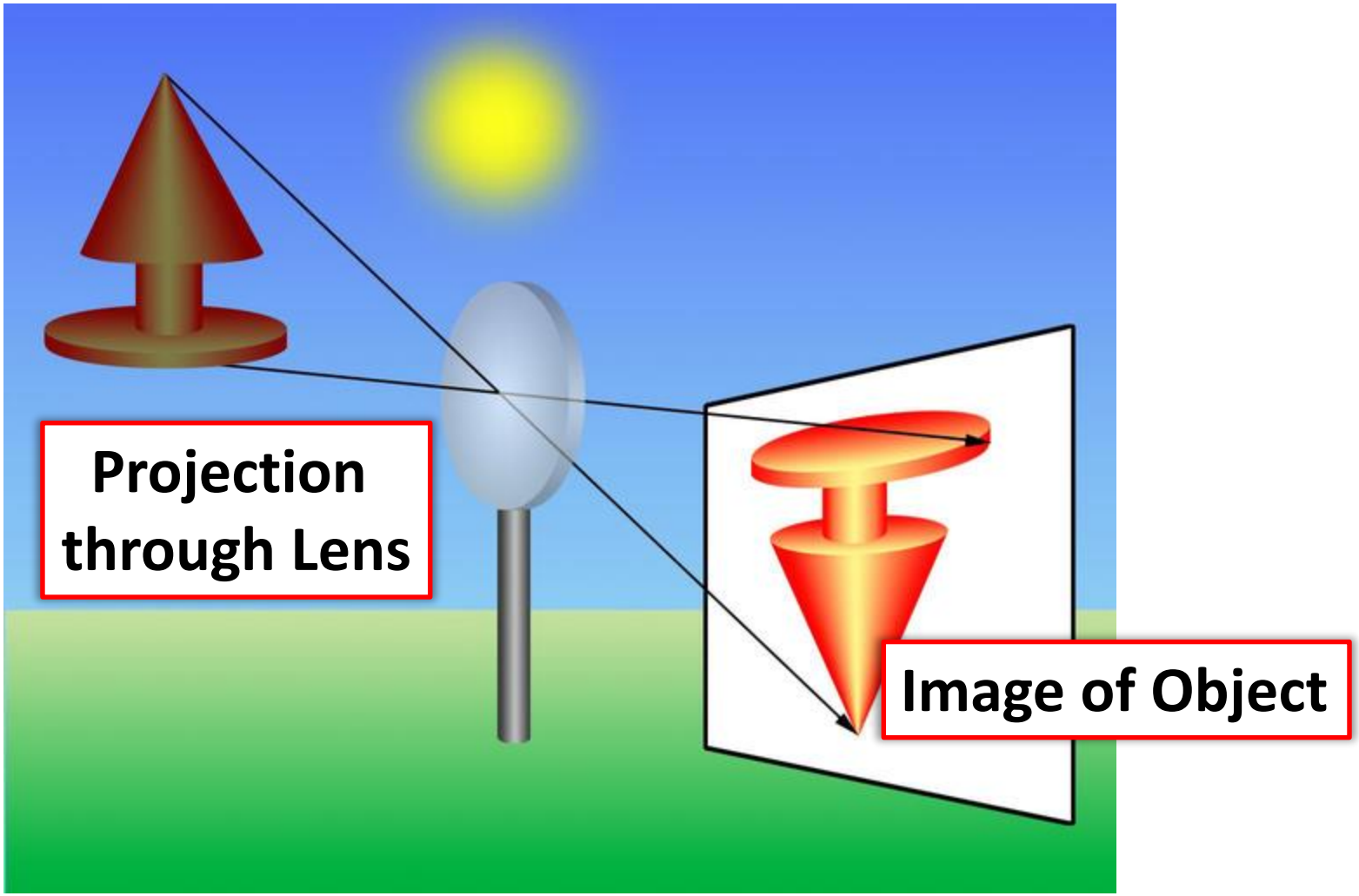
Imaging Principle



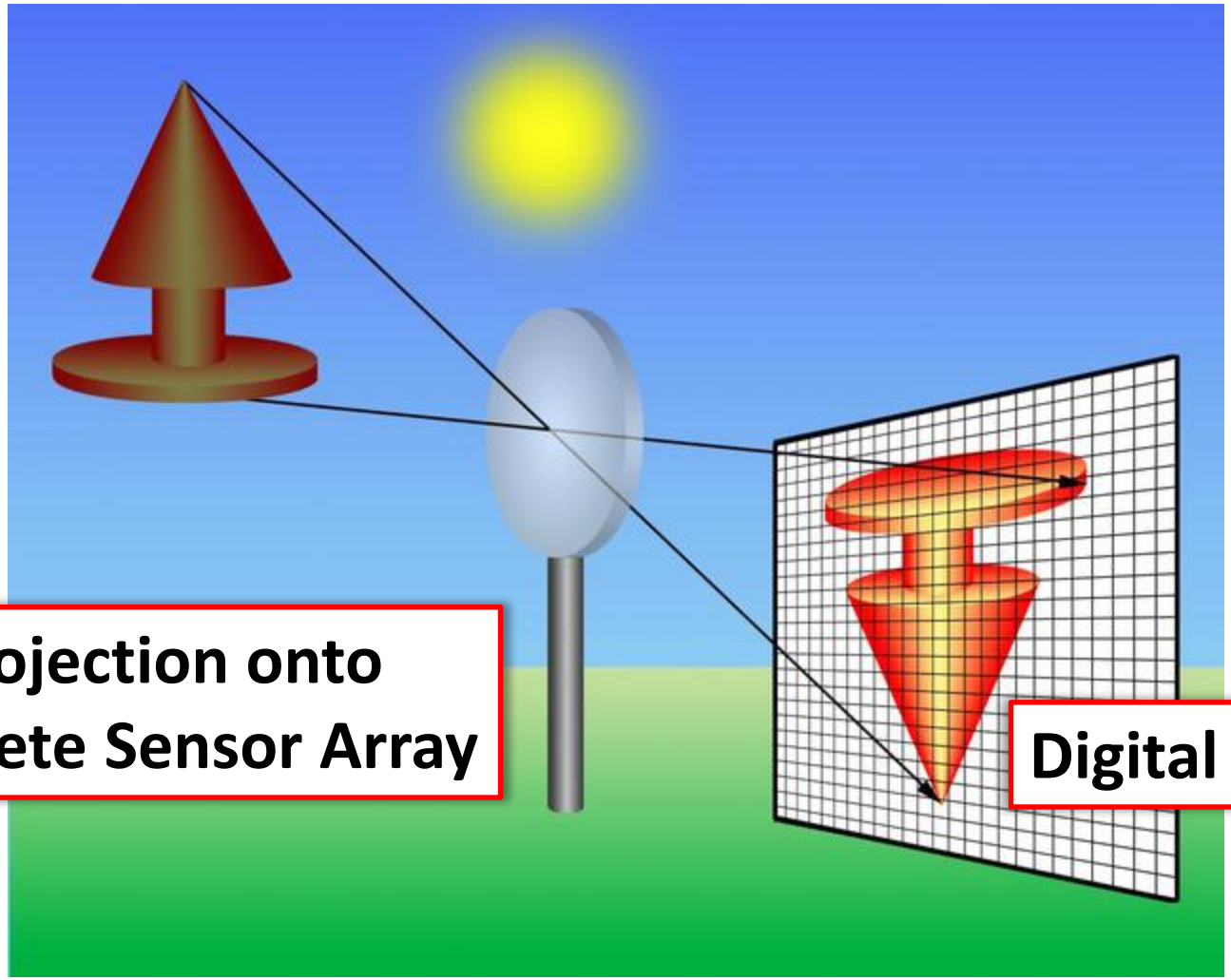
Imaging Principle



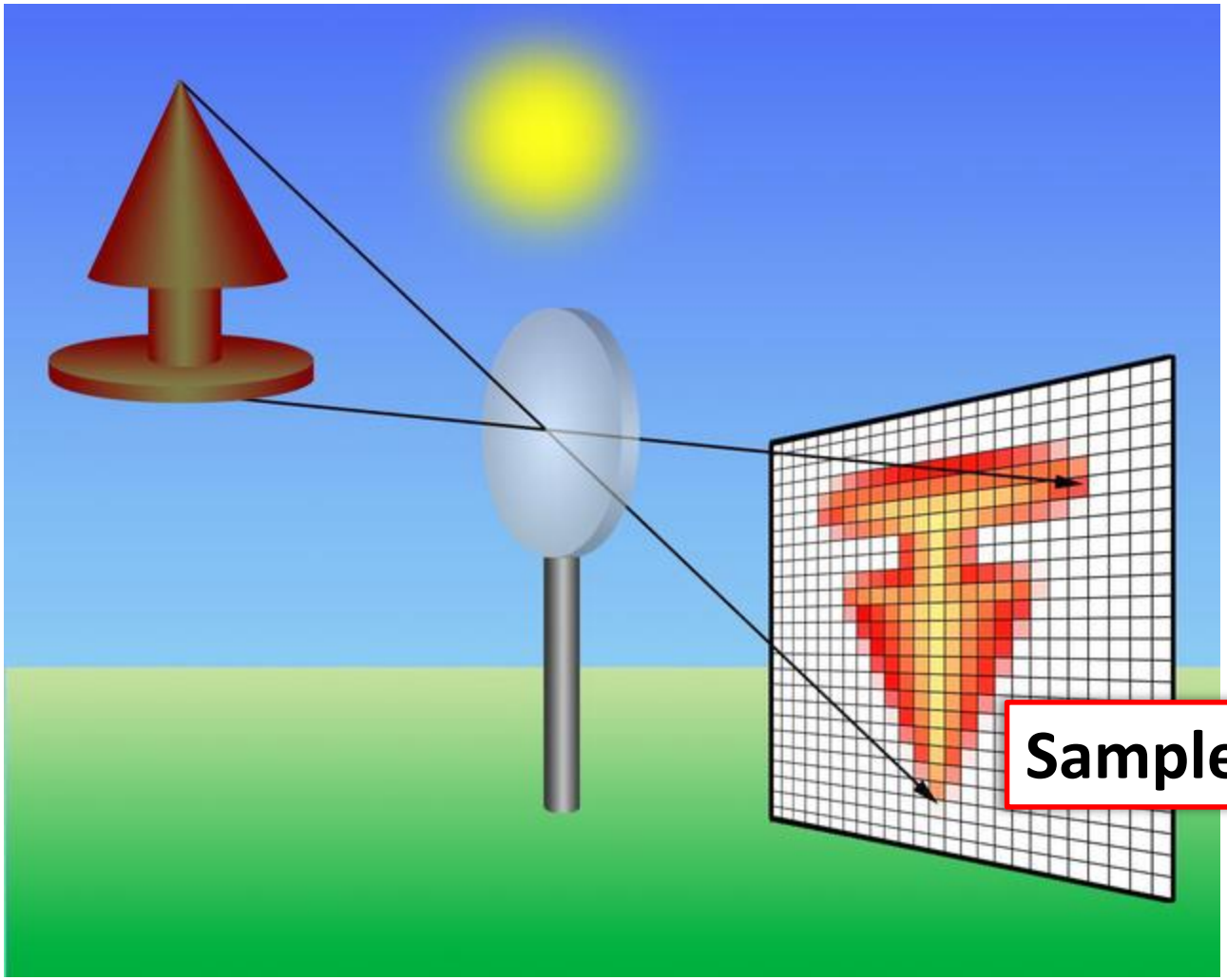
