1. Graphics Overview

Reading

Recommended:

- Foley et al., Sections 1.5, 4.2–4.5, 5.4.
- Hearn & Baker, Chapter 2.
- Woo et al., Chapters 1, 2, 8.
- Angel, Sections 1.6–1.8, Chapter 2.
- I. E. Sutherland. Sketchpad: a man-machine graphics communication system. *Proceedings of the Spring Joint Computer Conference*, 329–346, 1963.

Outline

1. High-level overview
2. Graphics hardware
3. Specifying a view in 2D
4. The 3D synthetic-camera model
5. Graphics API's
6. The geometry and graphics pipelines

Major areas in/around computer graphics

- Modeling
- Rendering
- Animation
- Visualization
- Hardware and I/O devices
- User interfaces
- Image processing
- Vision

Interdisciplinary!

- Computer science
- Mathematics
- Physics
- Engineering
- Biology
- Psychophysics
- Art

History

Crude graphics dates from early days of computing:

- Plotting on line printers
- Whirlwind Computer — 1950, MIT
  - CRT displays
- SAGE air-defense system — middle 1950's
  - "Command and control system"
  - Light pens
- Sketchpad — 1963, Ivan Sutherland
  - Constraint-based
  - Interaction techniques for choosing, pointing, drawing
  - Data structures for replicating components
  - Beginning of modern interactive graphics

Cathode ray tubes (CRTs)

Cathode ray tube (H&B, fig 2-4)

Consists of:

- Electron gun
- Electron focusing lens
- Horizontal and vertical deflection amplifiers
- Electron beam
- Anode (phosphor coating)

CRTs, cont.

Brightness comes from:

- Flourescence (fraction of μsec)
- Phosphorescence (10 to 60 μsec)

Different phosphors have different:

- colors
- persistence

Image needs to be *refreshed* to avoid *flicker*:

- Typically, need at least 60Hz
- Exact number depends on:
  - Image intensity
  - Ambient lighting
  - Wavelength
  - Size of screen
  - Observer

Calligraphic displays

Calligraphic display (H&B, fig 2-9)

"Calligraphic display" ≡ "Vector display"
≡ "Stroke display"
≡ "Random scan display"

Calligraphic displays, cont.

Used, for example, by:

- Sutherland's Sketchpad
- Asteroids video game
- Hypercube film

Raster displays

ras.ter, from radere, to scrape

History:

- Developed by Dick Shoup — early 70's, Xerox PARC
- Based on TV technology
- Has had huge impact on computer graphics

Raster displays, cont.

Raster display (H&B, fig 2-7)

Description:

- Electron beam traced in "raster scan order"
- Each left-to-right trace called a "scan line"
- Intensity of beam modulated according to "framebuffer" contents
- Each element of framebuffer associated with a "pixel" on the screen

Resolution

Resolution is determined by:

- Number of scan lines
- Number of pixels per scan line
- Number of bits per pixel

Examples:

Bitmapped display	$960 \times 1152 \times 1b$	1/8 MB
NTSC TV	$640 \times 480 \times 8b$	1/4 MB
Color workstation	$1280 \times 1024 \times 24b$	4 MB
Laser-printed page		
300dpi	$8.5 \times 11 \times 300^2 \times 1b$	1 MB
1200dpi	$8.5 \times 11 \times 1200^2 \times 1b$	17 MB
Film	$4500 \times 3000 \times 30b$	50 MB

Aspect ratio

"Frame aspect ratio" = horizontal / vertical size

TV	4 : 3
HDTV	16 : 9
Standard paper	$8.5 : 11 \approx 3 : 4$
35mm film	3 : 2
Panavision	2.35 : 1

"Pixel aspect ratio" = pixel width / pixel height
(Hopefully always 1!)

Interlacing

Raster scan (Foley, fig 1.3)

To reduce bandwidth in broadcast television, the refresh cycle is broken into two "fields":

- odd and even
- each lasting 1/60th second

Q: When does this work well?

