Today’s Agenda

Image Formation
Image Formation at a Glance

Exposure

This is light transport.

Illumination is generated at light sources, propagates thru world.

Interacts with objects in scene.

Elements of image formation:
- Illumination sources
- Objects
- Viewer (e.g., camera and eye)
- Attributes of materials

Reflection

Illumination

Absorption
The Response of Cones to Color

Three kinds of cones: S, L, and M

- S cones respond to blue
- M cones respond to green
- L cones respond to red

Response levels to illumination are

\[ s = \int S(\lambda)P(\lambda) d\lambda \]
\[ m = \int M(\lambda)P(\lambda) d\lambda \]
\[ l = \int L(\lambda)P(\lambda) d\lambda \]

- where \( s, m, l \) are scalars
- this implies that we humans perceive light as a 3-D space

And this 3D space is very complex

- for instance, it’s not Euclidean

![Graph showing response levels for S, M, and L cones.](image)
Humans Perceive a 3D Color Space

We can’t distinguish all distributions

- **metamers**: two colors with different spectral distributions but identical $s$, $m$, $l$ values
- these would look identical
The Simplest Camera: Pinhole Camera

Mount a piece of film in a lightproof box with a single pinhole in it

Pinhole focuses light on the film

- Lens degenerates to a point – no distortion
- One-to-one correspondence between 3D object point and 2D image point
- only select light ray can go through the hole (the hole is reduced to a point)
- note that image on film is flipped upside down
How to make pinhole camera?

- [http://www.exploratorium.edu/light_walk/camera_todo.html](http://www.exploratorium.edu/light_walk/camera_todo.html)
Synthetic Camera Model

- projector
- image plane
- projection of \( p \)
- center of projection
Only one coordinate system – camera coordinate system
The Equation of Perspective Projection

Cartesian coordinates:

- We have, by similar triangles, that

\[(x, y, z) \rightarrow (x', y', z') = (-f \frac{x}{z}, -f \frac{y}{z}, -f)\]

- Ignore the third coordinate, and assume the image plane is before the camera, we get

\[(x, y, z) \rightarrow (u, v) = \left(\frac{f}{z}x, \frac{f}{z}y\right)\]

3D object point \(\rightarrow\) 2D image point

The perspective projection is non-linear!
Properties of Perspective Projection

Points project to points

Lines project to lines

Planes project to the whole or half image
  • A plane may only has half of its area in the projection side

Scaling and foreshortening

Angles are not preserved
  • Parallel lines may be not projected to parallel lines unless they are parallel to the image plane

Degenerate cases
  • Line through focal point projects to a point.
  • Plane through focal point projects to line
The Structure of a Typical Camera

Film is light sensitive material

Lens focuses light on film

Aperture is the opening of lens
  • opening may vary in size
  • controls the total energy of incoming lights

Shutter restricts access to film
  • can open for variable periods
  • controls total energy that hits the film

Film is replaced by Charged Couple Device (CCD) in a digital camera
Refractive Lenses

Refraction happens when light rays travel between materials
  • results from dependence of the speed of light on the material

Real cameras use refractive lenses
  • typically made of glass or plastic
  • bend incoming light rays
  • parallel incoming rays converge on the focal point
Refractive Lenses
**Basic Optics: Thin Lens**

**Field of View:** \( \omega = 2 \arctan \frac{d}{f} \)

Depth of view (DOF) is inversely proportional to the focus length (f) and inversely proportional to the aperture (d)
Continuous representation needs to be sampled and quantized to generate a discrete representation.

Each image is represented by a rectangular grid of pixels $P[x,y]$

Each pixel $p$ will store a color value
- RGB triple for color images
- single value for grayscale (or monochrome) images
Basic Color Representation in Graphics

For each pixel, we will treat colors as a 3-D space of \((r, g, b)\) triples

- all colors will be composed from three primary colors: red, green, blue
- the value of each \((r, g, b)\) is between 0 and 1
- coefficients represent relative contribution of each primary

\[
\begin{align*}
\text{Red} & : (1,0,0) \\
\text{Green} & : (0,1,0) \\
\text{Blue} & : (0,0,1) \\
\end{align*}
\]

Colors along Red axis

\[
\begin{align*}
(0,0,0) & \quad \rightarrow \quad 0.6 \\
(0,0,1) & \quad \rightarrow \quad 0.0 \\
(0,1,0) & \quad \rightarrow \quad 0.8 \\
\end{align*}
\]

\((0.6, 0.0, 0.8)\)

Diagram showing the combination of red, green, and blue colors to produce a purple color.
Raster Image Representation

Can separate RGB color image into 3 distinct color channels
  • each by itself is a monochrome image
**Question**

What are those colors?