Imaging Principle



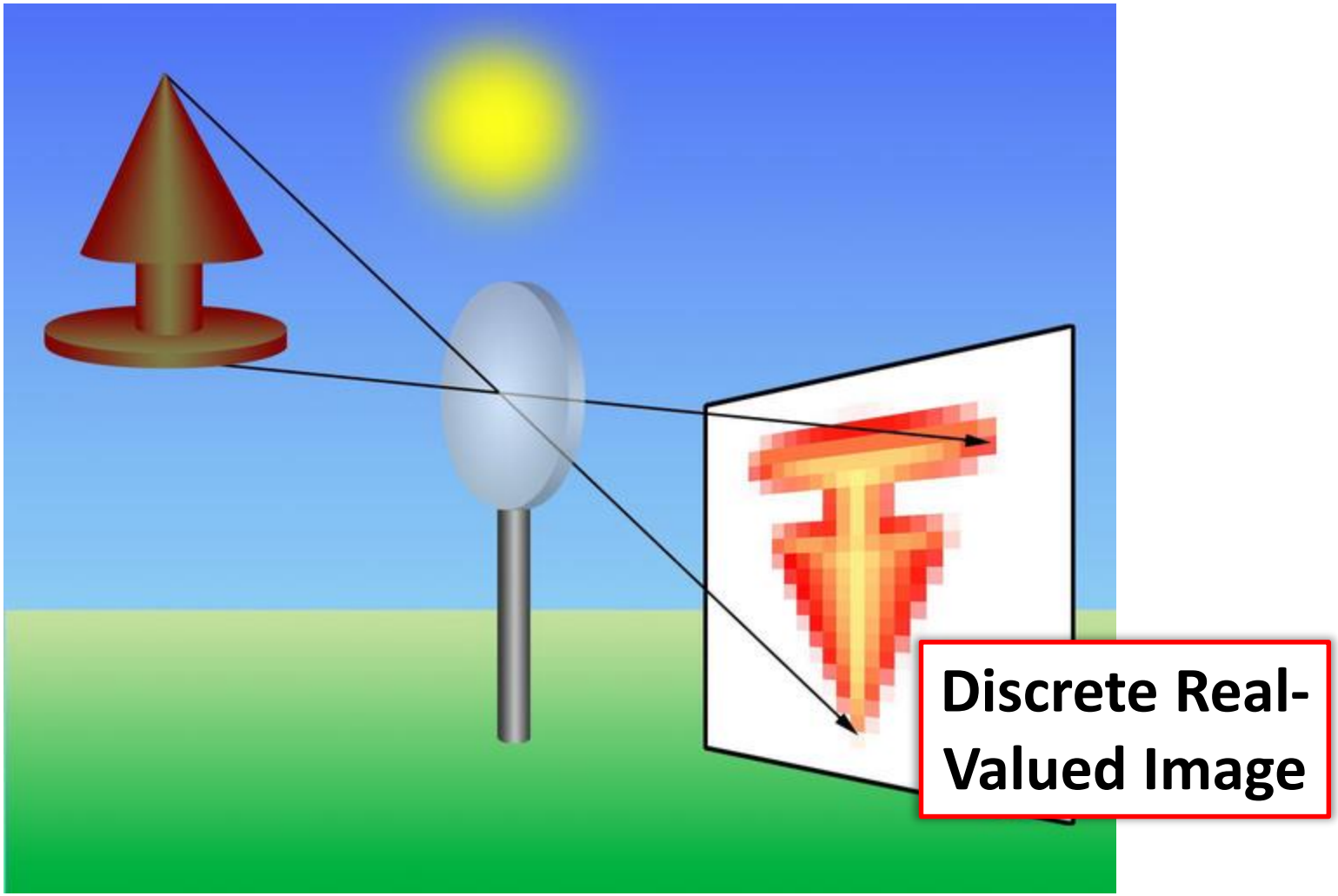
Imaging Principle



Imaging Principle

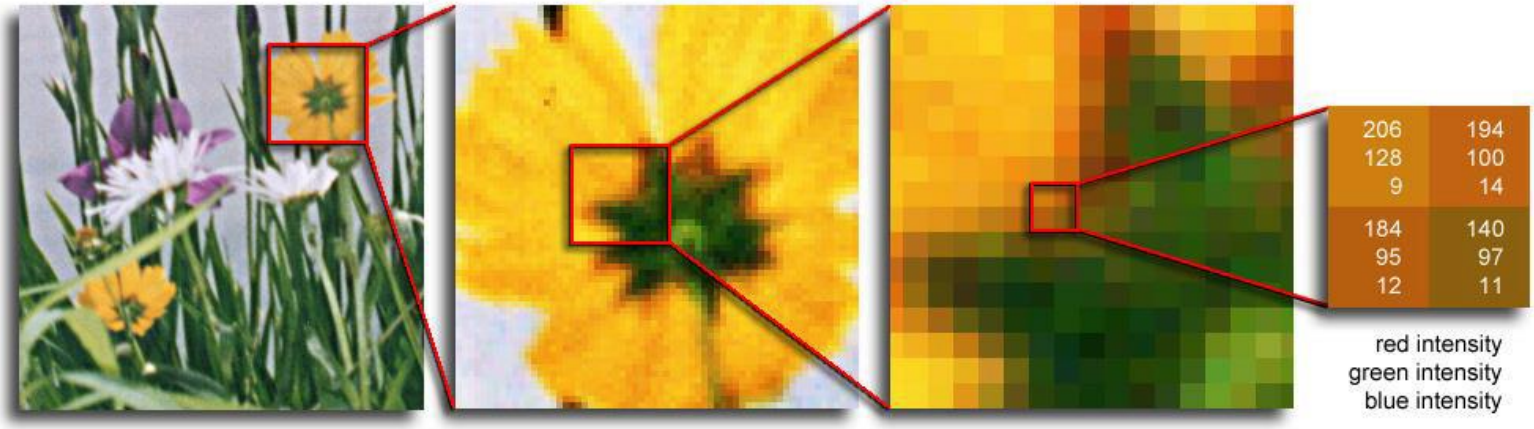


