CS230 : Computer Graphics
Lecture 2

Tamar Shinar
Computer Science & Engineering
UC Riverside
Raster Devices and Images
Raster Devices

- Raster displays show images as a rectangular array of pixels
- Most printers are also raster devices
  - Image is made by depositing ink at points on a grid
- Digital cameras - have image sensors made of grid of light-sensitive pixels (2D array)
- Scanner - linear array of pixels swept across page to create grid of pixels (1D array)
virtually all graphics system are raster based, meaning the image we see is a raster of pixels or a rectangular array of pixels. Here a raster scan device display an image as a set of discrete points across each scanline.
Displays are either transmissive or emissive.

One pixel of an LCD display:

(LEFT) In the off state the front polarizer blocks all the light that passes the back polarizer. In the on state the liquid crystal rotates the polarization of the light so it can pass through the front polarizer.

The degree of rotation can be adjusted by an applied voltage.

(RIGHT) LED display.
Raster Display

red, green, blue subpixels

get different colors by mixing red, green, and blue
this is from an LCD monitor
printers are also raster-based. image is made out of points on a grid
What is an image?

**Continuous image**

\[ I : R \to V \]

\( R \subset \mathbb{R}^2 \)

\( V = \mathbb{R}^+ \) (grayscale)

\( V = (\mathbb{R}^+)^3 \) (color)

An (continuous) image is a function defined over some 2D area, that maps points to intensity level.
What is an image?

**Sampled image**

\[ I : \mathbb{R} \rightarrow \mathbb{V} \]
\[ R \subset \mathbb{Z}^2 \]
\[ V = [0, 1] \quad \text{(grayscale)} \]
\[ V = [0, 1]^3 \quad \text{(color)} \]

\[ n_x = \text{number of columns} \]
\[ n_y = \text{number of rows} \]

\[ [-0.5, n_x - 0.5] \times [-0.5, n_y - 0.5] \]

Each pixel value represents the **average color** of the image over that pixel’s area.
A **raster image** is 2D array storing pixel values at each pixel (picture element)
3 numbers for color
alternative: **vector image** -- essentially a set of instructions for rendering an image
Bit depth - defined by device standards

<table>
<thead>
<tr>
<th>Bit-Depth</th>
<th>Number of Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 (monochrome)</td>
</tr>
<tr>
<td>2</td>
<td>4 (CGA)</td>
</tr>
<tr>
<td>4</td>
<td>16 (EGA)</td>
</tr>
<tr>
<td>8</td>
<td>256 (VGA)</td>
</tr>
<tr>
<td>16</td>
<td>65,536 (High Color, XGA)</td>
</tr>
<tr>
<td>24</td>
<td>16,777,216 (True Color, SVGA)</td>
</tr>
<tr>
<td>32</td>
<td>16,777,216 (True Color + Alpha Channel)</td>
</tr>
</tbody>
</table>

(Note alpha)

(Humans can perceive ~10,000,000 colors)

In practice, it is sufficient for pixels to have a bounded range e.g., [0,1]

They are represented in integers
Monitor Gamma

displayed intensity = (max intensity) $a^\gamma$

monitors convert pixel values, $a$, into displayed intensities
monitors are nonlinear with respect to input
Gamma Correction

displayed intensity = (max intensity)\left(a\frac{1}{\gamma}\right)^\gamma

find gamma using, e.g., checkboard
then gamma-correct the input

find gamma, so that you can give the monitor \(a^{\{1/\gamma\}}\)
- find \(a\) such that \(a^{\{\gamma\}} = .5\) through checkboard test and solve for \(\gamma\)
Color representation

**Additive color** – Primary colors are red, green, blue. Form a color by adding these. CRTs, projectors, LCD displays, positive film

**Subtractive color** – Form a color by filtering white light with cyan, magenta, and yellow filters. Printing, negative film
Compositing: two different interpretations: **pixel coverage** (fraction of pixel covered) and **blending**

\[ c = \alpha c_f + (1 - \alpha) c_b \]
Ray Tracing
up to 16 reflections per ray
shallow depth of field, area light sources, diffuse interreflection
Basic Algorithm

for each pixel

1. cast view ray:
compute view ray from camera through pixel into scene

2. intersect:
find intersection of ray with closest object

3. shade:
compute the color of the intersection point
Ray Tracing Program

for each pixel do
  compute viewing ray
  if ( ray hits an object with t in [0, inf] ) then
    compute $n$
    evaluate shading model and set pixel to that color
  else
    set pixel color to the background color
Object-oriented design

class Surface
{
    public:
        bool Intersection(RAY& ray)=0;
        Box Bounding_Box()=0;
}

Other objects: Ray, Light, Material, Camera, Film, World
Simple Ray Tracer