Q: What's a worst-case example?

Color CRT monitors

The shadow mask CRT (H&B, fig 2-10)

Most color monitors employ "shadow mask" technology:

- Uses "triads" of red, green, and blue phosphors at each pixel
- Uses three electron guns, one for each color
- Uses "shadow mask"

Such CRTs are also known as "RGB monitors."

Additive color mixing

The RGB cube

All colors on a monitor are produced using combinations of red, green, and blue.

A monitor that allows 256 voltage settings for each of R, G, B is known as a "full-color system."

The description of each color in framebuffer memory is known as a "channel."

Specifying color

The number of color choices depends on the amount of framebuffer storage allocated per pixel.

Q: How many colors can be displayed with:

- 3 bits per pixel?
- 8 bits per pixel?
- 24 bits per pixel?

Color tables

"Color tables" allow more color versatility when you only have a few bits per pixel.

Using a color table (H&B, fig 4-16)

Idea:

- Each framebuffer element stores an *index* into a color table, where the actual color is stored.
- Color table entries can be changed in software.

Color tables, cont.

Example: What would be a "good" choice of colors to put into a color table if the hardware has 3-bit indices?

Example: What would be a "good" choice of colors for displaying a greyscale image on "color-mapped" hardware with 8-bit indices?

Color tables on 24-bit systems

Even "full color" systems with 24 bits per pixel often use color tables.

In this case, there is a separate color table for each 8-bit channel:

Using color tables for each channel

Q: Why would you want this capability?

Perceptual light intensity

We perceive light intensity as we do sound: on a *logarithmic* scale.

Example: The perceived difference between 0.20 and 0.22 is the same as between 0.80 and _____.

Ideally, to display $n+1$ equally-spaced intensity levels, the intensity levels on the monitor should be spaced as (14-20):

$$\frac{I_1}{I_0} = \frac{I_2}{I_1} = \cdots = \frac{I_n}{I_{n-1}} = r$$

Example: Suppose $I_0 = 1/8$ and $n = 3$. What are the four intensity levels to be displayed?

For color systems, the values of I_0 and r may be different for each channel.

Gamma correction

Another problem with calculated intensities is related to the non-linearity of the monitor.

A typical "monitor response curve" looks like this:

(H&B, fig 14-31)

Q: How would you check if your monitor is linear?

Q: How might you use a color table to correct for any non-linearity?

Gamma correction, cont.

The monitor response curve is given by (14-26)

$$I = aV^{\gamma}$$

where:

- I is the displayed intensity
- V is the input voltage
- a, γ are constants that depend on the monitor

"Gamma correction": Use (14-27)

$$V_k = \left(\frac{I_k}{a}\right)^{1/\gamma}$$

where I_k is the intensity you would like to display

Gamma correction curve (H&B, fig 14-32)

Specifying a view

How do you specify a view of a scene?

The "window" and "viewport" (H&B, fig 6-1)

Most graphics systems let you specify:

- the part of a picture to display (the "window")
- the place to display that picture on the screen (the "viewport")

Specifying a view, cont.

Typically, the picture is defined in any convenient coordinate system — called "world coordinates."

The viewport is generally specified in coordinates in $([0,1],[0,1])$ — called "normalized device coordinates" (NDC).

Ultimately, these coordinates are mapped to integer pixel coordinates — also known as "device coordinates" or "screen coordinates."

The window-to-viewport transformation

(H&B, fig 6-5)

If we want to keep the same relative placement of (xw, yw) in the window to (xv, yv) in the viewport, then we must require (6-2):

The window-to-viewport transformation, cont.

Solving for viewport position (xv, yv), we have (6-3):

$$xv = xv_{\min} + (xw - xw_{\min})\, sx$$

$$yv = yv_{\min} + (yw - yw_{\min})\, sy$$

where the scaling factors are (6-4):

$$sx = \frac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}}$$

$$sy = \frac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}}$$

Q: What condition ensures that relative proportions are maintained in the window-to-viewport transformation?