Imaging Principle

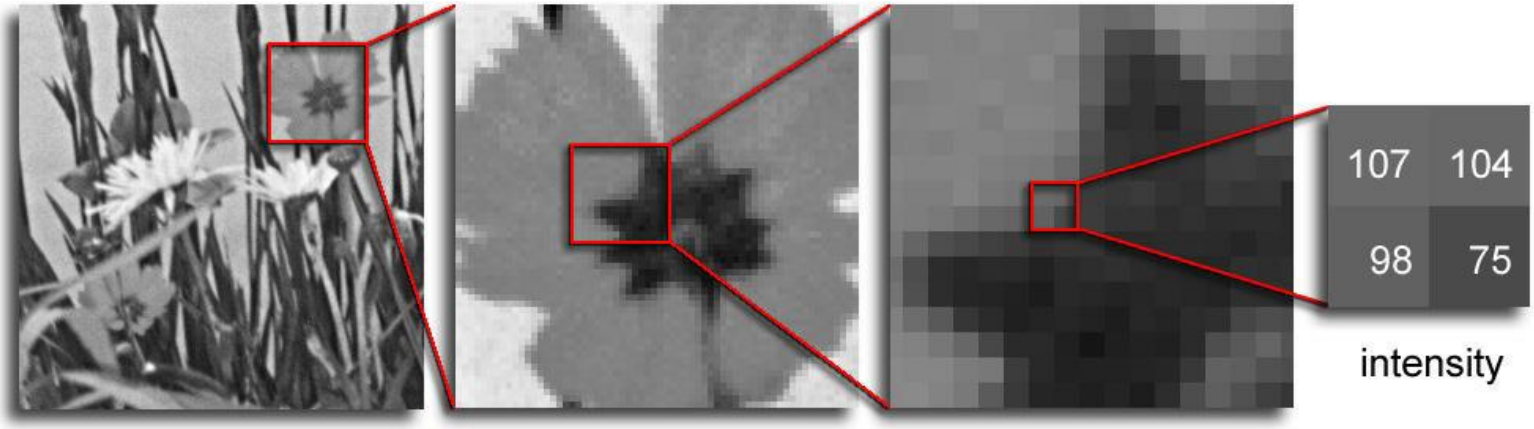


Digital Image

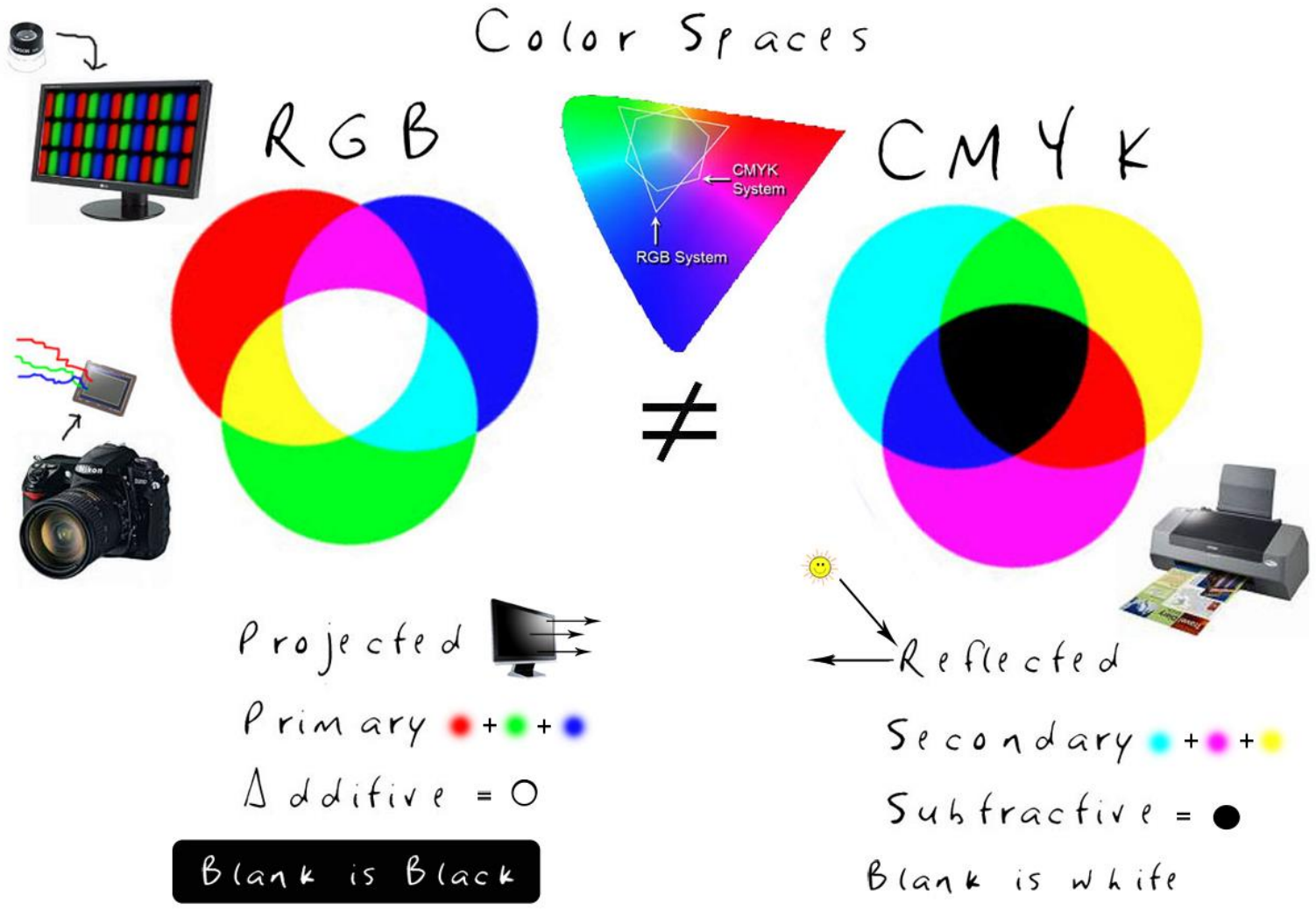
Color Image



Grey Image



Color Space


Color Spaces
RGB
CMYK
CMYK System
RGB System
≠
Projected
Primary + + =
Additive = O
Blank is Black
Reflected
Secondary + + =
Subtractive = ●
Blank is White