(r g b) = (0 0 0)

(r g b) = (1 1 1)
Generic Raster Display Systems

Graphics hardware maintains a 2-D array of pixels: the frame buffer
- values in the frame buffer control intensity of electron beams in CRT
- raster scan process is typically performed at 60–100 Hz

Frame buffers are characterized by
- resolution: dimensions in pixels (e.g. 1024 x 768)
- bit depth: # of bits per pixel (typically 8–24)

Color image and gray-scale image
- r=g=b → gray-scale image
- pixel intensity is used instead of RGB channels
Full-Color Displays

Each pixel contains 3 values, one for each of R, G, and B
- typically 24 bits/pixel = 8 bits/channel = values of 0–255
- integer values 0–255 correspond to floating points values 0–1
- integers are just more convenient in hardware implementation

Pixel values directly control intensity of electron beams
- R=0 implies red beam is off
- R=255 implies red beam at full intensity

24 bits/pixel generally considered “full-color”
- produces $2^{24} \approx 16$ million different colors
- high-end systems might support 36 bits/pixel or more
Color Display Via Lookup Tables

**Alternative to direct RGB values**
- single value per pixel
- typically 8 or 16 bits
- pixel value is an index into a color lookup table or palette

**Common when memory is scarce**
- can customize set of colors to image being displayed
  - 256 colors of your choice

**Also supports some handy tricks**
- can recolor entire image just by changing palette
- animating palette creates interesting effects (eg. glowing)
**Image Compositing**

Often want to combine a sequence of images together

- different parts of final image can come from different sources
- TV stations have been doing this for a long time

**Question:** how to handle the overlapped regions?
**Image Compositing**

Introduce a new alpha channel in addition to RGB channels

- the $\alpha$ value of a pixel indicates its transparency
  - if $\alpha=0$, pixel is totally transparent
  - if $\alpha=1$, pixel is totally opaque
- alternatively, can think of $\alpha$ as the fraction of the pixel actually covered by the stored color
- convenient to work with premultiplied colors

\[
P = \begin{bmatrix} r_p \\ g_p \\ b_p \\ \alpha_p \end{bmatrix} \implies P' = \begin{bmatrix} \alpha_p r_p \\ \alpha_p g_p \\ \alpha_p b_p \\ \alpha_p \end{bmatrix}
\]
Image Compositing

Compositing one image over another is most common choice
  - can think of each image drawn on a transparent plastic sheet
  - the final image is formed by stacking layers together

Given images $A$ & $B$, we can compute $C = A$ over $B$

$$C_{rgb} = \alpha_A A_{rgb} + (1 - \alpha_A) \alpha_B B_{rgb}$$

- if we pre-multiply $\alpha$ values, this simplifies to

$$C' = A' + (1 - \alpha_A)B'$$

This is only one possible compositing operator
  - there are in fact 12 possible ways of combining 2 images
Example: Image Compositing

Read RGBα values from frame buffer

Given RGB colors \( A = (0.8, 0.6, 1.0) \) and \( B = (1, 1, 1) \); \( \alpha_A = 0.5 \); \( \alpha_B = 0.2 \)

Premultiply: \( A' = \alpha_A A = (0.4, 0.3, 0.5) \) \hfill \( B' = \alpha_B B = (0.2, 0.2, 0.2) \)

\[
C' = A' + (1 - \alpha_A)B' = \begin{bmatrix}
0.5 \\
0.4 \\
0.6 \\
0.6
\end{bmatrix}
\]

\( \alpha_C = 0.6 \)

De-premultiply: \( C = C'/\alpha_C = (0.83, 0.67, 1.0) \)

Write \( C \) (RGBα values) back into frame buffer
Next Time: Basic Geometric Primitives

We’ll look at the simplest tools for representing geometry
  • lines, planes, triangles, and polygons

We’ll also look at some OpenGL basics
  • this will help you with your projects
Reading Assignment

Chapter 1 of Angel