The 3D synthetic-camera model

Imaging system (Angel, fig 1.17)

The "synthetic-camera model" is a paradigm for creating images of 3D geometry.

Involves two components, specified independently:

- objects (a.k.a. "geometry")
- viewer (a.k.a. "camera")

Imaging with the synthetic camera

Imaging (Angel, fig 1.19)

The image is rendered onto an "image plane" or "projection plane" (in front of camera).

"Projectors" emanate from the "center of projection" (COP), at the center of the lens.

The image of an object point P is at the intersection of the projector through P and the image plane.

Clipping

Clipping (Angel, fig 1.20)

We think of the image plane as having a finite (rectangular) extent.

Objects are "clipped" to a "clipping rectangle" or "clipping window."

Graphics API's

Application programmer's model of software (Angel, fig 1.22)

An "application programmer's interface" (API) provides an interface between the application code and the hardware.

Most popular graphics API's (OpenGL, PHIGS, GKS-3D) are based on the synthetic camera model.

Have functions to specify:

- objects
- the viewer
- light sources
- material properties

OpenGL objects

Most API's support several different geometric "primitives."

OpenGL provides:

- points (`GL_POINTS`)
- line segments (`GL_LINES`)
- polylines (`GL_LINE_STRIP`)
- unfilled polygons (`GL_LINE_LOOP`)
- filled polygons (`GL_POLYGON`)
- triangles (`GL_TRIANGLES`)
- quadrilaterals (`GL_QUADS`)
- filled polygons (`GL_POLYGON`)
- strips (`GL_TRIANGLE_STRIP`, `GL_QUAD_STRIP`)
- fans (`GL_TRIANGLE_FAN`)

It also lets you read and write pixels in the framebuffer.

Specifying a viewer

Camera specification (Angel, fig 1.23)

Camera specification requires four kinds of parameters:

- Position: The COP.
- Orientation: Rotations about axes with origin at COP.
- Focal length: Determines the size of the image on the film plane, or the "field of view."
- Film plane: Its width and height, and possibly orientation.

Specifying lights and materials

Light sources usually defined by:

- location
- strength
- color
- directionality

Materials usually defined by:

- various shading parameters
- texture maps

OpenGL rendering styles

OpenGL supports a variety of rendering styles:

- Wireframe
  - with depth-cueing
  - with antialiasing
- Visible polygons
  - with flat shading
  - with smooth ("Gouraud") shading
  - with texture maps and shadows
  - with motion blur
  - with atmospheric effects

The geometric pipeline

Geometric pipeline (Angel, fig 1.29)

Many commercial graphics workstations use a "pipeline" architecture, implemented in hardware, for processing geometry.

Works well because:

- Lots of data that is processed similarly
- Well-decomposed computation

Q: What's the downside of large pipelines?

The graphics pipeline

The pipeline metaphor can be extended to encompass just about everything we do in 3D graphics:

Animation
⋮
Modeling
⋮
Transformation
⋮
Clipping
⋮
Lighting and shading
⋮
Hidden surface
⋮
Projecting
⋮
Rasterizing
⋮
Compositing
⋮
Post-processing

Summary

Here's what you should take home from this lecture:

1. All the terms in quotations.
2. Sketchpad (1963) was the first interactive graphics system.
3. The basic components of a B&W or color CRT.
4. The difference between raster and calligraphic displays.
5. How to compute screen resolution.
6. The one-to-one correspondence between elements of framebuffer memory and pixels on the screen.
7. How red, green, and blue are mixed to form different colors.
8. How color tables work.
9. How to map from computed intensity to perceived intensity.
10. The reason for gamma correction and how it is computed.

11. How to compute the window-to-viewport transformation.
12. The basic idea of the synthetic-camera model and how its basic components are specified.
13. The basic concept of the geometric and graphics pipelines.