Light---Visible Light

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

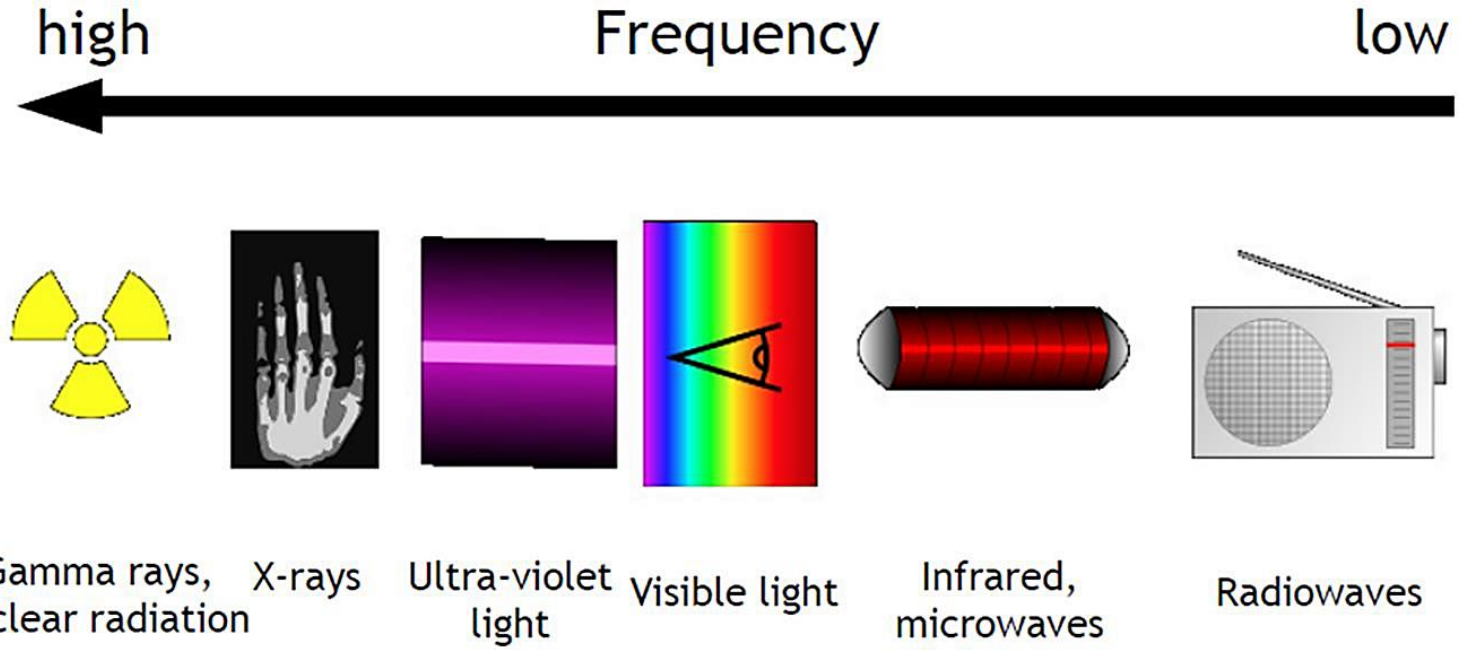
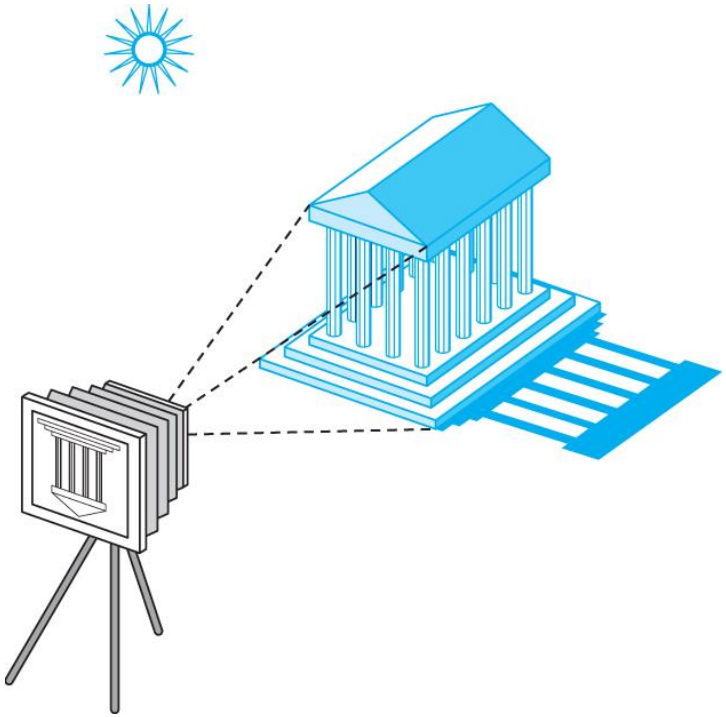


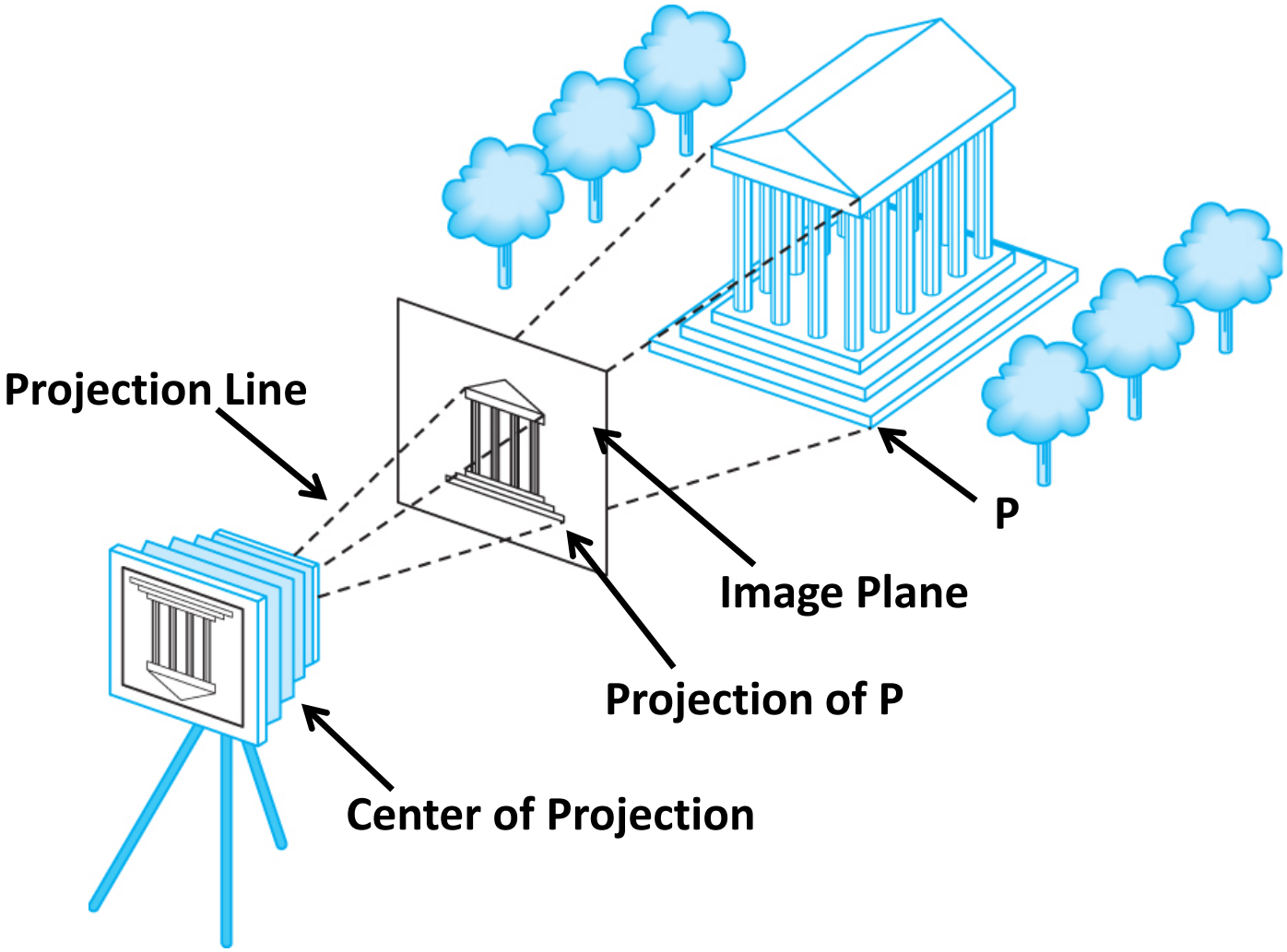
Image Synthetic Element

- Object
- Observer
- Light



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

Synthetic camera model



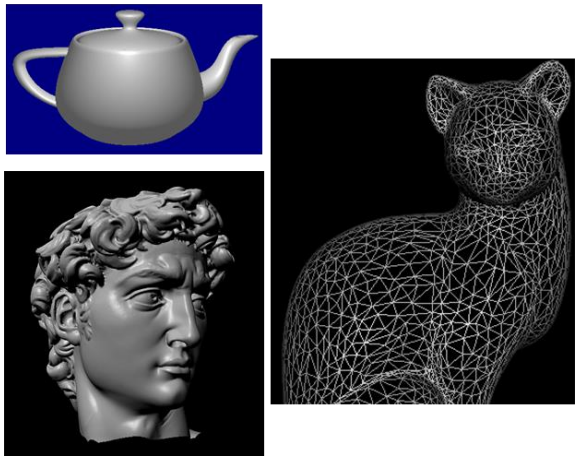
Advantages of Synthetic camera model

- Object (物体), Observer (观察者) and light (光源) is complete independent
- 2D graphics is a special case of 3D
- 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)

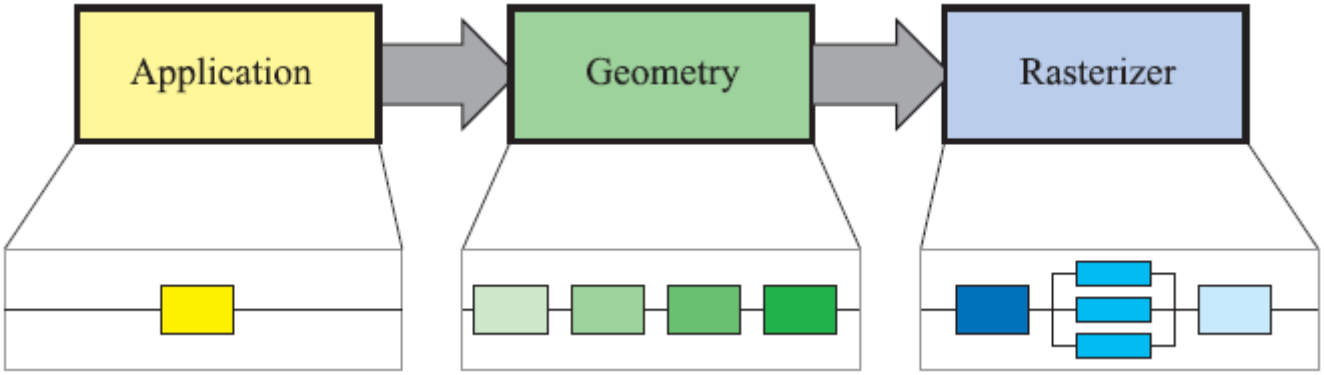
Demo - World War Z

Graphics Rendering Pipeline

- What is Rendering?
  - “Rendering is a process that takes as its input a set of objects and produces as its output an array of pixels.”

    -- 《Fundamentals of Computer Graphics (3rd Edition)》
- What is the rendering process (Graphic Pipeline)?
  - the sequence of steps used to create a 2D raster representation of a 3D scene.

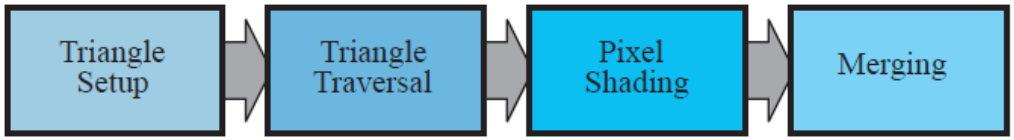
Rendering Pipeline Stages



Rendering Pipeline Stages

- Application
  - e.g. collision detection, global acceleration algorithms, animation, physics simulation….. (On CPU)
- Geometry
  - Deal with transforms, projection. Computes what is to be draw, how it should be drawn, and where it should be drawn (On GPU)

- Rasterizer
  - 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)

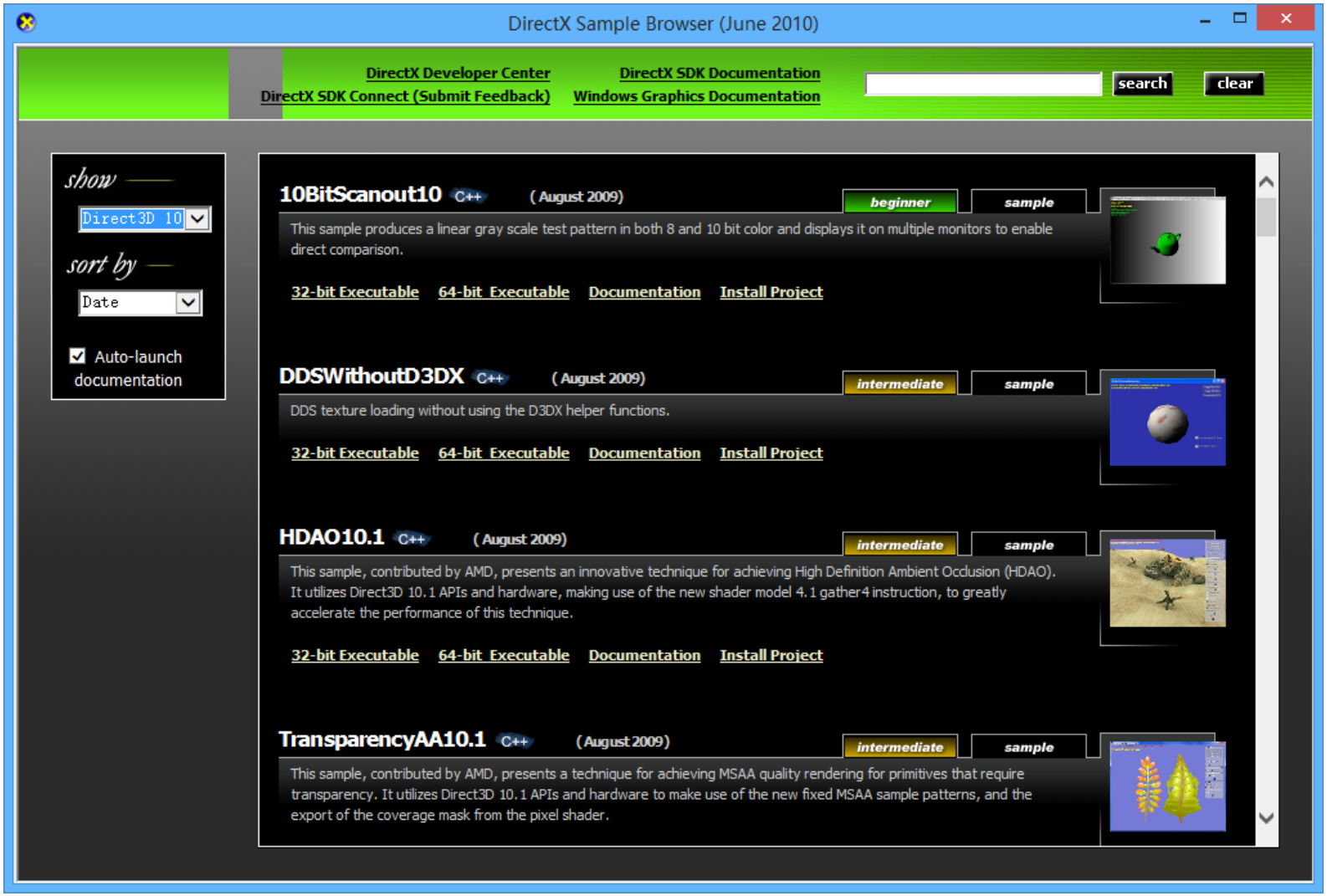


Rendering Pipeline

- Abstract model
  - sequence of operations to transform geometric model into digital image
  - graphics hardware workflow
- Underlying API model for programming graphics hardware
  - OpenGL
  - Direct 3D

- Many actual implementations

Direct 3D Samples

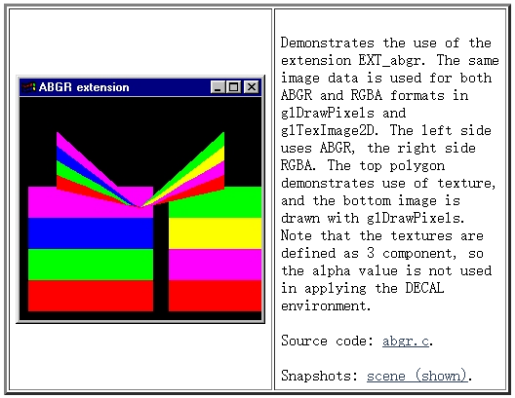


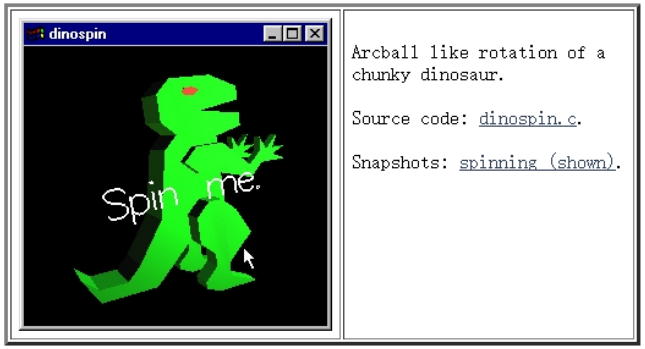
OpenGL Samples

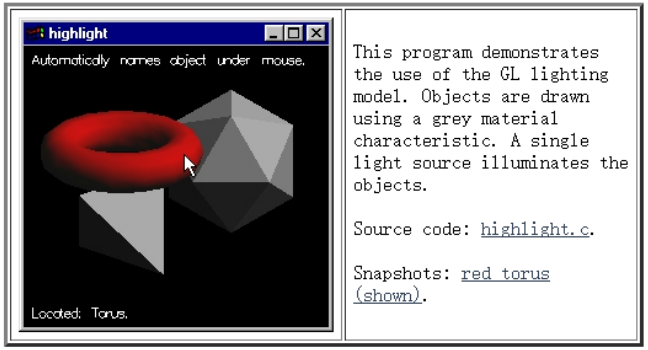
- Learn OpenGL demos

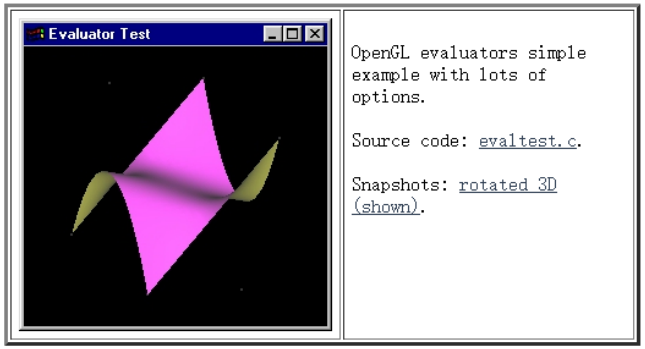
https://learnopengl.com/ or https://learnopengl-cn.github.io/

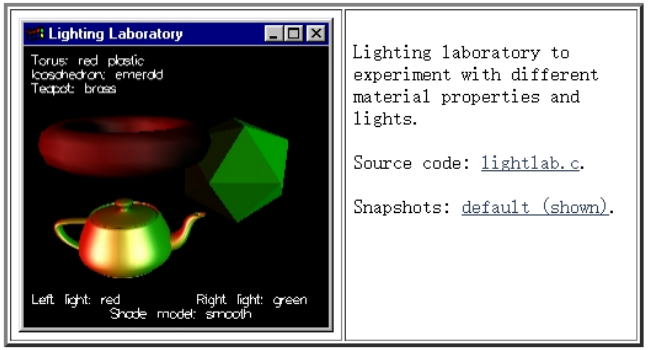
examples.zip











Samples

- **NVIDIA GameWorks™ Samples Overview**

  https://developer.nvidia.com/gameworks-samples-overview

NVIDIA GameWorks™ OpenGL Samples

Get the documentation or download the NVIDIA GameWorks™ OpenGL samples here:

Download ›

New Browse or Clone Source from GitHub:

GitHub ›

New samples in 2.0:

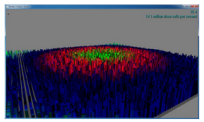
- Blended Antialiasing Sample
- Cascaded Shadow Mapping Sample
- Conservative Rasterization Sample
- Normal-Blended Decal Sample
- Weighted, Blended, Order-independent Transparency Sample

Bindless Graphics Sample

- Category: **Performance**

This sample demonstrates the large performance increase in OpenGL that is made possible by 'Bindless Graphics'. These extensions allow applications to draw large numbers of objects with only a few setup calls, rather than a few calls per object, thus reducing the driver overhead necessary to render highly populated scenery.

Docs ›

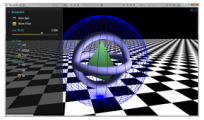


★NEW:Blended AA

- Category: **Performance, Visuals**

This sample implements a two-pass additive blending anti-aliasing technique using Target-Independent Rasterization (TIR), which should give comparable results to MSAA with a reduced memory footprint.

Docs ›

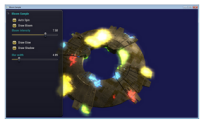


Bloom Sample

- Category: **Visuals**

This sample demonstrates creating a glow effect by post-processing the main scene. It heavily leverages FBO render targets across multiple steps/passes with custom effects processing shaders. It also integrates shadow mapping to demonstrate self-illumination cutting through the shadow effects.

Docs ›



NVIDIA GameWorks™ DirectX Samples

Get the documentation or download the NVIDIA GameWorks™ Direct3D samples here.

Download ›

New Browse or Clone Source from GitHub:

GitHub ›

New samples in 1.2:

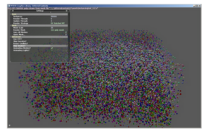
- Antialiased Deferred Rendering
- Motion Blur D3D Advanced Sample

D3D Deferred Contexts Sample

- Category: **Performance**

This sample shows how to use D3D11 Deferred Rendering contexts to lower the CPU overhead and improve performance when rendering large numbers of objects per frame, in situations where instancing is not feasible.

Docs ›

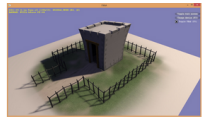


FXAA 3.11 Sample

- Category: **Performance**,**Visuals**

This sample presents a high performance and high quality screen-space software approximation to anti-aliasing called FXAA.

Docs ›

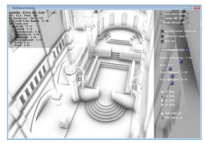


Deinterleaved Texturing Sample

- Category: **Performance**,**Visuals**

This sample demonstrates how a large, sparse and jittered post-processing filter (here a SSAO pass with a 4x4 random texture) can be made more cache-efficient by using a Deinterleaved Texturing approach.

Docs ›



OpenGL Rendering Pipeline